
# RELATÓRIO DE ESTÁGIO

**sobre**

# Tarântula – Sistema de Recolha de Documentos da Web

**realizado no**

## LASIGE – Laboratório de Sistemas Informáticos de Grande Escala

**por**

**Daniel Coelho Gomes**

**Lisboa, Julho de 2001**

**Universidade de Lisboa**

**Faculdade de Ciências**


**Relatório Final**

**Sobre**

# Tarântula – Sistema de Recolha de Documentos na WWW

**realizado no**

**LASIGE – Laboratório de Sistemas Informáticos de Grande Escala**

**Por**

**Daniel Coelho Gomes**

dcgomes@xldb.fc.ul.pt

*Coordenador* **FCUL:** ***Professor Pedro Antunes***

*Coordenador* **LASIGE:** ***Professor Mário Silva***

**Responsável pela FCUL: Professor Pedro Antunes**

**Responsável pelo Lasige: Professor Mário Silva**

## Lisboa, Julho de 2001

## Declaração

Daniel Coelho Gomes, aluno nº 21369 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Estágio Profissionalizante, intitulado Tarântula - Sistema de Recolha de Documentos na WWW , realizado no ano lectivo de 2000/2001 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, de Julho de 2001

_____

_____, supervisor do estágio profissionalizante de _____, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do estagiário realizado, intitulado _____.

Local, de Julho de 2001

_____

## Resumo

Este relatório descreve o estágio profissionalizante da Licenciatura em Informática da Faculdade de Ciências da Universidade de Lisboa, realizado no grupo XLDB do Laboratório de Sistemas Informáticos de Grande Escala.

O relatório, descreve como foi desenvolvido um novo módulo de recolha de informação da *Web*, denominado Tarântula, com características de configuração, integração e escalabilidade que permite que seja incluído como módulo de outros sistemas.

O Tarântula foi integrado em dois projectos em desenvolvimento na unidade de investigação, o TUMBA e o DROP:

- O TUMBA é um projecto de investigação que consiste no desenvolvimento de um motor de busca de documentos na WWW.
- O DROP é um sistema de recolha e armazenamento de publicações on-line, para depósito digital na Biblioteca Nacional, à semelhança do depósito legal que é efectuado para as publicações tradicionais impressas em papel.

# Índice

# Índice de Figuras

# 1 Introdução

Este relatório descreve o trabalho desenvolvido no âmbito do estágio profissionalizante da Licenciatura em Informática da Faculdade de Ciências da Universidade de Lisboa, que decorreu no grupo de investigação XDLB [62] do Laboratório de Sistemas Informáticos de Grande Escala [26].

Durante o estágio estive enquadrado numa equipa de Investigação e Desenvolvimento na área de Sistemas de Informação. O trabalho desenvolvido incidiu na análise e desenvolvimento de um módulo de recolha de documentos disponíveis na WWW. Este tipo de sistemas são vulgarmente designados por aranhas (*spiders*), pela forma como percorrem a teia (*World Wide Web*). O módulo de recolha (ou *spider*) desenvolvido durante o estágio apresenta características de adaptabilidade a diversas situações, pelo que foi baptizado de Tarântula [8], uma aranha que existe tanto em desertos como em florestas de todo o mundo e cuja longevidade é a maior das aranhas.

O Tarântula foi integrado em dois projectos que se encontram em desenvolvimento na unidade de investigação, o projecto de investigação TUMBA e o DROP desenvolvido para a Biblioteca Nacional.

## 1.1 Motivação

Nos últimos anos tem-se dado uma enorme vulgarização da utilização da Internet como forma privilegiada de comunicação e ferramenta de trabalho ou lazer. Esta vulgarização fez com que as dimensões da *World Wide Web* aumentassem também, existindo cada vez mais documentos on-line, uma vez que cada utilizador é hoje em dia um potencial publicador.

Em paralelo com a vulgarização da Internet na sociedade em geral, a comunidade cientifica e a industria informática empenham-se em criar novos sistemas que permitam gerir de alguma forma e tirar proveito de toda esta informação. Neste contexto, têm sido desenvolvidos, num processo que tenta acompanhar o crescimento da *web*, os sistemas de recolha, vulgarmente conhecidos como *robots*, *spiders* ou *crawlers*. Estes sistemas têm sido incluídos como subsistema em aplicações de diversas naturezas, principalmente nos motores de busca.

Convenciona-se hoje classificar os motores de busca em duas grandes categorias: convencional e especializado.

Os motores de busca convencionais actuam sobre grandes quantidades de documentos, recolhidos em massa da Internet. Face a esta grande quantidade de informação torna-se difícil obter resultados de pesquisa com a qualidade desejada [30].

Os motores de busca especializados em áreas temáticas [34,45], têm como objectivo a obtenção de maior relevância nos resultados de pesquisa. Estes actuam sobre colecções de documentos com características comuns, permitindo muitas vezes fazer análise semântica dos conteúdos ou da estrutura das páginas. O espaço de busca é assim reduzido e consequentemente os resultados devolvidos ao utilizador, com a vantagem de, à partida, serem todos da sua área temática de interesse. Os motores de busca especializados implicam que a recolha de documentos da WWW se faça respeitando restrições, de modo a só serem recolhidos documentos contendo informação relevante no contexto da área temática do motor de busca.

As abordagens apresentadas para a construção de motores de busca, apresentam um conjunto de características peculiares a cada uma delas. No entanto, independentemente da abordagem tomada para a construção de um motor de busca de documentos na WWW, é indispensável dispor de um módulo de recolha de documentos, que normalmente é desenvolvido especificamente para a aplicação a que se destina.

O desenvolvimento de um módulo de recolha ou *spider* pode parecer inicialmente um projecto simples, no entanto, a sua complexidade aumenta ao depararmos com a diversidade de protocolos, tipos de ficheiros ou sistemas de segurança que encontramos na Internet. As maiores dificuldades são devidas ao crescente desrespeito pelas normas estabelecidas, que obrigam a que o sistema seja tolerante a este tipo de situações e ao mesmo tempo robusto a informações erróneas, induzidas voluntariamente na Internet por indivíduos mal-intencionados com o objectivo de sabotar o funcionamento destes sistemas.

Surge assim, o interesse em sistemas como o Tarântula, capazes de serem integrados em várias aplicações *Web* e que realizem um conjunto de funções complexas que são comuns a esta variedade de aplicações.

## 1.2 Objectivos

O projecto de estágio destinou-se a criar um protótipo de um módulo de recolha genérico de documentos na WWW, com características de integração, configuração e escalabilidade que permitam que seja integrado facilmente como subsistema de uma aplicação, independentemente da sua natureza ou características particulares.

Este sistema desenvolvido, denominado Tarântula, guarda informação relativa aos documentos recolhidos permitindo a sua visualização e tratamento, consoante a finalidade a que se destinam no sistema em que se integram. As acções do módulo de recolha podem ser sempre controladas e monitorizadas, sendo possível interrompê-las e retomá-las de forma simples e eficiente.

O Tarântula foi integrado como módulo de recolha em dois projectos com objectivos e requisitos distintos, o TUMBA e o DROP:

O projecto DROP tem como objectivo desenvolver um sistema informático que permita a recolha, armazenamento duradouro e posterior consulta de um conjunto limitado e bem definido de publicações electrónicas on-line. A recolha é realizada de forma selectiva, para garantir a compilação das publicações especificadas como fazendo parte da colecção, evitando recolher publicações não interessantes. O sistema apresenta requisitos que colocam um grande desafio à sua realização, uma vez que se pretende guardar as colecções compiladas por um período de tempo teoricamente ilimitado, permitindo a sua constante consulta. O módulo de recolha de DROP tem a função de fazer uma recolha selectiva de publicações on-line, constituídas por conjuntos de documentos definidos pela Biblioteca Nacional. Estas recolhas de publicações, que são em alguns casos periódicas, depois de bibliotecários verificarem o seu interesse histórico, serão incluídas no sistema de armazenamento de publicações online da Biblioteca Nacional.

O TUMBA é um projecto de investigação do grupo XLDB para desenvolvimento de um motor de busca que, recorrendo a novas técnicas de recolha e indexação de documentos da WWW, venha a construir uma base de dados pesquisável da *Web* Portuguesa, permitindo aos utilizadores obter resultados com um grau de relevância superior ao oferecido pelos motores de busca existentes no mercado nacional. O módulo de recolha terá a tarefa de recolher a maior quantidade de documentos possível num intervalo de tempo

relativamente curto, para que sejam construídos índices, que permitam a pesquisa de termos em documentos da WWW.

## 1.3 Notação adoptada e terminologia

A notação adoptada neste documento determina que os termos que representam entidades/conceitos nos modelos elaborados são apresentados a itálico. Os termos originários da língua inglesa são traduzidos para português, ou representados a itálico caso não seja aplicável a tradução.

Na apresentação do trabalho desenvolvido na concepção do Tarântula são empregues termos que podem ser de interpretação subjectiva. O leitor interpreta um termo, consoante o sentido que lhe é mais familiar. Para um profissional de Informática, as palavras recolha ou colecção de documentos, podem ser consideradas sinónimos. Por sua vez, para um bibliotecário, estas palavras têm significados técnicos completamente distintos.

Para desambiguar este tipo de situações, apresento em seguida a terminologia empregue utilizada pelo nosso grupo de trabalho.

***Configuração de Publicação:*** *conjunto de predicados definidos por um utilizador do Tarântula, que exprimem restrições sobre os conteúdos disponíveis na WWW.*

***Publicação:*** *conjunto de conteúdos referenciados por URLs que respeitam uma Configuração de Publicação.*

Nos meios de comunicação tradicionais, o conteúdo de uma publicação é delimitado pelo seu meio de suporte físico, vulgarmente papel. Não é difícil descobrir quais são os artigos que compõem a edição de um determinado jornal, basta examinar o conteúdo das páginas onde está impresso.

Na WWW, porém, a definição de uma publicação é complexa, dada a variedade de formas de apresentação de informação e não existir uma norma para a definição do conceito de publicação. Vulgarmente, identificam-se as publicações online com o sítio onde estão disponíveis. No entanto, encontramos facilmente na WWW, publicações online cujo conteúdo está disponível em diversos sítios [39] e outras que contemplam apenas uma parte de um sítio [18]. Em última análise, qualquer

documento disponível na WWW, pode ser considerado uma publicação, quer seja uma simples página pessoal ou um jornal on-line.

Uma vez que não existe uma definição de publicação na WWW e o seu estudo ultrapassa o âmbito deste projecto, apresento um conceito lato de publicação na WWW, que permite ao utilizador do Tarântula definir os conteúdos que compõem uma publicação, consoante o seu critério.

***Item:*** *conjunto de documentos que compõem uma Publicação durante um determinado intervalo de tempo.*

Quando uma edição de um jornal é impressa, o seu conteúdo pode ser consultado por um período de tempo que depende exclusivamente da preservação do meio de suporte da impressão. Na WWW, o conceito de edição é vago, os conteúdos são disponibilizados durante um intervalo de tempo relativamente curto. Muitas publicações da *Web* são versões digitais de edições publicadas noutros media. O item tenta aproximar-se do conceito de edição transportado para a WWW. O utilizador através da configuração dos parâmetros que definem uma publicação, pode especificar um intervalo de tempo durante o qual, considera que os conteúdos de uma publicação, apresentam características comuns que justifiquem que sejam armazenados sob uma mesma unidade, o item.

***Documento:*** *conteúdo extraído da WWW, a partir de um URL.*

Na prática, trata-se da cópia do conteúdo de um URL, efectuada através do protocolo HTTP [58].

***Documento base****: O documento a partir do qual se inicia o processo de recolha de uma publicação, é denominado documento base.*

***Nome de Servidor:*** *URL atribuído à máquina que disponibiliza um conjunto de documentos na WWW.*

Na Internet, uma máquina é identificada pelo seu endereço IP [7]. No entanto, esta identificação é normalmente desconhecida do utilizador da WWW e os servidores *Web* identificados por um determinado IP, por vezes apresentam conteúdos diferentes consoante o nome utilizado para referir o servidor nos pedidos HTTP. Assim sendo, para uma maior aproximação do utilizador e simplicidade do sistema, considero que cada nome de servidor diferente encontrado num URL, corresponde a um servidor Web diferente, o que na realidade nem sempre acontece.

***Recolha ou colecção:*** *processo através do qual o sistema extrai e armazena um conjunto de documentos da WWW.*

As recolhas do Tarântula podem ser feitas a três níveis: item, servidor e documento, que serão detalhadas ao longo do capítulo 3.

***Tarefa:*** *conjunto de dados do sistema que especificam o processo de recolha de um item*

Conceptualmente, as tarefas correspondem às unidades de trabalho do processo de recolha.

***Tipo de documento:*** *meta-dado do documento que determina a forma do seu conteúdo.*

O Tarântula identifica o tipo de um documento, através do seu tipo MIME [15]. Esta informação é extremamente útil, pois permite uma escolha selectiva de documentos durante a recolha e a sua interpretação por *software* adequado, depois de recolhidos.

***Meta-dados:*** *dados relativos à execução de uma recolha*

Durante o processo de recolha o Tarântula recolhe e gera informação relativa às suas acções e aos documentos que recolhe, estes meta-dados são de grande importância para a análise da WWW e monitorização das recolhas.

## 1.4 Estrutura do Relatório

O relatório está organizado em 4 temas: enquadramento do estágio, descrição do sistema realizado, resultados e conclusões.

O enquadramento do estágio engloba os capítulos 1 e 2. No capitulo 1 é feita a introdução ao projecto desenvolvido durante o estágio. No capitulo 2 é apresentado o contexto tecnológico e profissional em que o mesmo decorreu.

A descrição do Tarântula é feita no capítulo 3, apresentando os casos de uso possíveis na utilização do sistema, um cenário de utilização do sistema, a descrição da arquitectura e finalmente a implementação escolhida para cada um dos componentes do sistema.

Os resultados do desenvolvimento do sistema são descritos no capítulo 4, que descreve as aplicações do sistema, o seu desempenho, pontos de congestão encontrados e apresenta os resultados do desenvolvimento em termos do sistema realizado e profissionais.

As conclusões incluem os capítulos 5 e 6. O capitulo 5 analisa a metodologia adoptada e a calendarização inicialmente proposta. No capítulo 6 apresento um sumário critico do trabalho realizado e as linhas de acção de trabalho futuro para melhoramento e evolução do sistema.

# 2 Contexto do trabalho

## 2.1 Integração na Instituição

O LaSIGE [26] (Laboratório de Sistemas Informáticos de Grande Escala) é uma unidade de investigação do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa, financiada pela Fundação da Ciência e Tecnologia [19]. A unidade é composta por professores e investigadores que se dividem em grupos com diferentes áreas de trabalho. Os seus principais objectivos são a realização de trabalhos de investigação no Departamento de Informática, formação e transferência de tecnologia nas áreas especificas de acção do LaSIGE.

O estágio foi realizado num dos grupos de trabalho que integram o LaSIGE, o XLDB [62], cujas linhas de acção se centram em sistemas de informação, com ênfase sobre a gestão em grande escala de dados distribuídos na Internet.

Para o desenvolvimento do projecto de estágio fui integrado numa equipa composta por quatro elementos, dois alunos de Mestrado, um Professor e um estagiário, que concentram esforços no desenvolvimento dos módulos que compõem os projectos DROP e TUMBA em desenvolvimento, na unidade. Durante o estágio desempenhei fundamentalmente funções de análise e programação dos módulos de recolha de informação destes dois sistemas.

## 2.2 Plano de desenvolvimento

O desenvolvimento do projecto dividiu-se nas seguintes grandes tarefas, realizadas seguindo uma metodologia de desenvolvimento em espiral [11] recorrendo a protótipos.

1. Análise de requisitos dos sistema a desenvolver
2. Elaboração do plano de estágio
3. Desenvolvimento do 1º protótipo - DROP
4. Desenvolvimento do 2º protótipo - DROP
5. Integração no projecto TUMBA

6. Desenvolvimento do sistema final - DROP

7. Elaboração do relatório final de estágio

8. Preparação e apresentação dos resultados do estágio profissionalizante

## 2.3 Sistemas de Recolha de Informação da *Web*

Um sistema de recolha de informação da *Web*, vulgarmente designado por *crawler*, é um componente de *software* conceptualmente simples, existente praticamente desde o início da Internet [6,35,12]. A sua actividade consiste num processo iterativo [24] de recolha de documentos da *Web*. Este processo inicia-se a partir de um conjunto de URLs, inseridos no sistema a partir de uma entidade externa. A partir destes URLs iniciais, é feita a recolha do conteúdo dos documentos por eles referidos. Segue-se uma análise dos conteúdos, para extracção das ligações para novos documentos (URLs) neles contidas, que serão por sua vez recolhidos na próxima iteração.

A complexidade de um *crawler* pode variar muito, consoante os requisitos do sistema onde estará integrado. Um *crawler* pode ser uma simples aplicação com carácter académico [5] ou um elaborado projecto de investigação [9,20]. Nos últimos anos, o investimento no desenvolvimento de *crawlers* cada vez mais sofisticados, tem sido feito principalmente pelo mercado dos motores de busca mundiais, onde são componentes cruciais. No entanto, o aumento dos investimentos não se tem reflectido, de forma proporcional, na evolução dos sistemas de *crawling*. Devido às características competitivas do mercado dos motores de busca, existem poucos artigos publicados sobre o assunto sendo normalmente bastante vagos, não constituindo uma base de conhecimento suficiente para o desenvolvimento de novos *crawlers*.

### *2.3.1 Requisitos de um crawler*

Os *crawlers* usados nos motores de busca mundiais têm requisitos de desempenho muito exigentes, uma vez que se pretende que façam recolhas de documentos à escala da rede mundial, num espaço de tempo relativamente curto.

Embora se pretenda que o *crawler* recolha a maior quantidade de documentos possível, esta não pode ser feita de forma indiscriminada, devendo respeitar normas de bom comportamento [28], tais como:

- Identificar o *crawler*, usando os campos disponibilizados pelo protocolo HTTP para esse fim;
- Não sobrecarregar servidores *Web*, evitando pedidos simultâneos ou sequenciais a um mesmo servidor;
- Não visitar servidores ou partes de servidores que não pretendam ser visitados por *crawlers*, respeitando o protocolo de exclusão de *robots* (*Robot Exclusion Protocol*) [29].

O *crawler* ao mesmo tempo que se mostra cordial deve ser robusto a situações nefastas para o seu desempenho, como por exemplo:

- Evitar recolher documentos repetidos.
- Não recolher informação fútil para os propósitos do sistema.
- Evitar *spider traps*. As *spider traps* são URLs que fazem com que o *crawler* faça recolhas infinitas num dado sítio da *Web*. As *spider traps* podem ser criadas por acidente, mas muitas vezes são propositadas. As motivações dos criadores das *spider traps* variam, podem ser criadas por puro vandalismo ou como retaliação contra *crawlers* usados pelos *Internet Marketeers*, que procuram endereços de *email* para posterior *spamming* de publicidade. Existem também sítios da *Web* onde são criadas as *traps*, para apanharem os *crawlers* dos motores de busca, de modo a aumentarem o *rating* de acesso ao sítio.

Um *crawler* deverá manter, de forma eficiente e acessível, informação acerca das suas acções de recolha de documentos na *Web*. Esta informação destina-se a permitir a monitorização permanente das suas acções e uma análise estatística da WWW, ou parte dela, à posteriori. A monitorização das acções do *crawler* é crucial para a sua operação, uma vez que dada a vastidão e diversidade da WWW, é impossível prever ou testar todas as situações que possam surgir. Por exemplo, o *crawler* do Google tentou recolher um jogo on-line [44]. Embora este problema tenha tido uma solução simples, só foi detectado após a recolha de dezenas de milhões de documentos da WWW.

### *2.3.2 Requisitos do Tarântula*

Os *crawlers* desenvolvidos para motores de busca mundiais, uma vez que se destinam a serem escaláveis para milhares de milhões de documentos, são desenvolvidos tendo em vista principalmente a sua capacidade de recolha em termos quantitativos. O Tarântula não apresenta requisitos quantitativos tão exigentes mas apresenta fortes requisitos de adaptabilidade e extensibilidade, uma vez que se destina a poder ser integrado em sistemas com diferentes finalidades.

Pretende-se que o Tarântula seja um *crawler*:

- escalável, possibilitando a expansão da sua capacidade de recolha a qualquer conjunto de domínios da *Web* Portuguesa, a um baixo custo.
- configurável, permitindo selectividade em relação às recolhas, face à constante mudança de tipos e estrutura de informação na WWW.
- integrável, apresentando características que permitam que seja integrado facilmente como subsistema de aplicações que necessitem de capturar periodicamente conjuntos de páginas *Web*, para vários tipos de tratamento de informação.

## 2.4 Tecnologias e ferramentas utilizadas

No decurso do estágio tive de recorrer a várias ferramentas e tecnologias para proceder ao desenvolvimento do Tarântula.

### *2.4.1 UML*

O UML [40] (*Unified Modeling Language*), é uma linguagem para especificação, visualização, construção e documentação de componentes de *software* orientados por objectos. A especificação do UML deriva da fusão de três metodologias de modelação de sistemas (Booch [1], OMT [3] e OOSE [21]), acrescentado ainda funcionalidades que nenhum dos modelos possuía.

O OMG (*Object Management Group*) [36] adoptou o UML, como norma para análise e desenho de aplicações orientadas por objectos, tendo como objectivo a redução da entropia na utilização de linguagens de modelação existente na industria Informática.

Existem actualmente no mercado várias ferramentas que auxiliam o desenvolvimento de *software* recorrendo ao UML. Neste projecto foram utilizadas duas destas ferramentas: o ArgoUML [10] e o Microsoft Visio [31], para criação dos modelos elaborados na análise e desenho do Tarântula.

### *2.4.2 HTML*

O HTML [60] (*HyperText Markup Language*) é a linguagem de publicação mais utilizada na *World Wide Web*. Trata-se de um formato não proprietário baseado no SGML (*Standard Generalized Markup Language*) [59], podendo ser criado a partir de uma grande variedade de ferramentas, desde um simples editor de texto até sofisticadas ferramentas de geração de código [27]. O HTML usa *tags* como <h1> e </h1> para estruturar texto em cabeçalhos, parágrafos, links, etc.

Para a geração de código HTML recorri ao Macromedia Dreamweaver 3 [27], que é um editor gráfico para criação e gestão de páginas e sítios *Web* que fornece sofisticadas ferramentas de desenho, assim como funcionalidades que facilitam o uso de HTML dinâmico. Previnem também problemas comuns com os *browsers* e plataformas mais populares. O *Dreamweaver* é altamente configurável, permitindo criar objectos, comandos, modificar menus, atalhos de teclado e até escrever código *Javascript* [55] para estender as funcionalidades da aplicação.

### *2.4.3 SQL*

Para manipulação dos dados mantidos no Tarântula recorri à linguagem de interrogação a bases de dados SQL (Structured Query Language) [42], que foi adoptada como standard da industria em 1986, posteriormente revista nas normas SQL-92 e SQL-99.

O *PostgreSQL* [37] é um sofisticado Sistema de Gestão de Bases de Dados relacional com objectos que suporta a quase totalidade das construções de SQL, incluindo sub-selects e funções ou tipos definidos pelo utilizador. O PostgreSQL é considerado o mais sofisticado sistema de gestão de base de dados disponivel em regime de *open-source*.

### *2.4.4 Java*

O Java [52] é um ambiente e ao mesmo tempo uma linguagem de programação de alto nivel, produzido pela Sun Microsystems, Inc. Trata-se de um dos mais importantes representantes da nova geração de linguagens orientadas por objectos e foi projectado para resolver os problemas de programação baseados no paradigma cliente-servidor [4].

A arquitectura do Java consiste resumidamente no seguinte: as aplicações escritas em Java são compiladas num código de *bytes* independente de arquitectura de HW ou SW. Esse código de *bytes* pode ser executado em qualquer plataforma que possua um interpretador Java (Java Virtual Machine).

O Java pelas características de modularidade e expansibilidade, permite que sejam acrescentadas novas funcionalidades, à configuração convencional do seu kit de desenvolvimento.

No desenvolvimento do Tarântula foi utilizado o kit de desenvolvimento jdk1.3 [46], adicionado dos módulos de desenvolvimento de *Servlets* [48] e JDBC [50] para comunicação com o PostgreSQL. As *Servlets* foram armazenadas no Tomcat e disponibilizadas através do Apache Web Server.

- *Servlets* são programas escritos em Java que aumentam as funcionalidades de servidores pedido/resposta, tais como os servidores *Web* capazes de executar programas Java. Uma *Servlet* reside num contentor existente no servidor *Web* e é executada no servidor a pedido de um cliente, devolvendo como resposta uma página HTML que pode ser a mostrada num *browser*.

- JDBC (*Java DataBase Connectivity*) é uma interface de acesso a bases de dados SQL. Permitindo construir aplicações que utilizem bases de dados, mantendo a independência de API's proprietárias. Normalmente, os fabricantes de SGDBs disponibilizam *drivers* JDBC que perm*Item* a comunicação de aplicações Java com os seus sistemas. No desenvolvimento deste projecto foi utilizada a *driver* jdbc7.0-1.2 para comunicação com o PostgreSQL.

- Apache Web Server [53] é um servidor (*Web*) HTTP, completo, robusto, *open-source* e gratuito. O seu desenvolvimento foi feito por colaboradores voluntários em todo o mundo, que comunicaram entre si através da Internet e da WWW.

- O Tomcat [54] é um contentor de *Servlets* e uma implementação de *Java Server Pages* [47], que pode ser usado como *stand-alone application* ou em conjunção com vários servidores *Web* populares, como é o caso neste projecto, em que é usado conjuntamente com o Apache.

O ambiente de desenvolvimento de aplicações em Java neste projecto, foi o *Forte for Java* [51], tratando-se o próprio de uma aplicação em Java.

### *2.4.5 JLex e JFLex*

Um analisador léxico parte uma sequência de caracteres em *tokens*. Construir um analisador léxico de raiz é uma tarefa penosa, mas muitas vezes necessária. Por isso foram desenvolvidas ferramentas para facilitar esta tarefa, os geradores de analisadores léxicos.

O *JLex* [16] e o *JFlex* [17] são geradores de analisadores léxicos escritos em Java e destinados a aplicações Java. O seu funcionamento baseia-se em receber um ficheiro de especificação e a partir dele construir o analisador léxico correspondente, na forma de uma *class* [52] Java. No inicio do desenvolvimento do Tarântula foi utilizado o Jlex para análise de conteúdos recolhidos no formato HTML, tendo sido depois substituído pelo JFlex v1.3.2, por este gerar analisadores mais rápidos.

### *2.4.6 Robots Exclusion Protocol*

O Robots Esclusion Protocol (REP) [29] é um mecanismo que permite aos servidores Web, indicarem que partes do servidor não deverão ser visitadas por *robots*.

O *robot* deverá estabelecer uma ligação HTTP com o servidor, a fim de recolher o ficheiro robots.txt, que deverá estar disponível na directoria raiz do sítio *Web*. Este ficheiro deverá respeitar a sintaxe do *REP.*, de modo a poder ser interpretado automaticamente pelo *robot*.

Embora o *Robots Exclusion Protocol* não seja uma norma da WWW, é muito utilizado, sendo aconselhado pela W3C [61]. Infelizmente, existem muitos administradores de sítios *Web* que o desconhecem ou que embora criem um ficheiro robots.txt, este não respeita a sintaxe definida pelo REP, existindo casos em que o conteúdo do ficheiro está escrito em linguagem natural, o que torna a sua interpretação por um *crawler* impossível.

# 3 Tarântula

Este capítulo apresenta o Tarântula. Inicialmente apresento os casos de uso do Tarântula, seguindo-se a descrição da arquitectura do sistema e do seu funcionamento. Finalmente discuto em seguida a implementação escolhida para cada um dos componentes do sistema.

## 3.1 Casos de uso

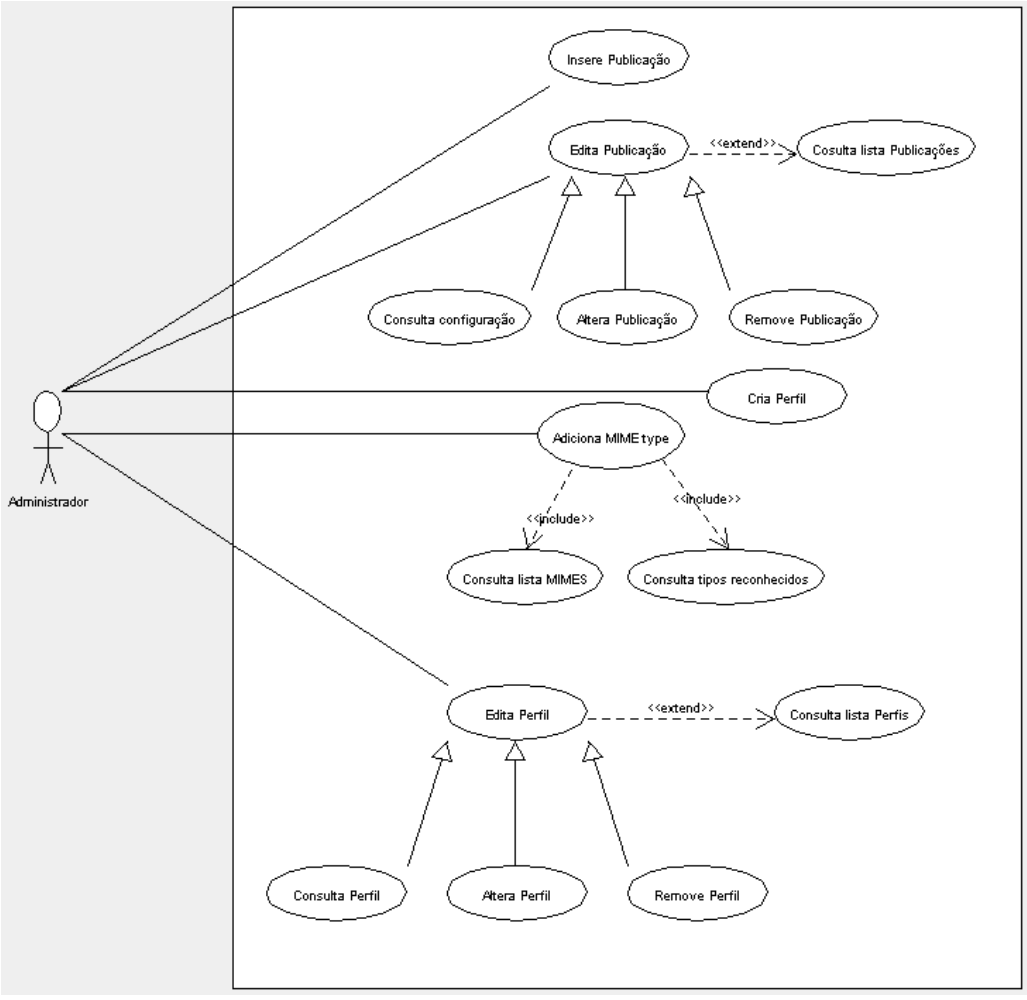



**Figura 3.1 Diagrama de casos de uso – UML. Casos de uso disponibilizados pela interface *web* a um utilizador humano quando este efectua a configuração do sistema.**

A Figura 3.1 representa o diagrama de casos de uso do Tarântula, existem casos de uso para:

- Gestão de informação, relativa à configuração das recolhas a efectuar das várias publicações, dispondo-se de casos para inserir, editar e consultar publicações.

- Gestão de tipos MIME [15] dos conteúdos recolhidos, uma vez que os tipos estão em constante mutação, surgindo novos tipos ao mesmo tempo que outros ficam obsoletos. Assim sendo, é fornecido um caso que permite acrescentar novos tipos ao sistema consoante os tipos de documentos que se pretendem recolher.

- Gestão de perfis de recolha, estes servem para facilitar a definição de configurações de recolha. No sistema, um conjunto de tipos MIME pode ser agrupado num destes perfis. Posteriormente, ao definir a configuração de uma publicação a recolher, em vez de se referir os tipos MIME a processar, indica-se o nome de um destes perfis. O Tarântula disponibiliza casos para inserir, editar e consultar perfis.

## 3.2 Cenário de utilização

Nesta secção apresento um exemplo de utilização para o caso de uso *Insere Publicação* (Figura 3.1), que inicia o processo de recolha de uma dada publicação. Neste exemplo, consideramos o sítio da Faculdade de Ciências da Universidade de Lisboa, incluindo todos os servidores do dominio *fc.ul.pt*, como sendo uma publicação.

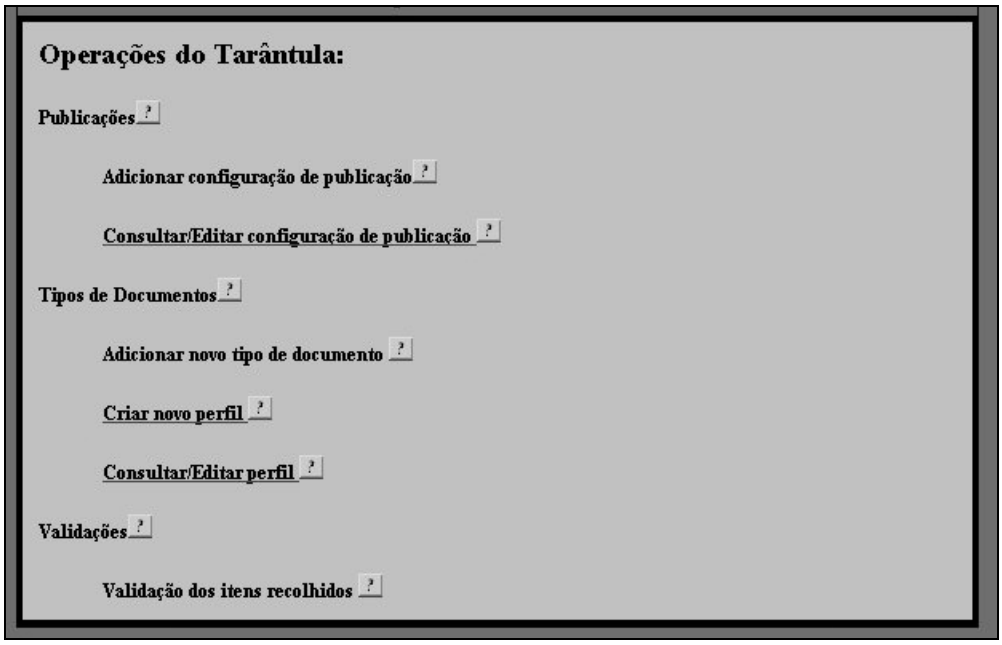



**Figura 3.2. Menu principal de operações do Tarântula. O utilizador pode optar por adicionar uma nova publicação ao sistema, alterar as configurações de uma existente, acrescentar um novo tipo de documento à base de dados de tipos reconhecidos pelo sistema, criar ou editar um *Perfil* de tipos de documentos ou validar os itens recolhidos.**

Na página de operações representada na Figura 3.2, temos as funcionalidades disponibilizadas pelo Tarântula. A operação *Validação dos itens recolhidos*, permite navegar pelos conteúdos dos itens recolhidos e atribuir-lhes um estado de validade. Esta operação embora esteja disponível, não faz parte do núcleo de funcionalidades do Tarântula, uma vez que a sua utilização impõe que tenha sido efectuada a conversão de ligações nos documentos e esta é opcional.

Iniciamos o exemplo de utilização escolhendo no menu principal de operações (Figura 3.2), a opção *Adicionar configuração de publicação*, a fim de definirmos a publicação através da inserção dos seus parâmetros de configuração (Figura 3.3 e Figura 3.4).

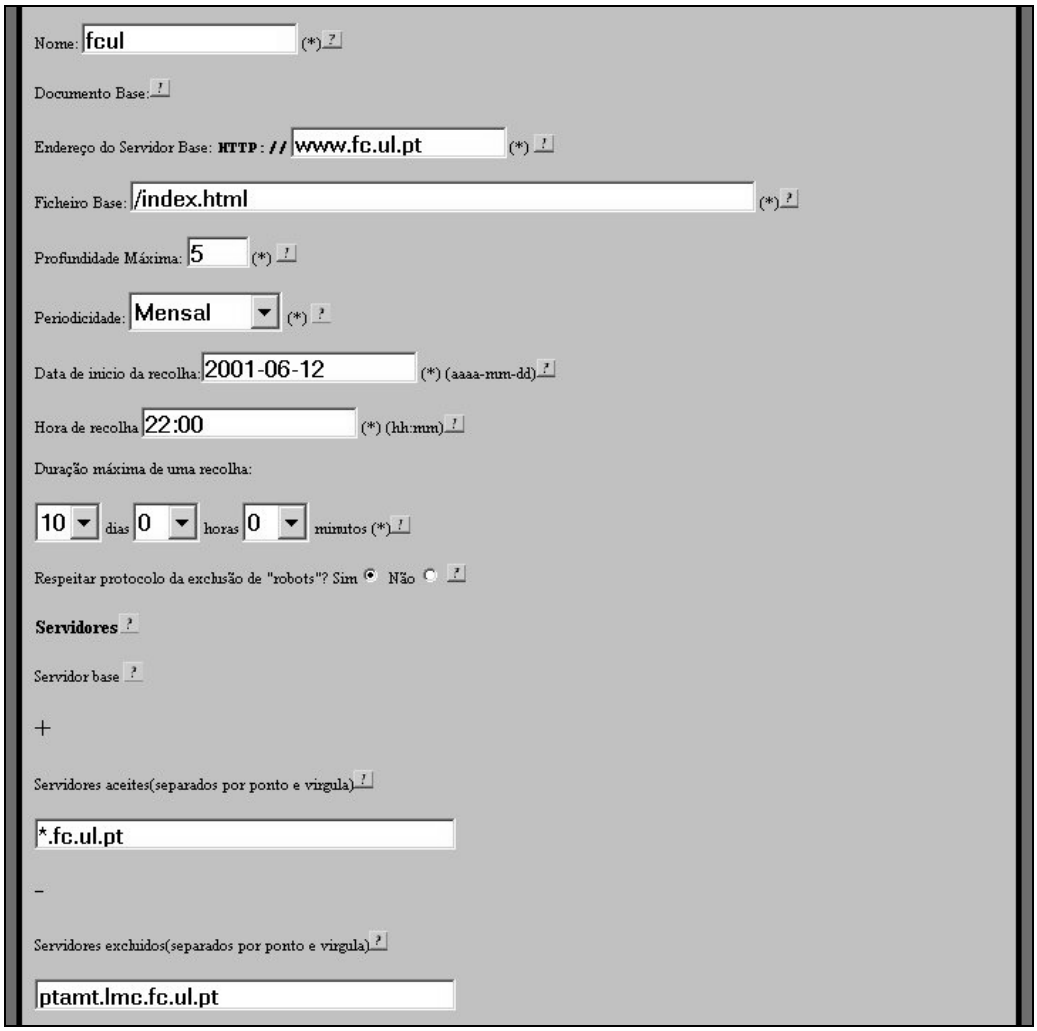



**Figura 3.3. Configuração dos parâmetros que definem a publicação FCUL (1ª parte)**

Os parâmetros definidos na Figura 3.3 determinam que a publicação *fcul* deverá ser recolhida a partir do documento base com o URL: *http://www.fc.ul.pt/index.html* até uma profundidade máxima de *5*. As recolhas da publicação deverão ser efectuadas *mensalmente*, a partir da data de *2001-06-12* pelas *22h00*, a duração máxima do processo de recolha será de *10* dias. Serão aceites documentos provenientes do servidor base (www.fc.ul.pt) e de todos os pertencentes ao domínio *fc.ul.pt*, à excepção do *ptamt.lmc.fc.ul.pt*, devendo ser sempre respeitado o protocolo de exclusão de *robots*.

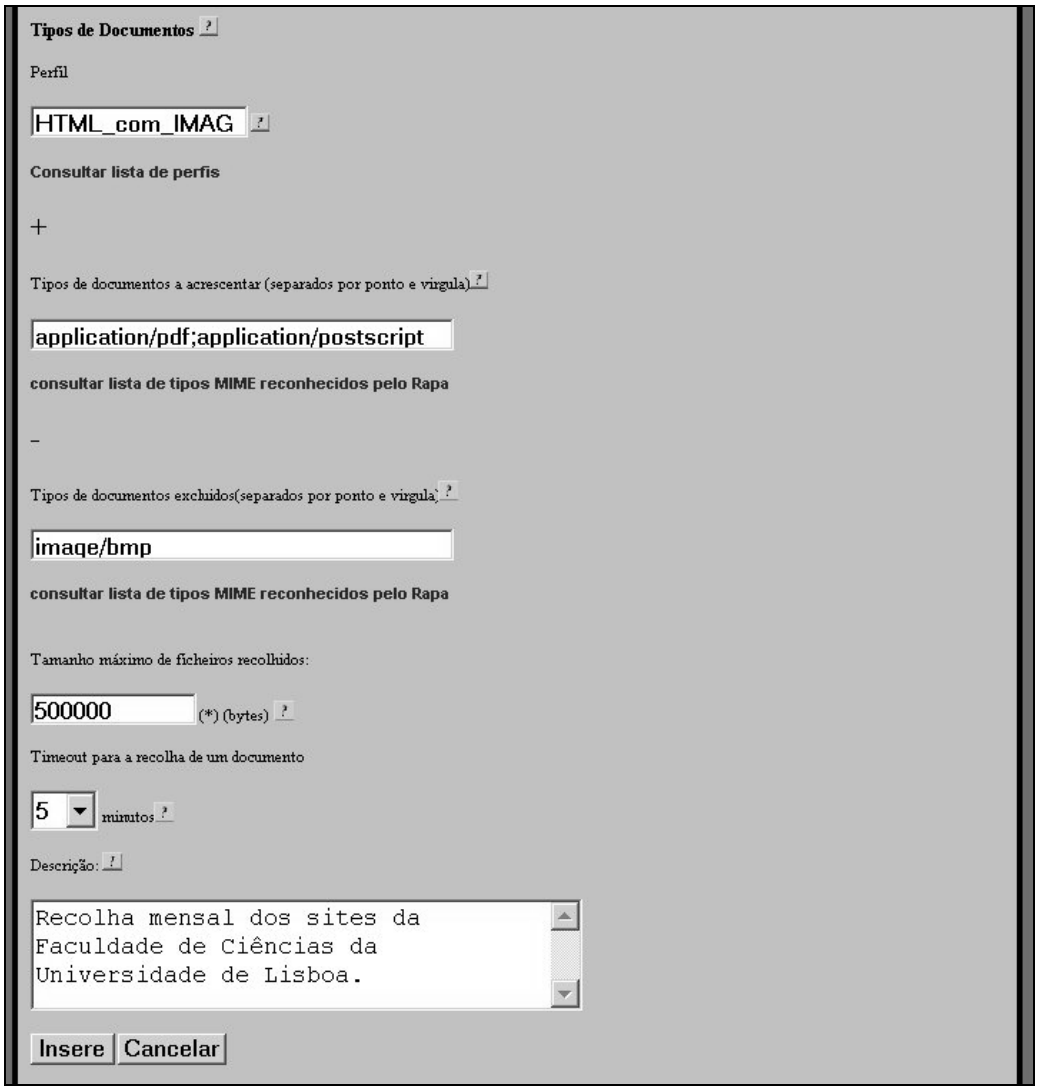



**Figura 3.4. Configuração dos parâmetros que definem a publicação FCUL (2ª parte)**

Os parâmetros definidos Figura 3.4 determinam que os documentos que compõem a publicação, devem ser de um dos tipos contidos no *Perfil HTML_com_IMAGENS* (à excepção do tipo *image/bmp*), acrescido dos tipos *application/pdf* e *application/postscript*. O tamanho de cada documento recolhido não deverá exceder os *500000* bytes (500Kb) e a sua recolha não poderá durar mais do que *5* minutos.

Após finalizar a configuração da publicação, o utilizador deverá clickar no botão *Insere*. Se as configurações publicação forem aceites, a publicação passará

a ser recolhida mensalmente, sendo criada uma directoria para cada item recolhido. O nome da directoria é baseado no nome da publicação e na data de recolha do item, de modo a que seja intuitiva a identificação do seu conteúdo.

## 3.3 Arquitectura

Na Figura 3.5 estão representados os componentes de *software* que compõem o Tarântula que são:

- A *Interface de Configuração e Gestão* que recebe configurações de publicações fornecidas pelos utilizadores.
- A base de *Dados de Configuração das Publicações* que armazena as configurações de publicações fornecidas pelos utilizadores.
- A base de *Dados de Execução* que armazena os dados relativos aos processos de recolha das publicações.
- O *Coordenador* que tem a responsabilidade de analisar periodicamente os *Dados de Configuração das Publicações*, a fim de identificar recolhas de publicações a efectuar. Ao identificar uma recolha, o *Coordenador* agenda uma nova tarefa no *Monitor de Tarefas*, acompanhando a sua evolução até ser dada por terminada.
- O *Monitor de Tarefas* que tem a função de controlar o acesso dos restantes componentes aos *Dados de Execução*, onde são armazenadas as tarefas.
- O *Coleccionador* que consulta periodicamente o *Monitor de Tarefas*, a fim de identificar documentos a recolher. Para efectuar a recolha dos documentos, o *Coleccionador* recorre a dois componentes, o *Interpretador do Protocolo de Exclusão* e o *Extractor de URLs*. Após a recolha de um documento, o *Coleccionador* armazena o seu conteúdo no sistema de ficheiros e actualiza os *meta-dados* no *Monitor de Tarefas*.
- O *Interpretador do Protocolo de Exclusão* é um componente que interpreta as restrições impostas pelos administradores dos sítios da *Web*, à recolha automática de documentos por *robots* (através do REP).
- O *Extractor de URLs*, analisa os conteúdos dos documentos HTML recolhidos e extrai as ligações (URLs) para outros documentos.

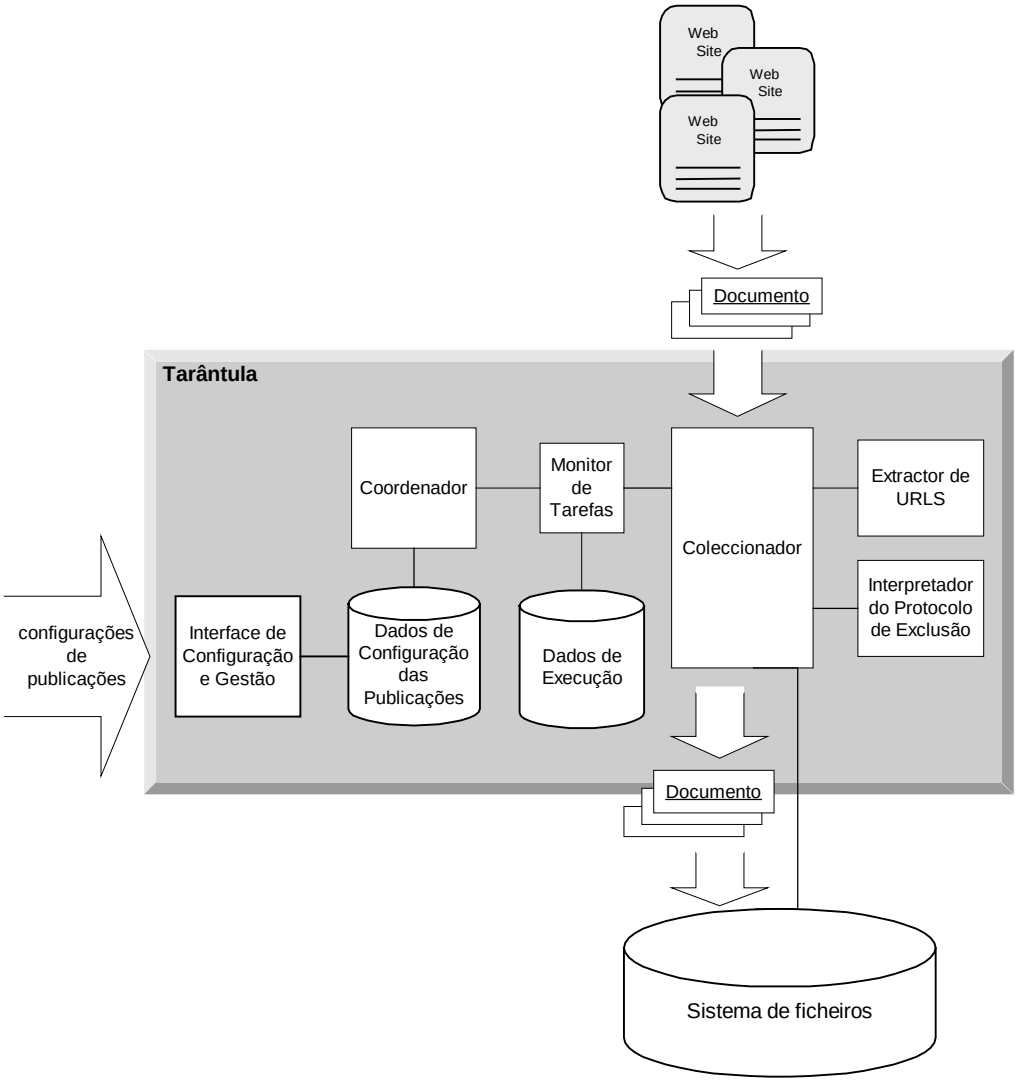



**Figura 3.5. Arquitectura do Tarântula. O sistema recebe configurações de publicações, a partir das quais rege o processo de recolha das publicações, transferindo-o da WWW para o sistema de ficheiros local.**

Existe ainda um componente adicional na Arquitectura do Tarântula, o *Conversor de Links*, que faz a conversão dos *links* entre os documentos de um item. Os documentos recolhidos contêm ligações para outros documentos, feitas através da referência para a sua localização na WWW. A fim de possibilitar uma navegação pelos documentos que compõem um item recolhido, semelhante à que é feita através dos documentos disponíveis na

*Web*, é necessário converter as ligações contidas nos documentos, para que estas referenciem os documentos recolhidos, e não os documentos disponíveis na *Web*.

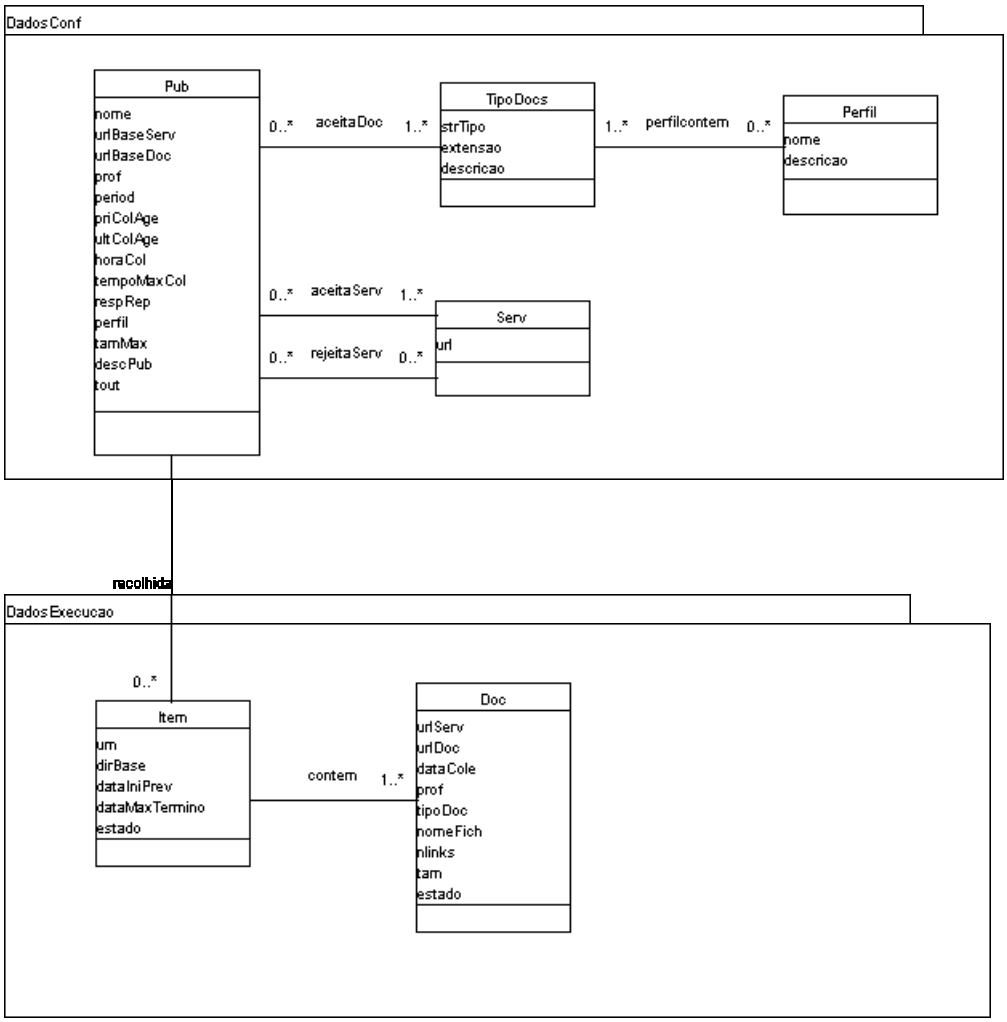



**Figura 3.6 Diagrama de *classes* – UML. Representação do modelo de dados dos *Dados de Configuração das Publicações* e dos *Dados de Execução*.**

O *Conversor de Links* não faz parte do núcleo de componentes do Tarântula, uma vez que se pretende que o Tarântula seja um sistema genérico e a alteração dos links contidos nos documentos, pode não ser desejada nalguns sistemas. Assim sendo, o recurso ao *Conversor de Links* é opcional, consoante as necessidades da aplicação em que o Tarântula se integra.

Nas subsecções que se seguem apresento os detalhes do funcionamento de cada um destes componentes.

### *3.3.1 Dados de Configuração das Publicações*

A informação relativa aos *Dados de Configuração das Publicações* é armazenada segundo o modelo de dados do pacote *DadosConf*, representado na Figura 3.6 (no apêndice encontra-se uma descrição detalhada deste modelo de dados).

Através da *classe Pub*, são guardados os atributos referentes à caracterização de cada publicação. Os seus valores são especificados na totalidade pelo utilizador, à excepção do atributo *ultcolage* (ultima colecção agendada), que é usado exclusivamente pelo sistema a fim de identificar a data da última recolha.

A *classe TipoDocs* representa os tipos de documentos que o sistema reconhece e as extensões de ficheiro que foram associadas a cada um deles pelo utilizador.

A *classe Perfil* representa os nomes (perfis) associados a conjuntos de tipos de documentos, estes perfis visam simplificar a interacção com o utilizador no processo de configuração das publicações através da *Interface de Configuração e Gestão*.

A *classe Serv* armazena URLs de servidores, cujos documentos poderão ser aceites ou rejeitados para uma publicação.

### *3.3.2 Dados de Execução*

Na Figura 3.6, o pacote *DadosExecucao* corresponde aos *Dados de Execução* relativos aos processos de recolha. Os dados das tarefas são representados pelas *classes*:

- *Item*, que representa a informação relativa à recolha de um item, obrigatoriamente associado às configurações de uma publicação.
- *Doc*, que representa a informação relativa aos documentos recolhidos e por recolher de cada item.

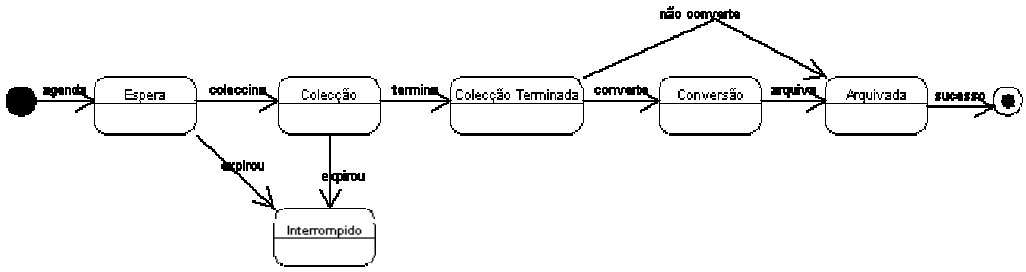



**Figura 3.7. Diagrama de estados da *classe* Item – UML. Sequência de estados de um objecto da classe Item desde o agendamento da tarefa até ao arquivo dos conteúdos recolhidos.**

Quando uma tarefa é agendada pelo *Coordenador*, é criada uma nova ocorrência de *Item*, que descreve o item a ser recolhido, e uma nova ocorrência de *Doc*, que contém o documento base, a partir do qual o *Coleccionador* vai iniciar a recolha.

Cada objecto de *Item* encontra-se inicialmente no estado de *Espera* (Figura 3.7), do qual transita quando for recolhido o documento base, passando a estar em *Colecção*. Quando não existirem mais documentos a recolher para o item, este passa para o estado de *Conversão*, durante o qual, é feita a conversão dos links contidos nos documentos HTML, a fim de permitir navegação pelos documentos do item. Finda a conversão, o item passa para o estado *Arquivado*. Em alternativa, o item pode ser imediatamente arquivado sem ser feita a conversão dos links dos documentos.

Cada item tem um prazo de validade definido pelos dados de configuração da publicação correspondente, durante o qual a recolha deve ser feita. Este prazo visa garantir que recolhas periódicas não se sobreponham no tempo. Caso o *Coordenador* detecte que a recolha de um item excedeu o seu prazo de validade, esta é interrompida, passando o item para o estado *Interrompido.* Os documentos recolhidos até à data da interrupção são mantidos, cabendo ao utilizador decidir o seu destino, podendo optar por estender o prazo de recolha do item.

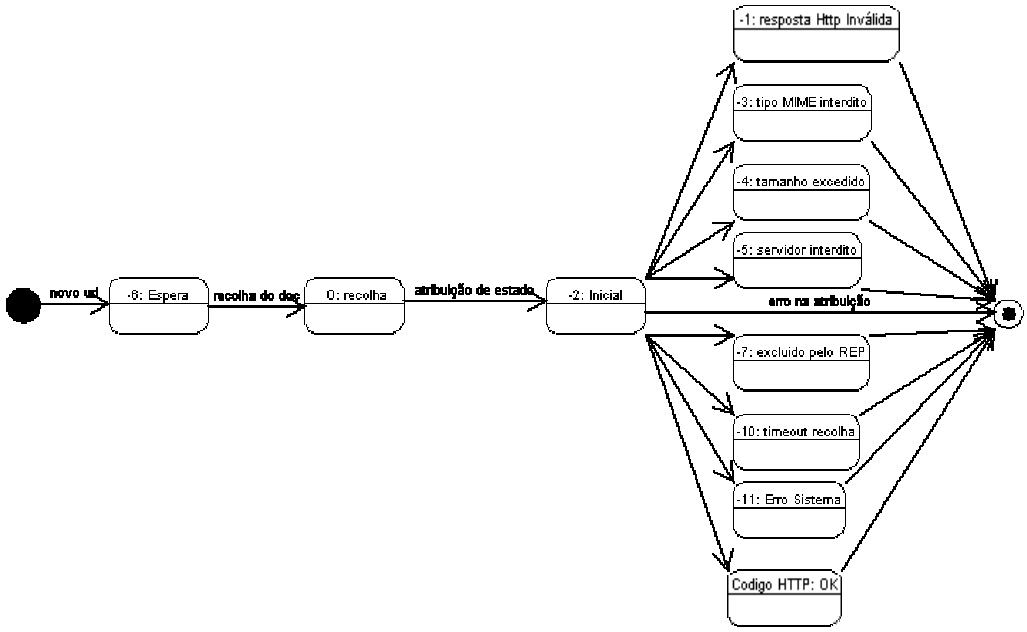



**Figura 3.8 Diagrama de estados da *classe Doc* – UML. Sequência de estados de uma instância da *classe Doc*, desde a inserção de um novo URL até ao termino da recolha do documento.**

Na Figura 3.8 estão representados os estados possíveis de uma ocorrência de *Doc* durante o processo de recolha. Cada vez que se pretende recolher um documento, é criada um novo objecto do tipo *Doc*, que armazena informação acerca da recolha do documento.

Inicialmente um documento encontra-se no estado de *espera* (*-6*). Quando o *Coleccionador* inicia o processo de recolha do documento, coloca-o no estado de *recolha* (*0*).

Durante a recolha podem dar-se situações de excepção que impossibilitam o processo normal de recolha do documento. Quando estas situações são detectadas, o documento passa para um estado descritivo da excepção ocorrida, que é acompanhado da escrita de uma mensagem descritiva da excepção para o ficheiro de *log* do sistema. Os estados correspondem a números negativos, no caso da excepção ter origem interna ao Tarântula. Quando o servidor *Web*, que é suposto alojar o documento que se pretende recolher, detecta alguma situação anómala ao tentar responder ao pedido efectuado, é atribuído um estado ao documento com o valor correspondente ao código HTTP, devolvido pelo servidor. Na Figura 3.8, os estados

correspondentes aos códigos HTTP foram substituídos por um estado único (*Codigo HTTP:OK*), para não sobrecarregar o diagrama.

### *3.3.3 Interface de Configuração e Gestão*

A Interface de Configuração e Gestão permite a um utilizador humano configurar e interagir com o sistema através de uma interface gráfica disponível a partir de um *browser,* validando os dados introduzidos e inserido-os nas estruturas de dados adequadas dos *Dados de Configuração das Publicações*.

O desenvolvimento de uma *Interface de Configuração e Gestão* completa e reutilizável em diversos contextos é um processo complexo e demorado, uma vez que requer grande interacção com os utilizadores finais. Assim sendo, as funcionalidades da interface implementadas têm um carácter simplesmente demonstrativo das capacidades básicas do sistema. A *Interface de Configuração e Gestão* deverá ser por conseguinte refeita consoante as necessidades do sistema que a vier a incorporar.

### *3.3.4 Monitor de Tarefas*

O *Monitor de Tarefas* oferece métodos de acesso aos *Dados de Execução*, permitindo o acesso concorrente e independente do sistema usado para o seu armazenamento.

### *3.3.5 Interpretador do Protocolo de Exclusão*

O *Interpretador do Protocolo de Exclusão* (*IPE*) detecta quais as partes de um servidor que podem ser visitadas por um *crawler*, através do *Robots Exclusion Protocol*. O *Interpretador* é invocado recebendo como parâmetro o nome de um servidor. Devolvendo um objecto de controlo de acesso, que permite analisar a partir do URL de um documento, se este pode ser recolhido. Se o *IPE.* não conseguir gerar o objecto de controlo, devido a desrespeito do protocolo de exclusão por parte do servidor *Web*, o Tarântula recolhe os documentos disponíveis sem restrições.

### *3.3.6 Extractor de URLs*

O *Extractor de URLs* analisa o conteúdo de um documento HTML e devolve a lista de ligações (URLs) para outros documentos.

Teoricamente, o *Extractor de URLs* consiste num componente de *software* que baseando-se na sintaxe do HTML definida no seu DTD [57], detecta as *tags* que contêm referências para outros documentos e guarda estas referências. Esta abordagem foi tomada no desenvolvimento da 1ª versão do extractor, mas os resultados obtidos não foram minimamente satisfatórios. O número de documentos que os servidores *web* indicam como sendo do tipo *text/html* que não respeitam o DTD do HTML ou que nem sequer contêm *tags* html, é muito elevado. Isto fez com que o extractor não conseguisse extrair todas as ligações contidas nestes documentos ou que desse erro. Embora estes documentos não respeitem a sintaxe do HTML, são visualizáveis na maior parte dos *browsers*. O utilizador do Tarântula pode assim ter a sensação de que o sistema não está a funcionar correctamente, uma vez que não recolhe todos os documentos por ele visualizados através do *browser*. Perante esta situação, o *Extractor de URLs* foi refeito para uma versão mais permissiva e flexível, que faz o mínimo de verificações à sintaxe dos documentos HTML. O *Extractor de URLs* é um componente delicado do sistema, uma vez que é impossível de testar todas as situações de excepção que possam surgir.

### *3.3.7 Conversor de Links*

À medida que a recolha de um item é efectuada, os documentos que a compõem vão sendo armazenados pelo Tarântula numa directoria do sistema de ficheiros. Terminada a recolha do item, temos uma directoria com os ficheiros correspondentes aos documentos recolhidos, mas a estrutura de navegação entre estes pode ter sido perdida, uma vez que as ligações entre os documentos pode ter sido feita através de URLs que dão a sua localização na WWW.

O *Conversor de Links* analisa os documentos (HTML) que contêm ligações e rescreve-os numa nova directoria, alterando apenas as ligações entre os documentos, de modo a permitir a navegabilidade pelos conteúdos do item, da forma o mais fiel possível à original.

### *3.3.8 Coordenador*

O *Coordenador* tem a função de orquestrar o funcionamento do sistema, através da análise e manipulação das suas estruturas de dados.

Quando é lançado, o *Coordenador* cria (caso não existam) duas directorias: "Itens" e "Arquivadas". Na directoria "Itens", o *Coleccionador* guardará os

documentos dos itens que estão em colecção. A directoria "Arquivadas" serve para guardar os itens cujas colecções já foram terminadas.

O *Coordenador* entra no ciclo de execução, do qual só sairá em caso de erro. A primeira acção do ciclo é analisar os *Dados de Configuração das Publicações*, a fim de detectar se existem publicações que devam ser recolhidas. Em caso afirmativo, é criada uma nova tarefa e uma nova sub-directoria dentro da directoria "Itens", para cada item a recolher. O passo seguinte é detectar colecções terminadas, atribuindo aos itens nessas circunstâncias o estado correspondente (Figura 3.7). Em seguida, é feita uma verificação dos prazos de validade dos itens, cuja colecção será interrompida no caso de terem sido excedidos. Finalmente o *Coordenador* arquiva os itens cujas recolhas foram terminadas, passando os seus conteúdos para uma sub-directoria dentro da directoria "Arquivadas" e fazendo opcionalmente a conversão de ligações entre os documentos.

### *3.3.9 Coleccionador*

O *Coleccionador* tem a função de recolher os documentos da WWW, armazená-los e guardar meta-dados referentes a essa recolha. O processo de recolha (ou colecção) dá-se a três níveis de execução: *Coleccionador* (*main*), servidor (*ServColect*) e documento (*DocColect*), descritos na Figura 3.9. Esta hierarquia de fios de execução, visa controlar a carga de pedidos a cada servidor e dar robustez ao sistema perante situações de excepção

O controlo de carga é feito lançando um fio de execução (*ServColect*) para cada servidor *Web* que contenha documentos a recolher. O *ServColect* após executar a recolha de um documento, dá preempção a outro fio de execução, só efectuando um novo pedido ao servidor passado um intervalo de tempo configurável (por defeito 1 segundo). Os pedidos são assim distribuídos pelos servidores *Web* de modo a não sobrecarregá-los com pedidos sucessivos.

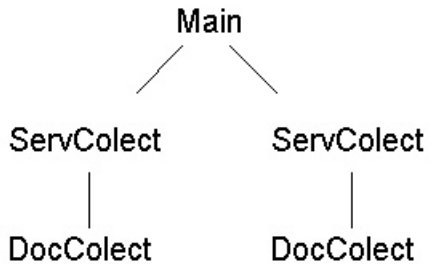



**Figura 3.9. Hierarquia de fios de execução do *Coleccionador*. O *Main* lança um *ServColect* para a recolha de documentos de cada servidor. O *ServColect* lança um *DocColect* cada vez que recolhe um documento.**

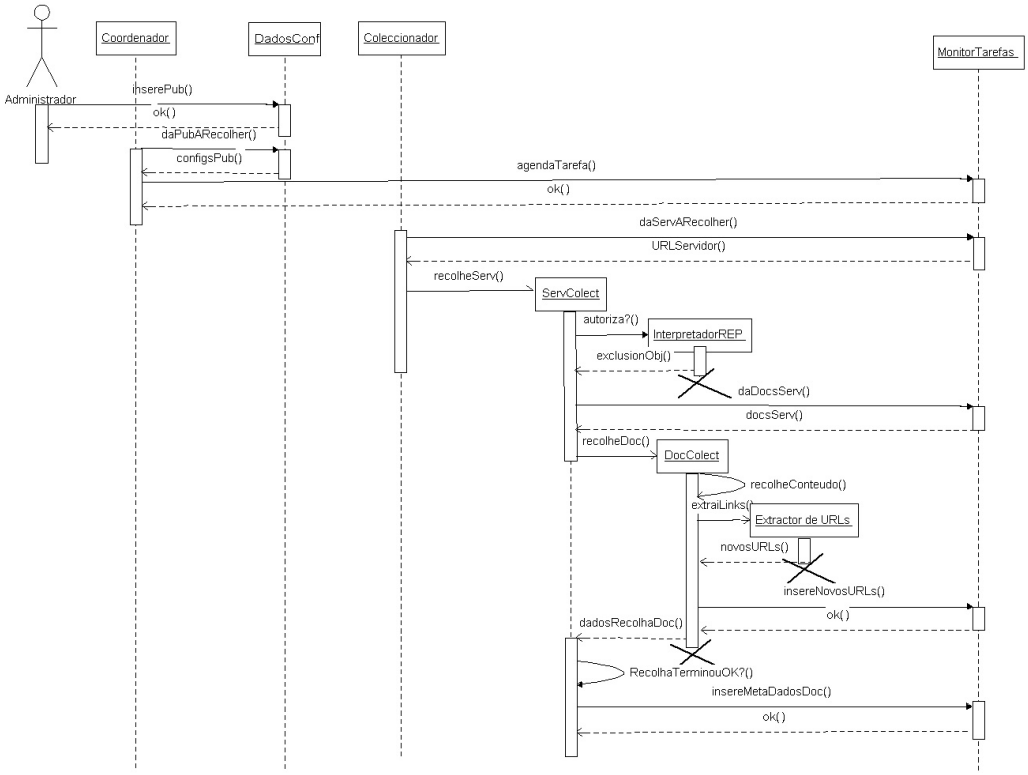



**Figura 3.10. Diagrama de Sequência – UML representativo da estrutura de execução (*thread*s) do *Coleccionador* ao efectuar a recolha de um documento de uma publicação.**

O sistema ganha robustez, através de um sistema de *timeouts* para a recolha de cada documento. A estrutura hierárquica de fios de execução permite que os *timeouts* de recolha sejam concretizados de uma forma eficaz e elegante. O funcionamento desta estrutura de execução é exemplificado na Figura 3.10.

O processo de recolha é despoletado pela inserção de uma publicação pelo utilizador. Quando o *Coordenador* detecta que deve ser feita a recolha da publicação, agenda uma nova tarefa no *Monitor de Tarefas*.

O *thread Main* do *Coleccionador* questiona periodicamente o *Monitor de Tarefas*, a fim de obter servidores *Web* que contenham documentos a recolher, lançando de forma assíncrona *threads ServColect* que irão efectuar a recolha dos documentos de cada servidor.

O *ServColect* inicia a sua execução invocando o *Interpretador do Protocolo de Exclusão*, para obter o objecto de controlo de acesso ao servidor. Em seguida pede ao *Monitor de Tarefas* uma lista de todos os documentos (e configurações de recolha respectivas) que deve recolher no servidor. Note-se que documentos alojados no mesmo servidor, podem pertencer a publicações diferentes e por conseguinte ter configurações de recolha diferentes. O *ServColect* lança sincronamente um novo fio de execução *DocColect*, que efectuará a recolha de um documento e espera que este termine dentro do *timeout*. Caso isto não aconteça, o *DocColect* é morto pelo *thread* pai (*ServColect*).

O *DocColect* recolhe o documento da WWW respeitando as restrições impostas e armazena-o na directoria do item a que pertence. O *Extractor de URLs* é invocado para efectuar a extracção de ligações contidas no documento recolhido, sendo estas inseridas para recolha no *Monitor de Tarefas*. O *DocColect* termina a sua execução devolvendo ao *ServColect* os meta-dados do documento.

O *ServColect* verifica se o documento foi recolhido dentro do limite de tempo estabelecido e insere os meta-dados no *Monitor de Tarefas*. O *ServColect* lança ciclicamente *threads DocColect*, (respeitando o intervalo mínimo entre pedidos aos mesmo servidor), até não existirem mais documentos a recolher no servidor.

## 3.4 Implementação

Nesta secção apresento as tecnologias empregues na realização de cada um dos componentes que compõem o sistema.

- ***Dados de Configuração das Publicações e Dados de Execução:*** Estes componentes foram desenvolvidos recorrendo à linguagem SQL e ao Sistema de Gestão de Bases de Dados PostgreSQL.

- ***Interface de Configuração e Gestão:*** foi desenvolvida recorrendo a HTML escrito manualmente e gerado através do Dreamweaver, e a Java Servlets contidas no TOMCAT e disponibilizadas através de um servidor *web* Apache. A *Interface de Configuração e Gestão* pode assim ser acedida remotamente através de um *browser*.

- ***Monitor de Tarefas:*** foi concretizado através de uma *class* Java, que estabelece uma ligação JDBC com o PostgreSQL.

- ***Interpretador do Protocolo de Exclusão:*** este componente é um analisador léxico desenvolvido em Java, gerado a partir de uma versão alterada do JLex. Foi utilizado o JLex e não o JFLex, porque as alterações efectuadas ao código do JLex, para que o analisador léxico gerado tivesse as características desejadas, seriam mais complexas no JFLex.

- ***Extractor de URLs e Conversor de Links:*** são analisadores léxicos escrito em Java, gerados a partir do JFLex. A máquina de estados para análise léxica dos documentos é semelhante em ambos, mas as acções realizadas para cada ligação encontrada no documento divergem.

- ***Coordenador e Coleccionador:*** são aplicações *multi-threaded* desenvolvidas em Java, que se encontram em permanente execução no sistema.

# 4 Resultados

Neste capitulo apresento os resultados obtidos nos testes de desempenho e integração efectuados ao Tarântula

Os testes de desempenho visaram detectar pontos de congestão e quantificar a diferença de desempenho do sistema sobre diferentes sistemas operativos.

Os testes de integração do Tarântula no DROP e no TUMBA, tiveram como objectivo provar que o sistema apresenta características que permitem que seja aplicado facilmente como subsistema.

Finalmente apresento as medidas relativas ao *software* desenvolvido no projecto.

## 4.1 Análise de desempenho

A maior parte dos componentes do Tarântula foram desenvolvidos em Java, o que permitiu efectuar testes de desempenho do sistema sobre sistemas operativos diferentes. Nos testes recorri a um Penthium 3 a 766MHz, com 256 Mb de memória para executar o código em Java, mantendo os componentes alojados no Postgres num servidor remoto. O teste de desempenho consistiu na recolha de um conjunto de documentos bem definido, alojado na rede local da Faculdade, de modo a evitar oscilações de largura de banda disponível durante as recolhas.

Os sistemas operativos em que o Tarântula foi testado foram o Microsoft Windows 2000 [32] e o RedHat Linux 6.2 [41], tendo o Tarântula executado uma média de **27** documentos por minuto sobre sistema da Microsoft, contra os **73** documentos por minuto obtidos sobre o RedHat (Figura 4.1). A máquina virtual Java da Sun usada no Linux disponibiliza a opção `-server` [49], destinada a aplicações servidor, tipicamente caracterizadas pela exigência de desempenho e pouca necessidade interfaces gráficas de utilização, esta opção foi utilizada nos testes efectuados e não está disponível na máquina virtual para Windows2000.

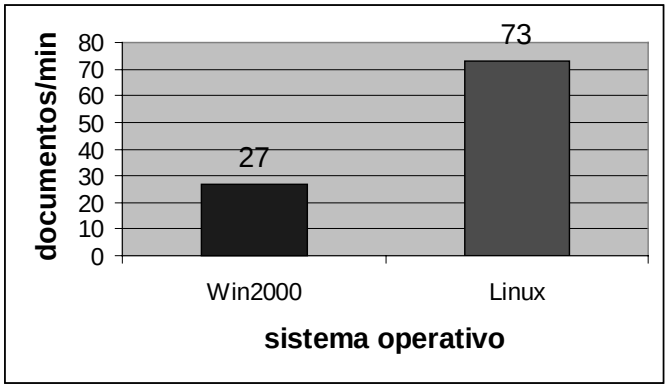



**Figura 4.1. Comparação de desempenho do Tarântula em Windows 2000 e RedHat Linux 6.2.**

## 4.2 Pontos de congestão

Independentemente do sistema operativo utilizado, identifiquei que o acesso aos *Dados de Execução* (implementados sob a forma de uma base de dados gerida pelo Postgres), é um forte ponto de congestão.

O Postgres demora em média 200 ms para efectuar a inserção de um tuplo na tabela *Doc* (Figura 3.6). Esta tabela guarda informação dos documentos a recolher e meta-dados dos documentos recolhidos. Considerando que a extracção de ligações de cada documento implica múltiplas inserções síncronas na tabela *Doc*, isto representa uma longo período de espera por parte das aplicações cliente e uma grande sobrecarga no SGBD. A demora do processo de inserção deve-se ao facto de o Postgres fazer *sync*, ou seja, cada inserção implica uma escrita em disco, em vez de armazenar em memória várias inserções e efectuar uma única escrita em bloco no disco. Esta característica é útil em sistemas transaccionais rigorosos (por exemplo gestão bancária), mas no caso do Tarântula torna-se prejudicial.

O desempenho do Postgres degrada-se gravemente com o aumento de informação na base de dados. A titulo de exemplo, a *query* efectuada à base de dados, que retorna quais os servidores que alojam documentos a recolher, é executada em cerca de 2 segundos com 76640 tuplos na base de dados, contra 1 minuto e 18 segundos com 562480 tuplos na base de dados. Tentei amenizar este problema recorrendo à criação de índices adicionais e à simplificação das *querys*, mas não consegui melhorias.

Estes resultados sugerem que um ganho de eficiência significativo só poderá ser conseguido com a substituição do PostgreSQL por:

- Um SGBD escalável
- Armazenamento dos dados em ficheiros indexados

## 4.3 Aplicações

### *4.3.1 DROP*


**Figura 4.2. Página de entrada no DROP**

- ***Integração***

A integração do Tarântula no DROP foi feita sem dificuldades, tendo sido escritas apenas mais 186 linhas de código. No entanto, foi necessário criar uma nova *Interface de Configuração e Gestão*, (incluindo ajudas e documentação online), que se adaptasse às necessidades do DROP, o que levou à escrita de mais 3953 linhas de código (excluindo páginas geradas através do Dreamweaver). Embora a geração deste elevado número de linhas de código

seja um acréscimo de trabalho durante o estágio, não denota dificuldades de integração do Tarântula, uma vez que os seus requisitos, definem que a *Interface de Configuração e Gestão* deve ser refeita consoante as necessidades do sistema em que se integra e este processo duplica tipicamente o número de linhas de código de um projecto [38].

A versão do Tarântula que passou a integrar o DROP, foi baptizada de RAPA (RecolhA de Publicações em LinhA- Figura 4.2).

- ***Configuração***

O Rapa foi inicialmente testado na recolha de um conjunto de 126 publicações online seleccionadas por bibliotecários. A recolha de cada uma das publicações foi limitada ao servidor base, (servidor que contem o documento base da publicação), e a uma profundidade máxima de 6, não tendo sido feitas restrições aos tipos e tamanhos dos documentos

- ***Estatísticas***

Para recolher as 126 publicações seleccionadas, o Rapa efectuou 60523 pedidos HTTP, tendo recolhido documentos em 72% deles. Pela análise da Figura 4.3 verifica-se que 19% dos pedidos se referem a documentos que não se encontravam alojados no servidor base, 4% são ligações quebradas (HTTP 404) e 5% a erros de sistema. Após analisar os ficheiros de *log*, verifiquei que os erros de sistema encontrados, foram principalmente devidos a ligações para servidores que deixaram de existir ou mudaram de nome. Este facto fez com que o Tarântula não tenha conseguido estabelecer ligações TCP/IP a estas máquinas e que tenha considerado que houve um erro de sistema, devido por exemplo a uma quebra de rede.

Uma estatística interessante, é que apenas uma publicação rejeitou o Rapa através do REP. Isto poderia dar-nos a perspectiva de que a maior parte dos publicadores e administradores dos sítios *Web* estariam receptivos a serem visitados por *robots*. No entanto, num estudo mais profundo verifiquei que a maior parte dos servidores visitados nem têm o ficheiro robots.txt utilizado pelo REP, como tal é provável que as publicações estejam livremente acessíveis a *robots* por os seus editores desconhecerem o REP.

A maior parte dos documentos recolhidos são do tipo *text/html*, *image/gif* ou *image/jpeg* (Figura 4.4), tendo tamanhos que variam entre os 2 Kb e os 32 Kb (Figura 4.5).

| Estado final | Significado | Nº Docs | % |
|---|---|---|---|
| 200 | HTTP: OK | 43322 | 72% |
| -5 | Servidor interdito | 11468 | 19% |
| -11 | Erro sistema | 2743 | 5% |
| 404 | HTTP: File Not Found | 2381 | 4% |
| -10 | Timeout recolha | 103 | 0% |
| -2 | Erro na atribuição de codigo | 181 | 0% |
| -4 | Tamanho excedido | 129 | 0% |
| -1 | Resposta http inválida | 110 | 0% |
| -3 | Tipo desconhecido | 52 | 0% |
| 500 | HTTP:Internal Server Error | 32 | 0% |
| -7 | Excluido peo REP | 1 | 0% |
| 403 | HTTP: Forbidden | 1 | 0% |
| | | 60523 | 100% |

**Figura 4.3. Resultados da integração no DROP – estados finais dos documentos. Os estados finais foram obtidos a partir dos códigos HTTP retornados aquando da recolha dos documentos e de códigos atribuídos pelo sistema.**

| MIME type | Number | Percent |
|---|---|---|
| Text/html | 34424 | 70% |
| Image/gif | 5448 | 11% |
| Unknown | 4478 | 9% |
| Image/jpeg | 4400 | 9% |
| Text/plain | 238 | 0% |
| Application/zip | 64 | 0% |
| Application/msword | 57 | 0% |
| Application/pdf | 45 | 0% |
| Image/png | 45 | 0% |
| Application/octet-stream | 28 | 0% |
| Audio/mpeg | 11 | 0% |
| Application/x-tcl | 4 | 0% |
| Application/vnd.ms-excel | 3 | 0% |
| Video/mpeg | 3 | 0% |
| Audio/x-pn-realaudio | 2 | 0% |
| Application/mac-binhex | 1 | 0% |
| Application/vnd.ms-powerpoint | 1 | 0% |
| Application/x-shockwave-flash | 1 | 0% |
| Audio/midi | 1 | 0% |
| Audio/x-wav | 1 | 0% |
| Image/bmp | 1 | 0% |
| | 49256 | 0% |

**Figura 4.4. Distribuição dos tipos de documentos recolhidos**

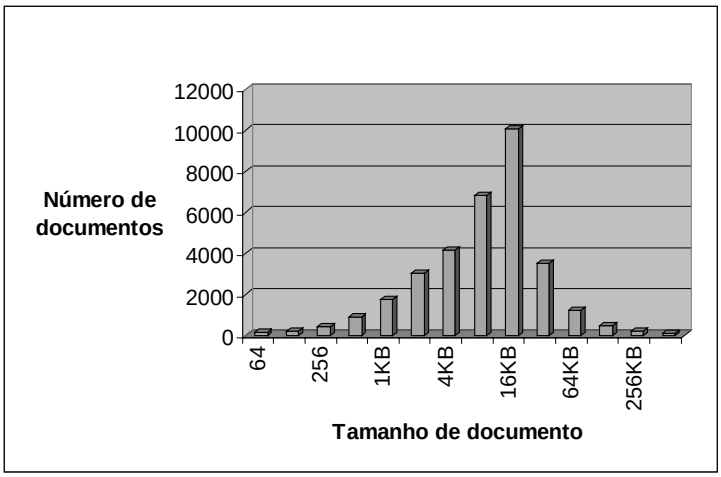



**Figura 4.5. Tamanhos dos documentos recolhidos.– relação entre o número de documentos recolhidos e o seu tamanho.**

A análise dos resultados obtidos nesta primeira recolha permitiu concluir que devido ao reduzido número de ligações quebradas encontradas (mesmo considerando as ligações para servidores inexistentes), a maior parte das publicações online portuguesas são mantidas cuidadosamente e são compostas por documentos pequenos e de tipos facilmente tratáveis.

### *4.3.2 TUMBA*


**Figura 4.6. Página de entrada no TUMBA.**

- ***Integração***

A integração do Tarântula no TUMBA (Figura 4.6) foi efectuada sem ser necessário gerar código adicional, uma vez que se trata do 1º protótipo do sistema e ainda não foram obtidos resultados concretos que justificassem a geração de código adicional.

Neste primeiro protótipo do TUMBA, construímos um motor de busca de termos sobre as páginas da *intranet* da Faculdade de Ciências da Universidade de Lisboa.

- ***Configuração***

O Tarântula foi configurado para recolher documentos do tipo *text/html* alojados em servidores no dominio *fc.ul.pt*, partindo do documento base do sítio da Faculdade de Ciências (URL: `http://www.fc.ul.pt/`) e até uma profundidade de 13.

| urlserv | ndocs |
|---|---|
| alf1.cii.fc.ul.pt | 22745 |
| ptmat.lmc.fc.ul.pt | 17607 |
| correio.cc.fc.ul.pt | 1113 |
| www.fc.ul.pt | 750 |
| www.di.fc.ul.pt | 633 |
| lmc.fc.ul.pt | 602 |
| hermite.cii.fc.ul.pt | 586 |
| astro.cc.fc.ul.pt | 569 |
| lince.cii.fc.ul.pt | 497 |
| appia.di.fc.ul.pt | 379 |
| fbp.lmc.fc.ul.pt | 297 |
| www.lmc.fc.ul.pt | 280 |
| www.navigators.di.fc.ul.pt | 206 |
| intranet.fc.ul.pt | 175 |
| www.dqb.cc.fc.ul.pt | 164 |
| mat.fc.ul.pt | 141 |
| npde99.lmc.fc.ul.pt | 141 |
| caul.cii.fc.ul.pt | 133 |
| cosmo.cii.fc.ul.pt | 133 |
| xldb.fc.ul.pt | 129 |
| gfm.cii.fc.ul.pt | 123 |
| cfmc.cii.fc.ul.pt | 116 |
| www.deio.fc.ul.pt | 100 |
| www.educ.fc.ul.pt | 98 |
| caravela.di.fc.ul.pt | 70 |
| alf3.cii.fc.ul.pt | 62 |
| mtcm2000.lmc.fc.ul.pt | 59 |
| spm.lmc.fc.ul.pt | 46 |

**Figura 4.7. Servidores visitados – lista dos servidores *Web* que alojam mais documentos da recolha do domínio *fc.ul.pt*.**

- ***Estatísticas***

A distribuição dos tamanhos dos documentos recolhidos varia pouco dos resultados obtidos nas recolhas efectuadas para o DROP. No entanto, pela análise da distribuição dos documentos pelos servidores *Web* (Figura 4.7), observa-se que grande parte dos documentos recolhidos são provenientes de apenas dois servidores: *alf1.cii.fc.ul.pt* e *ptmat.lmc.fc.ul.pt*. Uma visita a algumas páginas destes servidores, revelou que se tratam de bases de dados muito extensas de documentos HTML. Dado o elevado número de documentos alojados nestes servidores, interrompi arbitrariamente a sua recolha, uma vez que esta se estava a prolongar demasiado.

Hoje em dia os sítios tendem a estar estruturados de forma a que a informação mais relevante se encontre em profundidades pequenas, de modo a torná-la mais acessível aos leitores. As páginas que os utilizadores buscam

tipicamente, encontram-se até um profundidade de 3, (contada a partir da raiz do sítio). A maior parte dos documentos dos servidores, cuja recolha foi interrompida, situam-se a uma profundidade superior a 5, pelo que, em principio, recolhemos as páginas mais relevantes. No entanto, esta situação levanta o problema do controlo de profundidade por servidor, que seguindo as configurações inseridas não foi possível.

A análise dos resultados obtidos relevou que embora o Tarântula tenha cumprido a missão que lhe foi incumbida, não é usável em recolhas de grande escala, uma vez que demorou cerca de um dia a efectuar 76641 pedidos HTTP, sem nunca esgotar a parca largura de banda disponível, o que representa uma ínfima parte da WWW.

Os casos dos servidores com muitos documentos, alertou-me para o facto de que ao efectuar recolhas em grande escala é necessário ter configurações associadas aos servidores. O Tarântula deverá também suportar um mecanismo que permita gerir de forma eficiente as ligações encontradas para documentos alojados em novos servidores.

O primeiro protótipo do TUMBA suporta actualmente as pesquisas no sítio da FCUL estando também disponível em:

- http://xldb.fc.ul.pt/xldb.fc.ul.pt/tumba/tumba.htm
- http://xldb.fc.ul.pt/xldb.fc.ul.pt/tumba/index.html

## 4.4 Medidas

O *software* resultante do trabalho desenvolvido ao longo do estágio inclui 377 linhas de código em SQL, 629 linhas de código em HTML e 12818 linhas de código em Java.

Excluindo o código gerado automaticamente, obtemos um valor total de 12059 linhas de código escritas durante 6 meses, o que resulta numa produtividade média de 91 linhas de código por dia. Esta produtividade deverá ser enquadrada num contexto de investigação, que requereu permanentes consultas de documentação técnica e artigos científicos.

# 5 Metodologia e calendarização do trabalho

## 5.1 Metodologia

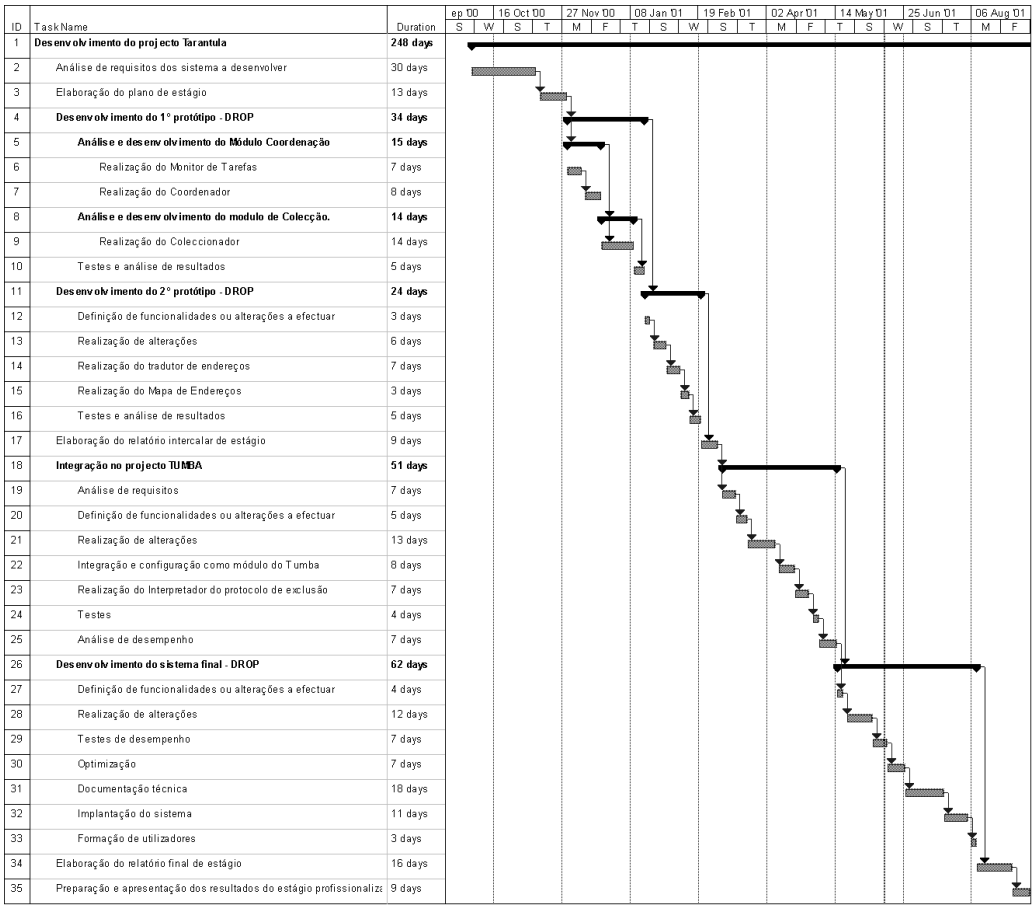
O Tarântula foi desenvolvido de acordo com o modelo em espiral [43], tendo sido desenvolvidos dois protótipos do sistema durante o estágio.

O primeiro protótipo foi desenvolvido exclusivamente para integração no projecto DROP (não tendo sido levados em conta os requisitos do TUMBA). Este protótipo não foi instalado nas instalações da Biblioteca Nacional, uma vez que se destinou principalmente a identificar requisitos e funcionalidades do sistema, que não eram bem conhecidos no início do projecto. O 1º protótipo do DROP permitiu que a BN, tomasse contacto directo com diversas questões relativas ao projecto, permitindo-lhe sugerir um conjunto de requisitos específicos para o início do desenvolvimento do segundo protótipo.

Após a conclusão do segundo protótipo, este foi integrado nas instalações da BN. Ocorreu em seguida uma fase de testes, ao fim da qual se identificaram um conjunto de *bugs* a retirar do sistema. O sistema final foi o resultante da rectificação destes *bugs*.

## 5.2 Calendarização do trabalho

A calendarização do trabalho (Figura 5.1) contempla o desenvolvimento do projecto ao longo de 12 meses, no entanto, o estágio profissionalizante tem a duração de 6 a 9 meses. Pelo que, após reunião com as partes intervenientes no estágio por parte da FCUL e do LaSIGE, tomou-se a decisão de considerar o trabalho de estágio terminado no fim da integração do 2º protótipo do Tarântula no projecto TUMBA. As restantes tarefas referidas na calendarização inicial serão realizadas fora do âmbito do estágio.



| ID | Task Name | Duration |
|---|---|---|
| 1 | **Desenvolvimento do projecto Tarantula** | **248 days** |
| 2 | Análise de requisitos dos sistema a desenvolver | 30 days |
| 3 | Elaboração do plano de estágio | 13 days |
| 4 | **Desenvolvimento do 1° protótipo - DROP** | **34 days** |
| 5 | **Análise e desenvolvimento do Módulo Coordenação** | **15 days** |
| 6 | Realização do Monitor de Tarefas | 7 days |
| 7 | Realização do Coordenador | 8 days |
| 8 | **Análise e desenvolvimento do modulo de Colecção.** | **14 days** |
| 9 | Realização do Coleccionador | 14 days |
| 10 | Testes e análise de resultados | 5 days |
| 11 | **Desenvolvimento do 2° protótipo - DROP** | **24 days** |
| 12 | Definição de funcionalidades ou alterações a efectuar | 3 days |
| 13 | Realização de alterações | 6 days |
| 14 | Realização do tradutor de endereços | 7 days |
| 15 | Realização do Mapa de Endereços | 3 days |
| 16 | Testes e análise de resultados | 5 days |
| 17 | Elaboração do relatório intercalar de estágio | 9 days |
| 18 | **Integração no projecto TUMBA** | **51 days** |
| 19 | Análise de requisitos | 7 days |
| 20 | Definição de funcionalidades ou alterações a efectuar | 5 days |
| 21 | Realização de alterações | 13 days |
| 22 | Integração e configuração como módulo do Tumba | 8 days |
| 23 | Realização do Interpretador do protocolo de exclusão | 7 days |
| 24 | Testes | 4 days |
| 25 | Análise de desempenho | 7 days |
| 26 | **Desenvolvimento do sistema final - DROP** | **62 days** |
| 27 | Definição de funcionalidades ou alterações a efectuar | 4 days |
| 28 | Realização de alterações | 12 days |
| 29 | Testes de desempenho | 7 days |
| 30 | Optimização | 7 days |
| 31 | Documentação técnica | 18 days |
| 32 | Implantação do sistema | 11 days |
| 33 | Formação de utilizadores | 3 days |
| 34 | Elaboração do relatório final de estágio | 16 days |
| 35 | Preparação e apresentação dos resultados do estágio profissionaliz | 9 days |



**Figura 5.1. Diagrama de Gantt referente à calendarização das tarefas descritas no plano de estágio.**

## 5.3 Análise crítica

Analisando o trabalho desenvolvido e o plano inicial, considero que a calendarização do projecto foi, no geral, cumprida. As suas grandes tarefas foram realizadas dentro do prazo estipulado, não causando atraso na realização do estágio.

No entanto, a duração de algumas tarefas foi subestimada, ao passo que a duração de outras foi sobrestimada. Em seguida, apresento as tarefas, em que encontrei maiores discrepâncias entre os valores de duração estimados e reais, acompanhadas dos factos que estiveram na sua origem.

### *5.3.1 Tarefas subestimadas*

- *Tarefa 2: Análise de requisitos do sistema a desenvolver*

  Esta tarefa demorou cerca do dobro do tempo inicialmente previsto, uma vez que, o primeiro protótipo do Tarântula destinava-se exclusivamente ao projecto DROP e o processo de análise de requisitos arrastou-se devido a sucessivos adiamentos e mudanças de posição por parte da BN.

- *Tarefa 16: Testes e análise de resultados*

  Os testes e análise de resultados, implicitamente incluem uma terceira tarefa, que foi menosprezada na calendarização: os melhoramentos. Os melhoramentos, são correcções de pequenos erros detectados na fase de testes, que não necessitam de ser incluídos nos requisitos do protótipo seguinte, mas que aumentaram muito a duração desta tarefa.

### *5.3.2 Tarefas sobrestimadas*

- *Tarefas 14 e 15: Realização do Tradutor de endereços e Mapa de endereços*

  Estes dois componentes foram suprimidos do sistema, uma vez que a sua realização se devia à suposição inicial, de que o acesso ao Servidor de Nomes [56], seria um dos principais pontos de congestão do sistema, o que na prática não se verificou.

- *Tarefa 18: Integração no projecto TUMBA*

  A duração total desta grande tarefa foi muito inferior ao inicialmente previsto, devido à maior parte das tarefas que a compõem terem sido realizadas ou simplificadas pelo desenvolvimento do 2º protótipo do DROP. Por exemplo, a *Tarefa 23 Realização do Interpretador do Protocolo de Exclusão*, por indicações da BN, passou a fazer parte dos requisitos de 2º protótipo do DROP, não sendo necessária a sua realização propositada para o TUMBA.

A conjunção dos ganhos e perdas de tempo das tarefas com duração subestimada e sobrestimada, juntamente com o facto de ter sido dado maior peso às estimativas pessimistas na elaboração do plano, permitiu que o projecto fosse realizado dentro dos prazos estabelecidos.

A decisão de recorrer a protótipos para o desenvolvimento do sistema, revelou-se extremamente produtiva, uma vez que permitiu responder aos requisitos peculiares de um projecto de investigação e ao mesmo tempo aos requisitos de um projecto convencional, resultando num processo de evolução gradual do sistema.

# 6 Conclusões e trabalho futuro

## 6.1 Sumário

Durante o estágio desenvolvi um protótipo do Tarântula segundo os requisitos inicialmente propostos. O Tarântula é um *crawler* que pode ser facilmente integrado e configurado como subsistema. A arquitectura do Tarântula é modular, permitindo a alteração e desenvolvimento de componentes de forma independente. As tecnologias usadas baseiam-se principalmente na plataforma Java e no sistema de gestão de base de dados relacionais PostgreSQL.

O Tarântula foi configurado e integrado no sistema de depósito digital DROP, posteriormente baptizado de RAPA, e no motor de busca TUMBA. Os resultados obtidos nos testes de integração do sistema foram satisfatórios, embora o desempenho com a implementação do Tarântula não permita responder às necessidades de sistemas de grande escala.

Devido ao ponto de congestão, (descrito no capitulo 4.2), causado pela utilização do PostgreSQL para implementação dos *Dados de Execução*, e por restrições de tempo e equipamento, não foi possível efectuar recolhas de grandes quantidades de informação da *Web*, que comprovassem a escalabilidade do Tarântula. Contudo, verificou-se que a recolha da *intranet* da FCUL foi efectuada em cerca de 24 horas, recorrendo somente a uma máquina para alojar todo o sistema. O Tarântula apresenta uma arquitectura modular com capacidades de distribuição, que visa expandir a capacidade de recolha do sistema através do aumento de máquinas que alojam o sistema.

A capacidade de recolha do sistema pode assim ser aumentada através de:

- Expansão do sistema com novos componentes de recolha (*Coleccionadores*), que permitem que se passe da recolha de um único domínio, para a recolha simultânea de todos os domínios da Web Portuguesa a baixo custo.

- Repartição da carga de processamento e armazenamento do sistema, por diversas máquinas, passando cada uma delas a alojar um único componente do sistema.

A calendarização no geral foi cumprida, não tendo existido grandes atrasos no projecto.

O desenvolvimento desta primeira versão do Tarântula teve resultados positivos, permitindo adquirir grande conhecimento acerca da WWW e da construção de *crawlers*. Ao longo do desenvolvimento do sistema deparei com situações que criam novos desafios, fazendo com que o Tarântula seja apenas o ponto de partida para trabalhos futuros, existindo ainda muito espaço para melhoramentos e novas funcionalidades.

## 6.2 Trabalho futuro

O trabalho a realizar posteriormente ao estágio visa a continuação do desenvolvimento dos dois projectos em que o Tarântula foi integrado: o DROP e o TUMBA. Estes projectos impõem alterações ao Tarântula, para que responda completamente às suas necessidades, pelo que o trabalho de desenvolvimento bifurca, na concretização do sistema final do RAPA para o DROP e na versão 2 do Tarântula estritamente direccionado para o TUMBA.

O trabalho futuro no RAPA consistirá principalmente no estudo e criação de interfaces de utilização e na detecção e correcção de *bugs*.

O Tarântula v.2 será um sistema distribuído de recolha completamente integrado no TUMBA, que estará em estrita colaboração com um repositório de dados. Esta nova arquitectura visa obter uma gestão e armazenamento organizado da informação recolhida da WWW, que resolva os problemas actuais de congestão e escalabilidade.[23,22,13]. O sistema incluirá novas funcionalidades, como a gestão de conteúdos repetidos e recolha de informação que permita aplicar algoritmos de *PageRank* [25] sobre os documentos recolhidos.

## 6.3 Resultados profissionais

A adaptação ao ambiente de trabalho, decorreu sem dificuldades desde o inicio do estágio, sendo facilitada pela colaboração das pessoas envolvidas nos projectos em que participei.

O trabalho desenvolvido durante o estágio permitiu-me solidificar conhecimentos adquiridos durante a Licenciatura, aplicando-os no desenvolvimento de projectos reais.

Um dos aspectos que mais apreciei deste estágio, foi a possibilidade de experimentar simultaneamente duas vertentes da Informática, a investigação, ao participar no desenvolvimento do TUMBA e a empresarial, com o desenvolvimento do DROP para a Biblioteca Nacional.

O estágio permitiu-me executar uma grande diversidade de tarefas, como o levantamento de requisitos, análise e modelação de sistemas, programação, optimização de bases de dados, testes ou interfaces pessoa-máquina, que me permitiram progredir em diversas direcções.

Pelas razões apresentadas considero-me satisfeito com os resultados obtidos e que o estágio foi crucial na minha carreira profissional.

# 7 Referências


1. "*Booch Object Model*".http://www.ecs.csun.edu/~nleuci/comp282/SS_fcpp_L11_BoochOM.html

2. "*DNS For Dummies: What is DNS?*".http://www.woz.org/pages/staff/auri/dns/What_Is_DNS.html

3. "*OMT Object Model: Index*". http://panoramix.univ-paris1.fr/CRINFO/dmrg/MEE/misop007/

4. "The Client/Server Paradigm" http://www.infosys.tuwien.ac.at/Teaching/Finished/MastersTheses/RPCmagiX/node10.html

5. *"The Java Program: Crawler.java"*. http://cs.fit.edu/~ryan/java/programs/*crawler*/*Crawler*-java.html

6. *"The Web Robots Pages"*. http://info.web*crawler*.com/mak/projects/robots/robots.html

7. 3COM – "*Understanding IP Addressing: Everything You Ever Wanted To Know*". http://www.3com.com/solutions/en_US/ncs/501302.html

8. A.R. Royo. "*Tarantulas (DesertUSA)*". http://www.desertusa.com/july96/du_taran.html

9. Allan Heydon and Marc Najork. "*Mercator: a Scalable, Extensible Web Crawler*". http://research.compaq.com/SRC/mercator/papers/www/paper.html

10. ArgoUML. http://argouml.tigris.org

11. Barry W. Boehm. "*A spiral model of software development and enhancement. Computer*", 21(5):61{72, 1988.

12. Brian Pinkerton. "*Finding What People Want: Experiences with the WebCrawler*". *In Proceedings of the Second International World Wide Web Conference, 1994.*

13. Craig E. Wills and Mikhail Mikhailov. "*Towards a Better Understanding of Web Resources and Server Responses for Improved Caching*". http://www8.org/w8-papers/2a-webserver/towards/towards.html

14. DEPARTAMENTO DE INFORMÁTICA, FCUL – "Guia do estagiário da Licenciatura em Informática", Outubro de 1997.

15. Earl Hood – *"MIME. (Multipurpose Internet Mail Extensions)*". http://www.nacs.uci.edu/indiv/ehood/MIME/MIME.html

16. Elliot Berk at Princeton UniversityJFlex – "JFlex – The Fast Scanner Generator for Java". http://www.jflex.de/

17. Elliot Joel Berk and C. Scott Ananian – "*JLex: A Lexical Analyzer Generator for Java(TM)*" http://www.cs.princeton.edu/~appel/modern/java/JLex/

18. Ferreira & Bento – "PCGuia – Tire o Máximo do seu computador em Casa" – http://www.fbnet.pt/pcg/

19. Fundação Ciência e Tecnologia. http://www.fct.mct.pt

20. "*Harvest*" – http://harvest.transarc.com

21. Ivar Jacobson – . "*Object-Oriented Analisys and Design Methods –12. OOSE*" http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo-12.html

22. Jun Hirai, Sriram Raghavan, Hector Garcia-Molina, Andreas Paepcke. "*WebBase: A repository of web pages*". http://www9.org/w9cdrom/296/296.html

23. Junghoo Cho and Hector Garcia-Molina. "*The evolution of the web and implications for na incremental crawler*", In VLDB 2000 Proceedings of 26th International Conference on Very Large Data Bases, September10-14, 2000, Cairo, Egypt. Pages 200-20229, 2000.

24. Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. "*Efficient crawling through URL ordering*". *In Proceedings of the Seventh International World Wide Wb Conference*, pages 161--172, April 1998.

25. Larry Page, Sergey Brin, R. Motwani, T. Winograd – "*The PageRank Citation Ranking: Bringing Order to the Web"*. http://dbpubs.stanford.edu/pub/showDoc.Fulltext?lang=en&doc=1998-8&format=pdf&compression=

26. LASIGE. http://lasige.di.fc.ul.pt

27. Macromedia – "*Macromedia Dreamweaver*" http://www.macromedia.com/*software*/dreamweaver/

28. Martijn Koster. "*Guidelines for Robot Writers"*. http://info.web*crawler*.com/mak/projects/robots/guidelines.html

29. Martjin Koster – "*A Standard for Robot Exclusion"* http://info.web*crawler*.com/mak/projects/robots/norobots.html

30. Michael Hersovici, Michael Jacovi, Yoelle S. Maarek, Dan Pelleg, Menachem Shtalhaim and Sigalit Ur. "*The shark-search algorithm – An application_ tailored Web site mapping"*. http://www7.scu.edu.au/programme/fullpapers/1849/com1849.htm

31. Microsoft – "*Microsoft Visio*". http://microsoft.com/office/visio/

32. Microsoft –"*Windows 2000 Home Page*" http://www.microsoft.com/windows2000/

33. Norman Noronha, João P.Campos, Daniel Gomes, Mário J. Silva and José Borbinha. A Deposit for Digital Collections. http://xldb.fc.ul.pt/referencias/paper_ecdl01.pdf

34. Nuno Maria. "*Theme-Based Retrieval of Web News"*. Dissertação para a obtenção do grau de Mestre em Informática. Faculdade de Ciências, Universidade de Lisboa. Julho de 2000. http://xldb.fc.ul.pt/ariadne/documentos/tese_final.pdf

35. Oliver A. McBryan. *GENVL and WWWW: Tools for taming the Web. In Proceedings if the First International World Wide Web Conference, pages 79—90, 1994*.

36. OMG – http://www.omg.org

37. PostgreSQL – *"a sophisticated Object Relational DBMS"*. http://www.postgresql.org.

38. Pressman, Roger S. – "*Software Engineering: A practitioner's approach".* Adaptado para a edição europeia por INCE, Darrel. 4ª edição. Eric M.Munson, 1997. ISBN 0-07-709411-5.

39. Publico S.A – "Publico.pt" – http://www.publico.pt

40. RATIONAL – "*OMG UML v. 1.3 specification*" http://www.rational.com/media/uml/post.pdf

41. RedHat Inc. – "*REDHAT.COM—Serving the Linux and Open Source Communities*". http://www.redhat.com/

42. Refsnes Data – "*SQL Tutorial*". http://www.w3schools.com/sql/

43. Roger S.Pressman. *Software* Engineering , "*A Practitioner's Approach McGraw Hill"*, 1997

44. Sergey Brin and Lawrence Page. *The anatomy of a large-scale hypertextual Web search engine In Proceedings of the Seventh International World Wide Web Conference, pages 107--117, April 1998*.

45. Soumen Chakrabarti, Martin van den Berg, Byron Dom. "*Focused crawling: a new approach to topic-specific Web resource discovery*". http://www8.org/w8-papers/5a-search-query/crawling/

46. SUN MICROSYSTEMS – "*Java 2 Platform SE v1.3.1*" http://java.sun.com/j2se/1.3/docs/api/

47. SUN MICROSYSTEMS – "*JavaServer Pages(TM) Technology*". http://java.sun.com/products/jsp/

48. SUN MICROSYSTEMS INC. – "*Java (TM) Servlet Technology*". http://java.sun.com/products/servlet/

49. SUN MICROSYSTEMS INC. – "*Java HotSpot VM Options*" http://java.sun.com/docs/hotspot/VMOptions.html

50. SUN MICROSYSTEMS INC. – "*JDBC (TM) Technology*". http://java.sun.com/products/jdbc/

51. SUN MICROSYSTEMS INC. – *Forte for Java*. http://www.sun.com/forte

52. SUN MICROSYSTEMS, INC. – "*What is the Java Platform?- Here's the quick guide"*. http://java.sun.com/nav/whatis

53. The Apache *Software* Foundation – "*Apache Project*" http://httpd.apache.org/

54. The Apache *Software* Foundation.- "*The Jakarta Site – Jakarta Tomcat*" http://jakarta.apache.org/tomcat/

55. The Web Design Resource – "*Javascript City*". http://www.javascriptcity.com/

56. TUMBA. http://xldb.fc.ul.pt/tumba

57. W3C – "*HTML 4.01 Specification*". http://www.w3.org/TR/html401/

58. W3C - "*HTTP- Hypertext Transfer Protocol Overview*" http://www.w3.org/Protocols/

59. W3C – "SGML". http://www.w3c.org/MarkUp/SGML

60. W3C –"HTML". http://www.w3c.org/Markup

61. W3C. http://w3c.org

62. XLDB. http://xldb.fc.ul.pt

# 8 Apêndice

**Descrição detalhada do modelo de dados.**

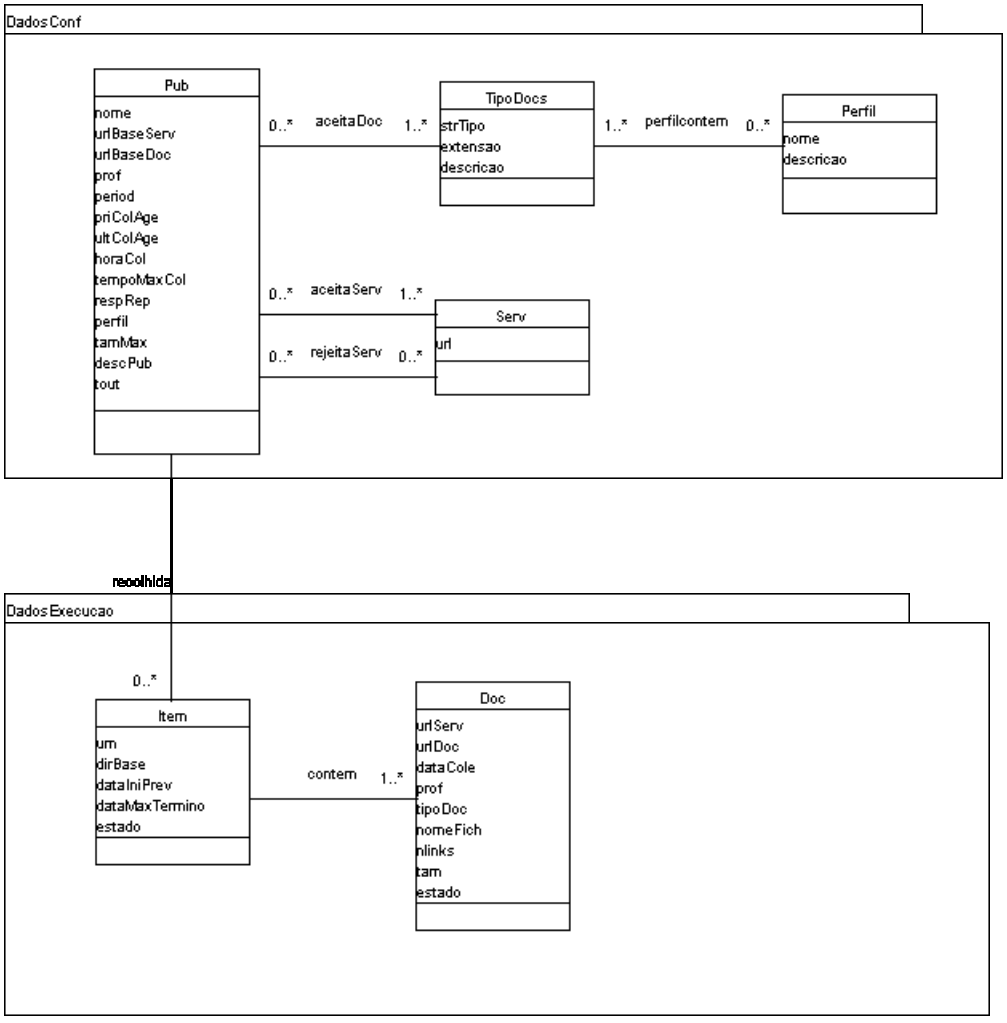



**Figura.1. Diagrama de classes – UML. Representação do modelo de dados dos Dados de Configuração das Publicações e dos Dados de Execução .**

*PACOTE: Dados de Configuração das Publicações*

**CLASSE: Pub**

Classe que armazena informação acerca de publicações

urnBase: varchar(100) – nome da publicação

urlBaseServ: varchar(100) – url do servidor que aloja o documento base

urlBaseDoc: varchar – localização do documento base no servidor.

prof: int – profundidade máxima de colecção

period: varchar(10) – periodicidade de colecção

priColAge: date – data da primeira colecção agendada

ultColAge: date - data da ultima colecção agendada

horaCol: time – hora de colecção

tempoMaxCol: int – tempo máximo de colecção (em minutos)

respREP: bool – indicação se o crawler deve respeitar o REP durante a colecção.

perfil: varchar – perfil de tipo de documentos

tamMax: int – tamanho máximo de ficheiros aceites (em bytes)

descPub: varchar – descrição da publicação

tout: int – tempo máximo para recolha de um documento (em mili-segundos)

**CLASSE: Serv**

Classe que armazena informação acerca de servidores

url: varchar – URL que identifica o servidor

**CLASSE: Perfil**

Classe que armazena os perfis definidos

nome: varchar – nome que identifica o perfil

descricao: varchar – descrição do perfil

**CLASSE: TipoDocs**

Classe que armazena descrições de tipos de documentos

strTipo : varchar – string do tipo MIME

extensao: varchar – extensão a atribuir aos ficheiros deste tipo

descricao: varchar(100) – descrição do tipo de documento

*PACOTE: Dados de Execução*

**CLASSE: Item**

Classe que armazena a informação relativa a uma recolha da publicação (item).

urn: varchar – nome gerado para o item

estado: varchar – estado da colecção do item

dirBase: varchar – directoria onde vão ser armazenados os conteúdos recolhidos

dataIniPrev: timestamp – data de inicio de colecção prevista

dataMaxTermino: timestamp – data máxima de termino da colecção do item

**CLASSE: Doc**

Classe que armazena informação acerca dos documentos que compõem um item

urlServ: varchar – URL do servidor que aloja o documento

urlDoc: varchar. – localização do documento base no servidor

dataCole: timestamp – data de colecção

prof: int – profundidade

estado: int – estado final de colecção

tipoDoc: varchar – tipo MIME

nomefich: varchar – nome do ficheiro que armazena o conteúdo do documento

nlinks: int – numero de links extraidos

tam: int - tamanho do ficheiro recolhido (em bytes)