FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Unificação e expansão da interface de pesquisa no webmail

Rui Alexandre Rodrigues Carneiro

Relatório de Projecto

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Jaime dos Santos Cardoso (Doutor)

7 de Julho de 2008

Unificação e expansão da interface de pesquisa no webmail

Rui Alexandre Rodrigues Carneiro

Relatório de Projecto

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Pedro Alexandre Guimarães Lobo Ferreira do Souto (Doutor)

Arguente: Paulo Alexandre Ribeiro Cortez (Doutor)

Vogal: Jaime dos Santos Cardoso (Doutor)

31 de Julho de 2008

Resumo

O *email* é um dos serviços electrónicos mais utilizado pela população em geral nos dias de hoje. Apesar de nos seus primórdios o *email* servir apenas para a recepção e envio de mensagens, actualmente, o número de funcionalidades que os sistemas de *email* fornecem é bastante superior. Esta situação rapidamente deu origem a um fenómeno denominado "Email Overload" que consiste sobrecarga de informação sobre os seus utilizadores ao ponto de estes não conseguirem gerir os seus *emails*.

A Portugalmail Comunicações S.A. é uma empresa que fornece serviço de *webmail* e que tem como objectivo tornar-se pioneira nesta área, estando, desta forma, sensível a este problema. Assim, propôs este projecto, Unificação e expansão da interface de pesquisa no webmail, com o intuito de obter uma implementação de uma possível solução, através da expansão do domínio da pesquisa actual em *email*, além da unificação de todas as pesquisas que existem actualmente (p.e. contactos, calendário, gestor de ficheiros).

Os objectivos deste trabalho consistem, resumidamente, no melhoramento do serviço de pesquisa em *email*, através da aplicação de uma ferramenta de Recuperação de Informação (indexação e pesquisa), e na criação de um módulo e interface de pesquisa. De entre isto, o aspecto em que este projecto assentou principalmente foi o alargamento do domínio de pesquisa ao conteúdo dos anexos dos *emails*. O restante desenvolvimento consistiu na adaptação do sistema de forma a suportar esta funcionalidade. Todas as decisões para este projecto foram meticulosamente ponderadas, tendo sido sujeitas a diversos testes.

O tempo, espaço e memória utilizados na indexação e pesquisa foram medidos em enúmeros testes e provaram a viabilidade da solução proposta. Os resultados obtidos foram considerados bastante satisfatórios por parte da empresa, uma vez que se mostraram ser complementamente comportáveis pelas capacidades desta. Como resultado deste projecto, obteve-se ainda um protótipo funcional de interface que permite uma unificação de todos os módulos existentes assim como a geração de novos tipos de informação.

Dada a complexidade envolvida na implementação das funcionalidades deste projecto, este incidiu na criação de bases sustentadas que permitem desenvolvimentos futuros. Contudo, é possível concluir que este projecto foi realizado com sucesso através da quantidade e qualidade dos requisitos implementados.

Uma das particularidades que mais valorizam este projecto é o facto de o código desenvolvido vir a ser englobado numa das próximas distribuições do projecto *open-source* Dovecot.

Abstract

Nowadays, the email is one of the most used electronics services. Despite its only purpose was originally the reception and send of online messages, today the number of functionalities has increased greatly. This rapidily originated a phenomenom usually known as Email Overload which consists in the overload of information the daily users have to deal with, causing problems in managing their emails.

Portugalmail Comunicações S.A. is a company which provides webmail service and, presently, has as goal to become a pioneer in the field. Therefore, PortugalMail proposed this project, Unifying and expanding the webmail search interface, in order to obtain a possible solution's implementation, through the expansion of the actual email search domain and the unification of all todays's search (e.g. contacts, calendar, files management).

This project's goals are the improvement of the email search engine, applying a tool of Information Retrieval (indexing and search), and the development of a searching module and interface. From all these goals, the main one is the extension of the search domain to the content of the emails' attachments. The remaining development consisted in adaptating all the system so that it suported this functionality. Every call made during this project was carefully contemplated, having been submited to several and distinct tests.

The time, space and memory that were necessary to the indexing and search were measured in several tests proving this solution's viability. The results were considered plenty satisfactory as the company can easily bear them. During this project an interface functional prototype which enables the unification of all the project's functionalities has also been developed.

Due to this project's complexity, its main purpose was to create some well funded bases that enabled future developments. However, it is still possible to realize this project success through the quantity and quality of the implemented requirements.

One valuable particularity of this project is the fact that the code developed will be covered in one of the open-source project Dovecot's following distributions.

Agradecimentos

Gostaria em primerio lugar de agradecer ao Prof. Jaime dos Santos Cardoso pela excelente orientação que prestou neste projecto e pelo apoio constante até ao último minuto da entrega,

A todos colaboradores da Portugalmail que me proporcionaram um óptimo ambiente para a realização do projecto, ajudando-me sempre que necessário,

Aos meus pais por todos os sacrifícios que passaram para que chegasse este dia,

A todos os meus amigos e familiares por todos os excelentes momentos passados,

E finalmente, à Inês pela paciência, carinho e apoio sempre dados.

Rui Carneiro

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ADSL	Asymmetric Digital Subscriber Line
AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DIMP	Dynamic Internet Messaging Program
FTS	Full Text Search
GPL	GNU General Public License
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
IMP	Internet Messaging Program
LDAP	Lightweight Directory Access Protocol
MIME	Multipurpose Internet Mail Extensions
PHP	PHP: Hypertext Preprocessor
PNL	Processamento de Linguagem Natural
POP	Post Office Protocol
RDF	Resource Description Framework
RI	Recuperação de Informação
STMP	Simple Mail Transfer Protocol
TCP/IP	Internet Protocol Suite
URIs	Uniform Resource Identifiers
XML	eXtensible Markup Language

Capítulo 1

Introdução

Neste capítulo será feita uma breve contextualização do projecto, apresentando a empresa Portugalmail, onde o projecto foi desenvolvido, e o seu novo sistema de Webmail, onde este projecto se insere. Será ainda descrito o projecto, apresentada a motivação para a sua realização e os resultados esperados no fim do mesmo. Por fim, será descrita a restante organização do documento.

1.1 Portugalmail

A Portugalmail – Comunicações S.A. é uma companhia que se baseia em tecnologias Web. Foi criada em 20 de Março de 1999 com o objectivo de criar o primeiro serviço de *email* português. Desde então, a Portugalmail alargou o seu leque de serviços *Web*, incluindo Acesso à Internet *ADSL*, Registo de domínios, Alojamento, Publicidade *Web*, um serviço de blogs internacional (www.blog.com), uma comunidade sobre futebol (www.futebolar.com) e consultoria de software de gestão *PHC* [1].

Figura 1.1: Logótipo da Portugalmail - Comunicações S.A.

Actualmente, a Portugalmail procura ser a referência em serviços de *webmail* oferecendo 2Gb de espaço para utilizadores individuais e uma plataforma dedicada de *email* para empresas.

[1] **PHC Software** - www.phc.pt

Media Capital, Onitelecom, Grupo Sonae, Grupo M. Coutinho, Copicanola e Seque, Banco de Investimento Global e AICCOPN são apenas alguns dos actuais clientes da Portugalmail.

1.2 Sistema Webmail

O serviço de *webmail* está actualmente dividido em duas diferentes categorias, utilizadores individuais (serviço gratuito) e plataformas profissionais (serviço pago).

Qualquer utilizador pode criar gratuitamente uma conta de *email* acedendo aos domínios portugalmail.pt ou portugalmail.com.

A plataforma empresarial é uma solução dedicada a empresas que pertendem fazer o *outsource* do seu sistema de *email* (gestão, manutenção e equipamento). Actualmente, a Portugalmail gere um grande número de empresas, desde pequenas companhias com menos de 100 utilizadores, até grandes empresas com mais de 300.000 utilizadores.

A Portugalmail procura estar sempre no topo ao nível do serviço *webmail* e a prova disso são as constantes inovações aos seus serviços. Há dois anos atrás, a Portugalmail desenvolveu uma nova interface *AJAX* para o seu serviço *webmail* baseado no trabalho de Rui Santos [San06], que permitiu à empresa atingir o nível de qualidade dos maiores serviços de *webmail* do mundo. Actualmente, a Portugalmail está a desenvolver uma nova plataforma completamente de raiz. Esta decisão foi suportada no trabalho de Felipe Ávila da Costa [Cos09], que concebeu todo um novo paradigma de interface de webmail, sendo algumas das novas funcionalidades a detecção de padrões em *emails*, interfaces adaptativas, integração com calendário, novo sistema de pesquisa e a funcionalidade central, o desenvolvimento de uma secretária virtual, que disponibilizaria o mesmo tipo de ajuda que uma secretária real.

1.3 O Projecto

1.3.1 Contexto e Motivação

O *email* é uma das aplicações criadas com mais sucesso até hoje. Contudo, hoje em dia, o *email* deixou de servir apenas como ferramenta de comunicação assíncrona e passou a incorporar muitas outras funcionalidades fora do âmbito original para o qual foi criado. Este tipo de funcionalidades passa pela gestão de tarefas, contactos (p.e. nomes, números, moradas), uso incorporado de calendário, *chat*, entre muitas outras aplicações variando de serviço para serviço. Com este alargamento dos serviços prestados pelo *webmail* actual, rapidamente se verificou um novo fenómeno denominado *Email Overload*. O primeiro uso deste termo veio de Whittaker e Sidner [WS96], que o descreveram como o fenómeno de o email estar a acopular diversas funcionalidades para o qual não tinha sido

desenvolvido. Contudo, o uso deste termo foi-se redefinindo ao longo do tempo e, actualmente, não significa apenas o facto de o email ser sobrecarregado com funcionalidades mas sim o facto de um utilizador não conseguir gerir o seu *email* (independentemente do fluxo de *emails* que este recebe).

Apesar de a pesquisa *web* ter evoluído extraordinariamente (p.e. a ascensão da *Google* [2] e o recente aparecimento da *Wolfram* [3]), a pesquisa em *email* continua, estranhamente, igual há 20 anos atrás. Os serviços actuais, à excepção da *Yahoo!* [4], possuem sistemas de pesquisa básicos onde apenas nos é permitido pesquisar pelo conteúdo de texto de um *email* não fazendo, assim, uso das mais avançadas tecnologias actualmente disponíveis e usadas noutros domínios (p.e. pesquisa na *Web*, *Desktop*, Warehouses).

Outra das motivações para este projecto é o facto de, nos serviços actuais de *email*, a pesquisa se encontrar espalhada pelos diferentes módulos do sistema. Um utilizador, caso deseje pesquisar por todos os *emails* de uma determinada pessoa, ao fazer a pesquisa "João", não pode, actualmente, saber quais os *emails* que recebeu do "João", quais os seus dados pessoais guardados nos contactos ou até quais os eventos do calendário em que este está inscrito. Para colmatar esta lacuna nos sistemas actuais, foi introduzida uma forte componente de unificação no sistema de pesquisa a desenvolver.

1.3.2 Objectivos

Em traços gerais, o grande objectivo deste projecto consiste em desenvolver um novo sistema de pesquisa que unifique todas as aplicações existentes na plataforma actual. Para que isto seja alcançado foram estabelecidos os seguintes objectivos:

- Fornecer a funcionalidade de pesquisa por texto integral em *emails*;
- Alargar os elementos disponíveis para pesquisa em *emails* (p.e. anexos, metadata)
- Escolha e configuração de uma ferramenta de Recuperação de Informação;
- Criação de um módulo de pesquisa no cliente actual que unifique todas as pesquisas;
- Criação de uma interface de pesquisa que sirva como demonstração da tecnologia desenvolvida.

Como podemos observar, os objectivos fundamentais deste trabalho são acrescentar algumas funcionalidades à pesquisa em *emails* e unificar as restantes pesquisas já existentes (com possível optimização das mesmas).

[2] **Google** - www.google.com
[3] **Wolfram** - www.wolfram.com
[4] **Yahoo!** - www.yahoo.com

1.3.3 Resultados Esperados

Como resultado deste projecto, espera-se que a pesquisa no sistema de *webmail* actual consiga abranger mais elementos de cada mensagem. Cada mensagem possui inúmeros elementos de informação que não são utilizados como, por exemplo, anexos, relevância, meta-informação, entre outros. Por outro lado, pretende-se uma unificação da pesquisa entre as várias aplicações existentes (p.e gestor de eventos, contactos, *email*, ficheiros, entre outros).

Pretende-se, no entanto, que tudo isto seja desenvolvido tendo em conta a plataforma actual e tendo em constante consideração a API utilizada. Deseja-se que o detalhe e desempenho das pesquisas seja de acordo com os padrões esperados pelos utilizadores deste serviço. A interface desenvolvida deve servir como demonstração da tecnologia de pesquisa utilizada e das suas capacidades de expansão futura.

1.4 Contribuições

Durante o desenvolvimento deste projecto foram várias as contribuições resultantes. Para a empresa, os objectivos atingidos resultaram nas seguintes contribuições:

- Sistema de pesquisa IMAP por texto integral.
- Suporte de indexação (e consequente pesquisa) de ficheiros doc, xsl, ppt (e pps) e pdf.
- Criação do módulo de pesquisa unificada para o novo sistema *webmail*.
- Criação de uma base sólida para o desenvolvimento do novo módulo de gestão de ficheiros.
- Criação de uma API de comunicação com o webservice Solr.

Este projecto fez surgir o interesse da comunidade envolvida no projecto *Open-Source* Dovecot [5] em especial do seu criador Timo Sirainen. Devido ao cariz inovador deste projecto surgiu a hipótese de este projecto poder contribuir de forma activa no desenvolvimento de novas funcionalidades para o projecto Dovecot. Esta proposta foi aceite e como consequência do trabalho realizado resultaram as seguintes contribuições para o projecto Dovecot:

- Correcção de alguns bugs no parsing de mensagem.
- Adicionada a funcionalidade de indexação de anexos (recorrendo a programas externos).

[5] **Dovecot** - www.dovecot.org

- Criação de uma API que permitirá futuros desenvolvimentos na pesquisa de texto integral.

1.5 Estrutura do Relatório

O presente documento encontra-se dividido em oito capítulos que descrevem todas as áreas envolvidas no desenvolvimento deste projecto. É fornecido também um conjunto de anexos que pretendem complementar a informação existente nos oito capítulos deste documento.

Após este capítulo introdutório, no Capítulo 2, Revisão Bibliográfia, é analisado o estado da arte das áreas mais relevantes para este projecto.

No Capítulo 3, Avaliação Tecnológica, são estudadas todas as tecnologias que serão parte integrante deste projecto. Serão feitos também alguns testes e *benchmarks* de forma a consolidar as opções tomadas a nível tecnológico.

No Capítulo 4, Especificação, é apresentada a arquitectura actual, descreve-se os vários tipos de requisitos da solução a implementar e apresentado um protótipo como solução final para o utilizador.

No Capítulo 5, Solução Proposta, são apresentadas as propostas para resolver os problemas em questão. É apresentada uma nova arquitectura e explicada cada alteração efectuada.

No Capítulo 6, Implementação, é feita uma descrição detalhada de todas as opções de implementação tomadas.

No Capítulo 7, Testes e Resultados, são feitos alguns testes para validar a eficiência da proposta implementada.

No Capítulo 8, Conclusões e Perspectivas de Trabalho Futuro, tiram-se conclusões acerca do trabalho desenvolvido, aspectos inovadores e limitações do projecto assim como perspectivas futuras de desenvolvimento.

Capítulo 2

Revisão Bibliográfica

Nesta secção irá ser feita uma descrição do estado de arte da ciência, em especial no que diz respeito aos problemas de actuais de gestão de *emails*. Irão ser revistos os mais importantes estudos na àrea da sumarização, filtragem de *email*, pesquisa semântica, *ranking* de pesquisa entre outros.

2.1 Problema

Estima-se que, em Agosto de 2008, o número de utilizadores de *email* eram de 1.3 mil milhões. Isto significa que uma em cada 5 pessoas na Terra utilizam *email*. Apesar deste ter sido desenhado como uma forma simples de troca de mensagens, hoje em dia é muito mais que isso. O *email* é agora utilizado para diversos fins como, por exemplo, calendarização de tarefas, marcação de encontros, gestão de tarefas e/ou contactos, entre outras funcionalidades que diferem de serviço para serviço.

Na Portugalmail, o sistema actual é constituído por três diferentes módulos: *email*, gestor de contactos e calendário. Cada um destes módulos possui um sistema de pesquisa específico, o que pode tornar complicado a sua utilização. Pertende-se, por isso, criar um sistema de pesquisa unificado entre os diferentes módulos actuais e os novos que irão ser criados para a nova plataforma. A pesquisa mais importante de todas é a pesquisa em *email* pelo que será onde irá recair grande parte deste trabalho. Espera-se que a nova interface de pesquisa abranja o máximo de elementos possíveis de uma mensagem (conteúdo, anexos, relevância, entre outros).

Este novo sistema de pesquisa deve ser integrado com a tecnologia existente na empresa (tal será abordado no capítulo seguinte).

2.2 Recuperação de Informação

Esta é a área global onde se enquadra o projecto. Recuperação de Informação (RI) é a ciência computacional que lida com a pesquisa de documentos, informação contida nesses documentos e meta-informação associadas a estes. A fonte de informação pode estar disponível sob forma de textos, sons, imagens ou dados. O processo de recuperação de informação começa quando um utilizador efectua uma pesquisa no sistema (p.e. pesquisar FEUP no *Google*). Em grande parte dos sistemas, nunca é retornado apenas um resultado, mas sim um conjunto deles ordenado por diferentes níveis de relevância.

Os testes que avaliam o desempenho destes sistemas baseiam-se em duas métricas estatísticas [Man07]:

$$Precisão = \frac{|documentos\ relevantes| \bigcap |documentos\ devolvidos|}{|documentos\ devolvidos|} \tag{2.1}$$

$$Sensibilidade = \frac{|documentos\ relevantes| \bigcap |documentos\ devolvidos|}{|documentos\ relevantes|} \tag{2.2}$$

Por Precisão entende-se fracção de documentos devolvidos que são relevantes para o utilizador. Por outro lado, Sensibilidade define a fracção de documentos relevantes que são devolvidos para a pesquisa em causa.

Um dos ramos mais recentes da recuperação de informação é a pesquisa de informação em dados multimédia. Isto é uma tarefa complexa pois a origem dos dados pode ser qualquer fonte multimédia (p.e. vídeo, imagem, som, etc). Actualmente, os maiores desafios nesta área passam, por exemplo, pela pesquisa semântica, sistemas de *feedback* ou detecção de conceitos em fontes multimédia [LSDJ06].

2.3 Data Mining

À medida que a quantidade de dados com que lidamos diariamente vai aumentando, aumenta também o interesse por ferramentas de *Data Mining*. *Data Mining* consiste no processamento de grandes quantidades de dados com o objectivo de detectar padrões que são invisíveis à partida. Este processo de análise divide-se em 3 partes: pré-processamento, extracção e validação.

Visto os *datasets* geralmente utilizados em *Data Mining* serem muito grandes, a extracção de dados pode demorar demasiado tempo. Devido a este factor é comum os *datasets* serem pré-processados de forma a diminuir o tamanho dos mesmos. Estas técnicas baseiam-se na selecção de atributos (alguns atributos podem ser repetidos ou irrelevantes para o resultado final), amostragem, descritização de valores contínuos, entre outros. A

grande importância desta etapa faz com que mais de metade do tempo de análise seja aqui utilizado.

Depois de pré-processado, existem diversos algortimos possíveis para proceder à extracção de conhecimento:

- Classificação — pretende catalogar determinados elementos (p.e. classificar um *email* como *spam*)
- *Clustering* — agrupa os elementos semelhantes em grupos (p.e. sumariar um conjunto de *emails* agrupando-os por semelhança)
- Regressão — método estatístico que permite inferir a relação de uma variável dependente (contínua) com variáveis independentes. O método de estimação mais amplamente utilizado é o método dos mínimos quadrados.
- Regras de Associação — pretendem encontrar relação entre os diferentes atributos do *dataset* (p.e. descobrir qual a influência que o estado social tem nos rendimentos mensais)

Na fase final, é suposto validar os resultados obtidos com a fase anterior. Para isso, é usado um *dataset* de teste com dados não existentes no *dataset* de treino (uso para a extracção de dados).

Este tipo de técnicas é largamente usada em diversas áreas da problemática do *Email Overload* que serão analisadas em maior detalhe em secções futuras.

2.4 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é um campo do estudo da inteligência artificial e da linguística que pretende dotar as máquinas de capacidade de compreensão e resposta da linguagem humana. Espera-se, por isso, num estado perfeito, que numa interacção pessoa-computador não tenha de existir adaptação do humano às limitações linguísticas do computador. Esta tecnologia é actualmente usada em diversas aplicações como, por exemplo, traduções automáticas, extracção de informação, compreensão e geração de linguagem natural, sumarização de texto, conversão de texto para fala ou legendagem automática.

O PLN está voltado para diferentes aspectos da linguística:

- **Fonética** — processamento da linguagem através do reconhecimento dos sons (não aplicável em todos os casos).
- **Morfológica** — estudo das diversas palavras contidas numa frase visando a sua classe gramatical. A morfologia trata as palavras quanto à sua estrutura, forma, flexão e classificação.

- **Sintática** — estuda os processos generativos ou combinatórios das frases das línguas naturais, tendo em vista especificar a sua estrutura interna e funcionamento.
- **Semântica** — o analisador semântico analisa o significado das estruturas criadas pelo analisador sintático.
- **Pragmática** — trata da análise do contexto em que a frase está inserido.

Das análises referidas anteriormente, a análise morfológica, sintática e semântica estão actualmente bem definidas para cada dialecto. No entanto, a fonética e a pragmática destacam-se pela sua dificuldade. Identificar a forma como as palavras são pronunciadas e o contexto em que estão inseridas é um dos desafios actuais da PLN.

2.5 Web Semântica

A *Web* Semântica é uma extensão da *Web* actual que permite que, recorrendo a regras semânticas, a *Web* seja percebida tanto por humanos como computadores [Din04]. Os elementos da Web Semântica são expressos em especificações formais.

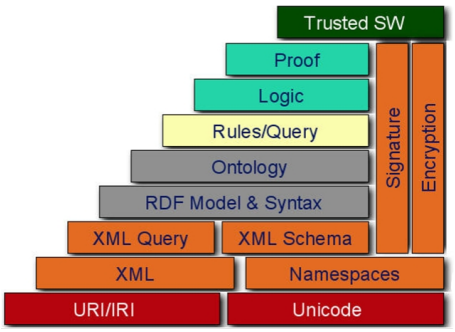

Figura 2.1: Estrutura da Web Semântica

Estas especificações são baseadas em RDF (*Resource Description Framework*) que é uma linguagem suportada em XML (eXtensible Markup Language) e URIs (*Uniform Resource Identifiers*) que permite representar informação sobre recursos na *Web*, sejam eles objectos recuperáveis (p.e. páginas Web) ou não recuperáveis mas apenas identificáveis (p.e. Pessoas, produtos à venda, preferências de utilizadores). O vocabulário genérico utilizado nos RDF é especificado por RDF Schemas que são uma extensão semântica ao primeiro formato e definem as classes, propriedades e seus significados. Para definir o vocabulário para um domínio específico são utilizadas ontologias de forma a formalizar o domínio das classes e propriedades definidas no RDF Schema.

2.6 Gestão de Email

Após a expansão do uso do *email* e das suas funcionalidades, a gestão deste passou a ser um dos principais factores de produtividade em empresas. Um dos principais problemas inerentes à gestão do *email* é um fenómeno denominado de *Email Overload*. Este termo foi usado pela primeira vez por Whittaker e Sidner [WS96] que o usaram para descrever o aumento de funcionalidades adjacentes ao *email* (calendário, gestão de ficheiros, contactos, entre outros), tornando-os, hoje em dia, nos denominados *Gestores de Informação Pessoal*. Contudo, o significado deste termo foi evoluindo, sendo hoje utilizado para descrever a percepção que utilizador tem de que perdeu o controlo sobre o seu próprio *email* [BH06, DK06]. Esta percepção por parte do utilizador tem diversos efeitos negativos no mesmo, passando pelo aumento dos níveis de *stress*, baixa de produtividade ou a perda do rasto de documentos.

Nas secções seguintes irão ser abordadas algumas técnicas para a resolução da problemática do *Email Overload*.

2.6.1 Sumarização

O sumário de um ou vários *emails* pode ser utilizado para facilitar a forma como os utilizadores lidam com a sobrecarga a que estão sujeitos. Estes devem permitir ao utilizador efectuar rapidamente uma triagem ao seu *email* e ajudar na recuperação da informação *ad hoc* e pesquisa [DWPP08].

Um dos métodos existentes para implementar esta solução é o uso de sistemas inteligentes de aprendizagem, usados de forma supervisionada ou não-supervisionada. Em 2001, *Muresan el al* [MTK01] apresentou um modelo baseado em aprendizagem supervisionada. Este sistema baseia-se em regras linguísticas definidas *a priori* e treinadas com base em anotações feitas individualmente pelo utilizador. Contudo, segundo *Dredze* [DWPP08], é irrealista esperar obter estas anotações em centenas de *emails*. Este sugere um sistema de aprendizagem não supervisionado onde são usados dois modelos conceptuais:

- LSA (*Latent Semantic Analysis*) — analisa as relações entre um conjunto de documentos e os termos que eles contêm, produzindo um conjunto de conceitos relacionados com os documentos e os termos;
- LDA (*Latent Dirichlet Allocation*) — um modelo hierárquico probabilístico *Bayesiano* [AR08].

Outro método interessante é o uso de técnicas de *clustering* que permitem agrupar os *emails* baseando-se na semelhança do conteúdo das frases nele contidas [WM04, ZDL08]. Um deles é o sugerido por *Zajic et al* [ZDL08] onde a aglomeração das frases é feita

através de um modelo linguístico ou de um modelo probabilístico denominado "Modelo oculto de *Markov*" (*Hidden Markov Model*).

2.6.2 Filtros

Outro caminho seguido para limitar a quantidade de informação com que o utilizador é confrontado quando utiliza o seu *email* é o uso de filtros.

Uma das suas utilizações é na criação automática de pastas no *email* recorrendo, por exemplo, à sumarização de *emails* como referido por Dredze [DWPP08]. Contudo, *Schuff et al* [STD06] sugeriram um sistema mais completo que não utiliza apenas o texto da mensagem mas também elementos como a data, emissor e receptor da mensagem. O sistema sugerido permite, ainda, que o utilizador altere os pesos dos diferentes atributos conforme o seu interesse pessoal. Todas estas variáveis são no fim utilizadas para, através de um algoritmo de *clustering*, separar as mensagens por pastas. *Schuff et al* [STD06] acreditam que assim, limitando a carga cognitiva do utilizador, mesmo que a quantidade de *emails* recebidos não páre de aumentar, é uma excelente forma de retardar a sensação de *Email Overload* e, assim, evitar quebras de produtividade.

Numa abordagem completamente diferente, *Kadoya et al* [KFA+04] propuseram um sistema de filtros baseados em prioridade temporal. Esta técnica tem como objectivo dar mais relevância aos *emails* que necessitam de resposta e ordená-los temporalmente. A ideia é saber quando um *email* foi recebido (através da informação contida no cabeçalho da mensagem), quando necessita de resposta (se necessitar) e calcular o intervalo de tempo entre estas duas datas que serve como valor de *ranking*. O sistema faz uso de técnicas de detecção de padrões de forma a identificar determinados padrões que indiciam a necessidade de resposta. *Kadoya et al* [KFA+04] sugerem para isso a criação de atributos compostos por 3 valores: palavras, categoria (verbo, substantivo ou adjectivo) e contexto em que está inserido (semântica).

2.6.3 Análise Forense

Devido à sua simplicidade e vulnarabilidade, a comunicação por email é usada abusivamente para a prática de diversos crimes. *Spam*, *phishing*, tráfico de drogas, pornografia infantil e assédio sexual são apenas alguns dos crimes mais comuns. Para a identificação dos responsáveis por estes crimes são necessárias ferramentas forenses que permitam ao investigador analisar *emails* suspeitos.

E-mail Mining Toolkit (EMT) é uma das *frameworks* que facilita o trabalho dos investigadores forenses [SHH+06]. Esta *framework* é capaz de calcular o perfil comportamental de um utilizador, baseando-se nas suas contas de *email*. Esta ferramenta, apesar de útil a gerar relatórios sumarizados (ver 2.6.1), não aborda a problemática da verificação de identidade nem da detecção de *emails* semelhantes.

Existem, no entanto, outros estudos que tentaram colmatar as lacunas do EMT. Em 2008, *Abbasi* e *Chen* [AC08] utilizaram a estilística linguistica para a detecção de identidades em *datasets online* como *email*, *instant messaging* ou comentários de *feedback*. Esta técnica envolve o estudo léxico, sintáctico, estrutural, de contexto e de atributos idiossincráticos como erros gramaticais, *misspellings* entre outras anomalias. Esta abordagem foi também seguida por *Iqbal et al* [IHFD08] e *Debbabi et al* [HDL+09]. Se, por um lado, *Iqbal et al* desenvolveram uma *framework* que aborda apenas o problema da autenticidade, já *Hadjidj et al* criaram uma *framework* mais completa que contempla, para além da estilística linguística, funcionalidades como a detecção de comportamentos desviantes (comparando com comportamentos normais) ou rastreamento geográfico (localizando o endereço físico de um IP).

2.6.4 SPAM

Apesar de ser um problema antigo, o *SPAM* tem-se tornado num dos maiores problemas da nossa sociedade não só pela quantidade mas também pelo crescente número de crimes que a ele estão envolvidos [WSWS08]. As estatísticas apontam para que 92-99% dos *emails* recebidos sejam *SPAM* (apesar de grande parte ser bloqueada ao nível do *SMTP*) [Cos09]. O *SPAM* em *email* tem 3 usos principais:

- Promoção de produtos e serviços — os *spammers* são pagos por empresas para promoção dos seus produtos. Normalmente, isto é feito através dos emails (mais recentemente em fóruns e blogs);
- Instalação de programas ilícitos — executam operações não desejadas pelo utilizador como a abertura de páginas *Web*, *pop-ups* de publicidade ou execução de *key logs* com o intuito de roubar informação do utilizador (palavras-passe, contas, informação pessoal, etc);
- *Pishing* — roubo de informação fazendo-se passar por uma entidade legítima (entidades bancárias por exemplo);

Algumas das técnicas para a resolução deste problema já foram abordadas em secções anteriores (ver 2.6.2 e 2.6.3). Contudo, o estudo de Hayati e Potdar [HP08] analisa um grande número de outros métodos de detecção e prevenção de *SPAM*. Filtros *bayesianos* ou de palavras chave, aproximação baseada em memória, programação genética ou processos *zombie*, que simulam um utilizador afectado que serve de modelo de teste, são apenas alguns dos métodos avaliados nesse trabalho. Apesar das técnicas de *anti-spam* terem aumentado em número e eficácia, os métodos utilizados pelos *spammers* também evoluiram. Actualmente, um dos métodos mais eficazes é o *SPAM* por imagens (Figura 2.2) que consiste em inserir o conteúdo do SPAM em imagens de forma a que seja "invisível" aos *anti-spam* e não ao olho humano.

Figura 2.2: Exemplo de *SPAM* numa imagem

As soluções para esta nova ameaça passam por análise de imagem, histogramas e análise de meta-dados. Apesar de ser possível em grande parte dos casos identificar e extrair o texto desenhado numa imagem, essa operação é demasiado custosa. Devido a esse facto, as técnicas usadas passam pelo cálculo das médias de cor e saturação, análise de texturas e teste aleatório de *pixels*. A abordagem do uso de meta-dados faz uso de dados como o tamanho do ficheiro, o tamanho da imagem e o formato da imagem, entre outros. Com os dados obtidos com as técnicas anteriores, podemos ainda criar histogramas de casos positivos de forma a permitir ao sistema identificar automaticamente um mail de *SPAM*.

2.6.5 Pesquisa Semântica

O termo "Pesquisa Semântica" é usado para descrever diversos tipos de pesquisa [Man07, ULM08, ULL+07, Din04] que se podem agrupar em duas diferentes áreas: Processamento de Linguagem Natural e Web Semântica (Meta-informação).

2.6.5.1 Processamento de Linguagem Natural

Um bom exemplo da utilização desta tecnologia é o trabalho desenvolvido por *Hao et al* [HLW+08]. Este permite categorizar e ordenar resultados provenientes do motor de pesquisa da *Google*. A similaridade entre resultados é obtida num algoritmo baseado no *WordNet* [1] que consiste numa grande base de dados lexical (da língua inglesa). Os resultados do trabalho de *Hao et al* mostraram que 72.7% dos dados foram correctamente classificados nos três primeiros lugares.

Por outro lado *Chantree* [Cha04] declara a ambiguidade como um dos maiores problemas (senão o maior) das análises de linguagem natural. Segundo este, apesar de grande parte das ferramentas de PLN terem excelentes resultados em domínios pequenos e restritos, a abertura destas análises a domínios mais gerais levanta o problema da ambiguidade num discurso. Existem três tipos de ambiguidades:

[1] **WordNet** - wordnet.princeton.edu

- **Lexical** — palavras com vários significados (palavras homónimas) e palavras diferentes com o mesmo significado (sinónimos);
- **Semântica** — várias interpretações de uma frase dependendo da combinação das palavras;
- **Pragmática** — frases com vários significados dependendo do contexto.

Segundo *Chantree* [Cha04] a ambiguidade existe e não deve ser sempre eliminada. Este sugere uma ferramenta de geração de linguagem natural (baseada no trabalho de *Shemtov* [She97]) que, com o *feedback* do utilizador, utiliza um sistema de aprendizagem que permite ao sistema decidir se uma determinada ambiguidade deve ser mantida ou não.

2.6.5.2 Web Semântica

Apesar de existir alguma investigação sobre Web Semântica [NJJY09], ferramentas [ULM08] e avaliação de aplicações [Man07, ULL+07], existe muito pouco sobre o domínio dos *emails*. Contrariando esta tendência, em 2003, *Taghva et al* [TBC+03] propuseram a criação de uma ontologia para a classificação de *emails*. O objectivo deste trabalho era dotar os *emails* de maior capacidade de análise do domínio e permitir uma partilha do domínio da informação entre diferentes pessoas e *software*. A ontologia criada pretendia abranger algumas característas do *email* para além daquelas que estão disponíveis no cabeçalho de cada mensagem (agente, destinatário, rememente, etc). Para caracterizar o *email*, foi criada uma nova classe na ontologia sobre as características do *email* e respectivas sub-características sobre o corpo, assunto e anexo da mensagem. Para classificar um *email* perante estas classes foi utilizado um método probabilístico, no caso, um classificador *Bayesiano*.

Ainda no mesmo ano, *Alon et al* [AHLM03] introduziram o termo *Semantic Email* e, em 2004, *McDowell et al* [MEH04] apresentaram a teoria e aplicações deste paradigma. *McDowell et al* afirmam que a visão da *Semantic Web* pode ser transitada para o domínio do *email* e descreveram quais as principais aplicações que o *Semantic Email* poderia ter. Os autores identificaram três actividades principais efectuadas no *email*, que são repetitivas e penosas para o utilizador, e que poderiam ser automatizadas com as seguintes técnicas.

- **Update** — uso do *email* para adicionar dados a alguma fonte (p.e. adicionar um "post" num blog através do *email*);
- **Query** — uso dos *emails* como repositório de informação (p.e. pesquisa de informação sobre um contacto)
- **Process** — uso do *Semantic Email* para automatizar processos que são, actualmente, feitos à mão.

Neste artigo é focada a terceira aplicação e exposto um modelo teórico de como é possivel inferir que estas automatizações podem ser feitas num tempo polinomial.

2.6.6 *Ranking*

O principal factor que demonstra ao utilizador a qualidade de uma pesquisa é a relevância dos seus resultados. Empresas como a *Google* ou *Yahoo!* mantêm o seu sistema de *ranking* bem guardado [Cha08].

Nesta secção, será abordada, para além do *ranking Web*, o *ranking* em *emails* e algumas técnicas de optimização de resultados.

2.6.6.1 *Web*

Um dos principais factores que leva os utilizadores a optarem por um determinado motor de pesquisa Web é a qualidade dos seus resultados. Apesar de escassas, existem algumas informações sobre diversos algoritmos utilizadados nesses motores de pesquisa. De seguida alguns dos algoritmos mais importantes no *web ranking*:

- PageRank — algoritmo de Ranking utilizado pela Google e patenteado pela Universidade de Stanford.
- HITS — O *Hyperlink-Induced Topic Search*, desenvolvido por Jon Kleinberg, é um algoritmo baseado na análise da rede que classifica páginas Web.
- TrushRank — Desenvolvido pela Universidade de Stanford em colaboração com a Yahoo! este algoritmo semi-automático permite a distinção entre boas páginas Web e *SPAM*.
- RankNet — "Ranking using Neural Net" é uma rede neuronal artificial que utiliza o *feedback* dos utilizadores para melhorar o ranking efectuado. Uma das aplicações que utiliza este algoritmo é o motor de pesquisa Bing [2].

Na secção 2.6.6.3 irá ser abordada uma outra série de artigos referentes ao *ranking* na web.

2.6.6.2 E-mail

Uma das principais diferenças entre a pesquisa na *Web* e no *email* é o alvo da pesquisa. É usual, na *Web*, pesquisarmos por termos pelos quais queremos ter mais informação. É raro, por outro lado, fazermos o mesmo no *email*. *MacDonald et al* [MO06] referem a pesquisa em *email* como sendo *known-item tasks*, ou seja, quando um utilizador efectua uma pesquisa fá-lo tendo em mente um determinado documento que pretende encontrar.

[2] **Bing** - www.bing.com

Tentando encontrar quais os factores que influenciam mais o ranking de um *email*, *MacDonald et al* [MO06] aplicaram um algoritmo de *Data-Mining* baseado num modelo estatístico denominado de *PL2F* num conjunto de 174.311 *emails*. Através destes resultados foi possível inferir o peso dos atributos num *email* (corpo, remetente, titulo, etc) e uma combinação de pares destes atributos, tendo o par *Atext* (texto dos *hyperlinks*) e *Unquoted* (parte citada do *email*) obtido o maior peso, ou seja, é o par mais relevante para o *ranking* de uma pesquisa em *emails*. Esta mesma abordagem é seguida por *Cohen et al* [CDZ07] mas para o caso de pesquisa em *Desktop*.

2.6.6.3 *Re-Finding*

Os sistemas de pesquisa são utilizados tanto para procurar nova informação como para reencontrar informação acedida anteriormente. De facto, estima-se que 30% das pesquisas efectuadas por um utilizador já foram feitas previamente mais do que uma vez [TAJP07] e que 80% das visitas a páginas *Web* são de utilizadores que já tinham visitado essa mesma página [CGJ+03]. Este comportamento despertou recentemente um maior interesse na área [OWHM07, TAAK04, All03, Voo01] incluindo por parte de algumas das maiores empresas do ramo como a Google [YC02, YC04] e a Yahoo [TAJP07].

Existem duas abordagem possíveis para a implementação de um sistema de *Re-Finding*:

- **Análise da Rede** — através do estudo do comportamento da rede podemos saber quais os resultados mais escolhidos pela comunidade, ou seja, que foram mais relevantes para os mesmos.
- **Análise do Utilizador** — todo o comportamento do utilizador é utilizado numa pesquisa futura que seja igual a uma já efectuada.

Algumas fontes [Cha08, YC02, YC04] sugerem que o sistema da *Google* utiliza os dados disponíveis da rede para refinar pesquisas repetidas. No caso do *PageRank*, quando um cálculo de relevância para dois documentos é igual, este dá vantagem ao mais popular. Contudo, a identificação de uma pesquisa repetida pode ir muito mais além da comparação crua das palavras chaves utilizadas. No estudo de *Teevan et al* [TAJP07] apenas 33% dos utilizadores (de um universo de 13.060 utilizadores da Yahoo) que efectuaram pesquisas repetidas utilizaram a mesma *string* em ambas as pesquisas. *Teevan et al* identificam mais três outras possibilidades de uso de *re-finding*:

- ***Overlapping-click*** — interrogações que, apesar de serem diferentes, têm uma lista de resultados semelhante.
- ***Equal-click*** — o utilizador escolhe o mesmo resultado para pesquisas diferentes. Apesar de formalmente as perguntas serem diferentes, para este utilizador, o resultado mais relevante é o mesmo em ambos.

- ***Navigational*** — pesquisas em que o utilizador escolhe um e só um resultado sempre que efectua essa pesquisa. Este comportamento (denominado de *orientered*) torna a reordenação complexa e é explicado em maior detalhe na secção 2.7.

Contudo, existem alguns problemas inerentes ao facto de se utilizar o resultado das massas para a optimização dos resultados. Facilmente os resultados se podem tornar viciados e desprovidos de informação nova o que pode ser considerado um problema para os motores de busca [Tee08].

De forma a contornar este problema, *Agichtein et al* e *Teevan* propõem uma abordagem mais pessoal ao sistema de refinamento das pesquisas. *Agichtein et al* efectou no seu trabalho [ABD06] alguns testes sobre o *BM25F* que foi o algoritmo mais eficaz na TREC [3] 2004. Usaram, ainda, uma outra série de algoritmos baseados no RankNet [4] (utilizado também no motor de pesquisas Bing [Neo09]) que implementam diferentes tipos de *Re-Ranking* baseados em diferentes combinações de informação disponível sobre o comportamento do utilizador:

- **Clickthough** — informação sobre os cliques que o utilizador faz (p.e. quantidade, frequência, ordem dos clicks, entre outros);
- **Browsing** — dados sobre a navegação do utilizador (p.e. tempo de visita, tempo num domínio, número de "saltos" até encontrar a solução, entre outros);
- **Query-Text** — informações relativas ao texto pesquisado (p.e. tamanho, palavras partilhadas entre a pesquisa e o título, resumo e *url* do resultado, entre outros).

Para avaliar todas as diferentes implementações, foram criadas diferentes métricas, baseadas numa qualificação dos utilizadores (seis níveis de "Perfeito" a "Mau"), tendo obtido resultados que evidenciaram melhorias até 31%.

Por outro lado, outros autores abordam, nesta questão, os factores humanos envolvidos (p.e. memória visual e cognitiva). *Obendorf et al* e *Teevan* defendem que qualquer alteração à primeira lista de resultados disponibilizada ao utilizador pode ser prejudicial [Tee08, OWHM07]. Ambos os autores defendem que deve existir consistência nos resultados devolvidos ao utilizador. Apesar disso, segundo *Teevan*, o *re-ranking* pode ser efectuado de forma a mostrar nova informação ao utilizador. Para isso, devem ser mantidos inalterados dois resultados: o primeiro resultado e o mais relevante. Estes dois resultados são os que mais influenciam a memória do utilizador e por isso não devem ser alterados de forma a manter uma consistência aparente ao utilizador. Apesar deste estudo se basear no domínio das pesquisas *Web*, *Teevan et al* suporta a ideia que o uso desta meta-informação pode ser facilmente utilizada para outros domínios, como é o caso do *email* [TAAK04].

[3] **TREC** - Text REtrieval Conference - Conferência sobre recuperação de informação
[4] **RankNet** - Rede Neuronal desenvolvida pela Microsoft

2.7 Categorização de Utilizadores

Um dos estudos que tentou categorizar os tipos de utilizadores que necessitam de pesquisar algo foi o estudo de *Jaime Teevan et al* [TAAK04]. Neste estudo, foram identificados dois tipos de utilização: *teleporting* e *orienterring*.

Por *teleporting* entende-se que são utilizadores que, quando necessitam de encontrar uma determinada informação, tentam obter directamente essa informação, utilizando, por exemplo, o ”Sinto-me com sorte“ nas pesquisas do *Google* ou indo directamente ao sítio desejado sem utilizar pesquisa.

Por outro lado, *orienterring* representa os utilizadores que fazem as suas pesquisas de forma sustentada, isto é, em vez de tentarem obter a informação directamente recorrem à contextualização e métodos passados de obtenção dessa informação. Um caso deste tipo de utilização é quando, por exemplo, para sabermos o número de telefone do gabinete do Prof. Jaime Cardoso, em vez de utilizarmos uma pesquisa global, contextualizamos o problema e seguimos uma ordem lógica na obtenção da informação (visitar site da FEUP, Departamento de Engenharia Electrotécnica e de Computadores, Professores, etc). Outro exemplo desta interacção é, por exemplo, quando um utilizador efectua uma pesquisa no *Google* e, passado um determinado espaço de tempo, refaz a mesma pergunta ao sistema, o utilizador tende a lembrar todo o caminho que fez até à informação que desejava.

Em termos gerais, os utilizadores comuns optam por uma pesquisa mais orientada. Isto deve-se a diversos factores como a sensação de localização, que se pode perder com uma pesquisa directa ou a memória cognitiva que os utilizadores têm de onde está a informação. Mesmo quando um utilizador utiliza um motor de pesquisa, este serve geralmente como base para uma pesquisa orientada. A pesquisa orientada revela-se ainda mais fiável que a pesquisa directa que falha em grande parte dos casos [TAAK04].

2.8 Clientes

2.8.1 Proprietários

Actualmente existem dezenas de clientes *email* disponíveis. Existem dois tipos de clientes bastantes distintos: clientes *Web* (*webmails*) ou clientes locais. Visto a quantidade de clientes disponíveis ser demasiado elevada, não foi viável fazer uma avaliação de todos os clientes. Foram, por isso, escolhidas os três *webmails* mais utilizados em 2008 segundo *Cselle* [Cse08] (ver figura 2.3).

O teste a estes três clientes foi puramente ao nível de funcionalidades disponibilizadas. A implementação em termos de tecnologias (*software* e *hardware*) utilizadas é geralmente coberta de grande sigilo. Apesar de todos possuirem pesquisa por Destinatário, Remetente e Assunto da mensagem, várias outras propriedades diferem.

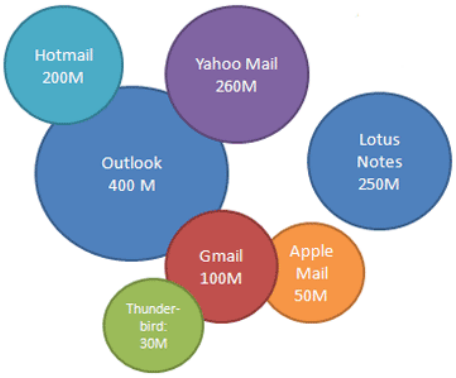

Figura 2.3: Distribuição de utilizadores de *email*

Nestes testes, não foram utilizados programas como *Outlook* ou *Thunderbird* devido à clara vantagem das aplicações de *software* a correr no cliente em relação às suas equivalentes que correm em servidor (p.e. capacidade de processamento ou largura de banda).

Interessante é o facto de o líder de mercado em pesquisas *Web*, a *Google*, possuir um sistema de pesquisa muito abaixo do seu rival *Yahoo!*. O sistema da *Yahoo!* é bastante bom para o uso de utilizadores comuns pois efectua bastantes operações de forma transparente. É de longe o sistema que fornece mais funcionalidades e opções aos seus utilizadores, ganhando assim aos seus mais directos adversários.

2.8.2 Open-Source

Em anterirores projectos internos da Portugalmail [San06, Cos09, Sil06] já foram realizadas enúmeras comparações entre clientes de *webmail* Da enorme lista de opções destacam-se aplicações como Zimbra, Horde, RoundCube, Scalix ou SquirrelMail.

Através da análise dos trabalhos anteriores em conjunto com algumas actualizações de informação desactualizada resultou a seguinte tabela:

Tabela 2.1: Comparação de plataformas *webmail* proprietárias

	Folders	**Rest. Temporais**	**Conteúdo de Anexos**	**Wildcards**
Gmail	Sim	Sim	Não	Sim
Hotmail	Não	Não	Não	Não
Yahoo	Sim	Sim	Sim	Sim

Tabela 2.2: Comparação de plataformas *webmail Open-Source* com funcionalidades de pesquisa

	Zimbra	**RoundCube**	**Horde**
Pesquisa por atributos	Vários	Vários	From / Subject / To
Pesquisa por conteudo de anexos	Versão Paga	Não	Não
Pesquisa por contactos	Sim	Não	Sim (Turba)
Guardar pesquisas	Sim	Não	Não

Capítulo 3

Avaliação Tecnológica

Neste capítulo são revistas as tecnologias mais relevantes para este projecto. São ainda explicadas em detalhe as diferentes escolhas tecnológicas que foram tomadas para a implementação deste projecto.

3.1 Tecnologias Utilizadas

Nesta secção irão ser descritas as principais tecnologias utilizadas na plataforma actual que irão ser parte integrante deste projecto. Entre estas tecnologias estão os serviços responsáveis por recepção, envio e armazenamento de *emails*, sistema de autenticação e cliente *webmail* utilizado.

3.1.1 Horde

Esta *framework* é escrita em PHP e fornece diversos serviços que possibilitam tornar o desenvolvimento Web mais simples disponibilizando, por exemplo, sistemas de autenticação, detecção de browsers, compressão de páginas ou validações. A arquitectura específica do Horde é baseada em aplicações. Uma das mais usadas nesta plataforma é o cliente de *webmail* IMP e sua versão AJAX denominada DIMP [1] (Dinamic IMP) que fornece acesso a contas de *email* através de IMAP ou POP3. A facilidade de costumização das aplicações existentes e a possibilidade de fácil desenvolvimento de novas aplicações faz com que a plataforma de *email* actual da Portugalmail use um vasto número de aplicações desta *framework*, no caso, DIMP como cliente de *email*, Sybil [Cos09] como gestor de contactos, Ingo [2] para gerir os filtros de *email*, Passwd [3] como

[1] **DIMP** - www.horde.org/dimp
[2] **Ingo** - www.horde.org/ingo
[3] **Passwd** - www.horde.org/passwd

gestor de passwords e o Kronolith [4] como calendário (ainda em desenvolvimento para a nova plataforma).

3.1.2 Dovecot

Servidor IMAP e POP3, o Dovecot é uma aplicação *Open-Source* para sistemas Linux (ou UNIX-like) desenvolvido por Timo Sirainen. Esta aplicação caracteriza-se por ser rápida, fácil de configurar, leve e segura (existindo até uma recompensa de 1000 Euros para quem descobrir uma falha de segurança na aplicação [5]). Assim como a Horde, esta aplicação é bastante modular e permite adicionar ou alterar facilmente funcionalidades recorrendo a *plugins*. Entre os *plugins* existentes os *plugins* de gestão de quotas de *email*, anti-spam e procura por texto integral são apenas alguns dos mais populares de toda uma panóplia de *plugins* que auxiliam a manutenção de um serviço deste tipo.

3.1.3 Postfix

Postfix [6] é uma aplicação *Open-Source* que permite o encaminhamento e entrega de *emails*. A rapidez, facilidade de manutenção e a segurança são apenas algumas das carateristicas mais relevantes desta aplicação. Na plataforma actual é esta aplicação a responsável pelo envio (através de SMTP) das mensagens dos utilizadores da Portugalmail.

3.1.4 MIME

Multipurpose Internet Mail Extensions ou o seu acrónimo MIME [7] é uma norma da internet para o formato das mensagens de correio electrónico. Grande parte dos *emails* trocados na Web são transmitidos por SMTP no formato MIME. Neste formato são definidas todas as propriedades da mensagem como o tipo de conteúdo (*content-type*), destinatários, remetentes, datas, entre outros.

3.1.5 LDAP

LDAP é o acrónimo de *Lightweight Directory Access Protocol*, é um protocolo para actualizar e pesquisar directórios que usam TCP/IP. Um directório LDAP consiste num conjunto de objectos com similares atributos organizados de uma forma lógica e hierárquica. Este sistema possibilita, em comparação com base de dados relacionais, uma melhor eficiência aos pedidos efectuados mas, por outro lado, um pior desempenho nas operações de inserção e alteração. No sistema actual da Portugalmail, todos as aplicações fazem uso desta tecnologia para se autenticarem perante o sistema.

[4] **Kronolith** - www.horde.org/kronolith
[5] www.dovecot.org/security.html
[6] **Postfix** - www.postfix.org
[7] **MIME** - http://tools.ietf.org/html/rfc2045

3.1.6 Maildir

Um dos formatos mais usados para guardar *emails* em disco. A vantagem de usar este sistema está no facto de este não requerer um bloqueio dos ficheiros para manter a integridade das mensagens. Todas as mensagens são guardadas com um identificador único num directório (geralmente denominado Maildir) com três sub-directorias: tmp, new e cur.

3.2 Biblioteca de Recuperação de Informação

Geralmente denominada de indexador ou motor de pesquisa, uma ferramenta de recuperação de informação é a parte mais importante num sistema de pesquisa. Assim sendo, tornou-se imprescindível uma análise rigorosa a todas as opções disponíveis de forma a poder sustentar a escolha da ferramenta com bases sólidas. Os requisitos que condicionaram esta análise serão abordados mais tarde no capítulo 6.3.1.

3.2.1 Comparação das Soluções Existentes

Apesar de existirem inúmeros motores de pesquisa disponíveis (ver listagem em Apêndice A.1), poucos são aqueles que respeitam os requisitos mínimos desta aplicação sendo eles a existência de API de indexação e pesquisa, boa documentação e comunidade, ser *Open-Source* e possuir licenças que sejam compatíveis com o modelo de negócio da Portugalmail (p.e. *GPL* [8]).

Numa primeira abordagem encontrar comparações (e/ou *benchmarks*), que permitissem tirar conclusões sobre qual a ferramenta mais indicada para o problema em questão, foi bastante complicado dado a escassez destas. Infelizmente, grande parte dos *benchmarks* comparativos entre ferramentas são bastante dúbios pois são sempre efectuados por uma das partes envolvida na comparação. Existem, no entanto, dois documentos que comparam diferentes bibliotecas. Apesar de um deles, escrito por *Hassler* [Has05], consistir meramente na comparação de funcionalidades, o outro, por outro lado, redigido por *Haye* [Hay], para além dessa análise, efectua testes de desenpenho nas ferramentas a nível de indexação e pesquisa. Na tabela 3.1, podemos ver alguma da informação genérica comparada e na tabela 3.2 algumas das funcionalidades mais importantes para este projecto.

O documento de *Haye* pretendia analisar qual o melhor motor de indexação e pesquisa a ser usado no desenvolvimento da ferramenta *eXtensible Text Framework* (XTF) [9] (entretando já terminada). De um conjunto de nove bibliotecas, três satisfizeram os requisitos impostos por *Haye* (que são bastante semelhantes aos deste projecto) sendo elas: *Lucene*,

[8] **GPL** - www.gnu.org/licenses/gpl.html
[9] **XTF** - www.cdlib.org/inside/projects/xtf

Tabela 3.1: Comparação das características genéricas entre diferentes bibliotecas RI

	Xapian	**Lucene**	**Swich-e**
Linguagem	C++	Java	C
Licença	GPL	Apache License 2.0	GPL
Formato de Documentos	TXT, PDF, HTML, PHP, PostScript	TXT, HTML	TXT, HTML, XML
API	Sim	Sim	Apenas para pesquisa
Documentação	Excelente	Adequada	Fraca (não disponíivel para indexação)
Comunidade	Muito Grande	Grande	Pequena

Xapian e *Zebra*. Nestas três bibliotecas, foram testadas as velocidades de indexação, pesquisa e tamanho ocupado pelo índice, tendo os resultados do *Xapian* destacando-se pela negativa.

Destes, os melhores resultados foram obtidos por parte de Lucene (velocidade das interrogações e *ranking* por proximidade) e de Zebra (velocidade de indexação e tamanho ocupado pelo índice).

3.2.2 Benchmark

Apesar dos resultados dos testes de *Haye* terem dado clara vantagem ao *Lucene* e *Zebra*, o facto dos testes terem sido efectuados em 2004 e entretanto enúmeras novas versões das bibliotecas terem sido disponibilizadas, tira alguma validade para os testes serem utilizados como referência actualmente. Assim sendo, decidiu-se efectuar um novo *benchmark* com as versões mais recentes das bibliotecas *Lucene* e *Xapian*. Apesar de a biblioteca *Zebra* ter obtido excelentes resultados nos testes de *Haye*, a ferramenta carece de qualquer tipo de *API*, pelo que não corresponde aos requisitos mínimos definidos para a biblioteca de recuperação de informação a utilizar neste projecto e, por conseguinte, irá ser ignorada nos seguintes testes de *benchmark*.

3.2.2.1 Condições de teste

Devido às limitações de tempo disponível, não foi possível efectuar os testes de *benchmark* no sistema onde este projecto se irá encontrar em funcionamento aquando da sua conclusão. Houve, no entanto, a tentativa de conseguir um nível de condições igual para todos os testes de forma a que, apesar de não existir informação quanto ao desempenho óptimo das ferramentas, o resultado fosse coerente e confiável. Nesta primeira fase de testes, foi usado um conjunto de dados que é utilizado nos testes internos da empresa.

Tabela 3.2: Comparação das funcionalidades das diferentes bibliotecas RI

	Xapian	**Lucene**	**Swich-e**
Wildcards (p.e. *portuga*, fe?p*)	Sim	Sim	Sim (no fim da palavra)
Operadores boleanos (p.e. *"and", "or", "not", "+", -"*)	Sim	Sim	Sim
Operadores de intervalos (p.e. $1 < x < 42$)	Não	Sim	Não
Steaming (pesquisar por *trabalho* devolve *trabalhador*, *trabalhos*, etc.)	Sim	Sim	Sim
Expansão de Interrogações (p.e. pesquisar portugal e o sistema sugerir *Porto* e *Lisboa* como refinamento da interrogação anterior)	Sim	Sim	Não
Resultados parciais (p.e. devolver apenas 10 resultados dos 20 existentes)	Sim	Sim	Sim
Interrogações por frases (p.e. "Faculdade de Engenharia" - todos os termos têm de aparecer seguidos)	Sim	Sim	Sim

Apesar de não ser muito rico em conteúdo (em comparação com outros conjuntos de dados), é um conjunto de dados próximo do que existe na realidade. Este é constituído por 8148 *emails* e prefaz um tamanho de 23.5 MB (sendo 99,8% dos *emails* menores que 1mb). O computador onde foram feitos todos os testes possui dois núcleos de 3.00 GHz cada e 4GB de memória.

3.2.2.2 Resultados

Para cada biblioteca foram efectuadas cinco medições dos tempos de indexação (ver anexo A.2.1). De forma a tentar eliminar possíveis resultados atípicos, optou-se por eliminar, desses cinco resultados, o pior e o melhor e fazer uma média aritemética com os restantes três valores. Os resultados obtidos estão listados na tabela 3.3.

Tabela 3.3: Resultados dos testes de indexação

	Xapian	**Lucene**	**Lucene (com optimização)**
real	32.209s	2.637s	2.616s
user	30.955s	3.092s	3.104s
sys	0.939s	0.239s	0.291s
Tamanho do índice	154Mb (655%)	5.2Mb (22%)	5.2Mb (22%)

Com base nestes resultados podemos verificar que *Lucene* é, de longe, o mais eficiente dos dois testados, conseguindo muito melhores tempos e muito menos espaço ocupado. Assim como já relatado em 2004 [Hay], o *Xapian* teve problemas com amostras com um elevado número de documentos (ainda que pequenos em tamanho). Estes valores levam a

crer que não existiram grandes evoluções no *Xapian* nos últimos 5 anos (ao contrário do anunciado pelos seus criadores).

Um facto curioso foi os resultados dos testes de *Lucene*, sem optimização, terem dado um tempo real médio superior aos testes com optimização. Isto deve-se, provalmente, a dois factores: pequeno tamanho do índice criado (pouca optimização seria necessária) e ao reduzido tempo das indexações (qualquer factor externo pode influenciar um pouco os tempos obtidos). Ainda assim, notou-se uma ligeira subida nos tempos de processamento do CPU (em user e sys).

Devido à grande ineficiência por parte do *Xapian*, decidiu-se não continuar a fazer testes de desempenho e eficácia da pesquisa. Em reunião informal com os responsáveis na empresa, concluiu-se que um aumento de 655% no tamanho dos índices seria incomportável para empresa, tornando-se o *Lucene* a única opção viável.

3.2.3 Benchmark dos Ports de Lucene

A implementação do motor do *Lucene* provou ser de excelente qualidade. Este facto levou a que a versão original, desenvolvida em Java, fosse implementada em diversas outras linguagens, respeitando a arquitectura original e tirando partido das potencialidades específicas de cada linguagem. Entre estas implementações encontram-se linguagens como C/C++, Perl, Python ou Ruby. Posto isto, tornou-se necessário efectuar uma pesquisa sobre características de cada uma destas bibliotecas de forma a decidir quais tinham potencial para ser testadas. Da listagem completa (ver anexo A.3) sobressaiu apenas a implementação em Ruby (denominada Ferret [Bal08]) pelo tamanho da sua comunidade, documentação e alegado bom desempenho [10].

3.2.3.1 Condições de teste

As condições destes testes serão em tudo semelhantes às condições dos testes anteriores 3.2.2.1. A única diferença será a inclusão de dois novos conjuntos de dados para que o teste possa ser mais fidedigno e completo. Os conjuntos são os seguintes:

- **Reuteurs** - Um conjunto de dados de 28MB de texto puro (sem *tags* ou outro qualquer informação não indexável) usado na área da categorização de textos.
- **Conjunto Real** - Este conjunto consiste numa amostra de uma conta de email real. É bem mais rica em conteúdo do que a amostra usada nos testes da empresa (textos mais longos, anexos, entre outros) e contém 2854 *emails* (438.2 MB).

3.2.3.2 Resultados

O resultado dos testes de indexação para cada uma das amostras foi o seguinte:

[10] http://ferret.davebalmain.com/trac/wiki/MyFirstBenchmark

Tabela 3.4: Resultados com a amostra de teste da empresa

	Ferret	**Lucene**
Tempo Real	8.134s	2.637s
Tamanho do índice	14.1MB (60%)	5.2MB (22.1%)

Tabela 3.5: Resultados com conjunto de dado da *Reuteurs*

	Ferret	**Lucene**
Tempo Real	1.798s	3.539s
Tamanho do índice	1.8MB (6.4%)	633KB (2.2%)

Tabela 3.6: Resultados com a amostra real

	Ferret	**Lucene**
Tempo Real	178.288s	110.906
Tamanho do índice	284MB (64.8%)	74.9MB (17.1%)

Os resultados obtidos revelaram alguns dados interessantes. Para todos os conjuntos de dados o *Ferret* gastou consideravelmente mais espaço que o *Lucene*. Isto não deveria suceder pois ambos respeitam a mesma arquitectura (supostamente) e consequentemente o tamanho e conteúdo dos índices deveria ser idêntico. Por outro lado, os resultados em termos de tempos de indexação revelam que, à semelhança do *Xapian*, o *Ferret* não consegue superar o *Lucene* em conjuntos de dados com elevado número de ficheiros. Contudo, no conjunto de dados da *Reuteurs*, o *Ferret* foi quase duas vezes mais rápido que *Lucene*.

Estas diferenças de tempos, e principalmente a diferença dos tamanhos dos índices, levou a suspeitar da integridade do índices criados. Para remover tal suspeita foram realizados mais alguns testes, desta vez sobre a qualidade dos resultados obtidos numa pesquisa sobre os índices criados. Os tempos que estas pesquisas demoram a fazer para índices deste tamanho são bastante pequenas (na ordem dos milisegundos) e, por isso, foram ignoradas para os testes seguintes. O conjunto de dados utilizado foi a amostra real pois os seus resultados eram os mais fáceis de validar.

Tabela 3.7: Resultados da pesquisa ao índice do conjunto dos *emails* reais

Query	**Lucene**	**Ferret**
Pedro *AND* Rui *AND* Paulo	76 resultados	76 resultados
lbd *OR* rcom *OR* laso	11 resultados	6 resultados
“Wrap up more than music”	1 resultado	1 resultado
Rui *AND* (facebook *OR* hi5)	97 resultados	89 resultados
password *AND* account	35 resultados	25 resultados
grupo *AND* (trabalho *OR* borga) *AND* (aulas *OR* cerveja)	5 resultados	5 resultados
eu *OR* tu *OR* ele *OR* nos *OR* eles	936 resultados	905 resultados
eu *AND* ele *AND* tu	89 resultados	79 resultados

Apesar de em ambos os testes ser usado o mesmo analisador por defeito (*StandardAnalyser*) e de não existir nenhuma lista de *Stop Words* [11], os resultados diferiram. Em algumas das interrogações testadas, os resultados devolvidos pelo *Ferret* eram substancialmente menos que os apresentados pelo *Lucene*. Para eliminar a hipótese de ser o *Lucene* que estava a ter resultados a mais, foram feitos testes para todas as interrogações mas, desta vez, usando uma pesquisa recursiva recorrendo à utilização de expressões regulares, tendo sido confirmada a validade de todos os resultados de *Lucene*.

Estes resultados vêm mostrar, possivelmente, uma das razões de o *Ferret* se encontrar ainda na versão 0.11.6, ou seja, muito longe de ser uma versão final.

Em todos os testes estudados e efectuados o Lucene superou a sua concorrência por grande margem. Lucene é uma API madura e com larga comunidade e documentação o que facilita o desenvolvimento num curto espaços de tempo (como é o caso). Apesar da supremacia, o desempenho de Lucene por ser facilmente melhorado através de diversas configurações, por exemplo, da Java Virtual Machine ou de outros pormenores para alto desempenho em Java [Dua08].

3.3 Integração da Ferramenta RI

Após os testes efectuados terem mostrado Lucene como a melhor biblioteca dentro do lote disponível, foi necessário estudar qual a melhor forma de integrar esta ferramenta com a plataforma actual.

O servidor IMAP actual, o Dovecot, é bastante flexível o que permite alterá-lo facilmente de forma a servir os propósitos desejados. Existem duas possibilidades de integração disponiveis entre Dovecot e Lucene já implementadas. Uma delas baseia-se na utilização de CLucene que consiste numa implementação de Lucene em C anteriormente descartada 3.2.3 devido à sua pequena comunidade e ligeira desactualização. A restante opção consiste num servidor de pesquisa, denominado Apache Solr, baseado nas bibliotecas de pesquisa Lucene. Segundo o criador do Dovecot a utilização de CLucene não é aconselhável.

> "The current Lucene implementation is a bit ugly and simultaneous access may give some errors. Use Solr instead if possible." (*Timo Sirainen*) [12]

Solr é um servidor (Java servlet) de pesquisa empresarial *Open-Source* baseado na biblioteca Lucene. Esta tem a particularidade de ser altamente configurável sem necessitar de muito esforço. Todas as configurações são definidas através de ficheiros XML que podem ser alterados para mudar a estrutura do índice, analizadores, *tokenizers*, filtros, entre

[11] Palavras que ajudam à construção das frases mas não têm qualquer valor para análises de PLN. Em português palavras como "a", "o", "da", "de", "que", etc.

[12] http://wiki.dovecot.org/Plugins/FTS/Lucene

outras funcionalidades. Apesar de já ser fornecido um largo conjunto de ferramentas de tratamento de dados é possivel re-implementar todas essas classes (Analyzer, FieldTypes, Tokenizer, etc) da forma que se deseje.

De todas as funcionalidades existentes na aplicação Solr as mais relevantes podem-se resumir na seguinte lista:

Faceted navigation Permite que, após uma pesquisa efectuada pelo utilizador, o resultado devolva não só o resultado mas também uma série de sub-pesquisas que permitam aprofundar a pesquisa anterior (p.e. o utilizador pesquisa na amazon por livros e o sistema sugere novas pesquisas, mais aprofundadas, como "romance", "terror", etc).

Highlight Consiste na pré-selecção do texto que é retornado no resultado de uma pesquisa. Por vezes, não é necessário retornar ao utilizador o texto completo mas apenas a parte onde foi encontrada a(s) palavra(s) pesquisada(s).

Respostas customizadas Esta funcionalidade possibilita uma fácil integração com interfaces Web pois existe a possibilidade de customizar o formato das respostas recebidas por HTTP. Para este projecto serão interessantes os formatos das respostas em XML (para o Dovecot) e PHP (para o Horde).

Interface de administração O Solr possui uma interface de adminstração em HTML o que permite que a manutenção do serviço seja facilmente efectuada através da Web.

Replicação Apesar do índice criado não conter informação única (esta encontra-se replicada em Maildir) o custo da indexação em termos de recursos é considerável. Assim sendo, a replicação pode ser um funcionalidade interessante de futuro.

Extensibilidade Sendo uma aplicação modular, esta suporta uma fácil expansibilidade através do uso *plugins*.

Pesquisa distribuída Em casos de grandes conjuntos de dados (como irá ser o caso desta plataforma) é possível efectuar pesquisas distribuídas de forma a acelerar o processo de pesquisa.

Cache Suporte para uso de *Cache* é outras das funcionalidades de optimização do desempenho.

Em termos de integração com o sistema actual Solr é na verdade a única opção válida. A facilidade de integração com Dovecot é um factor determinante para a sua escolha como sendo a ferramenta de indexação e pesquisa a ser usada neste projecto.

3.4 Extracção dos Formatos

A indexação do conteúdo de anexos é um dos objectivos mais importantes (senão o mais importante) deste trabalho. Para ser possível a indexação do texto contido em ficheiros anexados a um *email* é necessário, de alguma forma, extrair esse texto dos ficheiros. Esta tarefa pode influenciar bastante a performance do sistema e por isso a aplicação responsável por esta tarefa foi escolhida com bastante cautela. As opções passavam por extrair o conteúdo do lado do Dovecot ou enviar o anexo completo para o Solr e extrair aí o conteúdo.

A segunda opção era, em termos de desenvolvimento, claramente a mais fácil. Existem inúmeras bibliotecas e ferramentas em Java que permitem a extracção de um vasto número de formatos. Uma das ferramentas mais usada em conjunto com Lucene/Solr é a ferramenta Tika [13] desenvolvida também pela *Apache Software Foundation*. Ainda assim, esta opção teria demasiada influência negativa no desempenho geral do sistema. O facto de termos de enviar por HTTP, para além dos dados normais de um *email*, os ficheiros de anexo a cada indexação pretendida traria uma carga adicional aos servidores que facilmente poderia ser incomportável. A solução adoptada passou então por extrair o texto dos ficheiros de anexo antes de enviar para o indexador. Alguns testes com alguns formatos de ficheiros (doc e pdf) mostraram que o texto contido nestes ficheiros, em média, corresponde a menos de 10% do tamanho original do ficheiro.

Tendo sido decido que a extracção do conteúdo dos anexos seria efectuada no Dovecot, era agora necessário decidir que método seria o mais adequado para o efeito. Após alguma pesquisa, foram encontradas duas hipóteses viáveis: uso de bibliotecas de extracção de conteúdo de ficheiros ou uso de aplicações de extração exteriores. Devido à escassez de bibliotecas de extracção do conteúdo de diversos formatos em C e a problemas relacionados com a arquitectura de formatos como o PDF (explicado em maior detalhe na secção 6), a decisão passou pela utilização de programas externos ao Dovecot.

[13] http://lucene.apache.org/tika/

Capítulo 4

Especificação

Neste capítulo será feita toda a especificação deste projecto. Será efectuada a descrição da arquitectura do sistema actual, realizado o levantamento de requisitos funcionais e não funcionais e apresentado o protótipo desenvolvido.

4.1 Arquitectura Actual

A arquitectura actual da plataforma *webmail* da Portugalmail consiste numa variação da conhecida arquitectura LAMP (Linux, Apache, MySQL e PHP) conhecida devido ao uso de software livre, desempenho e escalabilidade. Esta variação consiste na troca de base de dados utilizada (PostgresSQL em vez de MySQL) criando assim o acrónimo LAPP.

Neste projecto é de particular interesse a comunicação entre as camadas de PHP (Horde Framework) e os servidores de tratamento e armazenamento de dados (Dovecot, Postfix, LDAP e Maildir) exemplificado na figura 4.1.

A Horde Framework é a responsável pela ponte entre os utilizadores e o sistema de envio, recepção e armazenamento. Todas as operações relativas às caixas de mensagens são efectuadas pelo Dovecot como, por exemplo, a criação de pastas, apagar mensagens ou até o efectuar de uma pesquisa. A comunicação entre Dovecot e Horde Framework é feita na plataforma da Portugalmail recorrendo exclusivamente ao protocolo de comunicação IMAP. Isto limita a pesquisa actual em termos de funcionalidades pois o protocolo é restricto. A única operação sobre *emails* que não é tratada pelo Dovecot é o envio de *emails*. O envio está ao cargo do Postfix que encaminha os *emails* para fora do domínio Portugalmail através de SMTP. Tanto o Dovecot como o Postfix utilizam o formato de ficheiros Maildir para guardar os respectivos *emails* recebidos e enviados. Todas as autenticações

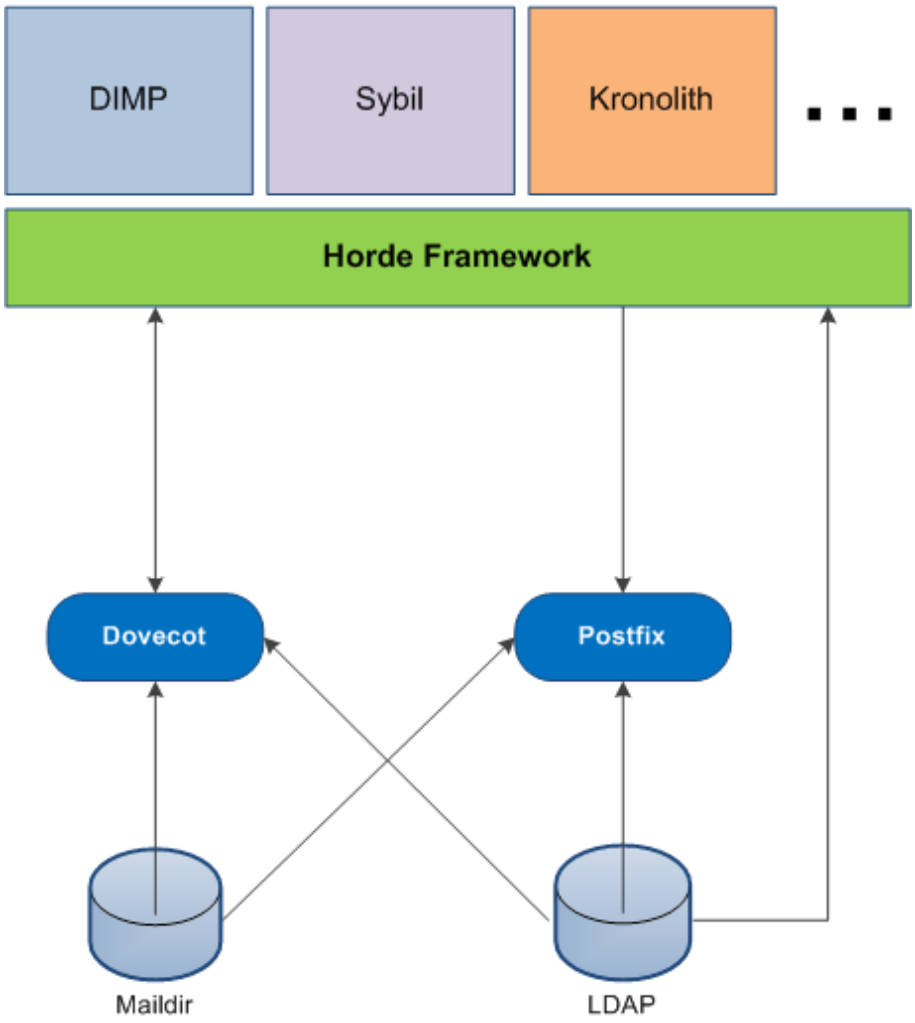

Figura 4.1: Arquitectura do Sistema Actual
aks

neste sistema são efectuadas através do protocolo LDAP que garante a validação das diferentes entidades.

4.2 Requisitos Funcionais

Os requisitos funcionais descrevem todos os processos que o sistema deverá efectuar quando terminado. O desenvolvimento deste projecto engloba múltiplas partes do sistema onde este irá ser inserido e inclui ainda a integração de todas elas. Cada uma destas partes irá utilizar as tecnologias anteriormente referidas (secção 3.1) desenvolvendo para cada uma delas as funcionalidades necessárias, usando os módulos já existentes ou criando novos, caso assim se justifique.

De forma a tornar a leitura e implementação das funcionalidades do sistema mais simples, os requisitos funcionais do projecto foram divididos em módulos. Estes basearam-se na arquitectura actual explicada na secção anterior e no futuro módulo de recuperação de informação responsável pela indexação e pesquisa do sistema, no caso, denominado de "Módulo Solr". Nas secções seguintes, serão definidos todos os requisitos levantados para cada módulo em específico.

4.2.1 Módulo Dovecot

Este é o módulo central do sistema pois é aqui que todas as mensagens são recebidas, tratadas e guardadas. O sistema central de *parse* dos diferentes blocos das mensagens irá ter de ser alterado. Actualmente, o Dovecot simplesmente ignora os *content-types* que não sejam texto. Toda a informação dos anexos (no formato BASE64) é descartada pelo *parser* actual não fazendo, por isso, qualquer uso da mesma. Será por isso necessário implementar o suporte para este tipo de conteúdos assim como para o *parse* de outros parâmetros MIME que não sejam ainda suportados.

Para além das alterações no núcleo do Dovecot, irá ser necessário implementar a extracção do conteúdo dos documentos em anexo, assim como a construção do pedido a enviar, suportando já as alterações que serão efectuadas no índice.

Os requisitos levantados para este módulo foram os seguintes:

Tabela 4.1: Lista de requisitos para o Módulo Dovecot

Requisito	**Prioridade**
Suporte do *Parser* da mensagem para os anexos das mensagens	Alta
Suporte do *Decoder* da mensagem para o *content-type* dos anexos da mensagem	Alta
Extracção do conteúdo de ficheiros no formato PDF	Alta
Extracção do conteúdo de ficheiros no formato Microsoft Word	Alta
Extracção do conteúdo de ficheiros no formato Microsoft PowerPoint	Alta
Extracção do conteúdo de ficheiros no formato Microsoft Excel	Média
Construção do novo pedido XML a enviar ao Solr (já com suporte de anexos)	Alta

4.2.2 Módulo Horde

É neste módulo que a forte componente de unificação deste projecto será implementada. Pretende-se que seja criada uma nova aplicação (a par do DIMP, Sybil e Kronolith) que se encarregue de fazer uma pesquisa global que incorpore todas as formas de informação da plataforma a ser desenvolvida: contactos, grupos, eventos, emails e anexos.

Para o utilizador, este será o único módulo que terá alguma visibilidade. Devido a isto, deve ser desenvolvida, para além de todo o sistema *backend*, uma interface que permita mostrar ao utilizador, de forma unificada, todas as pesquisas numa só de forma simples e rápida de utilizar.

Os requisitos levantados para este módulo foram os seguintes:

Tabela 4.2: Lista de requisitos para o Módulo Horde

Requisito	**Prioridade**
Criação de uma nova aplicação para o Horde que trate do *backend* da pesquisa	Alta
Criação de uma nova interface de pesquisa	Alta
Criação de uma API disponível para todas as aplicações Horde	Média
Suporte para pesquisa por *emails* (Solr/IMP)	Alta
Suporte para pesquisa por anexos (Solr)	Alta
Suporte para pesquisa por contactos (Sybil)	Média
Suporte para pesquisa por grupos (Sybil)	Média
Suporte para pesquisa por eventos (Kronolith)	Baixa
Suporte para pesquisa por frases	Média
Suporte para pesquisa avançada usando parâmetros como “+”, “-”, “not”, “or” ou “has attachment”	Baixa

4.2.3 Módulo Solr

O módulo Solr corresponde ao servidor de indexação e pesquisa do sistema. Será necessário criar uma estrutura para o índice utilizado de forma a que este suporte a indexação dos *emails* e respectivos anexos. Depois de decidida qual a estrutura a utilizar outras funcionalidades do Solr devem ser configuradas ou desenvolvidas.

Os requisitos levantados para este módulo foram os seguintes:

Tabela 4.3: Lista de requisitos para o Módulo Solr

Requisito	**Prioridade**
Criação de um índice que suporte múltiplos anexos	Alta
Suporte para a utilização de Hightlight	Alta
Suporte para a utilização de *Faceted Search*	Baixa
Implementação de analisadores de texto próprios (*Analizer*, *Tokenizer*, etc)	Baixa

4.3 Requisitos Não Funcionais

O número de requisitos não funcionais para um projecto desta dimensão é bastante elevado. Estes requisitos devem ser tidos em conta durante todo o desenvolvimento pois

espera-se que todos sejam cumpridos no final da implementação do projecto. Alguns dos mais importantes requisitos não funcionais serão descritos de seguida.

4.3.1 Eficiência

O sistema em desenvolvimento é um ponto crítico no desempenho global da plataforma. Espera-se, por isso, que todas as funcionalidades implementadas tenham em vista a utilização eficiente dos recursos do sistema, tais como memória, processamento e uso da largura de banda disponível. Apesar de a resposta do sistema para com o utilizador não necessitar de ser instantânea, espera-se que o tempo que uma pesquisa demora desde o *input* até à visualização dos resultados seja aceitável.

4.3.2 Multi-Língua

Sendo a nova plataforma de *webmail* um produto internacional, o sistema deve conseguir tratar diferentes *emails* escritos em diversas línguas. Em todos os módulos da aplicação deve existir suporte para um vasto número de línguas. Neste suporte inclui-se o tratamento do texto a indexar por parte da ferramenta de RI.

4.3.3 Modularidade

Todas as aplicações utilizadas no sistema actual respeitam o requisito da modularidade. Num sistema *Open-Source* como este, devido aos métodos de desenvolvimento em comunidade, é necessário que as aplicações desenvolvidas sejam o mais modulares possível de forma a minimizar os problemas de integração entre funcionalidades. As novas aplicações (ou módulos) criados devem, por isso, também respeitar este princípio. Assim, o código desenvolvido deverá ser escrito tendo em consideração que irão exisitr, quase garantidamente, futuros desenvolvimentos no mesmo.

4.3.4 Escalabilidade

Visto ser impossível determinar qual será o volume de informação envolvido na expansão da nova plataforma, é necessário que este seja capaz de lidar com um grande aumento no número de utilizadores sem grande esforço. Ainda assim é de realçar que o sistema actual da Portugalmail tem um tamanho aproximado de 15TB.

4.3.5 Segurança

Tendo o *email* os mais variados usos, a informação que neles está contida é por vezes de extrema confidencialidade. Todas as alterações aos módulos existentes ou criação de novos devem respeitar, quando possível, os sistemas de autenticação actuais, no caso, o

sistema LDAP. Mesmo que não seja possível utilizar o sistema LDAP actual, as funcionalidades desenvolvidas devem garantir que um utilizador do sistema acede, apenas e só, à informação que lhe pertence ou que está autorizado a ver.

4.3.6 Manutenção

Devido ao elevado número de utilizadores e de volume de dados tratados este sistema necessita de constante manutenção. Todas as opções tomadas durante o desenvolvimento desta aplicação devem ter este facto em consideração.

4.3.7 Usabilidade

Usabilidade é o atributo qualitativo que representa o quão uma interface é simples de usar. Esta análise envolve aspectos como a curva de aprendizagem (Quanto tempo leva um utilizador a completar tarefas básicas pela primeira vez?), eficiência (Depois de aprendida a interface, quanto tempo leva a completar tarefas?), memória (Os utilizadores conseguem facilmente lembrar de como utilizar a interface?), erros (O utilizador apercebe-se quando ocorre um erro?) e satifação (O utilizador sente satisfação a utilizar o sistema?).

4.4 Protótipo de Interface

Um dos objectivos deste trabalho é a criação de uma interface de pesquisa que sirva como demonstração da tecnologia de pesquisa utilizada e das suas capacidades de expansão futura. Para ajudar na concepção desta interface, foi desenvolvido um protótipo em papel, em colaboração com o *designer* da empresa e com o constante *feedback* das pessoas envolvidas no desenvolvimento da nova plataforma.

Este tipo de protótipos [Gom] tem a vantagem de conseguir estabelecer uma base sólida de conceitos e terminologias entre *developer*, *designer* e utilizador final. Permite ainda estabelecer todos os passos de navegação da aplicação, *layout* da página e respectivas funcionalidades. O facto de se usar papel permite efectuar alterações rapidamente a qualquer altura e eliminar o factor tecnologia em futuros testes de usabilidade.

Como desvantagens existe a dificuldade em copiar o comportamento de alguns elementos da interface (scrollbars, uso de cores, animações, etc), atraso no tempo disponível para o desenvolvimento (este protótipo levou três dias a ser construído) e o facto de não ser possível encontrar todos os problemas de usabilidade.

Visto a interface a construir ser apenas uma e ser uma aplicação inovadora onde existem muitos conceitos misturados (contactos, *email*, anexos e calendário), a utilização deste protótipo mostrou-se bastante útil a curto prazo, pois tornou a implementação desta interface bastante mais simples e rápida. O protótipo desenvolvido pode ser visto na figura 4.2.

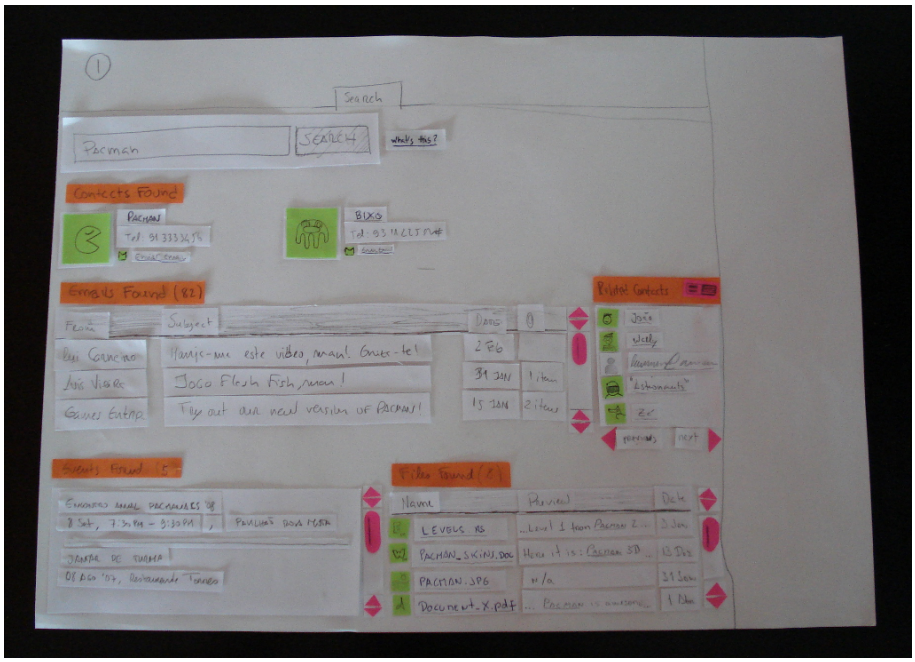

Figura 4.2: Interface do Módulo Sherlock

O objectivo desta interface passava pela unificação dos módulos actuais da plataforma: Contactos (Sybil), Calendário (Kronolite), Email (IMP/DIMP) e Gestor de Ficheiros (em desenvolvimento). Para isso, tentou-se que o resultado de uma pesquisa nesta interface pudesse ser apresentada toda numa única página, tendo em consideração que a resolução mínima suportada na nova plataforma é de 1024x768 [Cos09]. Toda esta construção teve em consideração o trabalho já efectuado na plataforma actual de forma a ser possível obter um alto nível de coerência entre as diferentes aplicações que a constituem.

Devido ao reduzido espaço e à potencial confusão que poderíamos estar a induzir ao utilizador pelo facto de ter quatro tipos de informação diferente, optou-se por ser o mais sintético possível ao mostrar apenas a informação mais relevante que um utilizador espera obter numa pesquisa. À semelhança de outros motores de busca (p.e. Google), este pretende que o utilizador saia o mais rápido possível da aplicação. Para isso, assim que utilizador carregue em qualquer item encontrado nesta interface, será redirecionado para o módulo respectivo (p.e. carrega num *email* que era um dos resultados da pesquisa e será redirecionado para o módulo IMP já a visionar o *email*).

Capítulo 5

Solução Proposta

Neste capítulo irá ser descrita a proposta para resolver os problemas existentes e criar uma aplicação que crie valor à plataforma onde este será inserido. Irá ser apresentada a arquitectura proposta para o sistema em geral e as principais decições para cada um dos módulos existentes nessa arquitectura.

5.1 Arquitectura Geral

A solução prosposta extende o modelo actual (ver figura 4.1) e está representado, esquematicamente, na Figura 5.1. A esta nova arquitectura foram adicionados três novos componentes: Módulo Solr, Índices e Módulo Sherlock.

O módulo Solr é o responsável pela indexação e pesquisa dos *emails* existentes. Para ser possível utilizar estes índices através de IMAP, existe um elo de comunicação entre Dovecot e Solr que utiliza o protocolo HTTP. No caso de pedidos de indexação, é enviado do Dovecot para o Solr um documento XML (respeitando o *Schema* existente no Solr) com o conteúdo do *email* a indexar. Por outro lado, no caso de pedidos de pesquisa, o pedido é feito através do *Request* de uma página específica enviando todos os parâmetros da pesquisa por variáveis *GET*. Em ambos os casos (pesquisa e indexação) existem respostas XML por parte do Solr.

Contudo, os pedidos de pesquisa não irão ser apenas efectuados para responder a pedidos IMAP. De forma a ser possível ter acesso completo à informação guardada no índice é necessário que a Horde Framework comunique com o Solr. Isto deve-se ao facto das novas funcionalidades desenvolvidas para este projecto (p.e. indexação do conteúdo de anexos) não serem IMAP standard. Para corresponder às necessidades exigidas por estas funcionalidades, os dados guardados para cada *email* aumentaram (ver secção 5.2) em relação ao índice original, de forma a ser possível aceder directamente ao Solr e não

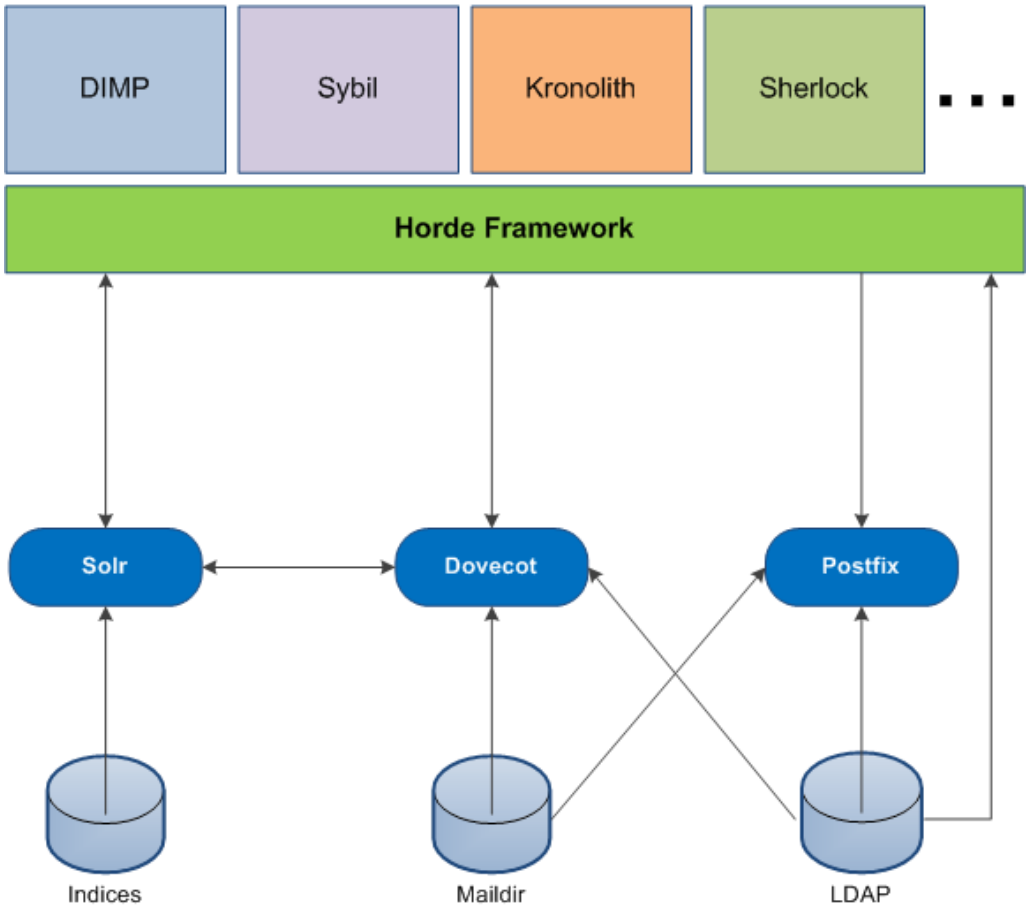

Figura 5.1: Nova arquitectura proposta

apenas recorrendo a IMAP. Assim sendo, foi criada uma nova aplicação na Horde Framework denominada Sherlock (ver secção 5.4). Esta é responsável pela comunicação entre Horde e Solr e as restantes pesquisas dos diferentes módulos existentes (p.e. IMP e Sybil) unificando assim, para o utilizador, a pesquisa existente.

5.2 Estrutura do Índice

Apesar do Dovecot já ter uma estrutura definida para o índice a criar no Solr, para garantir a implementação de todos os requisitos funcionais deste projecto, foi necessário adicionar alguns campos aos já existentes. A estrutura actual permite uma completa integração com Solr e tem a seguinte estrutura para cada documento:

id Identificador único da mensagem composto pelos campos que tornam um *email* único: uid, uidv, user e box. Este tem o formato `uid/uidv/user/box` pelo que nenhum

destes atributos deve conter o carácter /. Apesar de não trazer qualquer informação adicional, torna a identificação das mensagem mais eficiente.

uid Identificador único da mensagem ao nível de IMAP.

uidv Valor que permite saber se o UID ainda é um valor válido ou não. Este valor é alterado se a caixa de mensagens, por alguma razão, for recriada.

box A caixa de mensagens, geralmente demoninadas pastas, onde o *email* se encontra (p.e. Inbox).

ns Possibilita a associação de um *namespace* à mensagem. Isto permite que sejam aplicadas diversas operações sobre um *namespace*, o que poderá ter especial interesse em sistemas de grande escala. A existência deste campo no índice permite assim que qualquer operação efectuada sobre um *namespace* em Maildir possa também ser efectuada sobre o índice.

last_uid Consiste no UID da mensagem anterior a esta. Este campo existe apenas por questões de performance pois a determinação do UID anterior de uma determinada mensagem é largamente usada no funcionamento interno do Dovecot.

hdr Contém todos os headers da mensagem MIME (p.e. “From”, “To”, “Content-Type”, “Content-Disposition”, etc).

body Neste campo é guardado o texto da mensagem que é visível ao utilizador comum. O texto dos anexos (quer o seu conteúdo, quer o BASE64 do ficheiro) é excluido neste campo.

any Este campo não contêm qualquer informação adicional à restante estrutura. Consiste apenas na cópia dos campos Headers e Body. Esta funcionalidade particular do Solr permite que, com apenas uma pesquisa no índice seja possível pesquisar em múltiplos campos simultaneamente (no caso dois), aumentando assim a performance do sistema.

Tendo esta estrutura como base, foi necessário analisar qual a melhor forma de adicionar o conteúdo dos anexos à estrutura existente. Para cada documento indexado seria preciso, de alguma forma, relacionar todos os anexos desse documento (nome, conteúdo, etc). Em Solr existem vários tipos de campos desde os campos simples, campos de cópia (p.e. campo “Any” do índice) entre outros.

Uma das abordagens possíveis, seria acrescentar um novo tipo de campos à anterior estrutura que correspondesse a um anexo. Este campo deveria suportar valores múltiplos (funcionalidade disponível no Solr) de forma a suportar mais do que um anexo. Apesar de nesta estrutura a informação ser indexável, seria impossível uma eficiente pesquisa da

mesma. Apesar de ser possível descobrir se uma determinada pesquisa existe nos anexos, seria impossível determinar em qual dos campos esse *match* aconteceu. Imaginemos que existe um *email* com cinco anexos e procuramos a palavra "FEUP" que existe no documento 3 em anexo desse *email*. O resultado desta pesquisa seria um documento XML com os seus respectivos campos. Apesar de ser possível identificar em que texto foi encontrada a pesquisa efectuada através da funcionalidade de *Highlight* do Solr ou de uma simples expressão regular do lado de Horde ou Dovecot, não seria possível associar o nome do anexo (que está nos Headers) ao conteúdo do mesmo.

No entanto, existe uma abordagem que resolve este problema. Um dos tipos de campos disponibilizados pelo Solr é o *Dynamic Fields*. Estes campos são definidos usando padrões, através da utilização de *wildcards*, e são geralmente utilizados para especificar diferentes campos que têm propriedades semelhantes. Os campos criados para suportar a indexação de anexos foram os seguintes:

attach_* Campo dinâmico que representará todos os anexos relativos a uma mensagem. Este deverá ser completado conforme o nome do anexo. Espera-se que a aplicação que fizer um pedido de indexação (no caso o Dovecot) preencha de forma correcta o *wildcard* deste campo. Assim, o formato esperado neste campo será, por exemplo, `attach_ficheiro.pdf`.

any_attach À semelhança do campo "any" da estrutura base, este será também uma cópia. Neste caso, será a cópia do campo dinâmico definido anteriormente, o que significa que, para pesquisar em anexos, será apenas necessário pesquisar neste campo.

Espera-se, com esta estrutura, suportar de forma eficiente a implementação de todas as funcionalidades previstas para este projecto. As definições específicas do índice, definidas no capítulo 6, devem ter em conta que, apesar do módulo Dovecot não fazer uso dos novos campos especificados para a sua pesquisa (visto não serem IMAP standard), o funcionamento base do Dovecot deve-se manter. Por outro lado, as novas funcionalidades de pesquisa devem ser completamente funcionais para o módulo Horde.

5.3 Módulo Dovecot

Os requisitos funcionais para este módulo baseiam-se todos no suporte de pesquisa por texto integral em anexos. Sendo esta aplicação bastante complexa, o ponto de partida para o desenvolvimento destas funcionalidades específicas partiu do *plugin* responsável pela pesquisa por texto integral denominada de *Full Text Search Plugin* (*FTS Plugin*).

Segundo sugestão do criador do Dovecot, Timo Sirainen, a melhor metodologia seria analisar a função responsável pela transformação das mensagens em blocos legíveis

pelos *plugins* dos indexadores suportados pelo Dovecot. Depois de percebido o seu funcionamento, a análise deveria ser aprofundada até ao nível pretendido. Para esta implementação, foi necessário efectuar algumas alterações a um baixo nível do sistema. O diagrama da figura 6.1 pretende elucidar melhor sobre o funcionamento do *plugin* em questão.

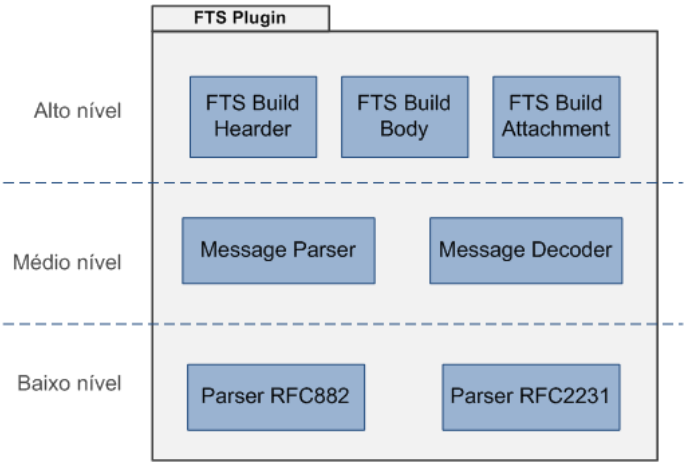

Figura 5.2: Arquitectura do *Full Text Search plugin*

O sistema base funciona a baixo nível com *parsers* que validam as mensagens relativamente aos protocolos RFC822 [1] e RFC2231 [2]. Após esta validação, são criadas todas as estruturas necessárias no nível médio da aplicação (através do *Message Decode* e *Parser*). Por fim, no nível mais elevado deste *plugin* são tratados os diversos tipos de blocos suportados, sendo estes os cabeçalhos e corpo das mensagems. Como solução proposta ao problema em questão, sugere-se a criação de um novo módulo de alto nível que trate devidamente os anexos das mensagens.

5.4 Módulo Sherlock

Um dos requisitos propostos para este trabalho é a criação de uma nova aplicação para o Horde que trate de todo o *backend* e interface de pesquisa da nova plataforma. Esta nova aplicação terá o nome *Sherlock*, inspirado nas capacidades da personagem de ficção Sherlock Holmes em pesquisa e na resolução de mistérios. Este módulo é o responsável pela componente de unificação a que este trabalho se propôs desde início. O seu funcionamento é bastante simples e consiste apenas em, após uma pesquisa efectuada pelo utilizador, recuperar a informação disponível nos diferentes módulos existentes, unificá-la e apresentá-la ao utilizador.

[1] **RFC822** - www.ietf.org/rfc/rfc822.txt
[2] **RFC2231** - www.ietf.org/rfc/rfc2231.txt

Uma das dificuldades destas operações prende-se com a forma como esta informação é armazenada. Cada módulo tem o seu próprio tipo de dados e armazena-os de forma diferente. Após a implementação do sistema de indexação e pesquisa Solr, existiram três tipos distintos de armazenamento:

Base de Dados Tanto o Sybil como o Kronolith guardam a sua informação em base de dados Postgres [3] em tabelas próprias.

Índice O Solr guarda o seu índice em disco com uma série de ficheiros correspondentes aos segmentos criados.

Maildir Todos os *emails* são guardados em disco neste formato, podendo ser acedidos pelo Dovecot através de IMAP.

Reunir diferentes tipos de dados numa pesquisa única é algo inovador nos sistemas *webmail*. Apesar disso, numa tentativa de mostrar as potencialidades do sistema de pesquisa, deverá ser adicionada uma nova componente (denominada "Contactos Relacionados") correspondendo aos contactos que estão envolvidos nos *emails* que foram retornados como resultado da pesquisa efectuada pelo utilizador. Isto permitirá ao utilizador pesquisar sobre um tema de conversa e saber quais os utilizadores que estiveram relacionados com a mesma.

Cada tipo de informação que este módulo irá disponibilizar ao utilizador envolve um ou mais pedidos. A forma como estes diferentes pedidos são efectuados segue uma determinada sequência, que pode ser consultada no diagrama de sequência contido na figura 5.3.

O módulo de pesquisa começa por efectuar a pesquisa inserida pelo utilizador ao servidor de indexação Solr. Este devolve como resposta um XML que contém todos os documentos resultantes da pesquisa efectuada. Para cada um dos documentos, a informação disponível é a especificada na secção anterior (ver secção 5.2). Apesar de neste resultado existir informação suficiente sobre os anexos (nome e conteúdo), a informação sobre os *emails* é insuficiente. No índice, é guardada apenas informação que permite identificar um *email* e não o seu cabeçalho ou corpo. Isto deve-se ao facto de esta informação já se encontrar guardada em Maildir e, caso fosse guardada também no índice, o espaço ocupado em disco na plataforma iria sofrer um grande aumento. Assim sendo, é necessário, após ter o id de cada *email* existente no resultado da pesquisa ao índice, fazer um pedido IMAP por esses id's de forma a ser possível mostrar ao utilizador informação mais relevante como assunto ou corpo da mensagem. Este pedido IMAP deve ser efectuado usando a aplicação Horde desenvolvida para o efeito, no caso, o IMP. Seguindo o mesmo

[3] **Postgres** - www.postgresql.org

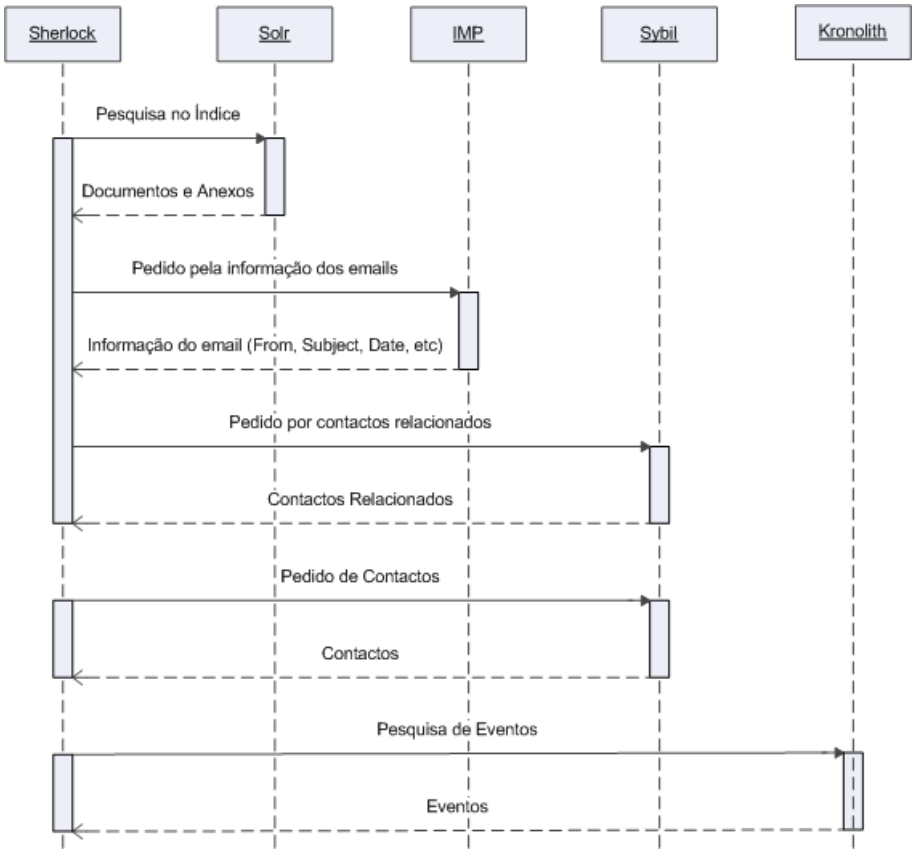

Figura 5.3: Diagrama de sequência dos pedidos efectuados pelo Módulo Sherlock

princípio de respeito à modularidade da Horde Framework, são pedidos todos os contactos relacionados (existentes nos destinatários) com as conversas em causa ao módulo Sybil.

Os restantes dois tipos de dados deverão utilizar as pesquisas já implementadas nos respectivos módulos. No caso dos contactos, o Sherlock deve utilizar a API disponibilizada pela aplicação Sybil responsável pela gestão destes. Por outro lado, no caso da pesquisa de eventos no calendário, a API utilizada deve ser a da aplicação Kronolith responsável pela integração deste na plataforma actual.

Capítulo 6

Implementação

Este capítulo pretende discutir as questões envolvidas na implementação de cada um dos módulos existentes neste projecto. Para cada um deles será explicado, em detalhe, a implementação de todas as funcionalidades assim como as decisões mais importantes tomadas ao longo da implementação.

6.1 Módulo Dovecot

A implementação actual do Dovecot já possui grande parte do projecto necessário para este trabalho. Contudo, apesar do Dovecot ser uma aplicação bastante modular e utilizar um sistema de *plugins*, as alterações necessárias para a implementação de todas as funcionalidades propostas envolvem também o sistema *Core* do Dovecot. Como já referido na secção 5.3 a metodologia a seguir passará pela análise do *FTS Plugin*, mais concretamente da função `fts_build_mail`. Toda a análise, implementação e detalhes de desenvolvimento deste módulo serão referidos nas secções seguintes.

6.1.1 Message Parser

O sistema actual tratava apenas de dois tipos de dados de um *email*: cabeçalho e corpo da mensagem. Mesmo no corpo da mensagem, apenas eram tratados alguns tipos de conteúdos. O *parsering* da mensagem é feito a baixo e médio nível. No primeiro, é feito o *parse* da mensagem ao nível das normas RFC822 e RFC2231. Neste nível são extraídos todos os dados e alocados em estrutura, de forma a que seja possível utilizar esta informação nas mais variadas partes internas do Dovecot. No nível médio, por outro lado, são usadas as estruturas de baixo nível anteriormente mencionadas.

Para os objectivos em causa apenas o nível médio irá interessar para este projecto. É a este nível, por exemplo, que são analisados os diferentes blocos de uma mensagem. Cada

bloco corresponde a uma parte da mensagem que possui as mesmas características, sendo exemplo disso mesmo os cabeçalhos da mensagem, o corpo da mensagem ou até blocos `"multipart/*"` que consistem, por exemplo, no corpo das mensagens adjacentes à original (p.e. "forwards", "replies").

Cada bloco tem as suas propriedades que definem que tipo de bloco se está a analisar. Para este projecto, as propriedades mais interessantes são as que um bloco do tipo anexo pode ter. De seguida, é apresentado um exemplo de cabeçalho relativo a um anexo de um *email*.

```
Content-Type: application/pdf;
     name="tese.pdf"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
     filename="tese.pdf"
```

Ao observar o exemplo anterior, podemos rapidamente perceber que o próximo bloco a este cabeçalho será um documento pdf, terá um *encoding* Base64 e o nome do ficheiro é `tese.pdf`. O sistema actual suporta apenas uma pequena parte de tipos de conteúdo, sendo eles:

- **message/rfc822** — indica que o *body* contém uma mensagem encapsulada com a sintaxe de uma mensagem RFC822.
- **text/*** — indica que o conteúdo do próximo bloco é apresentado numa forma textual.
- **multipart/*** — indica que esta mensagem contém um ou mais tipos diferentes de dados combinados num único corpo.

Todos os *content-types* não referidos na lista anterior são simplesmente ignorado pelo Dovecot. Foi, por isso, necessário adicionar os tipos de conteúdos necessários para implementar os requisitos funcionais que envolviam formatos de ficheiros específicos. Os novos tipos de conteúdos foram, por isso, os seguintes:

- **application/pdf** — Para suporte de ficheiros PDF
- **application/msword** — Para suporte de ficheiros Microsoft Word
- **application/ms-excel** — Para suporte de ficheiros Microsoft Excel
- **application/mspowerpoint** — Para suporte de ficheiros Microsoft PowerPoint

Na verdade, foram adicionados muitos outros tipos de conteúdo. Contudo, isto deve-se apenas ao facto de existirem múltiplos *Content-Types* para cada um dos formatos anteriormente listados [1].

Com a adição destes novos *content-types*, o sistema deixará de ignorar o conteúdo dos blocos relativos aos anexos. Para finalizar as alterações no *Message Parser* do Dovecot, era necessário apenas guardar mais um dado: o nome do ficheiro. No exemplo dado anteriormente, era possível verificar que o nome do ficheiro em anexo naquele bloco estava presente tanto no *Content-Type* como no *Content-Disposition*. Contudo, após a consulta da norma RFC2183 [2], descobriu-se que o nome do ficheiro no *Content-Type* é de carácter facultativo. Posto isso, foi necessário implementar de raiz um *parser* para o *Content-Disposition*, visto ainda não existir (nunca foi necessário este atributo para a implementação actual do Dovecot). Esta implementação foi, no entanto, bastante simples pois foi tomada como base a metodologia utilizada no *parser* já existente para o *Content-Type*.

Terminadas as alterações no *Message Parser*, é possível agora analisar os blocos das mensagens relativos a anexos (visto já não serem ignorados) e associar, caso isso se verifique, o nome do ficheiro em anexo num determinado bloco.

6.1.2 Extracção do Conteúdo de Anexos

Após guardada a informação necessária para cada bloco correspondente a um anexo, foi necessário proceder à extracção do conteúdo dos ficheiros em causa. Assim como o já referido na secção 3.4, a opção tomada baseou-se na utilização de programas externos ao Dovecot. Apesar de ser a opção mais fácil de implementar, esta traz algumas contrapartidas.

Alguns dos programas testados não utilizam o *standard input* como entrada (p.e. pdftotext [3]). Isto torna-se um problema na medida em que, para o correcto funcionamento do programa, será necessário a utilização de ficheiros temporários. O uso de tais ficheiros podem levantar diversos problemas quando esta plataforma for utilizada de forma exaustiva. Caso o código desenvolvido não seja robusto, existe grande potencial para a ocorrência de problemas relacionados com a manutenção do serviço (p.e. ficheiros temporários que não são apagados, problemas de leitura/escrita nos ficheiros, etc).

Apesar das desvantagens de utilizar programas externos serem claras, existiram dois factores que influenciaram bastante a decisão do seu uso. Uma delas deveu-se ao facto de, na ausência de bibliotecas de extracção em C, o tempo de implementação deste tipo de bibliotecas ser incomportável no tempo disponível para o projecto. Contudo, a razão que mais influenciou esta decisão foi a forma como alguns formatos de ficheiros são

[1] www.iana.org/assignments/media-types
[2] **RFC2183** - www.ietf.org/rfc/rfc2183.txt
[3] **pdftotext** - http://www.foolabs.com/xpdf/

estruturados. Os formatos a serem suportados por esta aplicação (DOC, XLS, PPT e PDF) são formatos binários e, ao contrário dos ficheiros de texto, não seguem uma estrutura linear. Alguns destes formatos, como o PDF [4], requerem, a determinado ponto da sua interpretação, ler o final do ficheiro. No formato PDF existem no final do ficheiro tabelas de indexação que permitem ao visualizador de PDF decompor os dados correctamente. Assim sendo, mesmo que o programa suportasse leitura através do *standard input*, teria, primeiro, de o ler completamente antes de o converter. O mesmo princípio se aplica à utilização de bibliotecas.

Para este projecto, foram escolhidos os seguintes programas para tratar os formatos existentes: pdftotext (pdf), catdoc (Microsoft Word), catppt(Microsoft PowerPoint) e xls2csv (Microsoft Excel).

Todo o código desenvolvido teve em consideração, ainda assim, possíveis alterações na arquitectura de extracção do conteúdo dos anexos, sendo, com a arquitectura actual, fácil a alteração da utilização de programas externos para a utilização de bibliotecas C. O modelo utilizado segue os passos do fluxograma da figura 6.1.

Figura 6.1: Fluxograma do funcionamento do sistema de extracção de anexos

[4] **PDF Specification** - www.adobe.com/devnet/pdf/pdf_reference.html

O único problema crítico desta implementação deveu-se à falta de suporte por parte do *Message Decoder* para blocos que não texto (`text/*`). Esta necessidade nunca se verificou pois o único tipo de dados em que havia necessidade de *decode* era, de facto, o texto (as restantes que usam *encoding* eram ignoradas). Este problema originou, até, a saida de uma nova revisão para a versão 1.2 do Dovecot [5].

6.1.3 Full-Text-Search Plugin

O *FTS Plugin* é o *plugin* genérico utilizado por todos os *plugins* que implementam a integração com indexadores. De momento, existe um interno ao Dovecot denominado de *Squat*, uma implementação de Lucene em C (CLucene) e o utilizado neste projecto, o Solr. Como o referido na secção 6.1 é neste *plugin* que o *email* é construído antes de ser enviado para o *backend* do *plugin* responsável pela integração com *Solr* (*Solr Plugin*).

O funcionamento original deste *plugin* é bastante simples. Cada mensagem é decomposta, através do *Message Parser*, nos seus respectivos blocos. Contudo, apenas os blocos de texto, ou seja, com *Content-Type* `text/*` ou `message/rfc822`, são aproveitados nesta construção. Se estes blocos tiverem o *Content-Type* desejado, serão enviados para o *Solr Plugin* como sendo *headers* (cabeçalhos) ou *body* (corpo).

Tendo adicionado um novo conjunto de *Content-Types* válidos, tornou-se necessário proceder a algumas alterações no funcionamento deste *plugin*. Assim que detectado um *Content-Type* que não seja texto, é verificado se é um formato conhecido e suportado. Caso o seja, são inicializadas as estruturas definidas na secção anterior e adicionado, depois do *decode*, cada pequena parte do bloco ao ficheiro temporário. No fim do bloco, este é enviado para o *Solr Plugin* com os parâmetros correspondentes ao tipo do bloco (*header*, *body* ou *attachment*).

A última parte necessária à indexação dos anexos é a construção do pedido XML a fazer ao Solr através de HTTP. Esta construção é feita pelo *Solr Plugin* e, para que este suportasse anexos, foi apenas necessário adicionar alguns novos argumentos como *input*, de forma a que fosse possível a correcta construção do XML a enviar.

6.2 Módulo Solr

Uma das particularidades de Solr em relação ao Lucene é que ao contrário de Lucene que consiste numa API de indexação e de pesquisa, Solr é uma ferramenta pronta a utilizar. Assim sendo, a implementação neste módulo será bastante reduzida passando apenas pela configuração do servidor de forma a que sirva os propósitos deste projecto.

[5] http://hg.dovecot.org/dovecot-1.2/rev/44548a7fb10d

6.2.1 Configurações

Após decidida qual a estrutura do índice a utilizar (ver 5.2), foi necessário definir quais as propriedades dos dois novos campos. A configuração dos novos campos pode ver vista na tabela 6.1.

Tabela 6.1: Configuração escolhida para os novos campos do índice

Field	type	indexed	stored	multiValued	required
attach_*	text	true	true	true	false
any_attach	text	true	false	true	false

Visto ser necessário poder pesquisar em ambos os campos, os dois novos campos foram definidos como indexáveis. Contudo, apenas o campo `attach_*` é guardado para além de indexado. Apesar do anexo também se encontrar guardado em Maildir (num bloco da mensagem com *encode* Base64), é imprescindível guardá-lo também no índice, de forma a que o conteúdo do anexo tenha fácil acesso através da aplicação criada no Horde.

Para além do aumento do espaço ocupado em disco, o único problema desta decisão é o grande aumento do tráfego do sistema caso sejam pesquisados ficheiros com anexos muito grandes. Contudo, este problema é facilmente resolvido usando a funcionalidade de *Hightlight* do Solr, já referida anteriormente na secção 3.3. Aplicando o *Highlight* ao campo `attach_*`, tem-se então apenas o conteúdo mais relevante do documento em causa, no caso, apenas as N palavras (sendo N um número configurável) antes e depois de onde foi encontrada a pesquisa efectuada.

6.3 Módulo Sherlock

A componente de unificação deste trabalho englobou duas das três aplicações existentes e relevantes para o utilizador, o DIMP/IMP e o Sybil. A aplicação Kronolith, responsável pelos calendários, foi excluída da unificação das pesquisas visto encontrar-se em actual desenvolvimento no âmbito de outra Tese de Mestrado a ser desenvolvida na empresa. Contudo, esta será, ainda assim, tida em conta na criação de todos os protótipo.

A criação da aplicação Sherlock baseou-se num *template* disponibilizado pela Horde Framework para a criação de novas aplicações denominado Skeleton. Todo o código foi desenvolvido tendo em conta que um dos requisitos deste módulo é a criação de uma API para futura utilização por parte de outras aplicações Horde.

6.3.1 Comunicação com Solr

Todo o processo de recolha da informação seguiu o modelo proposto na figura 5.3 do capítulo . Visto os pedidos ao Solr consistirem em pedidos HTTP, foi necessário construir

devidamente esse pedido. Existe uma grande quantidade de parâmetros que podem ser usados, estando na tabela 6.2 listados os que foram utilizados nesta construção.

Tabela 6.2: Parâmetros do pedido a enviar ao Solr

Parâmetro	**Descrição**
q	Único parâmetro obrigatório e que corresponde à pesquisa que o utilizador deseja fazer
start	Utilizado para a paginação de resultados pois permite indicar o *offset* do resultado completo, isto é, definir quando o conjunto de resultados deve começar. O valor por defeito é zero.
row	Utilizado, à semelhança do anterior, na paginação de resultados. Este define o máximo de documentos que serão devolvidos no resultado. O valor por defeito é 10
fl	Permite especificar quais os campos a serem retornados numa pesquisa.
wt	Tipo de linguagem em que o resultado é retornado (p.e. XML, Perl, PHP, etc)
hl	Para activar a funcionalidade de *Hightlight* é necessário especificar este parâmetro como `on` e o parâmetro `hl.fl` com o valor do campo a ser submetido esta funcionalidade.

Na sua maioria todos os parâmetros foram deixados com os seus valores por defeito. Os únicos em que tal não ocorreu foram o tipo de linguagem que, por questões de facilidade de integração, foi escolhido PHP (`wt=php`) e o *Highlight* que foi activo para o campo onde são guardados os textos contidos nos anexos. (`hl.fl=attach_*`)

6.3.2 Comunicação com Sybil

A implementação de todos as aplicações Horde são completamente integradas na Horde Framework. Isto permite que a comunicação entre elas seja bastante simples. Cada aplicação tem, para além das suas funções privadas, funções registadas na *framework*. Estas funções ficam, assim, disponíveis para que qualquer outra aplicação as utilize.

No caso da pesquisa por contactos e grupos, o módulo Sybil não possuia qualquer função registada na *framework*, pelo que foi necessário desenvolver uma função que se regista-se na *framework*. Após este passo, foi possível chamar estes novos métodos na aplicação Sherlock.

Os pedidos feitos à aplicação Sybil foram de três tipos: contactos, grupos e contactos relacionados. Para a pesquisa de contactos e grupos foi utilizada apenas a pesquisa actual da aplicação Sybil e apresentados os resultados. Contudo, para os contactos relacionados, a dependência de informação vinda de diversos locais levou a que não fosse tão simples quanto os primeiros. Ainda assim bastou, após recebidos os resultados do Solr e IMP, determinar quais os contactos (conhecidos ou não) que estavam presentes nos remetentes das mensagens em causa. Depois de descobertos os *emails* dos contactos, bastou fazer

um novo pedido à aplicação pela informação dos contactos a que os respectivos *emails* pertenciam.

6.3.3 Protótipo

Tendo todo a implementação *backend* do módulo Sherlock concluida foi bastante simples implementar um protótipo funcional. Baseado no protótipo em papel (ver 4.4) foi construido o protótipo representado na figura 6.2.

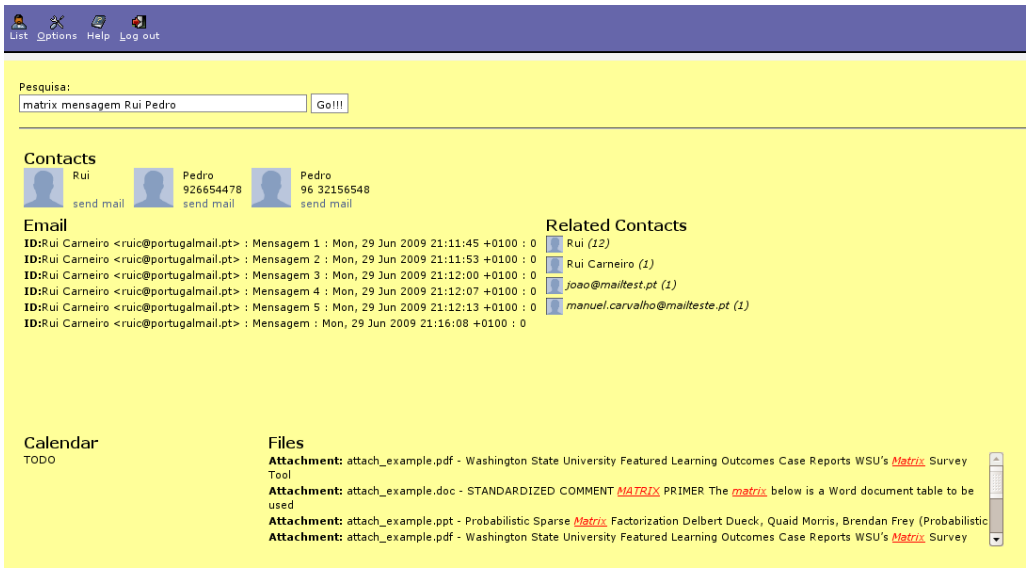

Figura 6.2: Protótipo implementado

Como o referido anteriormente a única componente não implementada foi o calendário. Todas as restantes funcionalidades estão implementadas faltando apenas implementar todas as questões relativas a design (p.e. HTML e CSS) e usabilidade (p.e. tolerância a erros). Contudo, estas decisões também já foram tomadas em conjunto com o *designer* da empresa. O resultado obtido pode ser visualizado na figura 6.3

Neste protótipo de alta resolução tentou-se enquadrar o design do módulo de pesquisa com o restante da plataforma. Questões de usabilidade como memorização, curva de aprendizagem e *feedback* das acções são apenas alguns dos aspectos considerados.

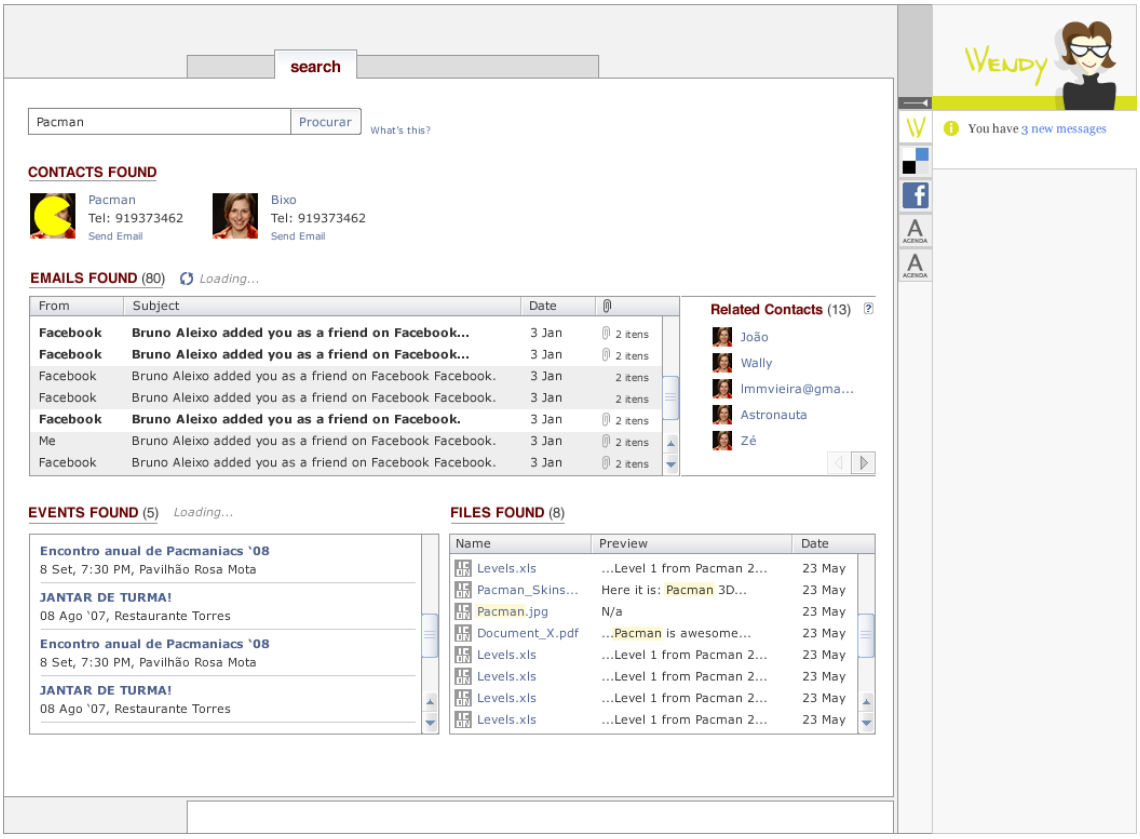

Figura 6.3: Protótipo de alta resolução

Capítulo 7

Testes e Resultados

Neste capítulo será abordados uma série de testes realizados sobre o sistema desenvolvido. Para estes testes foram utilizadas condições semelhantes às utilizadas nos *benchmarks* efectuados anteriormente (ver 3.2.2.1). Contudo, para estes testes, foi apenas utilizado o conjunto de *emails* que pertence a uma amostra real de um utilizador.

7.1 Indexação

A amostra utilizada para os testes de indexação foi a amostra que contém parte de uma conta real (ver Secção 3.2.3.1). Esta escolha deveu-se ao seu tamanho (a maior de todas as amostras utilizadas neste projecto) e devido a ser a única amostra rica em anexos. Do conjunto de 2854 *emails* que a constituem existe um total de 189 anexos suportados. A sua distribuição é a seguine:

- **Portable Document Format (PDF)** - 68 documentos
- **Microsoft Word (doc)** - 21 documentos
- **Microsoft Powerpoint (doc)** - 98 documentos
- **Microsoft Excel (xls)** - 2 documentos

Para este teste, a indexação deste conjunto de dados foi despoletada pelo Dovecot. Através de uma ligação `telnet` ao serviço de IMAP, foi feita uma pesquisa pelo termo `xxyyzz`. Os tempos obtidos nos três testes efectuados não diferiram muitos sendo a sua média de 37.53 segundos.

Tomando como certo que este tempo aumenta de forma linear com o aumento o tamanho da amostra, é possível obter uma estimativa de quando demoraria a indexação de todos os dados da Portugalmail. Todas as contas de *email* da Portugalmail ocupam

até à data cerca de 15TB. Apesar de não ser muito científico, é possível estimar que a indexação de todas estas contas levaria cerca de 15 dias de processamento. Infelizmente foi impossível, durante a escrita desta tese, efectuar testes nos servidores reais onde o sistema de indexação irá ser posto em funcionamento mais tarde. Estes testes teriam resultados muito melhores do que os obtidos devido às enormes diferenças de capacidade de processamento e memória entre os servidores empresariais comuns e o *desktop* onde foram efectuados estes testes.

Mesmo podendo ser bastante melhorados (com a utilização de um servidor dedicado), estes resultados foram encarados por parte dos responsáveis da empresa como bastante satisfatórios.

Apesar da aprovação, foram efectuados mais alguns testes de forma a testar quais as percentagens de ocupação de processamento e memória. Os resultados podem ser consultados nas Figuras 7.1 e 7.2.

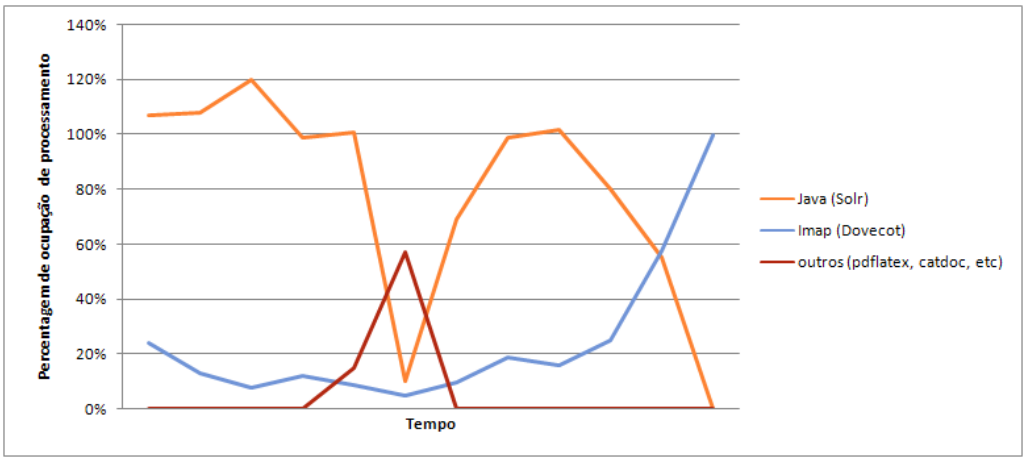

Figura 7.1: Evolução da utilização do processador durante a indexação

Cada uma das figuras representa a evolução do processo de indexação. O facto de na Figura 7.1 a percentagem de utilização ascender acima dos 100% deve-se ao facto de o comando utilizado para este teste ter sido o comando Linux `top`. Este comando toma como valor máximo de utilização 200% devido à utilização de um processador de dois núcleos nestes testes.

Destes dois testes, podemos concluir que a influência que a utilização de programas externos ao Dovecot para a extracção do seu conteúdo não foi tão negativa como se ponderou ser. Apesar de se notar por vezes a influência destes programas, a sua ocorrência é esporádica. O facto de ser visível na figura 7.1 a utilização destes programas foi até pura sorte visto o comando `top` só actualizar uma vez em cada 3 segundos.

Um dos factos interessantes observados foi o facto da memória utilizada ter sido bastante baixa. Apesar de, para o Dovecot, a pouca utilização de memória ser uma das

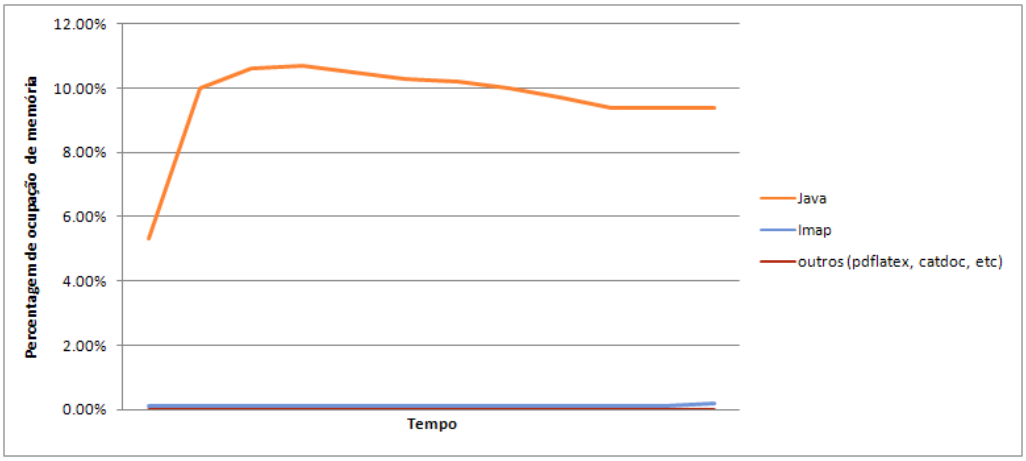

Figura 7.2: Evolução da utilização da memória durante a indexação

suas características, para o Solr está relacionado com as configurações utilizadas. É possível, para o Solr, definir uma série de parâmetros que alteram drasticamente o rácio de utilização processador/memória. Estas configurações não foram abordadas pois, a esta altura, é impossível determinar quais serão as necessidades do servidor dedicado para o Solr.

7.2 Pesquisa

Outro dos processos que foi necessário testar e avaliar foi a pesquisa sobre o índice criado durante a indexação. Sendo este um sistema que irá ter diversos utilizadores em simultâneo, foi testada a capacidade que este sistema tem de lidar com múltiplos pedidos simultâneos.

Os seguintes testes foram baseados em testes anteriormente feitos por Justin Leider [1] utilizando a *Apache HTTP server benchmarking tool* que permite fazer efectuar facilmente testes com múltiplos pedidos e diferentes concorrências (pedidos em simultâneo). Nestes testes foram utilizados vários níveis de concorrência de pedidos indo de 2 pedidos até 512 pedidos em simultâneo. O *script* utilizado pode ser consultado no Anexo .

Os resultados obtidos foram separados por dois tipos de resultados: tempo de reposta médio e número de pedidos tratados por segundo. Visto o *script* usado não mudar a sua pesquisa (no caso é sempre "Rui") foram feitos testes com e sem *Cache* activa.

De seguida, alguma informação mais detalhada sobre o pior caso, ou seja, sem Cache e concorrência de 512 pedidos.

[1] www.derivante.com/authors/justin-leider

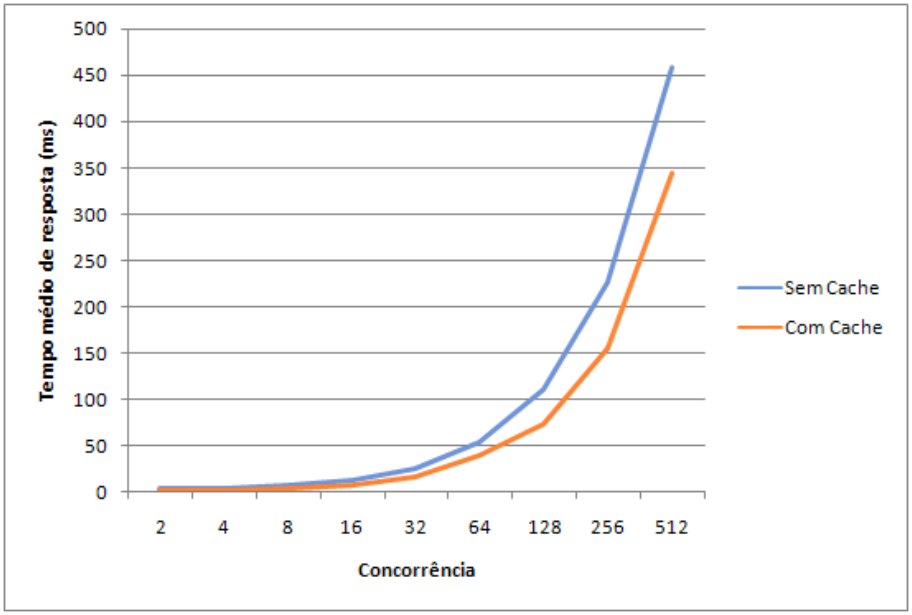

Figura 7.3: Tempo médio de resposta para os pedidos

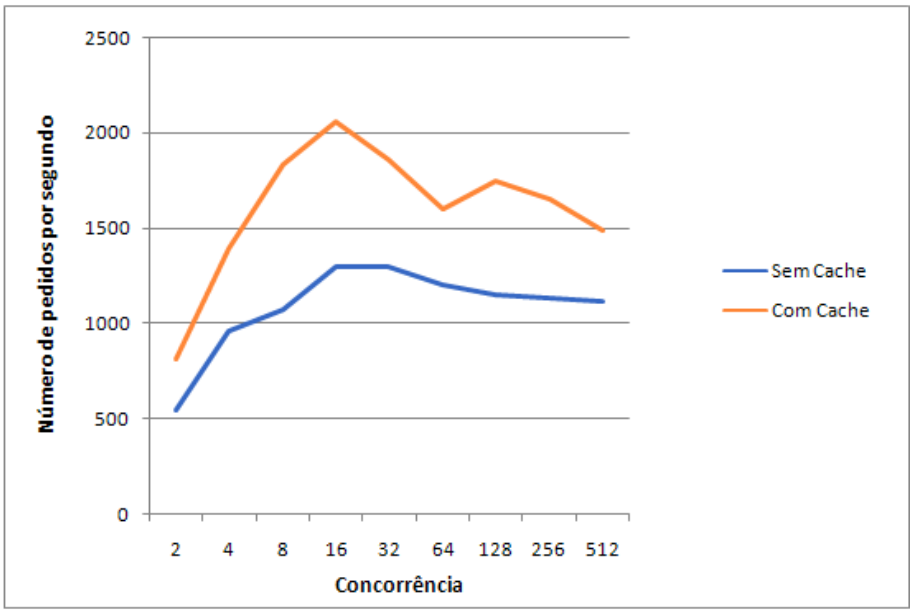

Figura 7.4: Número de pedidos tratados por segundo

Analisando a tabela 7.1 podemos observar que o tempo médio de resposta foi de 450ms e o sistema conseguiu responder a cerca de 1100 pedidos por segundo. Isto significa que, mesmo com a máquina actual de testes, seria possível responder a 1100 utilizadores em simultâneo num tempo bastante aceitável. Deve ser tido em conta, contudo, que os tempos foram obtidos localmente, ou seja, quase sem latências adicionais da rede.

Tabela 7.1: Resultados da pesquisa sem Cache e concorrência de 512 pedidos

Tempo do teste	`458.891 (sec)`
Pedidos respondidos	`512000`
Dados tranferidos	`3.6 GB`
Pedidos por segundo	`1115.73 [#/sec] (média)`
Tempo por pedido	`458.891 [ms] (média geral)`
Velocidade de tranferência	`8258.11 [Kbytes/sec] recebidos`

Ainda assim, não deixa de ser surpreendente como o sistema conseguiu tratar de 3.6 GB de informação em apenas 458 segundos.

7.3 Conclusões

Apesar de não terem sido tomadas muitas precausões sobre o desempenho do Solr, os sistema teve um desempenho bastante bom. Algumas opções críticas da implementação do suporte para anexos no Dovecot mostraram-se bastante eficientes. Ainda assim, deve-se ter em conta que existem muitas questões de optimização que devem ser tidas em conta antes da instalação deste sistema. Configurações da *Java Virtual Machine*, suporte *Multi-Core* no Solr e a utilização de bibliotecas eficientes para extração de anexos em Dovecot são apenas algumas das alterações que irão ter grande influência nos resultados obtidos.

Capítulo 8

Conclusões e Perspectivas de Trabalho Futuro

Neste último capítulo, são apresentadas algumas referências aos capítulos anteriores e as suas conclusões. São apresentadas conclusões sobre o trabalho que foi desenvolvido, em que medida este foi inovador, quais as suas limitações actuais e por fim quais as perspectivas futuras de desenvolvimento deste projecto.

8.1 Trabalho Desenvolvido

Os objectivos pretendidos no início deste projecto foram totalmente atingidos. Com este projecto, conseguiu-se dotar a plataforma actual da Portugalmail de um sistema de pesquisa por texto integral, que permite a pesquisa alargada de um novo elemento, no caso, o conteúdo de anexo. A ferramenta Recuperação de Informação foi também alvo de uma escolha acertada. Os testes efectuados e a experiência adquirida durante a implementação do projecto confirma as potencialidades que já tinham sido detectadas no início deste projecto.

Por outro lado, o novo módulo de pesquisa Sherlock, permite efectuar uma pequena amostra da tecnologia implementada, conseguindo ter o factor de unificação ao qual este projecto se tinha proposto.

Em termos de requisitos funcionais, foram todos correctamente implementados à excepção da pesquisa por eventos. Este facto deveu-se a esta aplicação se encontrar em actual desenvolvimento dentro da empresa, o que levou a considerar esta funcionalidade como trabalho futuro.

Todos os requisitos não funcionais foram respeitados à excepção da usabilidade do sistema que, por decisão em conjunto com a empresa, foi adiada para o final da Tese.

8.2 Inovações

Um dos motivos da escolha deste projecto foi o seu carácter inovador. Não existe nenhuma aplicação webmail que consiga unificar todas as suas pesquisas numa só de forma clara e simples. Mesmo alargando o domínio da pesquisa fora do *webmail* temos poucos exemplos de pesquisas que a façam eficientemente (à excepção do Google).

Outro factor inovador deste projecto foi a disponibilização de um sistema que permitisse a indexação do conteúdo de anexos. Apesar de este sistema existir na Yahoo! e em alguns sistemas pagos, este projecto foi o primeiro a contribuir com esta funcionalidade para um projecto *Open-Source*, no caso, o projecto Dovecot.

8.3 Limitações

Devido à complexidade deste projecto e ao tempo restrito para este trabalho, existem algumas limitações que, contudo, não afectam o correcto funcionamento do mesmo.

Estas limitações passam, por exemplo, pela forma como algumas decisões foram tomadas no Dovecot. O local onde são guardados os ficheiros temporários, o tamanho dos *buffers* e programas externos a usar deveriam ser, todos eles, configuráveis a partir dos ficheiros de configuração originais do Dovecot. Actualmente, todas estas definições são feitas através de código.

Devido a existirem algumas restrições quanto ao espaço ocupado pelos *emails* e respectivos índices, algumas das funcionalidades não foram implementadas da forma mais eficiente. O facto de ser incomportável para a empresa, por exemplo, guardar toda a informação nos índices faz com que o método de obtenção de informação complique e se torne consequentemente menos eficiente.

Outra das limitações prende-se com o facto da qualidade das pesquisas das aplicações Horde já existentes ser, por vezes, fraca. Dado o tempo excasso de desenvolvimento, foi incomportável a re-implementação da pesquisa em cada um dos módulos, o que poderá tornar a qualidade do produto final menor.

8.4 Perspectivas Futuras

Grande parte deste trabalho consiste na prova de um conceito e na criação de uma interface que prova-se ser inovadora e eficaz. Apesar deste objectivo ter sido conseguido, muito trabalho adicional pode ser desenvolvido tendo como base este projecto.

As capacidades de pesquisa desta aplicação, em conjunto com as actividades inteligentes da secretária virtual da plataforma, levam o desenvolvimento a outro nível. Funcionalidades como o *Faceted Navigation* poderiam ser implementadas no futuro, utilizando

a secretária virtual como elo de ligação entre a funcionalidade e o utilizador final. A secretária, após uma pesquisa, poderia sugerir novas pesquisas ao utilizador de forma a que este pudesse melhorar a sua pesquisa.

Outro possível melhoramento prende-se com o estudo efectuado do estado da arte neste projecto. Existe muito potencial em algumas tecnologias e técnicas estudadas, como é o caso do *Re-Raking*, alguns algorimtmos de *Data Mining* ou até o enverdar pela pesquisa semântica.

A nível de interface, durante a escrita desta tese, já estavam decididas as datas de implementação da mesma.

Por fim, todas as limitações existentes para este trabalho poderiam ser eliminadas à excepcção das limitações criadas pela própria empresa.

Referências

[ABD06] Eugene Agichtein, Eric Brill e Susan Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM.

[AC08] Ahmed Abbasi e Hsinchun Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Trans. Inf. Syst.*, 26(2):1–29, 2008.

[AHLM03] Oren Etzioni Alon, Alon Halevy, Henry Levy e Luke Mcdowell. Semantic email: Adding lightweight data manipulation capabilities to the email habitat. In *In Sixth Int. Workshop on the Web and Databases*, pages 12–13, 2003.

[All03] James Allan. Hard track overview in trec 2003, 2003.

[AR08] Rachit Arora e Balaraman Ravindran. Latent dirichlet allocation based multi-document summarization. pages 91–97, 2008.

[Bal08] David Balmain. *Ferret*. O'Reilly, 2008.

[BH06] Danyel Fisher Bergie Hogan. A scale for measuring email overload. Technical Report MSR-TR-2006-65, Microsoft Research, 2006.

[CDZ07] Sara Cohen, Carmel Domshlak e Naama Zwerdling. On ranking techniques for desktop search. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1183–1184, New York, NY, USA, 2007. ACM.

[CGJ+03] Andy Cockburn, Saul Greenberg, Steve Jones, Bruce Mckenzie e Michael Moyle. Improving web page revisitation: Analysis ,design ,and evaluation. *IT and Society*, 1:159–183, 2003.

[Cha04] Francis Chantree. Ambiguity management in natural language generation. 2004.

[Cha08] Timothy P. Chartier. A googol of information about google. *Computing in Science and Engg.*, 10(6):11–12, 2008.

[Cos09] Felipe Costa. New paradigm of webmail interfaces. Masterthesis, Faculdade de Engenharia da Universidade do Porto, 2009.

[Cse08] Gabor Cselle. Gabor hits send, 2008. http://www.gaborcselle.com/, consultado a 1 Maio de 2009.

[Din04] Luca Dini. Nlp technologies and the semantic web: Risks, opportunities and challenges. 2004.

[DK06] Laura A. Dabbish e Robert E. Kraut. Email overload at work: an analysis of factors associated with email strain. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 431–440, New York, NY, USA, 2006. ACM.

[Dua08] Miguel Duarte. Alto desempenho com java, 2008. Sapo Codebits 2008, http://codebits.sapo.pt/intra/s/speaker/27.

[DWPP08] Mark Dredze, Hanna M. Wallach, Danny Puller e Fernando Pereira. Generating summary keywords for emails using topics. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 199–206, New York, NY, USA, 2008. ACM.

[Gom] Ivo Gomes. Protipagem em papel. 3º Seminário de Usabilidade APPU, Lisboa.

[Has05] Vesna Hassler. Open source libraries for information retrieval. *IEEE Softw.*, 22(5):78–82, 2005.

[Hay] Martin Haye. Cross-instance search system - search engine comparison. Technical report. disponivel em http://www.cdlib.org/inside/projects/xtf/Search_Engine_Comparison.pdf.

[HDL+09] Rachid Hadjidj, Mourad Debbabi, Hakim Lounis, Farkhund Iqbal, Adam Szporer e Djamel Benredjem. Towards an integrated e-mail forensic analysis framework. *Digital Investigation*, 5(3-4):124–137, 2009.

[HLW+08] Tianyong Hao, Zhi Lu, Shitong Wang, Tiansong Zou, Shenhua GU e Liu Wenyin. Categorizing and ranking search engine's results by semantic similarity. In *ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 284–288, New York, NY, USA, 2008. ACM.

[HP08] Pedram Hayati e Vidyasagar Potdar. Evaluation of spam detection and prevention frameworks for email and image spam: a state of art. pages 520–527, 2008.

[IHFD08] F. Iqbal, R. Hadjidj, B. C. M. Fung e M. Debbabi. A novel approach of mining write-prints for authorship attribution in e-mail forensics. *Digital Investigation*, 5(1):42–51, September 2008.

[KFA+04] Yuki Kadoya, Masao Fuketa, Elsayed Atlam, Kazuhiro Morita, Shinkaku Kashiji e Jun-ichi Aoe. An efficient e-mail filtering using time priority measurement. *Inf. Sci. Inf. Comput. Sci.*, 166(1-4):213–229, 2004.

[LSDJ06] Michael S. Lew, Nicu Sebe, Chabane Djeraba e Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, 2006.

[Man07] Christoph Mangold. A survey and classification of semantic search approaches. *Int. J. Metadata Semant. Ontologies*, 2(1):23–34, 2007.

[MEH04] Luke Mcdowell, Oren Etzioni e Alon Halevy. Semantic email: Theory and applications. *J. Web Semantics*, 2:153–183, 2004.

[MO06] Craig Macdonald e Iadh Ounis. Combining fields in known-item email search. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 675–676, New York, NY, USA, 2006. ACM.

[MTK01] Smaranda Muresan, Evelyne Tzoukermann e Judith L. Klavans. Combining linguistic and machine learning techniques for email summarization. In *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[Neo09] Ranknet: How bing works, 2009. http://neotracks.blogspot.com/2009/06/ranknethow-bing-works.html, consultado a 15 Junho de 2009.

[NJJY09] Xiaomin Ning, Hai Jin, Weijia Jia e Pingpeng Yuan. Practical and effective ir-style keyword search over semantic web. *Inf. Process. Manage.*, 45(2):263–271, 2009.

[OWHM07] Hartmut Obendorf, Harald Weinreich, Eelco Herder e Matthias Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 597–606, New York, NY, USA, 2007. ACM.

[San06] Rui Santos. Análise e implementação de um interface ajax para webmail. Masterthesis, Faculdade de Engenharia da Universidade do Porto, 2006.

[She97] Hadar Shemtov. *Ambiguity Management in Natural Language Generation*. PhD thesis, Stanford, CA, USA, 1997. Adviser-Kay, Martin.

[SHH+06] Salvatore J. Stolfo, Shlomo Hershkop, Chia-Wei Hu, Wei-Jen Li, Olivier Nimeskern e Ke Wang. Behavior-based modeling and its application to email analysis. *ACM Trans. Internet Technol.*, 6(2):187–221, 2006.

[Sil06] Daniela Silva. Análise e desenvolvimento do interface de gestão de correio electrónico empresarial portugalmail. Masterthesis, Faculdade de Engenharia da Universidade do Porto, 2006.

[STD06] David Schuff, Ozgur Turetken e John D'Arcy. A multi-attribute, multi-weight clustering approach to managing ë-mail overload. *Decis. Support Syst.*, 42(3):1350–1365, 2006.

[TAAK04] Jaime Teevan, Christine Alvarado, Mark S. Ackerman e David R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422, New York, NY, USA, 2004. ACM.

[TAJP07] Jaime Teevan, Eytan Adar, Rosie Jones e Michael A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158, New York, NY, USA, 2007. ACM.

[TBC+03] Kazem Taghva, Julie Borsack, Jeffrey Coombs, Allen Condit, Steve Lumos e Tom Nartker. Ontology-based classification of email. In *ITCC '03: Proceedings of the International Conference on Information Technology: Computers and Communications*, page 194, Washington, DC, USA, 2003. IEEE Computer Society.

[Tee08] Jaime Teevan. How people recall, recognize, and reuse search results. *ACM Trans. Inf. Syst.*, 26(4):1–27, 2008.

[ULL+07] Victoria Uren, Yuangui Lei, Vanessa Lopez, Haiming Liu, Enrico Motta e Marina Giordanino. The usability of semantic search tools: A review. *Knowl. Eng. Rev.*, 22(4):361–377, 2007.

[ULM08] Victoria S. Uren, Yuangui Lei e Enrico Motta. Semsearch: Refining semantic search. In *ESWC*, pages 874–878, 2008.

[Voo01] Ellen M. Voorhees. Overview of TREC 2001. In *Text REtrieval Conference*, 2001.

[WM04] Stephen Wan e Kathy McKeown. Generating overview summaries of ongoing email thread discussions. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 549, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[WS96] Steve Whittaker e Candace Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 276–283, New York, NY, USA, 1996. ACM.

[WSWS08] Chun Wei, Alan Sprague, Gary Warner e Anthony Skjellum. Mining spam email to identify common origins for forensic application. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1433–1437, New York, NY, USA, 2008. ACM.

[YC02] Behnak Yaltaghian e Mark Chignell. Re-ranking search results using network analysis a case study with google: a case study with google. In *CASCON '02: Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, page 14. IBM Press, 2002.

[YC04] Behnak Yaltaghian e Mark H. Chignell. Effect of different network analysis strategies on search engine re-ranking. In *CASCON '04: Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, pages 308–317. IBM Press, 2004.

[ZDL08] David M. Zajic, Bonnie J. Dorr e Jimmy Lin. Single-document and multi-document summarization techniques for email threads using sentence compression. *Inf. Process. Manage.*, 44(4):1600–1610, 2008.

Anexo A

Bibliotecas de Recuperação de Informação

A.1 Listagem de Bibliotecas Analisadas

Para a indexação e pesquisa da informação contida nos emails (e seus conteúdos) é necessario um motor de pesquisa que permita a indexação da informação e posterior pesquisa. Assim, foi feita uma análise aos principais motores de busca Open-Source existentes. Para a escolha foram considerados aspectos como: formatos de ficheiros suportados, linguagens de programação, escalabilidade, documentação, etc.

De seguida estão listados os motores de pesquisa que se evidenciaram por algumas das suas carateristicas (após uma pré-selecção de mais de 20 ferramentas). Em baixo, estão listadas as particularidades mais relevantes (pela positiva e negativa).

Egothor

Linguagem	Java
Documentação	Apenas a v1.x tem documentação em ingles. A v2.x ainda está em desenvolvimento
Comunidade	Fraca
Prós	Tem melhor performance que o Lucene
Contras	Apesar de ser java é algo complexo Diz ser fácil acrescentar novos formatos de ficheiros Não existe limite nem de documentos a indexar nem de tamanho máximo
URL	http://egothor.sourceforge.net

Lucene

Linguagem	Java
Documentação	Excelente
Comunidade	Muito Grande
Prós	É o mais usado e o que tem maior comunidade.
Contras	Existe em quase qualquer linguagem (C e PHP por exemplo) o que pode permitir boa integração com o dovecot e horde; Diz ser fácil acrescentar novos formatos de ficheiros; Não existe limite nem de documentos a indexar nem de tamanho máximo.
URL	http://egothor.sourceforge.net

mnoGoSearch

Linguagem	C e PHP
Documentação	Muito Fraca
Comunidade	Fraca
Prós	Possibilidade de associar parsers externos facilmente
Contras	Apesar de possuir features interessantes estas não são necessárias para este trabalho
URL	www.mnogosearch.org/

SWISH-E

Linguagem	C e Perl
Documentação	Apenas para pesquisa
Comunidade	Grande
Prós	Leque de funcionalidades interessantes
Contras	Têm diversos problemas com volumes de dados maiores que 2Gb
URL	http://swish-e.org

Xapian

Linguagem	C++
Documentação	Excelente (doxygen)
Comunidade	Grande
Prós	Boa escabilidade. Afirmam que em índices de 1.5 Tb fazem pesquisas em menos de 1 segundo; Existência de outras funcionalidades interessantes como “Simultaneous search and update” que pode ser bastante útil para o projecto.
Contras	Alguns estudos afirmam um suposto mau desempenho em indexação
URL	http://xapian.org

Beagle

Linguagem	C#
Documentação	Muito Fraca
Comunidade	Pequena
Prós	Suporta uma grande quantidade de documentos
Contras	Vários problemas relatados com falta de performance
URL	www.gnome.org/projects/beagle

Wumpus

Linguagem	C++
Documentação	Muito Fraca
Comunidade	Pequena
Prós	Possui suporte para alguns dos formatos de ficheiros mais conhecidos..
Contras	Problemas de compilação na versão mais recente do GCC (gcc 4.3.2)
URL	www.wumpus-search.org

Zebra

Linguagem	C
Documentação	Muito Fraca
Comunidade	Média
Prós	Bastante eficiente na versão 1.x e 4 a 5 vezes mais rápido na última versão
Contras	Não possui API, qualquer alteração tem de ser feita na sua source.
URL	www.indexdata.dk/zebra

A.2 Benchmarks

Nesta secção, serão listados os tempos obtidos no *benchmark* efectuado para a indexação de diferentes amostras e bibliotecas.

A.2.1 Amostra interna da Portugalmail

Esta amostra consiste num *dataset* de 8148 emails que são usados regularmente nos testes internos da Portugalmail.

Lucene (sem optimizador)

	Time 1	Time 2	Time 3	Time 4	Time 5
real	0m2.853s	0m2.623s	0m2.596s	0m2.630s	0m2.539s
user	0m3.172s	0m3.128s	0m3.132s	0m3.052s	0m2.824s
sys	0m0.260s	0m0.332s	0m0.276s	0m0.264s	0m0.228s
index size	5.2mb				

Lucene (com optimizador)

	Time 1	Time 2	Time 3	Time 4	Time 5
real	0m2.435s	0m2.888s	0m2.588s	0m2.553s	0m2.770s
user	0m2.792s	0m3.332s	0m3.068s	0m3.080s	0m3.128s
sys	0m0.284s	0m0.288s	0m0.284s	0m0.236s	0m0.196s
index size	5.2mb				

Xapian (sem optimizador)

	Time 1	Time 2	Time 3	Time 4	Time 5
real	0m32.144s	0m32.123s	0m32.359s	0m32.368s	0m32.093s
user	0m30.926s	0m30.946s	0m30.994s	0m31.086s	0m30.926s
sys	0m1.008s	0m0.928s	0m0.880s	0m0.872s	0m0.992s
index size	154Mb				

A.3 Lucene Ports

De seguida serão listados todos os *ports* de Lucene que foram avaliados para este projecto.

Lucy

Linguagem	C, Perl e Ruby
Documentação	Muito Fraca
Comunidade	Razoável
URL	http://lucene.apache.org/lucy/

pyLucene

Linguagem	Python
Documentação	Boa
Comunidade	Razoável
Contras	Não é autónoma (utiliza uma *java virtual machine* embebida)
URL	http://lucene.apache.org/pylucene/

Clucene

Linguagem	C++
Documentação	Muito Boa
Comunidade	Pequena
Prós	Eficiência
Contras	Desactualizada (p.e não compila no mais recente g++)
URL	http://clucene.wiki.sourceforge.net/

Ferret

Linguagem	C e Ruby
Documentação	Muito Boa
Comunidade	Grande
Prós	Existência de um livro sobre a ferramenta.
URL	http://ferret.davebalmain.com/

Plucene

Linguagem	Perl
Documentação	Boa
Comunidade	Pequena
Contras	Problemas de desempenho
URL	http://search.cpan.org/dist/Plucene/

Lucene.Net

Linguagem	C#
Documentação	Muito Boa
Comunidade	Grande
Contras	Dependência da framework .Net
URL	http://incubator.apache.org/lucene.net/

Lucene4c

Linguagem	C
Documentação	Boa
Comunidade	Pequena
Contras	Ainda não existe suporte para indexação (apenas pesquisa).
URL	http://incubator.apache.org/lucene4c/

ZendFramework

Linguagem	C++
Documentação	Muito Boa
Comunidade	Razoável
Prós	Fácil integração
Contras	Problemas de desempenho
URL	http://framework.zend.com/

Mutis

Linguagem	Delphi
Contras	Problemas terminou
URL	http://sourceforge.net/projects/mutis

A.4 Solr Schema

Passar o Schema todo para aqui com as minhas alterações

```
<fields>
 <field name="id" type="string" indexed="true" stored="true" required="true" />
 <field name="uid" type="slong" indexed="true" stored="true" required="true" />
 <field name="uidv" type="long" indexed="true" stored="true" required="true" />
 <field name="box" type="string" indexed="true" stored="true" required="true" />
 <field name="user" type="string" indexed="true" stored="true" required="true" />
 <field name="ns" type="string" indexed="true" stored="true" required="false" />
 <field name="last_uid" type="boolean" indexed="true" stored="false" />
 <field name="hdr" type="text" indexed="true" stored="false" />
 <field name="body" type="text" indexed="true" stored="false" />
 <field name="any" type="text" indexed="true" stored="false" multiValued="true" />
 <dynamicField name="attach_*" type="text" indexed="true" stored="true"
        required="false" />
 <field name="any_attach" type="text" indexed="true" stored="false"
        multiValued="true" required="false" />
</fields>

<copyField source="hdr" dest="any" />
<copyField source="body" dest="any" />
<copyField source="attach_*" dest="any_attach" />
<uniqueKey>id</uniqueKey>
```

A.5 Benchmark da Pesquisa do Sistema Final

Script Utilizado

```
 #!/bin/bash
echo "" > results.log
for C in 2 4 8 16 32 64 128 256 512
do
N=$(($C*1000))
echo "ab -n$N -c$C" >> results.log
ab -n$N -c$C 'http://localhost:8080/solr/select?q=Rui&start=0&rows=20' >> results.log
done
```