INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA

Portal de Emprego Inteligente

HENRIQUE MANUEL MOREIRA GONÇALVES
Outubro de 2014

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Portal de Emprego Inteligente

Henrique Manuel Moreira Gonçalves

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Tecnologias do Conhecimento e Decisão

Orientador: Professor Doutor Nuno Malheiro

Júri:

Presidente:

[Nome do Presidente, Escola]

Vogais:

[Nome do Vogal1, Escola]

[Nome do Vogal2, Escola] (até 4 vogais)

Porto, Outubro 2014

Dedicatória

Dedico este trabalho à minha mulher Daniela, pelo sentido e luz que traz à minha vida.

Resumo

A importância da internet é hoje uma realidade cuja facilidade que nos traz de aceder a produtos ou serviços, informação ou até mesmo aproximar pessoas, torna-a ainda mais indispensável. Cada vez mais, a nossa vida é feita através internet. Seja uma simples consulta de informação de horário de funcionamento de uma loja, a compra de produtos de uma loja usando plataformas de venda on-line, transações bancárias ou até operações fiscais, a internet faz parte das nossas vidas. Até novas áreas de negócio surgem com a massificação do uso da internet.

Naturalmente, o surgimento de plataformas que repliquem o mundo real no mundo virtual torna-se bastante óbvio e cada vez mais desejado.

A pesquisa de emprego é algo bastante comum no mundo real. Naturalmente, com a internet, surgiram e surgem plataformas dedicadas a esta área. As empresas que disponibilizam empregos recorrem-se destas plataformas pois, estão ao alcance de muitos utilizadores e, geralmente, são gratuitas, juntando o melhor de dois mundos.

A necessidade de atingir com maior eficácia o publico alvo leva a que surjam plataformas com maior granularidade de áreas de emprego ou então especializadas em determinadas áreas. Contudo, a pesquisa nestas plataformas fica aquém do desejado pois não tem em consideração a relevância de um emprego para o utilizador apresentando resultados irrelevantes.

No sentido de oferecer um novo paradigma de pesquisa de empregos, criou-se uma plataforma, dotada de conhecimento, que estende a pesquisa o tipo de pesquisa tradicional obtendo mais resultados com muita relevância para o utilizador.

Palavras-chave: Sistemas Periciais, Relaxamento de Restrições, Relevância de resultados

Abstract

The internet relevance is a reality that easily allow us to access products or services, information and even brings people togheter, makes it even more indispensable. Our day to day actions are more and more made online. Be it a simple search for a store opening hours, shopping, bank transactions or even governement actions, the internet is there. New business areas emerge from the internet.

Naturally, new platforms, that somehow replicate the real world in the virtual one, is somewhat obvious and wanted.

Searching for a job is very common in the real world. With the internet, plataforms taylored for this field emerged. Many companys display the job openings using these platforms as they open for many users and they are free, wich is the best of both worlds.

With the necessity to target an audience with greater efficiency, new platforms are developed with more granularity in jobs fields or are developed targeting a specific field. Although, searching jobs in these platforms, falls short from the desired results as they don't take in consideration the user preferences.

Trying to improve this situation, a new job searching platform was developed, knowledge based, that improves search by relaxing query contraints and calculating the result relevance.

Keywords: Expert Systems, Query Relaxation, Result Relevance

Agradecimentos

Aos meus pais, Gabriel e Fernanda, por tudo o que fazem e fizeram por mim ao longo de todos estes anos, pelas dificuldades que passaram para me ajudarem a encontrar o meu caminho;

Ao meu irmão Gabriel pela amizade e amor incondicional com que sempre acompanhou a minha vida;

A minha mulher, Daniela, por ser a minha força inspiradora;

A minha tia Lucinda e ao José Neves pelo grande exemplo de vida que são e por ter a felicidade de os ter como uns segundos pais;

Ao Tiago Pinho por toda a amizade e acompanhamento que sempre me dedicou;

Ao meu orientador, Professor Doutor Nuno Malheiro, por todo apoio, dedicação e paciência depositados para que pudesse concluir este trabalho;

Ao Hugo Correia, Ricardo Moreira e Nuno Silva pelo acompanhamento deste e de outros projetos;

Ao Ricardo Ferreira e João Silveira por todo o apoio e acompanhamento ao longo do Mestrado;

A toda a minha família que sempre me acompanhou;

A todos os que direta ou indiretamente contribuíram para a realização deste trabalho.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Fórmulas

Lista de Blocos de Código

Acrónimos

Lista de Acrónimos

CV	*Curriculum Vitae*
JSON	*Javascript Object Notation*
REST	*Representational State Transfer*
SOAP	*Simple Object Access Protocol*
SOA	*Service Oriented Architecture*
XML	*Extensible Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
URI	*Uniform resource identifier*
YAML	*YAML Ain't Markup Language*
CSV	*Comma-separated Value*
SQL	*Structured Query Language*
NoSQL	*Not only SQL*
EJB	*Enterprise Java Bean*
DTO	*Data Transfer Object*
J2EE	*Java Enterprise Edition*
WSDL	*Web Services Description Language*
UDDI	*Universal Description Discovery and Integration*
RPC	*Remote Procedure Call*
IT	*Information Technology*

1 Introdução

Ao longo das últimas décadas, o mundo tem vindo a assistir a uma nova era na história da humanidade que veio influenciar a forma como os povos se ligam e comunicam. Com o decorrer do tempo esta será, possivelmente a época que será marcada na história como a época de transição da contemporânea para a próxima.

Fizeram-se grandes avanços tecnológicos que melhoraram a vida das pessoas, que num curto espaço de tempo viram a sua esperança média de vida aumentar significativamente, assim como a melhoria dos transportes, o acesso a informação que outrora não estava a acessível tão rápido e tão facilmente, entre as mais diversas áreas que também foram abrangidas.

Contudo, nos últimos anos, a Internet foi das tecnologias que mais influenciou a vida do cidadão comum. Inicialmente, esta era usada para partilha de informação usando um modelo bastante simples mas com o passar dos tempos, foi-se desenvolvendo para uma enorme rede que liga milhões de pessoas em qualquer parte do mundo.

As aplicações *web* acompanharam também esta evolução, sendo atualmente muito mais dinâmicas e interativas. Não vai há muito tempo que as aplicações *web* não passavam de páginas onde a informação era disponibilizada de um modo estático não suportando grande interatividade com o utilizador. Atualmente as aplicações tem crescido em tamanho, complexidade e no número de tarefas que permitem efetuar, permitindo a interação com o utilizador, onde é possível realizar as mais diversas operações. Realizar operações bancárias, comprar em lojas virtuais ou publicar em redes sociais, são tarefas consideradas normais no dia-a-dia do utilizador. Com este crescimento, cresce também a necessidade das infraestruturas de responderem as estes novos desafios, havendo necessidade de desenvolver novas metodologias, linguagens de programação, métodos de armazenamento de dados e novos componentes de hardware cada vez mais eficazes.

Com o crescimento das operações, outrora realizadas em papel, agora realizadas num suporte online, ou de novos serviços, surge naturalmente a necessidade de dotar os sistemas de novas tecnologias capazes de lidar com mais informação sabendo o que fazer com ela de forma a

optimizar a aplicação e melhorar a experiência do utilizador. Aos poucos, começaram a introduzir novas componentes nas aplicações para dotar o sistema de alguma "inteligência", como sugestões em lojas de comércio electrónico. Este seria um primeiro passo para a otimização da experiência do utilizador. Esta característica começou a ser cada vez mais partilhada pelas diversas aplicações, sendo cada vez mais melhorada, até ao ponto de o próprio sistema saber apresentar produtos relacionados sem que estes tenham de ser especificados em bases de dados pelos responsáveis pela aplicação. Esta característica de sugestões é usada comummente pelo Homem no dia-a-dia e é uma mais valia quando aplicada às ferramentas por ele usadas, mais ainda com pouca expressividade noutras aplicações *web*.

A procura de emprego tem vindo a aumentar diariamente, em especial na vertente *online*. Existem diversas plataformas de oferta de emprego, onde é possível pesquisar ou publicar anúncios. No entanto, estas soluções são limitadas no toca à procura eficiente de propostas de emprego para um perfil de utilizador.

O trabalho aqui apresentado centra-se na construção da base de uma aplicação de pesquisa e criação de ofertas de emprego e de uma ferramenta de sugestões de empregos relevantes para o utilizador, recorrendo a novas metodologias e tecnologias, referenciando os métodos usados.

1.1 Enquadramento

A pesquisa de emprego é algo bastante comum no mundo real. Naturalmente, com a internet, surgiram e surgem plataformas dedicadas a esta área. As empresas que disponibilizam empregos recorrem a estas plataformas pois estão ao alcance de muitos utilizadores e, geralmente são gratuitas, juntando o melhor de dois mundos.

A necessidade de atingir com maior eficácia o publico alvo leva a que surjam plataformas com maior granularidade de áreas de emprego ou então especializadas em determinadas áreas. Contudo, a pesquisa nestas plataformas fica aquém do desejado pois não tem em consideração a relevância de um emprego para o utilizador. Assumem simplesmente todos os empregos de uma área para uma localização, tipicamente separada por distritos.

Este trabalho foi desenvolvido no âmbito do Mestrado em Engenharia Informática, ramo de Tecnologias do Conhecimento e Decisão, motivado pela necessidade de um sistema de ofertas de emprego dotado de conhecimento de cada área de emprego, capaz de calcular a relevância de um emprego em função dos atributos disponíveis como sendo o salário, distância e localização geográfica.

1.2 Objectivos

O presente trabalho tem como objectivo a implementação de uma plataforma de pesquisa de ofertas de emprego capaz de apresentar ao utilizador as ofertas mais relevantes em função dos seus atributos como a categoria do emprego, a distância e o salário, optimizando assim a sua pesquisa.

A plataforma pretende-se capaz de evoluir e estar disponível noutras nos mais diversos meios, seja pelo *browser*, aplicações de telemóvel ou como um motor para integração noutros sistemas. Assim, uma arquitetura orientada a serviços (*web services*) facilitará a integração pelos diferentes meios.

Para se obter mais resultados, para além dos restringidos inicialmente pelo utilizador para que seja possível avaliar a relevância dos mesmos posteriormente, é necessário relaxar as restrições baseando em conhecimento adquirido previamente

O cálculo da relevância de um emprego é feito a partir da aproximação de cada atributo do mesmo à pesquisa efectuada inicialmente. Este cálculo, assim como o relaxamento de restrições, baseia-se em conhecimento adquirido junto de um perito tornando este sistema um sistema baseado em conhecimento.

1.3 Planeamento

O planeamento deste trabalho foi estruturado em 4 fases.

1. Requisitos
    a. Levantamento de requisitos
    b. Estado da arte
2. Investigação
    a. Arquiteturas
    b. Sugestões e relevância de dados de uma pesquisa
3. Protótipo
    a. Implementação dos *web services*
    b. Implementação do calculo de relevância
    c. Desenvolvimento da *interface*
4. Elaboração da dissertação
    a. Redação final da dissertação

Contudo a redação final da tese foi sendo efetuada juntamente com as diversas partes da investigação.

1.4 Estrutura da dissertação

Este documento encontra-se organizado em cinco capítulos.

No primeiro capítulo é apresentada a Introdução. Este contém uma visão generalizada das motivações que levaram a realizar o trabalho apresentado

O segundo capítulo apresenta o estado da arte, no que diz respeito aos sistemas existentes, as suas forças e fraquezas, métodos de pesquisa e suas restrições e arquitetura de *software*.

No terceiro capítulo é descreve-se como se encontra desenvolvido o Portal de Emprego Inteligente. É contemplada a sua arquitetura e como é efetuada uma pesquisa e calculada a relevância dos resultados encontrados.

O quarto capítulo descreve os resultados obtidos com a arquitetura do Portal assim como os resultados obtidos numa pesquisa de emprego utilizando o Portal de Emprego Inteligente comparativamente com uma pesquisa tradcional.

No quinto capítulo são apresentadas as conclusões do trabalho realizado, uma recapitulação dos resultados obtidos, as críticas ao trabalho, assim como perspectivas e trabalho futuro a realizar para melhorar a plataforma.

2 Estado da arte

A pesquisa de emprego sempre fez parte da vida do ser humano. Ao longo dos anos assistiram-se a diferentes evoluções sobre a forma como as ofertas chegam aos possíveis novos colaboradores. Desde anúncios publicados em montras de lojas a semanários com secções específicas de ofertas. A pesquisa era feita nas mais diversas fontes pois não existia um ponto central de referencia.

Com o advento das novas tecnologias, um passo natural consistiu na representação online das ofertas que se encontravam em papel. Com o passar do tempo e o aumento de utilizadores na internet, a criação de portais de emprego foi uma evolução natural, permitindo aos utilizadores efetuar pesquisas dos anúncios presentes nas bases de dados dos mesmos.

Da mesma forma, estes portais foram evoluindo, permitindo também às empresas colocar os seus próprios anúncios, constituindo uma ferramenta bastante relevante no recrutamento pois é fácil e bastante acessível, se não mesmo gratuita.

Começaram a ser desenvolvidos mais portais, dedicando-se a anúncios das diferentes áreas ou então a centrar-se em pequenos nichos, tentando destacar-se dessa forma.

2.1 Um aplicação típica de ofertas de emprego

Para construir uma nova aplicação *web* de ofertas de emprego, ou de outra área, é necessário efetuar-se uma análise às já existentes, de forma a poder traçar um "perfil" básico de uma aplicação e suas funcionalidades.

Assim, um aplicação de ofertas de emprego, tipicamente, apresenta as seguintes funcionalidades:

2.1.1 Pesquisa

As aplicações permitem a pesquisa de emprego por zona geográfica, apenas com detalhe até ao distrito, por categoria profissional ou área de emprego, e permitem também o uso de palavras-chave.

2.1.2 Outras funcionalidades

As aplicações que permitem o registo quer de candidatos, quer de empresas, apresentam mais algumas ferramentas disponíveis ao utilizador.

Aplicações que permitem o registo de utilizadores, guardam informação pessoal sobre este, permitem guardar o seu CV para que este possa ser consultado pelas empresas também registadas no sistema, têm também um sistema de mensagens e notificam aos utilizadores novas ofertas no sistema.

As aplicações que permitem o registo de empresas apresentam também poucas funcionalidades, destacando-se o registo de novas ofertas de emprego e a possibilidade de pesquisar CVs.

Estas funcionalidades não são de grande relevância para o Portal de Emprego Inteligente.

2.2 Aplicações existentes em Portugal

Existem diversas aplicações *web* destinadas à pesquisa de ofertas de emprego em Portugal sejam portais específicos ou apenas agregadores de ofertas de outros portais. Com uma rápida pesquisa num motor busca como o Google, encontram-se referencias para diversos sites, uns com maior dimensão e relevância, outros mais pequenos ou até mesmo *blogs* com algumas ofertas de emprego.

2.2.1 Portais em Portugal

Dos portais portugueses existentes destacam-se os mais maiores e mais usados.

2.2.1.1 Net Empregos

Este portal abrange várias áreas de emprego e permite o registo quer de utilizadores quer de empresas.

Perfil de pesquisa de anúncios por zona geográfica, categoria profissional e palavras chave.

2.2.1.2 ITJobs

Este portal de empregos é específico para a área das Tecnologias de Informação. É bastante limitado em termos de funcionalidades pois apenas permite pesquisar empregos por zona geográfica ou palavras chave, sem apresentar mais nenhum filtro.

Perfil de pesquisa de anúncios por palavras chave ou por localidade

2.2.1.3 InfoJobs

A pesquisa no InfoJobs é a bastante simples, não filtrando com grande precisão os resultados.

Perfil de pesquisa de anúncios por zona geográfica e categoria profissional.

2.2.1.4 Empregos online

Neste portal é obrigatório registo para que sejam apresentados os contactos do anunciante. Permite alguns filtros na pesquisa, contudo apenas se pode usar um de cada vez.

Perfil de pesquisa de anúncios por zona geográfica, categorias de emprego e palavras chave

2.2.1.5 Carga de trabalhos

O Carga de trabalhos é mais vocacionado para a área de comunicação (design, jornalismo, publicidade, marketing, multimédia) e também de programação. Apenas é possível pesquisar por categorias de emprego.

2.2.1.6 Expresso Emprego

O portal Expresso Emprego apresenta, para além dos atributos mais comuns de pesquisa de emprego uma pesquisa avançada onde se pode acrescentar alguns filtros.

Perfil de pesquisa de anúncios por zona geográfica, categorias de emprego, palavras chave, função (pesquisa avançada), sector (pesquisa avançada) e intervalo de tempo (pesquisa avançada – últimos 7 dias)

2.2.1.7 Portal Emprego

O Portal Emprego apresenta alguns filtros habituais de pesquisa.

Perfil de pesquisa de anúncios zona geográfica, tipo de emprego, categorias de emprego e palavras chave.

2.3 Evolução das aplicações

Com a massificação do uso de tecnologias de informação, onde cada vez mais humanos e máquinas comunicam entre si, gerando mais e mais novas informações, naturalmente, surge a necessidade de ter acesso a estas e explora-las.

O rápido crescimento de dados disponível em base de dados, apresentam novos desafios na sua exploração, quer a nível de infraestruturas com mais capacidade de armazenamento e disponibilidade em vários locais e máquinas, quer a nível da sua exploração, para que seja mais rápida e para que a informação obtida seja o que se procura.

Na interação homem-máquina. existem cada vez mais aplicações disponíveis, com diversas funcionalidades, passando o mundo real para o virtual, surgindo assim lojas online, serviços online, portais das diversas áreas que constituem o dia-a-dia do ser humano, entre uma infinidade de aplicações que permitem que os seus utilizadores obtenham o que desejam, e, naturalmente, gerando mais e mais informação.

A riqueza de informação presente numa base de dados de uma aplicação, como as referidas anteriormente, é difícil de explorar pelos seus utilizadores, pois geralmente, apenas disponibilizam acesso indireto a dados representados num formulário de pesquisa convencional. Quando um utilizador faz uma pesquisa usando um formulário deste género, geralmente está limitado à partida no âmbito da mesma pois, as opções existentes num formulário deste género, restringindo os resultados obtidos.

Pode considerar-se que este é o comportamento esperado numa aplicação, mas, nem sempre pode corresponder ao melhor resultado. O primeiro resultado obtido numa pesquisa nem sempre é o mais relevante.

Há muitas situações em que é difícil de traçar-se uma linha onde um resultado deixa de pertencer a uma categoria e passa pertencer a outra. Usando um exemplo simples, típico do dia-a-dia, havendo necessidade numa aplicação de categorizar eventos pela temperatura ambiente por exemplo, esta pode ser categorizada, por: dos zero graus aos 10 graus celsius ser uma temperatura fria, dos 10 graus aos 19 graus celsius amena e dos 20 graus em diante quente. Assim, pode fazer-se pesquisas sobre determinados eventos em função da categoria da temperatura obter eventos "quentes", "frios", "amenos" ou as demais categorias que se considerarem necessárias. Mas, na realidade, dizer-se que aos 19 graus celsius é ameno e aos 20 graus celsius é quente, assim como afirmar-se que 20 graus ou 30 graus celsius é igualmente quente é na verdade incorreto pois é uma representação demasiado "quadrada" do mundo real. Pode optar-se por aumentar o número de categorias diminuindo a amplitude térmica que estas representam mas, isso acarretaria outros problemas que implicitamente dificultaria uma pesquisa. O ideal seria de alguma forma fazer o que os seres humanos fazem inconscientemente. Em vez de dizer-se que dos 19 graus para os 20 graus celsius a temperatura muda para “quente”, seria mais correto dizer-se que começa a “ficar mais quente”.

Este problema é válido para imensas áreas cuja representação da realidade não pode ser determinada absolutamente. Acrescenta-se a este problema outras situações como a pesquisa por algo com diferentes características cujo o seu conjunto determina a sua relevância.

2.4 Relaxamento de restrições

Tipicamente, quando se faz uma pesquisa num sistema ou plataforma comum, de referir que não se está a falar de ambientes ditos "inteligentes", a informação recebida resulta em dois cenários: é devolvida uma lista com os resultados que pertençam aquela pesquisa onde um ou mais critérios satisfaçam a situação, dependendo se a pesquisa é booleana ou não; ou é devolvida uma lista vazia, cenário conhecido também como *failing query*.

Uma situação que daqui advém, e que não é à primeira vista evidente, reside no facto de que se se aumentar o rácio de pesquisa, ou seja, se se forem alterando parâmetros (relaxando-os) de forma a abranger áreas semelhantes à pesquisa inicial, nova informação pode ser obtida e esta revelar-se igualmente satisfatória e relevante.

Assim, um utilizador irá, naturalmente, realizar novas pesquisas relaxando as restrições manualmente para obter resultados aproximados. Esta tarefa torna-se entediante, frustrante e consome muito tempo com as combinações das diferentes restrições. Um relaxamento de restrições exagerado pode retornar uma lista enorme de resultados que, muitas vezes, podem nem ser relevantes.

A modificação de *queries*, ou relaxamento de restrições, é estudado há muito tempo em bases de dados e obtenção de informação [Baeza-Yates and Ribeiro-Neto,1999] [Chakrabarti et al.,2003] [Chaudhuri,1990] [Chaudhuri and Gravano,1999] [Godfrey,1998] [Janas,1979].

A investigação realizada neste âmbito tenta responder a dois problemas do modelo de pesquisa precisa: o problema da resposta vazia, ou *failing query*; o problema de muitas respostas.

Existem diversos sistemas propostos para adereçar os problemas de reposta vazia ou muitas repostas como CO-OP [Kaplan,1982], CoBase [Chu et al.,1996a] [Chu et al.,1996b] ou LOQR [Muslea, 2004] com diferentes tipos de funcionamento. Uns sistemas propõe o relaxamento de restrições automático, quer quando obtém resposta vazia efetuam nova *query* com as restrições mais abrangentes (relaxadas) ou efetuam imediatamente a primeira *query* já relaxada. Estes sistemas implicam que haja conhecimento que deve ser adquirido previamente (offline) acabando por também limitar um pouco a sua capacidade.

Outros sistemas como o CoBase ou o LOQR, usam técnicas de *machine learning* para efetuar o relaxamento de restrições automaticamente, sem estarem limitados a conhecimento prévio da área.

2.4.1 Relaxamento de restrições nas plataformas de emprego em Portugal

As plataformas de pesquisa de emprego em Portugal apresentam apenas uma pesquisa direta pela área de emprego e a sua localização. A localização está categorizada por distrito não tendo grande granularidade. Existem plataformas que são especializadas numa determinada área, como, por exemplo, o ITJobs para a área de informática, Carga de Trabalhos na área de design e comunicação. O ITJobs apenas têm empregos relacionados pela área, não categorizados. Na carga de trabalhos já existe alguma granularidade na área de emprego subdividindo um pouco mais as suas categorias. Contudo, mesmo nestes casos é uma divisão ainda com pouco detalhe obtendo-se resultados irrelevantes. Estes resultados apresentados aos utilizadores são, à priori, restringidos pelo distrito. A sua ordem da apresentação é pelo resultado ou entrada mais recente na base de dados. Assim o emprego que é apresentado em primeiro lugar pode até nem ter nada a ver com o que se pretende encontrar.

Considerando a dispersão da população pelo país, é evidente que não habitam simplesmente no centro dos respectivos distritos mas, nas suas periferias também. É esperado que, os indivíduos residentes nestes áreas pesquisem também empregos nos distritos circundantes. Esta é uma situação em que o relaxamento de restrições é feito pelo utilizador.

2.5 Sistemas Periciais

Os sistemas periciais têm por objectivo resolver problemas complexos de maneira idêntica à utilizada pelos peritos humanos. Num Sistema Pericial, o conhecimento é obtido a partir de um ou mais peritos.

O conhecimento, presente num sistema pericial, é um conjunto integrado de factos e relações que quando devidamente interpretado, produz um desempenho eficiente.

Existem diversos Sistemas Periciais nas mais diversas áreas. O DENDRAL, primeiro sistema pericial [Lindsay et al., 1980], trata espectros de massa e respostas magnéticas ao nível nuclear de modo a fornecer informação sobre a estrutura molecular de compósitos desconhecidos. MYCIN, Sistema Pericial mais conhecido, detém conhecimento sobre parâmetros de análises sanguíneas para diagnosticar automaticamente problemas relacionados com infecções bacterianas [Davis et al., 1977]. Muitos outros sistemas existem para resolver os mais diversos problemas.

O perito é o elemento central num sistema pericial. É o especialista que detém competência e conhecimento profundo sobre um domínio e sabe quando, onde, como e porquê. Alguns contratempos podem surgir no momento de o perito fornecer a informação pois pode ter

dificuldade em expressar-se, ter receio de explicitar o conhecimento, não acreditar no projeto entre outros demais problemas.

Um sistema pericial que envolva vários peritos pode apresentar uma maior cobertura do domínio.

Um sistema pericial apresenta vantagens perante um perito dado este ser um ser humano e estar sujeito a *stress* ou bloquear o raciocínio ou ser alguém bastante ocupado.

2.6 Arquitetura de software

O processo de definir formalmente a estrutura de um sistema, as suas funções, propriedades e *interfaces*, constitui uma arquitetura. Durante este processo o sistema é decomposto nos seus elementos estruturais, subsistemas e partes, tendo em conta a produtividade de desenvolvimento, flexibilidade e sua escalabilidade de forma a acomodar facilmente novas funcionalidades com pouco impacto no sistema e a um custo reduzido. Com uma boa decomposição do sistema, facilmente se obtêm poucas dependências entre as partes, simplificando o problema, dividindo-o em pequenas partes que podem ser usadas separadamente. Esta característica é conhecida como *Loose coupling*.

2.6.1 Porque é que uma arquitetura é importante?

Tal como qualquer estrutura complexa, o *software* deve ser construído com uma base sólida. Se não houver uma preparação para cenários chave, problemas comuns, consequências a longo prazo, o produto pode estar em risco. Se esta estrutura for pobre, o *software* torna-se instável, difícil de manter e pouco tolerante a novos desenvolvimentos.

O sistemas devem ser desenhados levando em consideração o utilizador, o sistema e infraestrutura e o objectivo do negócio.

Um arquitetura centra-se nos elementos e componentes globais do sistema, como estes são usados e como interagem. Algoritmos, estruturas de dados, detalhes de componentes individuais são aspectos de *design*. Contudo, faz sentido combinar ambas as áreas pois em alguns casos as decisões afectam a arquitetura assim como outras afectam o design.

2.6.2 Os objetivos de uma arquitetura

Um arquitetura de software tem por objetivo fazer a ligação entre os requisitos da área do negócio com os requisitos técnicos por meio de *use cases* para encontrar forma de os implementar. Uma boa arquitetura é suficientemente flexível para suportar as alterações que ocorreram com as tecnologias de *hardware* e *software*, assim como as de requisitos de

negócio e outros cenários. As arquiteturas devem expor a estrutura do sistema mas esconder os detalhes da sua implementação.

2.7 Serviços

Um serviço é um componente de software que pode ser acedido através de uma ligação e que providencia as suas funcionalidades ao cliente. Este deve funcionar de uma forma independente do estado de outros serviços excepto nos casos de serviços compostos (*composite services*) e deve possuir também uma *interface* bem definida.

2.8 Caso de estudo

A empresa *InterLás*, grande empresa de plásticos no sector alimentar, decide comprar uma pequena e jovem promissora empresa, a *MundLás* que se dedica ao fabrico de plásticos para o ramo automóvel e frio. Entretanto, a *InterLás*, adotando uma estratégia de gestão, decide comprar a empresa *LásSupply*, que é uma empresa fornecedora de matéria prima e também fornecedora da *InterLás*.

As três empresas encontram-se em localizações geográficas distintas. Cada empresa tem implementado o seu próprio sistema de gestão com a sua própria arquitetura, assim como os diferentes sistemas de distribuição, bases de dados de clientes, de materiais e de funcionários. Isto origina que cada sistema esteja montado possivelmente em formatos diferentes.

Após a fusão das três empresas, houve a necessidade de interligar os sistemas para poder haver uma gestão central. Foi então necessário criar um sistema facilmente escalável, para que no futuro, se existir uma nova expansão, não seja necessário refazer o sistema, apenas desenvolver novas funcionalidades para juntar ao existente. Ao implementar um sistema capaz de integrar novas funcionalidades e que comunique facilmente com os restantes do grupo, aumenta-se também o valor das empresas.

Para adereçar este problema, foram desenvolvidos novas camadas em cima das aplicações existentes de forma a expor a informação para as outras aplicações sem implicar grandes alterações nas existentes. Foram então desenvolvidos serviços que, em cada aplicação sabem receber e guardar dados e os transmitem para os restantes, num formato que todas conhecem.

2.9 SOA

SOA traduzindo-se em arquitetura orientada a serviços, é um tipo de arquitetura de software onde as funcionalidades implementadas são disponibilizadas na forma de serviços. O termo SOA refere-se à arquitetura de um sistema e não à sua implementação.

SOA é uma evolução de computação distribuída baseado no paradigma de pergunta/resposta para aplicações síncronas ou assíncronas. A lógica do negócio ou funções são modularizadas e apresentadas como serviços para as aplicações clientes. O que é a chave para estes serviços é o *Loose coupling* pois o interface do serviço é independente da sua implementação. Programadores desenvolvem aplicações utilizando serviços, sem conhecerem sequer qual é a sua implementação [Lin et al., 2011] [Endrei et al., 2004].

Tal como demonstrado no caso de estudo, a realidade nas empresas é que a infraestrutura de IT é diferente, seja a nível de máquinas, sistema operativo e/ou aplicações. Algumas aplicações são usadas para o próprio negócio levando a que construir um sistema de raiz não seja uma opção. As empresas devem responder rapidamente a mudanças de negócio, potenciar investimentos existentes para responder a novos requisitos, suportar novos canais de comunicação com clientes, parceiros e fornecedores. Dado à natureza de *Loose coupling* associada ao *SOA*, permite a que as empresas desenvolvam novos serviços ou melhorem os existentes, de forma granular, para responder a novos requisitos, é providenciado a opção de fazer com que os serviços sejam consumidos em diferentes canais, salvaguardando a infraestrutura IT existente.

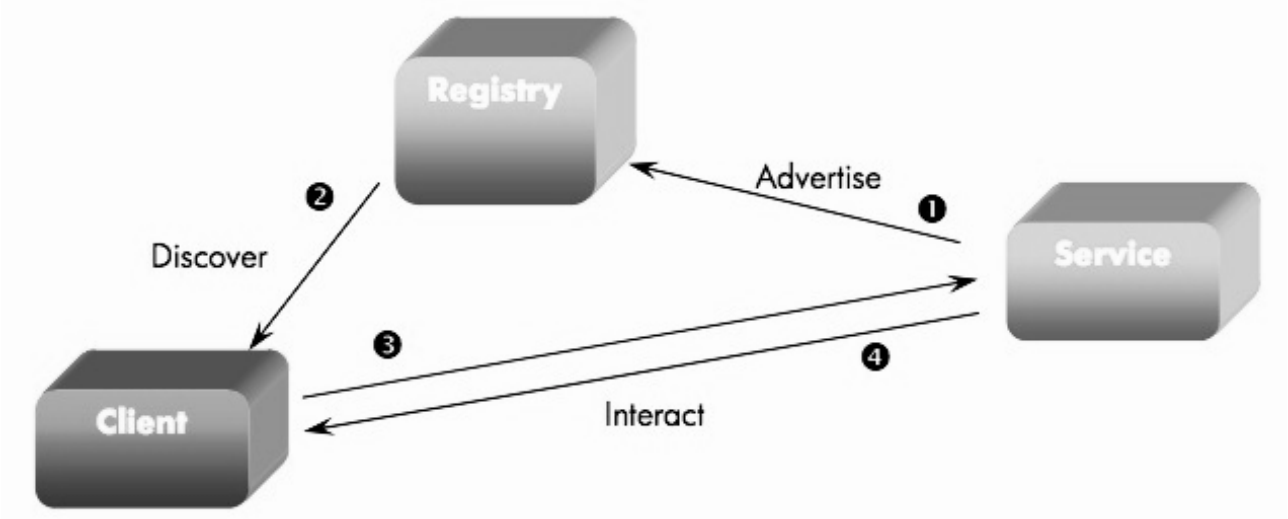

Figura 1 – Ciclo de interação de um pedido

A Figura 1 apresenta um ciclo de interação simples que começa com o serviço a dar-se a conhecer ao registo de serviços (1). Um potencial cliente, que pode ser ou não um serviço, vai procurar no registo por um serviço que satisfaça as suas necessidades (2). O cliente, após selecionar o serviço desejado envia-lhe um pedido, usando para isso um protocolo que seja

reconhecido por ambos (3). Por último o serviço responde, seja com o resultado, seja com uma mensagem de erro, ao cliente (4) [Endrei et al., 2004].

2.9.1 Infraestrutura

Para correr e manter aplicações SOA é necessário ter uma infraestrutura que suporte todos os *standards* relevantes e *runtime containers*.

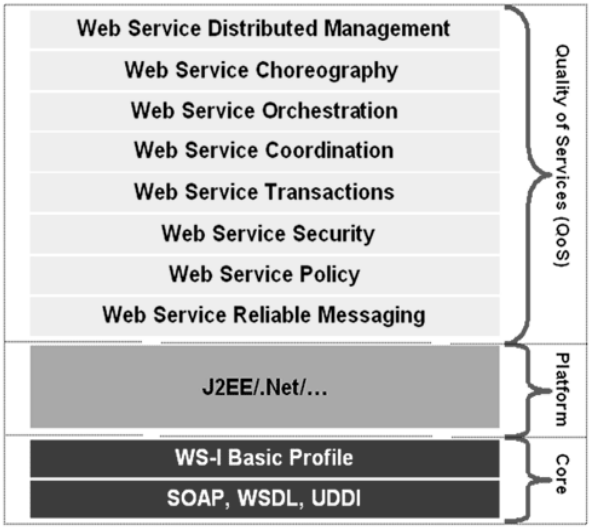

Figura 2 – Infraestrutura típica de SOA

SOAP, *WSDL*, *UDDI* são peças fundamentais numa infraestutura SOA. WSDL é usado para descrever o serviço, UDDI (*Universal Description Discovery and Integration*) é usando para registar e para o *look up* dos serviços, SOAP é a camada de transporte entre o serviço e o cliente.

WS-I Basic Profile está a tornar-se uma base nuclear necessária para testar a interoperabilidade dos serviços através de diferentes plataformas e tecnologias.

As plataformas J2EE e *.Net* são dominantes no desenvolvimento de aplicações SOA mas, não são as únicas. J2EE disponibiliza uma plataforma para os programadores já bastante madura, escalável, fiável e com grande performance com especificações para o *binding* de XML, como *JAX-WS* ou *JAX-RS*, que facilitam o desenvolvimento e *deploy* de *Web Services*.

Alguns sistemas críticos de empresas requerem outras necessidades como segurança e fiabilidade. Uma série de parâmetros devem ser usado de forma a garantir a qualidade do serviço.

2.9.2 Web Services

Hoje em dia é muito comum aceder-se à *web* a partir de um *web browser,* que fornece um *interface* orientado ao utilizador humano, para informação. Quando um utilizador efetua um pedido de uma página, este é tratado por um servidor *web* remoto que responde com a informação formatada em HTML. Este formato permite ao browser apresentar a informação usando uma seleção de fontes, cores, imagens, que torna mais fácil e apelativa a sua visualização por parte do utilizador humano.

Web services são componentes de software que providenciam informação às aplicações e não aos humanos através de um interface orientado para a aplicação. Esta informação é estruturada em XML para que possa ser analisada e processada facilmente.

Os *web services* publicam os detalhes das suas funções e interfaces mas mantêm a sua implementação privada. Além disto, um cliente e um serviço que usem o mesmo protocolo de comunicação podem interagir livremente independentemente das plataformas em que se encontram ou das linguagens nas quais foram programados.

Apesar de um *web service* suportar qualquer protocolo de comunicação, o mais frequente é SOAP através de HTTP ou HTTPS. Isto torna os *webservices* apelativos dado que HTTP e HTTPS encontram-se em todo o lado e normalmente não levantam problemas à *firewall* e permitem tráfego bidirecional.

2.9.3 REST *vs* SOAP

Web Services são o ponto chave de integração de diferentes aplicações pertencentes a diferentes plataformas, linguagens e sistemas. Tipicamente, estes *Web Services* eram construídos usando SOAP (*Simple Object Access Protocol*) que revolucionou RPC (*Remote Procedure Call)* e *Loose Coupling*, mas, recentemente, tem-se considerado que existe um método melhor para construir *Web Services* na forma de Transferência de Estado Representacional (REST).

REST é considerado mais como uma filosofia antiga do que uma nova tecnologia. A filosofia REST defende que os princípios e protocolos existentes da Web são suficientes para construir *Web Services* robustos. Isto significa que programadores que entendam HTML e XML são capazes de começar a desenvolver imediatamente sem necessitarem de usar ferramentas que vão para além do que utilizam normalmente para desenvolver uma aplicação Web.

Numa arquitetura REST, os recursos chave são identificados. Estes podem ser entidades, coleções ou o que arquiteto considere ser essencial ter o seu próprio URI [Berners-Lee et al., 1997] [Berners-Lee et al., 1998]. REST significa que cada URL único representa um objecto. Para se executar alguma operação sobre um objeto usam-se os métodos *standard*, neste caso os verbos HTTP, GET, POST, PUT e DELETE [Chappel,2009]. Assim, se quisermos obter os dados do cliente X numa aplicação basta apenas fazer um GET a /cliente/X.

REST tem sido cada vez mais adotado globalmente por várias empresas para os seus *web services*, tais como, *Twitter*, *eBay*, *Amazon*, *Flickr*, entre outros. Contudo, SOAP continua a ser amplamente utilizado na integração de várias aplicações empresariais assim como em sistemas legados. A Google continua a implementar os seus serviços usando SOAP.

Web services REST são mais "leves" pois não têm muito XML extra, os resultados são facilmente entendidos por humanos e são fáceis de construir. Em contra partida, *web services* SOAP são fáceis de consumir, são mais rígidos pois seguem regras assim como a verificação de tipos e existem várias ferramentas para auxiliar o desenvolvimento.

No que respeita à simplicidade e flexibilidade, a vantagem de REST deve-se ao facto de usar uma interface já bem conhecida, o URI. Pegando no exemplo do cliente apresentado em cima (http://localhost/cliente/x), qualquer cliente ou servidor com suporte HTTP pode facilmente fazer uma chamada ao serviço como o comando HTTP GET. A resposta HTTP resultante dessa chamada seriam apenas uns cabeçalhos e o conteúdo do objeto ou então um XML. Facilmente se consegue perceber como criar ou modificar o URI para aceder a um recurso diferente. Ao contrário desta simplicidade em REST, SOAP requer um conhecimento específico de um novo XML e ferramentas capazes de fazer pedidos e de processar as respostas.

Para estes pedidos e respostas de SOAP, é necessário usar-se um XML *wrapper* em cada pedido [Endrei et al., 2004]. Isto leva a que cada pedido SOAP requer até dez vezes mais bytes para uma resposta equivalente em REST. Tal como SOAP, REST também precisa de um documento que as definições dos serviços tais como os parâmetros de input e de output mas, a vantagem de REST é que este é suficientemente flexível para escrever o WSDL com o mesmo tipo de declaração formal ou apenas uma descrição dos serviços facilmente interpretada por humanos.

Outro assunto de debate entre REST e SOAP é a segurança. É considerado que enviar uma chamada SOAP através de portas HTTP é uma boa forma de assegurar os *web services* através dos limites da organização. Contudo diz-se que este tipo de procedimento tem falhas pois uma chamada típica de SOAP faz um POST para comunicar com um determinado serviço e, ao não se analisar a chamada, também dado que consome mais recursos e é mais demorado, não há forma de saber se esse pedido apenas quer receber dados ou apagar entradas nas tabelas. Neste caso, REST leva vantagem dado que um pedido GET pode ser sempre considerado seguro pois, por definição, não consegue alterar dados. Os pedidos REST podem ser feitos por HTTP ou HTTPS e o administrador ou *firewall* podem *analisar* a intenção de cada pedido, analisando o seu comando HTTP.

Apesar de apresentar muitas vantagens, REST não é perfeito nem a solução para todos os *web services*. Dados que devem ser mantidos seguros, nunca devem ser enviados como parâmetro no URI. Quantidades enormes de dados, como os detalhes completos de uma compra, podem ser pesados e exceder a dimensão do URI. No que diz respeito a anexos, SOAP transporta todos os binários sem falhas.

Enviar e receber mensagens SOAP nem sempre é a melhor maneira de as aplicações se comunicarem. Muitas vezes, um simples pedido REST e uma resposta em texto funciona, poupando tempo e recursos no processo. É importante analisar o problema, implementar com REST e recorrer ao SOAP quando necessário, ajudando assim a manter o desenvolvimento simples e acessível.

O serviço REST desenvolvido responde pelos seus *URIs* ou *endpoints* em *JavaScript Object Notation* (JSON). JSON suporta que os objetos e *arrays* estejam contidos noutros objetos e *arrays*. Tal como outros formatos como XML, amplamente usado, YAML ou CSV, JSON é independente da linguagem. Foi desenhado para ser facilmente lido por humanos e para transmitir dados entre aplicações.

JSON em relação ao XML tem pouco *overhead* pois é mais simples e mais pequeno, não sendo tão ambíguo na representação da informação. Ambos os formatos são os mais simples e usados na representação da informação.

Comparativamente JSON é um formato mais leve. Se se quiser descrever os dados de uma pessoa em JSON, seria uma estrutura como:

```
{
    "firstName": "João",
    "lastName": "Pereira",
    "age": 25,
    "address": {
        "streetAdd": "R. Dr. A. Bernardino de Almeida, 431",
        "city": "Porto",
        "postalCode": "4200-072"
    },
    "phoneNumber": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "+351 22 83 21 159"
        }
    ]
}
```

Código 1 - Exemplo estrutura Person em JSON

Em XML, pode apresenta de diversas formas, como por exemplo:

```
<person>
  <firstName>João</firstName>
  <lastName>Pereira</lastName>
  <age>25</age>
  <address>
    <streetAdd>R. Dr. A. Bernardino de Almeida, 431</streetAddress>
    <city>Porto</city>
    <postalCode>4200-072</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber type="home">+351 22 83 40 500</phoneNumber>
    <phoneNumber type="fax">+351 22 83 21 159</phoneNumber>
  </phoneNumbers>
</person>
```

Código 2 - Exemplo estrutura Person em XML

ou então como:

```
<person firstName="João" lastName="Pereira" age="25">
  <address streetAdd="R. Dr. A. Bernardino de Almeida, 431" city="Porto"
postalCode="4200-072" />
  <phoneNumbers>
     <phoneNumber type="home" number="+351 22 83 40 500"/>
     <phoneNumber type="fax"  number="+351 22 83 21 159"/>
  </phoneNumbers>
</person>
```

Código 3 - Exemplo estrutura Person em XML

Os três exemplos contém a mesma informação mas, em XML há alternativas para a representar, o que implica o formato de comunicação estar bem definido para que o objeto final de XML tenha o formato esperado.

2.10 Bases de Dados

Aplicações *web*, comummente, necessitam de guardar dados, sejam utilizadores, estados, perfis, informação de serviços externos, etc., para o seu funcionamento. Esta informação pode ser armazenada nos servidores de diversas maneiras desde ficheiros simples a bases de dados complexas.

A forma como os dados são guardados e recuperados é uma componente fundamental da aplicação pois vai influenciar a sua arquitetura e funcionamento. Tendo em vista o desenvolvimento de um portal Web torna-se claro e espectável que este cresça em dimensão e em número de utilizadores. Com este crescimento torna-se necessário ter mais do que uma máquina a operá-lo para poder servir todos os pedidos e suportar maior carga. Com o

aumento do número de máquinas, está associada uma rede servidores cada vez mais complexa, capaz de saber encontrar e guardar a informação que o utilizador requisita. Contudo, esta informação não pode apenas estar armazenada numa máquina pois criaria um *bottleneck* enorme se todas tentassem aceder simultaneamente.

Existem diversas formas de armazenar dados. O armazenamento de dados num ficheiro de texto é um exemplo mas, a pesquisa neste tipo de armazenamento de dados é mais complexa e morosa.

Com base nestas premissas, uma boa e tradicional solução, passa por armazenar os dados numa Base de Dados que seja facilmente escalável, dando maior flexibilidade e rapidez à aplicação.

2.10.1 SQL *vs* NoSQL

Atualmente, em aplicações *web* uma base de dados é uma componente imprescindível. Contudo, apesar de haverem diferentes tipos de organizações de bases de dados, SQL e NoSQL são os tipos mais devido à sua robustez e simplicidade.

Uma base de dados não é mais do que um conjunto de informação relacionada.

A linguagem SQL tem raízes muito antigas. Em 1970, um investigador da IBM, Dr. E. F. Codd [Beaulieu,2009] propôs que os dados fossem representados como tabelas e, tem sido até então, uma metodologia amplamente adoptada. O SQL é a linguagem usada para gerar, manipular e receber dados de uma base de dados relacional. Estas bases de dados são muito populares pois, quando propriamente desenhadas, manipulam facilmente enormes quantidades de dados e, com o passar dos anos, novas funcionalidades foram desenvolvidas para responder aos novos problemas a resolver.

O SQL anda de mão dadas com o modelo relacional pois, o resultado de uma *query* SQL é uma tabela, tabela esta cuja informação é originada a partir da informação de uma ou mais tabelas.

Nos últimos anos, com o advento da internet, as bases de dados relacionais foram usadas de uma forma que não era possível prever. As aplicações que consistiam num ambiente pequeno, aos poucos começavam a replicar dados para poder servir os utilizadores e, passado algum tempo, quando já não era possível afinar mais as *queries* ou aumentar a complexidade da máquina com *harwdare*, tornava-se inevitável distribuir as aplicações por vários *clusters*. Esta tarefa, neste contexto, torna-se extremamente complexa e dispendiosa.

Neste sentido, e por forma a abordar este problema, a comunidade *Open Source*, desenvolveu uma nova abordagem para as bases de dados das modernas aplicações *web*. Esta abordagem propõe a utilização de pares chave / valor para os dados em vez de complicadas estruturas SQL. Surgiram assim as bases de dados NoSQL que não têm uma estrutura definida que

prenda a aplicação ao modelo, não necessitando de se despender tempo de desenvolvimento caso este seja alterado.

Diversas empresas desenvolveram as suas abordagens ao NoSQL originando produtos como MongoDB, CouchDB, Cassandra, Hadoop, BigTable, entre outros, que fazem parte das arquiteturas de diversas aplicações, sendo das mais famosas a usar este tipo de tecnologia a Google, Facebook, Twitter, Digg, FourSquare.

Este tipo bases de dados são extremamente poderosos, flexíveis e facilmente escaláveis pois foram concebidas tendo em conta as necessidades das aplicações *web* atuais. Permitem as mesmas operações das bases de dados tradicionais e têm as mesmas componentes como indexação, ordenação, etc., exceto as operações de relação entre tabelas como UNION, JOIN, etc., pois não são relacionais. Contudo, estas relações estão presentes nos dados mas com uma abordagem diferente.

A abordagem de MongoDB à representação de dados vem substituir o conceito de linha de tabela, *row*, por documento, *document* [Chodorow and Dirolf,2010]. Com este conceito, é possível representar relações complexas no mesmo documento, sendo esta abordagem muito próxima do atual desenvolvimento orientado a objetos.

MongoDB é uma base de dados facilmente escalável pois foi desenhada para tal. À medida que a aplicações crescem, basta acrescentar máquinas ao *cluster* e a base de dados organizará tudo automaticamente, espalhando os documentos pelas diferentes máquinas, fazendo a sua gestão. Isto leva a que se poupe muito esforço na programação pois os programadores apenas têm de se concentrar na aplicação e não em escalá-la.

MongoDB apresenta algumas *features* que não se encontram em bases de dados tradicionais como suporte para *MapReduce* e *indexes* geoespaciais. Esta característica torna o MongoDB cada vez mais vantajoso para as aplicações pois, coordenadas como atributo de uma tabela, começa a ser um tipo de informação cada vez mais recorrente, seja através de aplicações para dispositivos móveis como para aplicações *web*. Outra característica do MongoDB é o uso de funções de *javascript* em vez de *stored procedures*.

Este tipo de representação de informação é mais uma das razões que fazem do MongoDB uma base de dados extremamente rápida, pois, as relações e *queries* complexas não existem e, esta rapidez, torna-se uma mais valia para um sistema que se espera ter grande crescimento.

Tipicamente, numa base de dados relacional, a informação relaciona-se entre si através de chaves estrangeiras e *ids* que nos indicam a localização de mais informação. Usando como exemplo um portal de emprego, uma oferta de emprego terá como atributo um *id* para a categoria em que essa oferta se insere, sendo fácil, através de uma pesquisa simples obter informação sobre a mesma. Em MongoDB ou bases de dados idênticas (NoSQL), esta relação não existe, implicando que terá de ser armazenada de outra forma. Guardar *ids* de outros documentos, como *categoria*, *distrito*, etc., numa oferta de emprego e posteriormente consultar todos os documentos para obter a restante informação da oferta para construir o

seu objeto torna-se bastante moroso e impraticável, acabando por se assemelhar a uma base de dados relacional. A solução comummente usada passa por replicar toda a informação dos restantes documentos no documento da oferta. Assim, uma oferta de emprego tem guardado tanto o *id* da categoria, como o seu nome, descrição, entre os demais atributos que este possua. Para os restantes atributos como distrito, cidade, etc., o procedimento repete-se.

É perfeitamente visível que ao fazer-se uma pesquisa na base de dados, estamos apenas a pesquisar numa coleção de documentos, não tendo de os relacionar. Isto torna a pesquisa muito mais rápida.

Contudo, esta situação origina um crescimento muito rápido da base de dados devido ao facto de ter de se replicar muita informação por cada documento introduzido. É uma situação que é facilmente ultrapassada pois apenas requer mais espaço em disco dos servidores, que é uma componente bastante acessível.

O maior problema associado a este tipo de bases de dados é relativo a sistemas complexos empresariais e transacionais. Nestes casos, a representação deste tipo de dados numa base de dados NoSQL torna-se impraticável pois o documento seria enormíssimo e bastante complicado de o manter. Outra questão levanta-se relativamente à integridade dos dados pois não há verificação nenhuma de inserção ou remoção de elementos das coleções, o que por um lado torna os processos mais rápidos, torna-os menos fiáveis.

O formato em que os dados são guardados e transferidos no MongoDB é BSON, que é uma serialização binária de documentos tipo JSON, Binary JSON. Isto permite que a integração de dados sejam rápida e fácil.

2.11 Cálculo de distâncias no planeta Terra

O cálculo entre duas coordenadas terrestres deve ter-se em conta não apenas a distância entre os dois pontos mas também a curvatura da Terra. Para simplificar, considera-se a Terra como sendo esférica, quando na verdade é mais elipsoidal [Milbert and Smith,1996] [Williams,2004], o que faz com que usar um modelo esférico origine alguns erros.

Outra condição a aplicar na distância entre dois pontos na Terra, considerando o percurso terrestre é a topografia do terreno e a trajetória a efetuar pois a distância entre duas coordenadas apenas nos indica o percurso em linha recta, o que na prática nunca acontece. Isto implica ter conhecimento específico sobre o terreno e seus percursos fazendo com que este cenário tenha sido afastado do âmbito desta plataforma pois o erro proveniente do cálculo direto, em poucas ocasiões, poderá estimar mal a relevância de um resultado de pesquisa.

Existem diversas formas de calcular a distância entre duas coordenadas terrestres. Isto entra no campo da geometria esférica.

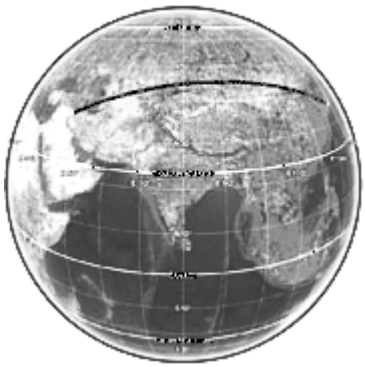

Figura 3 – Trajetória demonstrando uma distância no planeta Terra

2.11.1.1 Haversine

A fórmula de *Haversine* [Sinnot,1984] é uma equação usada na navegação para calcular distâncias entre dois pontos de uma esfera a partir de latitude e longitude ignorando a topografia do terreno.

$$haversin\left(\frac{d}{r}\right) = haversin(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\, haversin(\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1) \quad (1)$$

$onde$

$$haversin(\phi) = \sin^2\left(\frac{\phi}{2}\right) = \frac{1 - \cos(\phi)}{2}$$

d – distância entre dois pontos

r – raio da esfera

ϕ_1 e ϕ_2 – latitude do ponto 1 e latitude do ponto 2

$\boldsymbol{\lambda}_1$ e $\boldsymbol{\lambda}_2$ - longitude do ponto 1 e longitude do ponto 2

Esta formula extrai bons resultados mesmo em pequenas distâncias.

2.11.1.2 Lei esférica dos cossenos

A lei esférica dos cossenos [Brummelen,2009] é outro método para calcular a distancia entre coordenadas numa esfera, bastante preciso, destacando-se da fórmula de *Haversine* na simplicidade da sua implementação.

$$d = \operatorname{acos}(\sin(\phi 1) * \sin(\phi 2) + \cos(\phi 1) * \cos(\phi 2) * \cos(\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1)) * r \quad (2)$$

$onde$

d – distância entre dois pontos

r – raio da esfera

ϕ_1 e ϕ_2 – latitude do ponto 1 e latitude do ponto 2

$\boldsymbol{\lambda}_1$ e $\boldsymbol{\lambda}_2$ - longitude do ponto 1 e longitude do ponto 2

2.11.1.3 Teorema de Pitágoras

Os algoritmos previamente apresentados usam várias funções matemáticas para calcular resultados o que em determinadas situações, onde a performance é bastante importante e a precisão pode ser descurada, uma fórmula menos complexa para calcular a distância entre pontos pode apresentar uma vantagem relativamente a outras.

Um exemplo é o teorema de Pitágoras esférico [Brummelen,2009] [Júnior,2013] que apresenta bons resultados para pequenas distâncias.

$$d = r * \sqrt{(x^2 + y^2)} \quad (3)$$

$onde$

$x = (\lambda_2 - \lambda_1) * \cos((\phi_1 + \phi_2)/2)$

$y = \phi_2 - \phi_1$

d – distância entre dois pontos

r – raio da esfera

ϕ_1 e ϕ_2 – latitude do ponto 1 e latitude do ponto 2

$\boldsymbol{\lambda}_1$ e $\boldsymbol{\lambda}_2$ - longitude do ponto 1 e longitude do ponto 2

Esta fórmula tem um custo computacional bastante inferior, não recorre a tantas funções trigonométricas mas, apresenta um erro maior.

3 Desenvolvimento do Portal de Emprego Inteligente

3.1 Arquitetura do Portal de Emprego Inteligente

A arquitetura do Portal de Emprego Inteligente divide-se em três grandes áreas: *Backend*, *FrontEnd* e Base de Dados.

A camada *Backend* é a camada principal da aplicação pois é nesta que se encontra toda a lógica do Portal relativa à comunicação com a base de dados e lógica dos diferentes cálculos a serem efectuados. É nesta camada que se encontra toda a "lógica de negócio". Esta camada encontra-se desenvolvida na linguagem Java para beneficiar de uma linguagem fortemente tipada e aberta. Acenta num *Application Server JBoss* e disponibiliza um conjunto de serviços REST passíveis de serem consultados por diferentes *frontends*.

Esta camada divide-se em três camadas. O acesso à área de Backend pelos diferentes plataformas e ou aplicações é feito fazendo pedidos REST. Este pedidos são recebidos pela camada REST (como demonstrado na Figura 4). Se o pedido não for válido, como um pedido a um método inválido (fazer um POST a um método GET, estado conhecido como *405 Method not allowed*) ou um método inexistente (estado *404 Not found*), esta camada abstrai essa lógica da restante aplicação respondendo imediatamente.

Sendo um pedido válido, as variáveis presentes no mesmo, sejam parâmetros de URL, campos de um formulário por POST ou um objecto JSON no corpo do pedido, esta camada é responsável por criar uma nova instância de um objeto correspondente ao pedido. Este objeto, na sua forma mais básica, é apenas um DTO.

Posteriormente, esta camada é responsável por chamar um Enterprise JavaBeans (EJB), correspondente à área do pedido, para o executar.

O EJB permite padronizar de forma simples objetos Java, sendo possível gerir o seu ciclo de vida, estado e dando-lhes escalabilidade. É uma aplicação padrão de Java Enterprise Edition (J2EE) e é controlado por este. O J2EE mantém uma *pool* de instâncias de EJBs prontas a serem usados por um cliente. O cliente que usa o EJB não precisa de saber como este funciona internamente.

O Portal de Emprego Inteligente separa os EJBs por camadas de serviço, ou seja, existe um EJB responsável pelas operações relativas a empresas, outro pelas operações ao nível do utilizador e por fim um EJB responsável pelos empregos.

O EJB de empresas, dispõe de métodos para criar um empresa nova (novo registo) e outro para obter informações sobre uma determinada empresa.

O EJB de utilizadores é responsável pelo registo de novos utilizadores, pela sua atualização e também pelo *login*. Contudo, o *login* de uma empresa é feita também a partir deste EJB.

Por último, o EJB de empregos. É neste EJB que se encontram todas as operações relativas a empregos. A criação de um novo emprego, por parte uma empresa, a obtenção de um emprego a partir da sua identificação e listagem de categorias de empregos são métodos que se encontram disponíveis neste EJB. A pesquisa de emprego e calculo da relevância dos resultados obtidos, é efectuado também neste EJB. Foram desenvolvidas classes auxiliares, que são usadas para a pesquisa e cálculo de relevância de empregos. O método de pesquisa recorre-se destas classes, de forma a ficar com um código mais limpo, perceptível e mais modular.

Os EJBs são instanciados no momento é que um pedido válido é feito à camada REST.

Por último, existe a camada de acesso a dados. Esta camada é a responsável pela obtenção de dados da base de dados. Existem três classes homólogas às descritas nos EJBs, uma para empresas, outra para utilizadores e outra para empregos.

Cada classe presente na camada de dados assume que toda a informação foi tratada no EJB e que todas as condições foram tratadas. Apenas recebe parâmetros finais de pesquisa (não trata dados). Assim, as classes podem ser utilizadas por diferentes classes e objetos pois abstraem-se de onde de onde são instanciadas. Cada classe sabe injetar as suas dependências, nomeadamente as configurações da base de dados, libertando assim outras classes (como os EJBs) de terem de conhecer as suas configurações, facilitando a evolução da plataforma.

A criação de uma camada de EJBs oferece muita versatilidade ao sistema. O desenvolvimento de um novo cliente, passível de comunicar com EJBs, assente no mesmo *application server* permite que este cliente aceda aos métodos expostos nas interfaces remotas do EJB. Assim, um cliente pode beneficiar deste motor, usando diretamente o EJB, sem necessitar de fazer um pedido à camada REST, o que representa menos, pelo menos, um pedido HTTP efectuado, evitando verificações necessárias, tornando o acesso mais leve e rápido.

A camada de *web services* do Portal de Emprego Inteligente aceita os pedidos e devolve as respostas usando JSON.

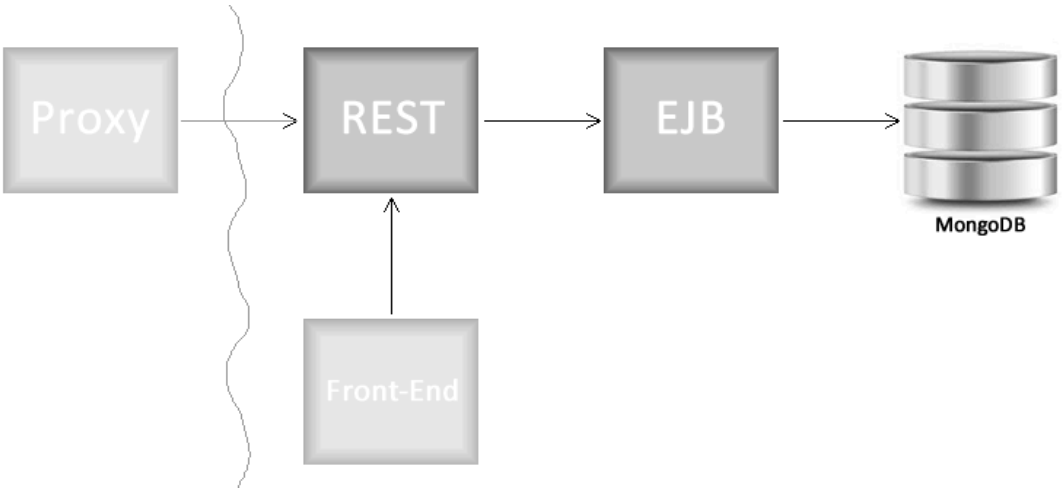

Figura 4 – Arquitetura do Portal de Emprego Inteligente

Na Base de Dados é onde se encontra guardada toda a informação do Portal, desde configurações, registos de utilizadores e empresas e de empregos. Dada a necessidade de se ter um acesso a dados muito rápido, a base de dados é MongoDB cuja a estrutura por documentos se torna uma mais-valia para registo e obtenção de informações, eliminando *queries* complexas (por sua vez mais lentas) e possibilitando ainda o uso de ferramentas já incluídas na sua estrutura base como o uso de atributos de geo-localização, tornando muito fácil a extração de dados em função de coordenadas.

Por último, a camada de *frontend*, camada visual, é a responsável pela interação dos utilizadores com o sistema. Os registos, *logins*, pesquisas, entre outras funcionalidades, é realizador a partir desta camada *web*.

A camada de *frontend* foi desenvolvida em *python* usando uma *micro-framework* (*Flask*). A rezão que leva a utilização de uma framework é que esta abstrai de implementações de mais baixo nível, protege mais a aplicação a plataforma de vulnerabilidades básicas (como SQL Injection) e evita o "re-inventar da roda" pois módulos genéricos já se encontram desenvolvidos. Isto torna o desenvolvimento mais rápido e melhor pois as operações tradicionalmente desenvolvidas, como acesso a dados, mecanismos de autenticação, de gestão de sessões, geração de endpoints, entre outros desenvolvimentos base, já se encontram implementados.

Outra vantagem da utilização de *frameworks* livres prende-se também com o facto de que estas são desenvolvidas por grandes comunidades a nível mundial. Isto leva a que novos desenvolvimento sejam disponibilizados muito rapidamente assim como a detecção e resolução de problemas seja igualmente tratado pela comunidade.

O uso de uma *micro-framework* no Portal de Emprego Inteligente, em vez de uma *framework* maior deve-se a que, o portal pouco beneficiaria das suas funcionalidades dado que um grande parte da plataforma encontra-se desenvolvida ao nível do *backend*. Assim, a *micro-framework*, apenas disponibiliza o ambiente gráfico (apresenta as vistas) aos utilizadores e, efetua pedidos ao *backend* consoante as ações dos utilizadores.

Esta estrutura permite desenvolver outros *frontends*, seja para aplicações para dispositivos móveis ou outros, usando a mesma camada de serviços (*backend*), mantendo assim todas as funcionalidades disponíveis e a sua integridade.

Esta arquitetura, por si só, acrescenta segurança à aplicação pois, usando uma linguagem tipada (neste caso Java) separada numa camada diferente (*backend-frontend*), diminui a facilidade de ataques mais elementares, não expondo nunca a estrutura da aplicação ou base de dados. Toda a estrutura encontra-se protegida no servidor, não permitindo acesso externo. O único ponto de acesso do exterior é a partir do *frontend*.

Todos os clientes a serem construídos, *frontends*, *backoffices*, entre outros, que não se encontrem no mesmo servidor, e que necessitem de acesso ao *backend*, podem faze-lo através de um *proxy*. Esta abordagem é segura pois, este *proxy*, para além das suas camadas de segurança (como possíveis *tokens* de autorização ou outros), apenas tem acesso a algumas partes do sistema e a determinadas funcionalidades que estarão disponíveis.

3.2 Domínio

Para construir um Portal de Emprego Inteligente é necessário ter conhecimento sobre as diferentes áreas. Nesta fase do projeto, apenas foi considerada a área de Informática e os seus principais ramos.

Para a aquisição deste conhecimento foram utilizadas diversas fontes como outros portais de emprego portugueses, empregadores e trabalhadores da área, das quais se destaca um perito da mesma, Hugo Correia. Devido não só à sua experiência como arquiteto de sistemas em plataformas de elevada dimensão, *team-leader*, e também como programador, foi possível construir e validar a base de conhecimento desenvolvida.

Durante as sessões de aquisição de conhecimento com o perito tentou identificar-se as vantagens e problemas das plataformas existentes e de que forma isto pode contribuir para o desenvolvimento da plataforma. Foram identificados diversos problemas das quais se destacaram:

- falta de granularidade nas categorias de emprego providenciadas pelas outras plataformas,

- pesquisa por localização ser pouco eficaz,
- relevância de um resultado ficar aquém do esperado pelo utilizador.

Deu-se então início ao planeamento do estudo da informação necessária. Definiu-se que seria necessário entender de que forma se poderia colmatar o problema da localização do utilizador, não restringindo apenas pela divisão administrativa de concelhos ou distritos.

Outra etapa identificada seria a construção de uma árvore com as diversas categorias e subcategorias mais relevantes de empregos da área de Informática em Portugal. Posteriormente à elaboração da árvore seria necessário entender de que forma é que cada categoria se relaciona com as restantes.

Como última etapa seria necessário identificar de que forma deveria ser calculada a relevância de um resultado em função da distância ao utilizador tendo em conta os valor máximo de distância desejável por si inserido previamente, assim como calcular a relevância do salário também em função do que foi definido pelo utilizador durante a pesquisa.

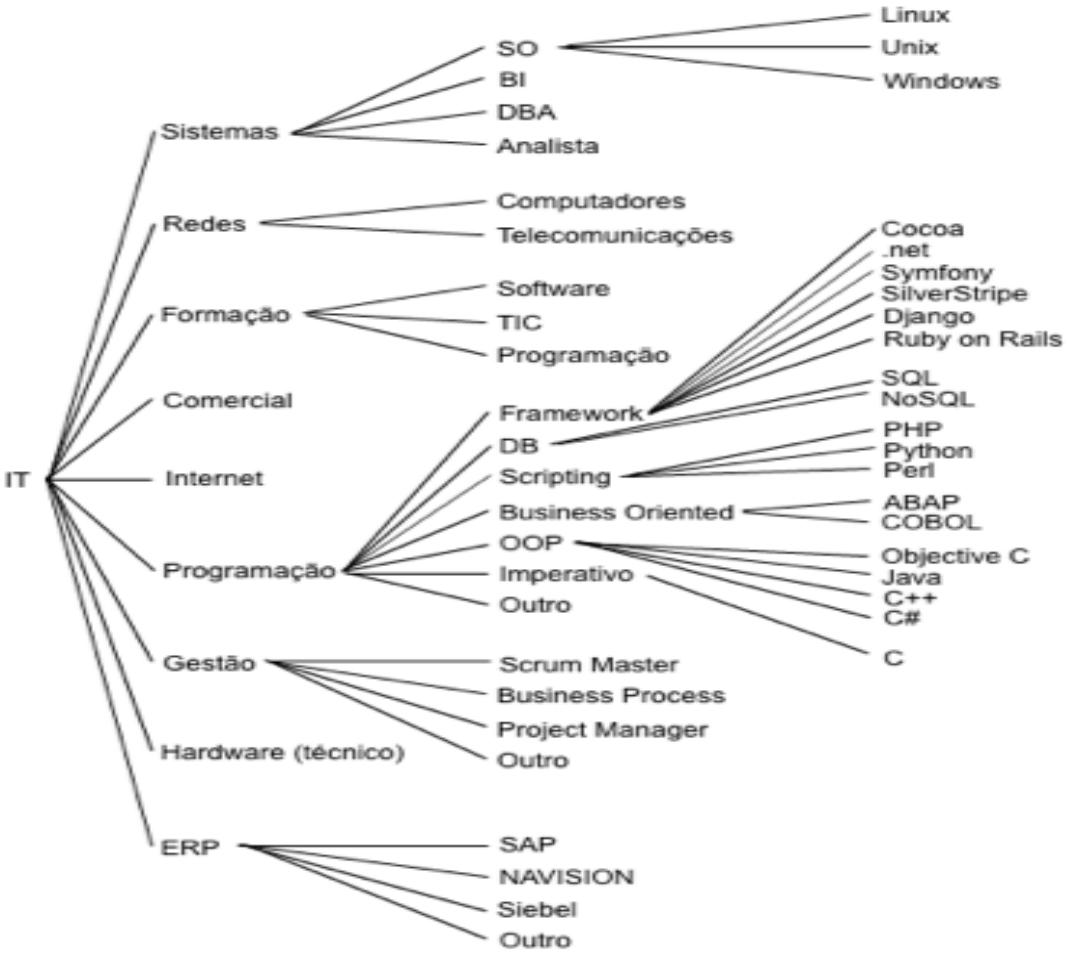

Figura 5 - Categorias de emprego na área IT

3.3 Relaxamento de restrições

No âmbito do desenvolvimento do Portal de Emprego Inteligente, a pesquisa de ofertas de emprego beneficia com a implementação deste mecanismo.

De modo a abranger mais resultados, o atributo distância, foi relaxado para o dobro do indicado pelo utilizador. O atributo salário foi também relaxado com aritmética simples, para valores entre menos 25% e mais 25% do salário indicado pelo utilizador. Dado que muitas vezes as empresas anunciantes optam por não o apresentar, este não é um atributo discriminatório, devolvendo-se na pesquisa as ofertas de emprego que não o apresentem. Estes valores para relaxar os respectivos atributos foram selecionados após alguma análise dos mesmos, que se consideram devolver documentos relacionados com a pesquisa do utilizador.

Relaxar as categorias de uma oferta de emprego, implica conhecer mais sobre a área em que este se insere. Foi com este problema em mente que se criou a árvore referida na Figura 5. Usando esta árvore, para cada subárea selecionada pelo utilizador para a pesquisa, é relaxada, pesquisando todas as ofertas de emprego que tenham como subárea as irmãs desta, ou seja, a partir do pai desta subárea, são selecionados todos os filhos e pesquisado os anúncios com as diferentes subáreas. Esta árvore foi dividida tendo em consideração a proximidade de nó e respectivos filhos. Contudo, um utilizador pode selecionar várias áreas e serão todas pesquisadas em simultâneo.

Após este processo, espera-se que sejam devolvidos muitos mais documentos do que numa simples pesquisa. Contudo, nem todos os documentos são relevantes para o utilizador pois alguns podem ser bastante díspares da pesquisa inicial.

É então necessário calcular para cada documento a sua relevância relativamente à pesquisa efectuada pelo utilizador e apresentar os resultados ordenados por este valor.

O MongoDB dispõe de uma função geo-espacial, *$geoWithin* [Chodorow and Dirolf,2010], para obter documentos cujas coordenadas se encontrem dentro de uma figura geométrica definida para a pesquisa. O Portal de Emprego Inteligente aproveita esta função *$geoWithin* para extrair todos os documentos de emprego que se encontrem dentro de um círculo, com centro nas coordenadas definidas pelo utilizador, cujo raio corresponde à distância (relaxada) definida pelo utilizador.

3.4 Cálculo da relevância dos documentos encontrados

Ao relaxar a pesquisa do utilizador, alguns dos resultados devolvidos podem ser pouco relevantes, logo um benefício que podia ser apresentado ao utilizador seria mostrar em percentagem a relevância individual de cada resulta. É então é necessário calcular para cada

documento da base de dados, a sua relevância para o utilizador em função dos atributos e dos respectivos pesos.

3.4.1 Calculo da distancia

Para efetuar o cálculo da distância entre a localização do utilizador e do resultado pesquisado, o sistema recorre-se do teorema de Pitágoras. A opção por usar este modelo menos preciso, deve-se a que a performance é bastante importante dado que são calculadas distâncias entre o utilizador e todos os resultados de pesquisa apresentados para calcular a sua relevância, que em determinadas situações podem comprometer a plataforma e, também porque este atributo corresponde a mais uma informação apresentada ao utilizador, não comprometendo a decisão final do mesmo.

A relevância da distância para o utilizador é calculada segundo algumas regras predefinidas. Dado que as restrições da pesquisa são relaxadas previamente, alguns dos resultados obtidos apresentam uma distância que vai além da definida pelo utilizador. Um método que se poderia adoptar seria fazer o cálculo direto, usando uma regra de três simples, entre o pedido do utilizador com máximo relaxado que se poderia obter. Isto implicaria que um resultado que se encontrasse à distância máxima definida pelo utilizador teria uma relevância bastante baixa (de 50%). Este facto deve ser considerado incorreto pois é uma distância que o utilizador está disposto a aceitar. Assim sendo, considera-se que um resultado que se encontre entre o utilizador e a distância que este definiu, terá uma relevância situada entre os 100% e 85% respectivamente. Com isto espera-se conseguir um compromisso em que um resultado à distância máxima seja bastante relevante.

3.4.2 Cálculo da relevância de um salário

O cálculo da relevância do atributo salário é efectuado numa proporção direta com o valor apresentado; baseia-se em aritmética simples. Se o valor do resultado apresentado for maior ou igual que o valor indicado pelo utilizador, este atributo é considerado com uma relevância de 100%.

Para os restantes cenários é feita uma regra de três simples entre o salário da pesquisa e do resultado. Desta forma consegue-se uma boa aproximação das espectativas do utilizador.

Num cenário real mais variáveis entram em consideração para este atributo nomeadamente, e por estar a tratar-se de um portal de emprego, as regalias seriam um factor em consideração assim como a proporção entre salário e distância.

3.4.3 Cálculo da relevância de categorias

Exposto o domínio, apresentado na figura 5, torna-se mais fácil identificar cada categoria e subcategoria, assim como a sua proximidade.

Para expor este conhecimento para plataforma, é necessário converter a árvore em diversas matrizes com a distância entre categoria e subcategoria.

As seguintes tabelas expõem o conhecimento extraído. Em cada célula é apresentado o valor (em percentagem) da semelhança / relevância entre as diferentes subcategorias.

Tabela 1 - Semelhança Sistemas

	SO	**BI**	**DBA**	**Analista**
SO	100	80	80	90
BI	80	100	70	80
DBA	80	70	100	70
Analista	90	80	70	100

Tabela 2 - Semelhança Sistemas Operativos

	Linux	**Unix**	**Windows**
Linux	100	95	75
Unix	95	100	75
Windows	75	75	100

Tabela 3 - Semelhança Redes

	Computadores	**Telecomunicações**
Computadores	100	80
Telecomunicações	80	100

Tabela 4 - Semelhança Formação

	Software	**TIC**	**Programação**
Software	100	85	70
TIC	85	100	75
Programação	70	75	100

Tabela 5 - Semelhança Gestão

	Scrum	**Business Process**	**Project Manager**	**Outro**
Scrum	100	80	85	85
Business Process	80	100	75	75
Project Management	85	75	100	75
Outro	85	75	75	100

Tabela 6 - Semelhança ERP

	SAD	**NAVISION**	**Siebel**	**Outro**
SAD	100	90	90	80
NAVISION	90	100	90	80
Siebel	90	90	100	80
Outro	80	80	80	100

Tabela 7 - Semelhança Programação

	Framework	**DB**	**Scripting**	**Business Oriented**	**OOP**	**Imperativo**	**Outro**
Framework	100	60	80	50	80	70	50
DB	60	100	70	50	70	80	40
Scripting	80	70	100	60	95	90	70
Business Oriented	50	50	60	100	60	60	45
OOP	80	70	95	60	100	80	70
Imperativo	70	80	90	60	80	100	70
Outro	50	40	70	45	70	70	100

Tabela 8 - Semelhança Programação (Frameworks)

	Cocoa	.net	Symfony	SilverStripe	Django	Ruby on Rails
Cocoa	100	80	65	65	65	65
.net	80	100	65	65	65	65
Symfony	65	65	100	85	90	90
SilverStripe	65	65	85	100	85	85
Django	65	65	90	85	100	90
Ruby on Rails	65	65	90	85	90	100

Tabela 9 - Semelhança Bases de Dados

	SQL	NoSQL
SQL	100	90
NoSQL	90	100

Tabela 10 - Semelhança Programação (Scripting)

	PHP	Python	Perl
PHP	100	85	80
Python	85	100	80
Perl	80	85	100

Tabela 11 - Semelhança Programação (Business Oriented)

	ABAP	COBOL
ABAP	100	90
COBOL	90	100

Tabela 12 - Semelhança Programação (OOP)

	Objective C	Java	C++	C#
Objective C	100	80	80	80
Java	80	100	90	95
C++	80	90	100	85
C#	80	95	85	100

Tabela 13 - Semelhança Programação (Imperativo)

	C
C	100

Durante o calculo da relevância de um resultado é pesquisado, usando estas tabelas, a relação entre os atributos pesquisados e os atributos obtidos no resultado.

3.4.4 Peso de cada atributo

Na tomada de decisão o ser o humano pondera as diversas soluções apresentadas e toma a sua opção conforme o que lhe considera ser mais interessante ou conveniente. Assim, analisando este comportamento, verificamos que cada atributo tem um peso menor, maior ou igual a outros que entrem na equação. Isto explica que em cenários semelhantes pessoas diferentes tomam decisões distintas pois para tal baseiam-se nas suas experiências e cultura [Cayrol and Barrère, 1997].

Desde cedo que nas tecnologias de informação se tentam aproximar o ambiente de uma aplicação ou sistema ao ambiente natural e humano. Um exemplo simples mas clássico é apresentado pelo ambiente de trabalho dos sistemas operativos mais usados no mundo inteiro. Cada vez mais estudos são feitos no processamento de linguagem natural e semântica para extrair contexto de um texto ou de comportamento humano para avaliar digitalmente o estado de espírito de alguém. Mesmo a criação de algoritmos inspirados na natureza, algoritmos genéticos, como *Ant colony optimization* inspirada no comportamento real das colónias de formigas na sua busca por comida e é usado para resolver problemas de optimização [Dorigo et al.,2006], ou *Artificial bee colony algorithm* algoritmo meta-heurístico de optimização baseado numa população (enxame) de soluções candidatas (partículas) com a habilidade de interagir entre si e com o meio ambiente [Kennedy and Eberhart, 1995] [Poli,2007], são usados para obter melhores resultados.

Tendo em consideração este tipo de comportamentos o cálculo da relevância de cada atributo de um documento deve ser também ponderado para obter um resultado final de relevância em função do peso de cada atributo.

Num fase inicial são atribuídos valores por defeito, valores estimados inicialmente na peritagem para o sistema, para efetuar a pesquisa. Assim que são devolvidos os diversos documentos da base de dados, para cada atributo de cada documento é calculada a sua relevância e, após ter todos os valores para o documento, estes são pesados para calcular a relevância final do documento. Os valores por defeito da plataforma consideram que a relevância de um resultado deve dar mais importância à categoria do mesmo, seguido do salário e finalmente da distância. A soma total dos pesos é de dez valores. Assim a categoria é

pesada inicialmente com cinco valores, o salário com três e por último a distância com dois valores.

3.4.4.1 Apresentado um exemplo prático deste cálculo

Num cenário em que a categoria do resultado tem uma relevância de 90%, o salário de 100% e a distância de 65%, a relevância deste resultado seria de 88% o que dado os dados iniciais se enquadra bastante bem com o interesse que o utilizador poderá ter no mesmo, apesar de a distância ser um pouco além da definida inicialmente pelo mesmo.
Contudo, estes atributos podem ser afinados pelo utilizador durante a pesquisa ou então, inferidos pelo sistema em função do histórico das pesquisas efectuadas.

4 Portal de Emprego Inteligente

4.1 A arquitetura

Centralizar a lógica de negócio numa camada de serviços revela-se muito vantajoso.

Hoje em dia, o uso de telemóveis inteligentes cresce exponencialmente, e, com eles, cresce o número de aplicações disponíveis para os mesmos. Outras plataformas como, televisões inteligentes, caixas de distribuidores de televisão, entre outros, estão também dotados de acesso à internet e com cada vez mais aplicações disponíveis.

Ao ter uma camada de serviços que expõe as funcionalidades existentes no Portal de Emprego Inteligente, o desenvolvimento para outras plataformas torna-se muito simples. Tal como no caso do *frontend*, apenas é necessário desenvolver um novo cliente para a plataforma em que se quer disponibilizar o Portal, reaproveitando assim todo o motor do mesmo.

No desenvolvimento para novas plataformas, não é necessário conhecer tão pouco, como o portal funciona. Basta conhecer os *endpoints*, que pedidos são aceites e como, e que tipo de resposta se espera do serviço.

Esta arquitetura traz consigo outras vantagens, nomeadamente no que diz respeito à evolução do Portal.

A manutenção do Portal, como melhorias na performance, correção de erros ou até mesmo, uma mudança no paradigma interno do *backend*, permite, mantendo-se os mesmos *endpoints*, que o serviço funcione sempre como esperado. As aplicações de *frontend* não são afectadas com estes desenvolvimentos mas beneficiam dos mesmos.

Novas funcionalidades desenvolvidas ao nível do *backend*, ficam expostas imediatamente para todos os *frontends*, necessitando de apenas de se desenvolver ou atualizar uma vista.

Usando *standards* como, no caso do Portal de Emprego Inteligente, o JSON, permite que o desenvolvimento usando o Portal tenha uma curva de aprendizagem reduzida. Assim, se se quiser expor as funcionalidades para terceiros, espera-se que seja de fácil adopção.

4.2 A base de dados

As Bases de Dados *NoSQL* diferem bastante das *SQL*. Como não há uma estrutura em que os objetos se relacionem, é necessário fazer, caso necessário, as relações à posteriori assim como validar a integridade referencial dos dados. A maneira de se abordar uma aplicação assente em *NoSQL* é bastante diferente podendo no início gerar alguma confusão no planeamento.

Organizar uma aplicação bastante dependente de relações nas bases de dados, como uma aplicação de faturação ou uma estrutura empresarial, é uma tarefa ingrata e até mesmo quase impossível. Contudo, outras aplicações, como o Portal de Emprego Inteligente, podem beneficiar das vantagens de usar uma base de dados *NoSQL*.

Existem diferentes tipos de bases de Dados *NoSQL* como Cassandra cuja organização é feita por famílias de colunas, Redis onde os dados são guardados em pares chave-valor ou como o MongoDB usada no Portal de Emprego Inteligente, cuja organização é efectuada em documentos. Para o Portal de Emprego Inteligente a organização por documentos tornou-se bastante vantajosa pois permite que facilmente se altere a sua estrutura, sem ter necessidade de se mexer na base de dados. Novos documentos com nova estrutura podem ser facilmente guardados em coleções já existentes. Assim acrescentar um campo ou referência nova é quase imediato. Cada emprego corresponde a um documento guardado em Base de Dados e nesse documento são guardadas todas as informações relativas ao mesmo eliminando relações com categorias, empresa, etc., tornando o acesso ao mesmo muito rápido.

Dado o relaxamento de restrições, que implica extrair muito mais documentos da base de dados para que possam ser analisados e calculada a sua relevância, a rapidez de acesso torna-se crucial no processo. Outra vantagem no uso de MongoDB reside nas funções disponibilizadas pelo mesmo. Hoje em dia cada vez mais há necessidade de, em diferentes plataformas ou portais, obter a localização de algo e, começa a ser cada vez mais comum, as aplicações apresentarem resultados em função da proximidade do utilizador. O MongoDB traz incluído capacidades de geo-referênciação. Este mecanismo permite guardar documentos com uma localização (latitude e longitude). O que torna vantajoso é que a extração da informação pode ser efectuada usando uma localização base e obter os pontos de interesse próximos.

A extração da informação da base de dados em função de coordenadas torna-se simples e extremamente rápida com o uso de funções já disponíveis no MongoDB, revelando-se uma boa aposta para o desenvolvimento da plataforma.

4.3 Resultados de uma pesquisa

Para testar a plataforma foi necessário criar várias entradas na base de dados tanto de empregos como de utilizadores.

Para realizar os testes e comparar resultados obtidos, o Portal de Emprego Inteligente disponibiliza também uma pesquisa tradicional, por distrito. Esta pesquisa engloba a área disponível no Portal, Tecnologias de Informação, e um distrito.

Abstraindo os valores internos do Portal de Emprego Inteligente, como identificadores de área e subárea de emprego, localização, simplificando a visualização da lista, o título dos resultados apresentados são genéricos identificando cada subárea de acordo com o domínio estabelecido assim como é apresentado o concelho do emprego.

4.3.1 Perfil do utilizador

O utilizador usado para o teste tem definido os parâmetros: localização em Santo Tirso, nas coordenadas 41.344246, -8.4849193; distância (ou raio) pretendido de quinze quilómetros e preferências nas áreas de *PHP*, *Python* e *Java*.

A utilização de uma localização prende-se por se encontrar no limite do distrito do Porto, fazendo fronteira com o distrito de Braga. Utilizadores nestas situações tendem a efetuar mais do que uma pesquisa para poderem alternar entre distritos.

4.3.2 Pesquisa por distrito

Usando a pesquisa por distrito, os resultados obtidos são ordenados por data, estando a entrada de emprego mais recente da base de dados, no topo da lista.

Tabela 14 - Lista de ofertas de emprego para o distrito do Porto

Título	**Concelho/ Freguesia**
Gestor de Projeto	Porto
Analista BD	Porto
Comercial IT	Matosinhos
Programador C#	Porto
Programador C#	Porto
Programador Python	Porto
Programador PHP	Porto
Programador Python	S. Félix da Marinha

Tabela 15 - Lista de ofertas de emprego para o distrito de Braga

Título	**Concelho/ Freguesia**
Programador Obj-C	Famalicão
Programador Python	Famalicão
Programador C#	Braga
Programador C#	Famalicão
Analista BD	Braga
Programador Java	Braga

Nos exemplos apresentados nas tabelas 14 e 15, torna-se evidente que se obtiveram resultados que não se aplicam aos interesses do utilizador, como Analista BD, Gestor de Projeto e Comercial IT.

A apresentação do concelho ou freguesia é já algo mais do que o habitual numa aplicação de ofertas de emprego, que apenas olha ao distrito, levando muitas vezes a que o utilizador tenha de abrir a oferta para perceber se ainda interessa ou não. O exemplo mais crasso é a oferta para *Python* em S. Félix da Marinha. A distância do utilizador à localidade é excessiva, dado ser uma localidade no limite oposto do distrito do Porto, na fronteira com Aveiro.

Estas pesquisas pouco ou nada se relacionam com as preferências do utilizador

4.3.3 Resultados de uma pesquisa usando as preferências do utilizador

Usando o Portal de Emprego Inteligente com os parâmetros acima definidos obtiveram-se resultados ordenados pela sua relevância por ordem descendente.

Tabela 16 - Lista de ofertas de emprego usando o Portal de Emprego Inteligente

Título	**Concelho/Freguesia**	**Relevância**
Programador Python	Famalicao	94.74%
Programador C#	Famalicao	92.24%
Programador Java	Braga	84.76%
Programador Obj-C	Famalicao	84.74%
Programador PHP	Porto	83.61%
Programador Python	Porto	83.61%
Programador C#	Braga	82.26%
Programador C#	Porto	81.11%
Programador C#	Porto	81.11%

As entradas de emprego que não interessam ao utilizador, não são apresentadas. Os empregos referidos anteriormente, como sem interesse, Analista BD, Comercial IT e mesmo o emprego em S. Félix da Marinha, não são apresentados.

Contudo, apesar de não estar nas preferências do utilizador, a empregos das áreas C# e Objective C, são também apresentados. Isto deve-se dado a proximidade das categorias definidas pelo utilizador com os resultados da base de dados.

Apesar da diferença mínima, de apenas 0.02%, o emprego apresentado em terceiro lugar na lista, *Programador Java, Braga*, teve precedência sobre o quarto emprego que, apesar de ser mais próximo geograficamente do utilizador, é numa área que não consta na sua lista de preferências. É um emprego que foi obtido com relaxando as restrições do utilizador e calculando a sua relevância.

5 Conclusões

A aposta em plataformas inteligentes é a tendência de evolução da área das Tecnologias de Informação. A internet começou por ser uma rede em que os utilizadores tinham acesso a informação estática em diferentes sítios. Esta informação estava disponível apenas numa direção.

As aplicações entretanto evoluíram, gerando o que se chama atualmente a *Web 2.0*. Nesta nova geração, os utilizadores são igualmente responsáveis pela informação que se encontra disponível na internet. O maior exemplo deste advento é a *Wikipédia*. Iniciada em Janeiro de 2001, qualquer utilizador é livre de contribuir com informação para o crescimento da mesma. Enciclopédias como Encarta da Microsoft acabaram por tornar-se menos populares e serem descontinuadas.

Hoje em dia é espectável que as plataformas permitam comunicação dos dois lados. Há cada vez mais informação a ser gerada diariamente e novos desafios a precisarem de ser respondidos.

O relacionamento entre informação começa a ser cada vez mais valioso e relevante.

O Portal de Emprego Inteligente revela-se como uma evolução natural de portais de emprego. Apesar de amostras de teste pequenos, os resultados obtidos revelam-se mais satisfatórios comparativamente com as pesquisas usuais de emprego.

O relaxamento de restrições possibilita que se obtenham mais informação que revela dados bastante próximos do que se pretende pesquisar.

Os resultados obtidos, com o relaxamento das áreas de interesse do utilizado no Portal de Emprego Inteligente, são mais direcionados para as preferências do utilizador. Tipicamente, num portal de emprego convencional, as áreas de emprego são bastante genéricas abrangendo muitas subáreas que muitas vezes pouco têm a ver com o perfil utilizador. No

Portal de Emprego Inteligente, os resultados obtidos correspondem às preferências do utilizador, tendo em conta outras subáreas que são, no seu domínio, semelhantes.

Os cálculos da distância até ao emprego revelam-se bastantes vantajosos no sentido em que os resultados obtidos são mais satisfatórios. Contudo, este cálculo é feito descurando um pouco a precisão. Apenas tem em atenção a distância entre os pontos no numa esfera e o seu raio (neste caso o raio do Planeta Terra). Este cálculo engloba erros no sentido em que o Planeta Terra não é uma esfera. Considerando que a distância é calculada em linha recta (distância de um ponto ao outro) é descurada a topografia do terreno que influencia diretamente as vias de comunicação disponíveis entre os pontos. Também não é tido em conta outros meios de acesso como os meios de transporte, que também podem ser determinantes no acesso a uma localização, mesmo que esta seja relevante mas de difícil acesso.

Um dos aspectos fundamentais de um emprego é também o seu salário. Em plataformas doutros países, o atributo salário é apresentado num intervalo de valores e tanto pode ser anualmente como mensalmente, dependendo de país para país. Contudo, este atributo, em Portugal, tende a ser desvalorizado. Tipicamente um anúncio de emprego, excepto o caso de bolsas ou estágios profissionais remunerados, situações em que os valores são conhecidos, a informação disponível sobre este atributo costuma ser genérico referindo-se apenas a um pacote salarial atrativo e de acordo com a experiência demonstrado. Isto leva a que o atributo salário seja um atributo fraco para o cálculo da relevância de um emprego e foi retirado da equação.

A Plataforma de Emprego Inteligente pode ainda evoluir em vários factores.

Considerando este sistema como também pericial, implica que, à partida, o conhecimento sobre as áreas de emprego presentes, seja obtido previamente junto de um perito da mesma. Apesar de conseguir obter-se algo representante da realidade, há determinadas áreas que evoluem muito rapidamente, como as Tecnologias de Informação, dificultando o acompanhamento e atualização da plataforma.

Novos atributos podem ser considerados para o Portal de Emprego Inteligente para criar uma pesquisa ainda mais rica e relevante. A duração de emprego (*part-time*, *full-time* entre outros) seria uma mais valia a considerar num próximo desenvolvimento na plataforma.

Em diversas aplicações atuais, em especial lojas online, é habitual serem apresentadas ao utilizador sugestões de artigos que este poderá eventualmente gostar. Contudo, este tipo de sugestões baseia-se simplesmente em artigos já pré-determinados pelos responsáveis pela gestão da aplicação, que entendem ser um artigo relacionado, guardando essa informação estaticamente em bases de dados para todos os utilizadores, não havendo uma aprendizagem com a utilização da aplicação por parte do utilizador. Isto implica que, muitas vezes, o artigo proposto, não é relevante, havendo um esforço de gestão desperdiçado.

Com o intuito de apresentar ao utilizador automaticamente ofertas de emprego relevantes em função dos seus interesses, implementar um mecanismo que regista a atividade do utilizador durante a pesquisa na aplicação, em que por cada pesquisa que o utilizador efetue, as categorias pesquisadas são registadas na base de dados com um par (*id*,*n*), onde *id* corresponde ao *id* da categoria e *n* ao número de vezes que esta foi pesquisada e associar ao utilizador, apresentaria com mais precisão anúncios de emprego relevantes. Assim, vai-se mantendo o interesse do utilizador em determinadas áreas, havendo também a hipótese de posteriormente se desenvolver um novo mecanismo de sugestões para áreas que o utilizador já não pesquisa há algum tempo, permitindo determinar se ainda mantém o interesse nas mesmas.

O Portal de Emprego Inteligente apresenta uma nova forma de pesquisar emprego capaz de escalar com facilidade mantendo sempre a vantagem de uma pesquisa mais direcionada para o utilizador. Como em todos os sistemas, há sempre melhorias que podem ser realizadas quer em termos de algoritmos quer em termos de funcionalidades.

O mecanismo existente para o calculo de relevância de um emprego foi desenvolvido para que seja facilmente adaptado a outras situações. Sendo hoje em dia cada vez mais comum a pesquisa de recursos humanos pelas empresas usando plataformas sociais disponíveis na internet, como o *LinkedIN*, o Portal de Emprego Inteligente pode tirar vantagens deste cenário. Uma funcionalidade a ser desenvolvida seria efetuar a pesquisa a utilizadores em função do emprego. O cálculo da relevância é feito por utilizador por emprego, logo facilmente pode ser acrescentada esta funcionalidade tornando-se uma mais valia para a plataforma.

A relevância de um emprego é calculada em função dos seus atributos comparativamente com os do utilizador, considerando ainda o peso de cada um. Logo, um utilizador considerando um atributo mais importante do que outro obterá uma lista diferente de empregos (ou pelo menos apresentados noutra ordem). O Portal de Emprego Inteligente calcula a relevância considerando os pesos contudo, esta possibilidade não foi implementada no Interface do utilizador. Seria um pormenor interessante para utilizadores mais experientes.

O Portal de Emprego Inteligente poderia beneficiar de um agregador de empregos, capaz de pesquisar noutros portais em Portugal. Por cada emprego encontrado, para cada área contemplada no Portal de Empregos Inteligente, este poderia ser categorizado e adicionado à base de dados mesmo que apenas a informação para o cálculo de relevância fazendo posteriormente referência ao anúncio original. Seria uma mais valia dado que nem todas as empresas colocam o mesmo anúncio em todos os portais disponíveis, aumentando assim as ofertas ao utilizador colmatando algumas lacunas de outros agregadores de emprego existentes. Contudo, este desenvolvimento é bastante dispendioso e teria de se considerar se o seu desenvolvimento é justificado.

Referências

[Baeza-Yates and Ribeiro-Neto,1999] Baeza-Yates R., Ribeiro-Neto B. Modern Information Retrieval. Addison-Wesley, 1999.

[Beaulieu,2009] Beaulieu A., 2009. Learning SQL, O'Reilly Media, 2nd Edition.

[Berners-Lee et al., 1997] Berners-Lee T., Fielding R., Gettys J., Mogul J., Frystyk, H. Hypertext transfer protocol - HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt [último acesso: Março 2014]

[Berners-Lee et al., 1998] Berners-Lee T., Fielding R., Masinter L. Uniform resource identifiers (URI): Generic syntax, http://www.ietf.org/rfc/rfc2396.txt [último acesso: Março 2014]

[Brummelen,2009] Brummelen G. The mathematics of the heavens and the earth: The Early History of Trigonometry, 2009.

[Cayrol and Barrère, 1997] Cayrol A., Barrère P., 1997. Guia de PNL. Novas técnicas para o desenvolvimento pessoal e profissional. Editora Record, 2ª edição

[Chakrabarti et al.,2003] Chakrabarti K., Ortega M., Mehrotra S., Porkaew K. Evaluating refined queries in top-k retrieval systems. IEEE Transactions on Knowledge and Data Engineering, 2003.

[Chappel,2009] Chappel D. SOAP vs. REST: Complements or Competitors? In Proceedings of Developer Summit, California, 2009.

[Chaudhuri,1990] Chaudhuri S. Generalization and a framework for query modification. Sixth International Conference on Data Engineering, California, 1990.

[Chaudhuri and Gravano,1999] Chaudhuri S., Gravano L. Evaluating top-k selection queries. In Proceedings of the Conference on Very Large Databases, p. 397–410, 1999.

[Chodorow and Dirolf,2010] Chodorow K., Dirolf M., 2010. MongoDB: The Definitive Guide. O'Reilly Media.

[Chu et al.,1996a] Chu W., Chiang K., Hsu C., Yau H. An error-based conceptual clustering method for providing approximate query answers, p.216–230, 1996.

[Chu et al.,1996b] Chu W., Yang H., Chiang K., Minock M., Chow G., Larson C. Cobase: A scalable and extensible cooperative information system. Journal of Intelligence Information Systems, p.223–59, 1996.

[Davis et al., 1977] Davis R., Buchanan B., Shortliffe E. Production rules as a representation for a knowledge-based consultation program. Artificial Intelligence,1977.

[Dorigo et al.,2006] Dorigo M., Birattari M., Stützle T. Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique, 2006

[Endrei et al., 2004] Endrei M., Ang J., Arsanjani A., Chua S., Comte P. Patterns: Service-Oriented Architecture and Web Services, IBM, 2004.

[Godfrey,1998] Godfrey P. Relaxation in Web Search: A New Paradigm for Search by Boolean Queries. Army Research Laboratory, 1998.

[Janas,1979] Janas J. Towards more informative user interfaces. Fifth International Conference on Very Large Data Bases, Brazil, 1979.

[Júnior,2013] Júnior, E. Aplicações do Teorema de Pitágoras: Trigonometria Esférica, 2013.

[Kaplan,1982] Kaplan S., Cooperative responses from a portable natural

language query system, 1982.
[Kennedy and Eberhart, 1995] Kennedy J., Eberhart R. Particle Swarm Optimization, 1995.
[Lin et al., 2011] Lin G., Desmond K., Htoon N., Thuat N. A Fresh Graduate's Guide to Software Development Tools and Technologies, Cap. 4, 2011.
[Lindsay et al., 1980] Lindsay R., Buchanan, B., Feigenbaum E., Lederberg J. Applications of Artificial Intelligence to Chemistry: The DENDRAL Project. McGraw-Hill, New York, 1980.
[Milbert and Smith,1996] Milbert D., Smith D. Converting GPS Height into NAVD88 Elevation with the GEOID96 Geoid Height Model, http://www.ngs.noaa.gov/PUBS_LIB/gislis96.html [último acesso: Agosto 2014]
[Muslea, 2004] Muslea I. Machine Learning for Online Query Relaxation, SRI International, 2004.
[Poli,2007] Poli R. An analysis of publications on particle swarm optimisation applications. Technical Report, 2007.
[Sinnot,1984] Sinnott R. Virtues of the Haversine, Sky & Telescope magazine, 1984.
[Williams,2004] Williams, D. Earth Fact Sheet, http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html [último acesso: Agosto 2014]