Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Gerador Automático de Arquivos HTML de Ajuda para Aplicação em Educação a Distância (GAAHA)

Paulo Sérgio Salla Sá

Orientação: Prof. Dr. Dilvan de Abreu Moreira

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação de São Carlos - USP, para a obtenção do título de Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional.

USP - SÃO CARLOS
2000

Dedico este trabalho a minha esposa e aos meus filhos pelo respeito e compreensão demonstrados neste período.

“You can put all the pretty toolbars you want, but if your web site doesn't have a good search engine, you're going to frustrate people.”

(Network World, February 8, 1999)

Agradecimentos:

Ao Professor Dilvan, meu orientador, pelo apoio na definição do trabalho e orientação, pela extrema confiança em mim depositada, pelas ajudas inestimáveis, pela compreensão e paciência com a minha situação particular.

Aos demais professores do ICMC, que colaboraram diretamente para minha formação acadêmica no mestrado.

Aos amigos de casa: Emerson, Claúdio Hirose, Marcos, Andre, Flávia e todos do grupo de desenvolvimento.

Ao meu amigo Emerson de Presidente Prudente, pelo apoio e pelas várias "injeções" de animo nas horas difíceis e de desconsolo.

Os meus agradecimentos aos amigos Walter, Samuel, Ari, Sueli, João Luiz e Tony pelas contribuições de valores inestimáveis, alem de proporem soluções em circunstâncias por vezes penosas, que tanto me estimularam.

Ao pessoal do Polo Computacional de Rio Claro, que tanto me ajudaram, motivaram e estimularam nos momentos difíceis: Fábio, Lazinho, Leandro, Sidney e Rejane.

À Marcia C. Bueno e Gláucia M. S. Cristianini pela ajuda com a bibliografia.

Às funcionárias da Biblioteca do ICMC e da Unesp de Rio Claro.

Às funcionárias da secretaria do ICMC pela atenção e respeito sempre presente nos esclarecimentos.

À minha mãe, sogra, irmãos e cunhados pelo apoio e compreensão dos vários churrascos e feijoadas que não pude participar devido a dedicação no desenvolvimento deste projeto.

E um eterno agradecimento pelo amor, paciência e compreensão que minha esposa e os meus três filhos, mesmo sendo crianças, tiveram comigo. Principalmente nos momentos de intenso nervosismo, conseqüência dos árduos momentos e estresse que passei durante o período deste mestrado.

Sumário.

Lista de Figuras.

Lista de Tabelas.

Lista de Abreviaturas.

ABED – *Associação Brasileira de Educação a Distancia*

API - *Aplication Programming Interface*

CGI – *Common Gateway Interface*

DBMS - *Database Management System*

DLL - *Dynamic Link Librarys*

DNS - *Domain Name Service*

EAD – *Educação a Distância*

e-Mail – *Electronic Mail*

FTP – *File Transfer Protocol*

GAAHA – *Gerador Automático de Arquivo Html de Ajuda*

GUI – *Graphical User Interface*

HTML – *HyperText Mark-up Language*

HTTP – *HyperText Transfer Protocol*

IR - *Information Retrieval*

JDBC – *Java Database Connectivity*

OOP – *Object Oriented Programming*

SGML - *Standard Generalized Markup Language*

SQL - *Structured Query Language*

SSI - *Server Side Includes*

TCP/IP – *Transmission Control Protocol/Internet Protocol*

URL – *Uniform Resource Locator*

VRML – Virtual reality Modelling Language

WWW – *World-Wide Web*

WAIS - *Wide Area Information Service*

Resumo.

O objetivo deste trabalho é o desenvolvimento de um software: o Gerador Automático de Arquivo Html de Ajuda (GAAHA) para a geração de arquivos de indexação e ajuda em HTML baseado na análise automática de documentos. Esses arquivos de indexação e ajuda têm uma dinâmica muito rápida de atualizações de seu conteúdo. Este trabalho focaliza principalmente: a manutenção de arquivos de ajuda para documentos de cursos para Educação a Distância (via WWW) e a criação de um formato de arquivos de ajuda (Help files) eficiente, para facilitar e agilizar a pesquisa de tópicos ou palavras nesses cursos. O aluno, que participa de um treinamento ou curso de educação a distância, poderá aproveitar o sistema de ajuda para pesquisa sobre tópicos relacionados ao curso ou treinamento em questão.

O programa GAAHA é uma solução para a análise do conteúdo de cursos residentes em *Web Sites* e posterior criação de um mapa gráfico preciso, intuitivo e interativo. O GAAHA tem um robô de "parseamento" responsável pela geração de uma vista hierárquica de todos os *links* de um curso e uma base de dados indexada que é utilizada para pesquisas de palavras-chave entradas pelo usuário. Este sistema é composto por três subsistemas: um agente *crawler* para a indexação das informações, um agente *mentor* que é responsável pela pesquisa no banco de dados, e um cliente WWW para interface com os usuários.

Nessa tese também é apresentado um breve histórico das ferramentas e métodos que foram utilizados para o desenvolvimento desse software e uma breve explanação do conceito de educação a distância. Esta última com o objetivo de esclarecer o quão importante é, para um aluno que participa deste tipo de educação, ter uma ferramenta de pesquisa rápida e eficiente.

Abstract.

The main goal of this project is the development of a computer program: the Automatic Generator of HTML Help Files (in Portuguese GAAHA) for the generation of index and help files in HTML based in the automatic analyses of documents. Index and help files have a very fast rate of changes in their contents. This work is main focus is in the maintenance of help files for distant learning courses documents and the creation of an efficient format for help files to ease topics and words searches in those courses. The student taking part in a distant learning course can take advantage of this help system to research about topics related to the course.

The GAAHA system is a solution for the analyses of the contents of courses hosted in a Web sites and the generation of precise, intuitive and interactive maps and index data. It has a softbot for parsing the course's files to generate a hierarchical view of all links of the course and an indexed database used for searching key words entered by users. The whole system is divided in three parts: a crawler agent for information fetching and indexing, a *mentor* agent for searches in the databank, and a WWW user interface.

Also in this work is shown a brief history of tools and methods used in the software development and a brief explanation of the concept of distant learning. The goal of this explanation is to show how important is, for students taking part in this kind of learning, to have access to fast and efficient searching and indexing tools.

1. Introdução.

A área de informática tem sido caracterizada como uma daquelas em que os profissionais mais têm buscado aprimoramento constante e dinâmico, como forma de resposta à competitividade e dinamismo do mercado. Em especial cada vez mais tem sido requerida uma visão abrangente da informática em geral e dos processos de gestão de recursos de informática em particular, tanto para profissionais acadêmicos como aqueles do mercado não acadêmico.

Com o avanço das tecnologias de comunicação e com a disseminação da Internet, cada vez mais se torna imprescindível a elaboração de projetos que possam se beneficiar deste potencial. Nesta linha de raciocínio, tem-se desenvolvido muitas aplicações na área comercial, de pesquisa, de entretenimento e de Educação a Distância (EAD), que se beneficiam do padrão *HyperText Mark-up Language* (HTML) como meio de documentação e treinamento. Este padrão oferece características atraentes, tais como a possibilidade de se escrever um material uma vez e usá-lo em qualquer plataforma. Ele também é mais fácil de trabalhar, possibilitando adicionar às páginas detalhes sutis, como aplicações Java e clipes de som.

Esses documentos, escritos em HTML, tem uma evolução de conteúdo muito dinâmica, *softwares* tem novas versões e cursos (de educação a distância ou treinamento) mudam seus conteúdos freqüentemente. Em adição, os programas de EAD vem se intensificando e se tornando uma tendência marcante nos últimos tempos, inovando a forma de aprendizagem e possibilitando vários tipos de interação entre alunos e professores. Este tipo de aplicação se apresenta como uma das maneiras para se resolver o problema da concentração de potencialidades em alguns centros de excelência e da difusão dessas potencialidades a locais que não possuem tanta tecnologia e estejam geograficamente distantes.

A dinâmica rápida de mudança de conteúdo da documentação HTML, cria o problema de como manter esses documentos indexados corretamente e como manter arquivos de ajuda (*Help Files*) eficientes para os usuários deste material.

O objetivo deste projeto, é justamente resolver o problema exposto acima, focalizando principalmente o campo da educação e treinamento a distância, através do desenvolvimento de um *software*: o Gerador Automático de Arquivo Html de Ajuda (GAAHA) para a geração de arquivos de indexação e ajuda em HTML, oferecendo assim, uma maior rapidez, eficiência e elegância na procura e visualização de palavras e tópicos desejados.

2. Educação a Distância (EAD).

2.1. Caracterização.

A Educação a Distância é mais antiga do que parece, pois já contabiliza mais de um século de existência. Seus primórdios remontam ao ano de 1881 quando William Rainey Harper, primeiro reitor e fundador da Universidade de Chicago, ofereceu, com absoluto sucesso, um curso de Hebreu por correspondência. Em 1889 o Queen's College do Canadá deu início a uma série de cursos a distância, registrando sempre grande procura pelos mesmos motivos, principalmente, seu baixo custo e às grandes distâncias que separam os centros urbanos daquele país [1].

A EAD é caracterizada:

- Pela separação do professor e do aluno no espaço e/ou tempo [2] [3] [4].
- O controle da aprendizagem é realizado mais intensamente pelo aluno do que pelo professor instrutor [5].
- A interação entre alunos e professores é mediada por alguma forma de tecnologia [2] [6] [7].

"A Educação a Distância é uma aprendizagem planejada que normalmente ocorre em um local diferente do tradicional e como resultado requer projeto de curso e técnicas instrucionais especiais, métodos especiais de comunicação eletrônica ou outra tecnologia, bem como sistemas organizacionais e administrativos especiais" [8].

As tecnologias utilizadas para fornecer informação na EAD, tem o objetivo de promover interação entre professor/aluno e fornecer o *feedback* para o aluno a distância [4].

Desde a época do seu surgimento até os tempos atuais, a Educação a Distância foi sendo desenvolvida utilizando-se dos mais variados ferramentais pedagógicos possíveis, dependendo de fatores tais como: as características da escola e dos professores, o tipo de curso ministrado, da distribuição geográfica entre escola e alunos e, principalmente, a tecnologia disponível e a relação custo/benefício para o uso da mesma. Em função, principalmente da tecnologia de transmissão de informação adotada, a evolução da Educação a Distância pode ser dividida em três fases cronológicas, ou gerações [9] [10].

A primeira foi a geração textual, que se baseou no auto-aprendizado com suporte apenas em simples textos impressos, o que ocorreu até a década de 1960. A segunda foi a

geração analógica, que se baseou no auto-aprendizado com suporte em textos impressos intensamente complementados com recursos tecnológicos de multimídia, tais como gravações de vídeo e áudio, o que ocorreu entre as décadas de 1960 e de 1980. A terceira é a atual geração digital que se baseia no auto-aprendizado com suporte quase que exclusivamente em recursos tecnológicos altamente diferenciados, que podem ser distinguidos pelos seguintes fatores [11] [12] :

- A eficiência e o baixo custo dos modernos sistemas de telecomunicação digital e via satélite.
- A alta interatividade e o baixo custo dos modernos computadores pessoais.
- A amplitude e o custo acessível das redes computacionais locais e remotas, tais como as Intranets e a Internet.

O uso do ferramental pedagógico atualmente disponível pela EAD, permite o oferecimento de condições assíncronas de aprendizado, que podem, e devem ser combinadas com o ferramental do sistema convencional, permitindo uma combinação estreita de grande flexibilidade e alta eficiência no aprendizado final. Uma outra particularidade da EAD é que as modernas tecnologias, atualmente disponíveis, permitem o oferecimento de múltiplas combinações de ferramentas pedagógicas, modernas e tradicionais, com inegável e significativo melhoramento da relação custo-benefício de implantação e manutenção dos programas de pós-graduação nestes moldes [1].

2.2. História da Educação a Distância.

O primeiro curso por correspondência, nasceu na Inglaterra no final do século XIX sendo Sir Issac Pitman, da empresa *Correspondence Colleges*, o principal responsável por esta forma de transmissão de conhecimentos. Este primeiro curso por correspondência, tal como os que lhe seguiram, foi criado com o intuito de poder dar formação a um grupo de pessoas que, por motivos geográficos, de comodidade, econômicos ou sociais, não podiam deslocar-se aos tradicionais locais de ensino [13].

Com o decorrer dos anos, a evolução tecnológica foi também pondo a descoberto a dificuldade que os técnicos, das mais diversas áreas laborais, tinham em se adequar às novas tecnologias adotadas e a educação a distância foi a solução mais rentável e eficaz para resolver esse grande problema [13].

O emprego da informática em um sistema de EAD pode viabilizar melhorias na qualidade de ensino em larga escala e a custos reduzidos. A interatividade pode ser incorporada neste processo através da hipermídia interativa, a qual integra textos, imagem, vídeos, fotos, som e animação, proporcionando um ambiente de ensino atraente, envolvente e multissensorial [14].

Tecnologias anteriores de EAD como correio, TV e vídeo, têm tendências para um ensino com pouca interação. Ao contrário da WWW, por exemplo, que é uma ferramenta descentralizada e que cria o potencial para novas formas de interação entre alunos, reativando a crença de que "a tecnologia contribui com a educação" [15].

Em finais dos anos 60 eis que surge a Internet, e tudo se torna ainda mais simples, pois as distâncias são substancialmente encurtadas, até que em 1993, com o *World Wide Web*, se atinge o ponto mais alto neste desenvolvimento. Este ambiente visual com amplas potencialidades, permite que se utilizem inúmeras tecnologias de apoio a educação a distância e o crescimento espetacular existente nos últimos anos é um reflexo disso.

2.3. Porque Ensinar a Distância.

Segundo Matos [13], as principais características e vantagens da educação a distância podem resumir-se da seguinte forma:

- É abrangida uma maior diversidade de alunos, que de outra forma seria impossível.
- O estudante e o professor normalmente não se encontram no mesmo espaço físico, o que é vantajoso para casos onde seria muito dispendioso que isso acontecesse.
- O estudante não se desloca aos locais tradicionais de ensino, exceto nos casos em que seja necessário algum apoio de material existente em laboratórios ou em oficinas.
- Os horários praticados pelo aluno não são rígidos, muito pelo contrário, são bastante flexíveis permitindo o aproveitamento do tempo livre.
- O ritmo adotado pelo aluno é de sua exclusiva responsabilidade.
- Os temas de aprendizagem são mais vastos do que nos tradicionais locais de ensino.
- É possível receber contribuições de pessoas que, por razões de disponibilidade, não o poderiam fazer em um sistema tradicional.

Todas estas vantagens podem ainda ser acrescidas se houver uma "humanização" do espaço ou meio onde são lecionados os cursos, tornando-os o mais interativos possível e aperfeiçoando-os com constantes remodelações e atualizações de forma a que o ambiente seja dinâmico e não um espaço onde simplesmente é publicada informação. Deve também haver o cuidado de expor claramente os conteúdos para não suscitarem dúvidas e, no caso de estas existirem, devem ser prontamente retiradas e esclarecidas.

3. Internet.

A Internet é uma rede de redes, interligando computadores no mundo todo, como mostra a figura 1. Em julho de 1996 o número de máquinas conectadas era de aproximadamente 10 milhões, e a estimativa é que no final de 2000 possa haver 200 milhões de usuários no planeta, conforme o estudo de Ravet e Layte [16].

A Internet iniciou um novo conceito na comunicação, possibilitando a transmissão de textos, arquivos, imagens e sons, dependendo da capacidade do equipamento utilizado.

Segundo Ravet e Layte [16], a Internet é composta por:

- Uma grande quantidade de informação disponível: milhões de páginas de textos e gráficos, mas também som, vídeo, animação, simulação e programas de computador que podem ser copiados da rede para cada computador com um *click* do mouse.
- Informação distribuída: nós podemos comunicar, co-produzir, cooperar, co-aprender, interagir.
- Informação em tempo real: a distribuição da informação é imediata.
- Simulação distribuída: também é possível várias pessoas participando de uma simulação de locais diferentes.
- Informações disponíveis na *World Wide Web* vêm crescendo com grande velocidade [17].

Com o uso de "seminários virtuais", emergiram toda uma série de questões que chamam a atenção não só pela necessidade de um formato específico para cursos pela rede, mas também pela peculiaridade da linguagem e das mensagens que este meio propicia. Laaser [18] destaca que suas características especiais impedem que possam ser considerados equivalentes aos seminários presenciais, entretanto se apresentam como uma opção adicional que pode resultar no enriquecimento do aprendizado, se seu potencial for explorado adequadamente.

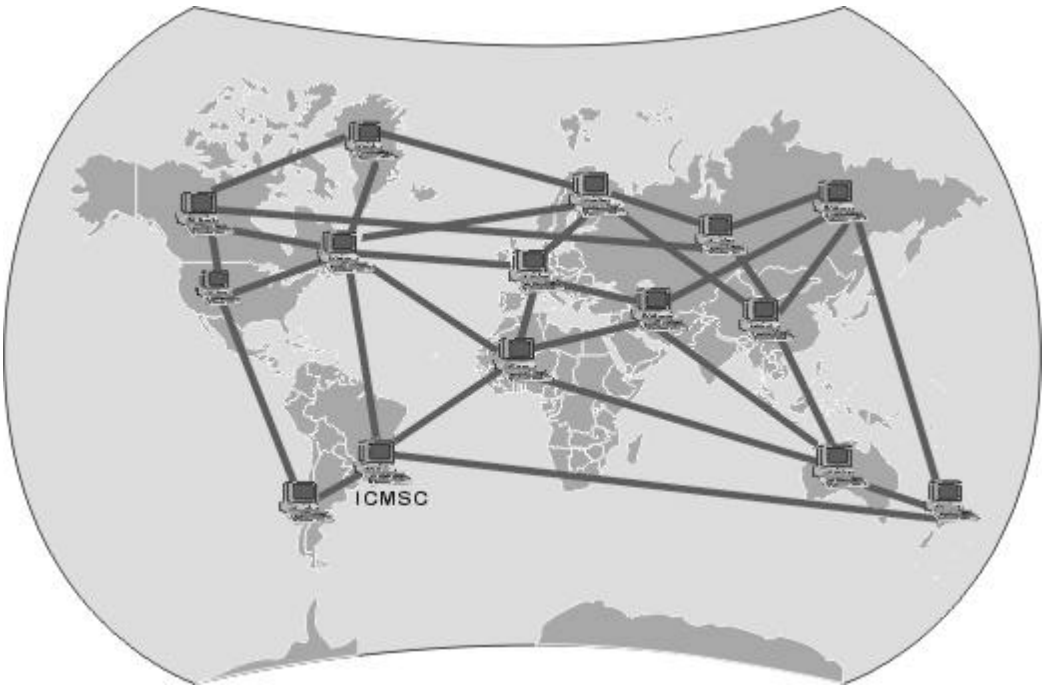

Figura 1 - Internet.

Entre os exemplos práticos desta nova aprendizagem, está o relato de Maki e Maki [19] que transformaram um curso de Introdução em Psicologia na North Dakota State University, que costumava atender mais de 400 alunos por trimestre em um curso via Internet, que substituiu as palestras por leituras de textos e atividades semanais. O curso foi oferecido para alunos fora e dentro do campus, utilizando *Web-browser*, *e-mail*, um livro texto e o *software* de demonstração/simulação que acompanhava o livro. O *site* continha informações sobre o curso, tais como: o sistema de avaliação e como se comunicar com o professor, relatório de desempenho do aluno, sugestões e novidades.

O respectivo curso contou com encontros presenciais para orientação e trabalhos individuais e em grupo, e 71% dos alunos afirmou que participaria de outro curso *on-line*, sendo que 15% não aprovaram a experiência (o restante foi neutro).

Agostinho, Lefoe e Hedberg [20] relatam a experiência de um curso *on-line* sobre aprendizado colaborativo, oferecido pela pós-graduação em aprendizado baseado em tecnologia da University of Wollongong, Austrália, para o qual foram criados vários espaços, onde os alunos poderiam acessar a agenda semanal, informações sobre o curso, avaliações, arquivos, mensagens do professor, referências, local de trabalho e fórum de discussão.

Os autores sugerem que os estudantes sejam treinados para o uso da tecnologia, que haja suporte técnico permanente e que, nas primeiras semanas de curso não sejam solicitados

trabalhos que façam parte do conceito final. Ressaltam a maior demanda de trabalho de um curso *on-line* e na importância de apoio para o professor nas questões técnicas.

Em todos os trabalhos mencionados, a necessidade de espaços para os alunos se comunicarem em tempo real e disponibilizarem trabalhos preparados previamente, o cuidado e o tempo necessário para desenvolver o curso e acompanhar os alunos estão presentes. As observações sobre a importância de suporte pedagógico e técnico também são citadas.

3.1. Educação a Distância na Internet.

Em primeiro lugar, vale realçar que o aparecimento de novas tecnologias para a Internet (especialmente para WWW), assim como o aumento constante da largura de banda disponível, permitem utilizar novas ferramentas mais poderosas, que não existiam há alguns anos atrás. Com este panorama favorável, a Internet vai certamente tornar-se no principal veículo utilizado para a educação a distância.

Segundo Matos [13], as vantagens inerentes a este ambiente cada vez mais familiar são muitas, sendo as mais importantes:

- Ensino centralizado no aluno.
  - O ensino é centralizado no aluno e não no professor, o que fomenta a colaboração entre os estudantes e proporciona um método menos rígido de aprendizagem, tornando-a mais interativa e mais interessante.
- Conveniência.
  - Os horários de trabalho e o local de ensino são bastante flexíveis permitindo o desenvolvimento de atividades paralelas por parte do aluno, como por exemplo, exercer uma profissão e, nas horas vagas, fazer o seu curso sem ter problemas de incompatibilidades .
- Ferramentas muito fáceis de utilizar.
  - As ferramentas (computador, clientes dos diversos serviços Internet utilizados, etc.), normalmente utilizadas para o apoio do aluno, são extremamente simples de utilizar e permitem uma rápida ambientação às mesmas.

- Desenvolvimento dos ambientes virtuais rápido e fácil.
  - As empresas e universidades que disponibilizam cursos *on-line* também têm a sua tarefa facilitada devido a existência de ferramentas de composição de criação das páginas *Web*. Hoje em dia, a construção de um *site* na Internet, ainda que com as devidas exceções, demora muito pouco tempo a projetar e ainda menos a concretizar.
- Aproveitamento dos recursos já existentes.
  - A grande quantidade de recursos atualmente disponível na Internet permite que o estudo de um determinado tema se possa reduzir a uma série de *links* para material já existentes, o que poupa imenso tempo. A facilidade de encontrar os mesmos, também aumenta de dia para dia com a implementação de poderosos motores de procura gratuitos, destinados a todos os usuários em nível mundial.
- Fácil alteração dos conteúdos temáticos.
  - O formato eletrônico dos documentos permite alterações simples e sem grandes custos de tempo, o que é uma grande vantagem quando há falta de recursos humanos disponíveis para o fazerem.
- Informação e tecnologias em formato padrão.
  - Neste ambiente virtual, tudo o que é publicado e todas as tecnologias utilizadas são neste momento acessíveis em qualquer parte do globo e num formato uniformizado, para que não haja incompatibilidades impeditivas do bom funcionamento deste sistema.

Com todas estas vantagens, percebe-se que o futuro da educação a distância, passa pela Internet. A maior tendência visível é a de um crescimento exponencial para os anos que se avizinham.

3.1.1. Fases de um Sistema de Educação a Distância.

Um curso ensinado via Internet, bem como em outro sistema de educação a distância, deve ser muito bem preparado para que o aluno não tenha qualquer tipo de dúvidas na aquisição de conhecimentos. Uma boa estruturação do mesmo, permite que a sua implementação inicial, bem como qualquer alteração que seja necessária, se torne um

problema de fácil resolução, sem grande perda de tempo e ocupação de recursos.

O desenvolvimento instrucional é essencial para a EAD. Fornece um processo e ambiente de trabalho para planejamento, desenvolvimento e adaptação da instrução, baseada nas necessidades do aluno e nos requisitos do conteúdo. Este processo consiste de algumas etapas básicas de projeto, desenvolvimento, avaliação e revisão [21]. Seguir os princípios do desenvolvimento instrucional, exibidos na figura 2, não superará todos os obstáculos encontrados no desenvolvimento de programa eficazes de EAD, afirma Willis [21], entretanto, fornecerá procedimentos para direcionar as mudanças instrucionais que certamente ocorrerão.

Este método processa-se em quatro fases, que resumindo, se podem explicar da seguinte forma: no início definem-se as prioridades, objetivos e qual vai ser o público alvo; em seguida faz-se uma estruturação dos conteúdos a ensinar, tentando aproveitar tudo o que já existe publicado; numa fase posterior avalia-se o que está pronto e tenta-se acrescentar algo que pareça necessário; por fim, retificam-se alguns erros que possam existir e, visto este processo ser iterativo, caso seja necessário fazer alguma alteração mais tarde, reinicia-se o processo [13].

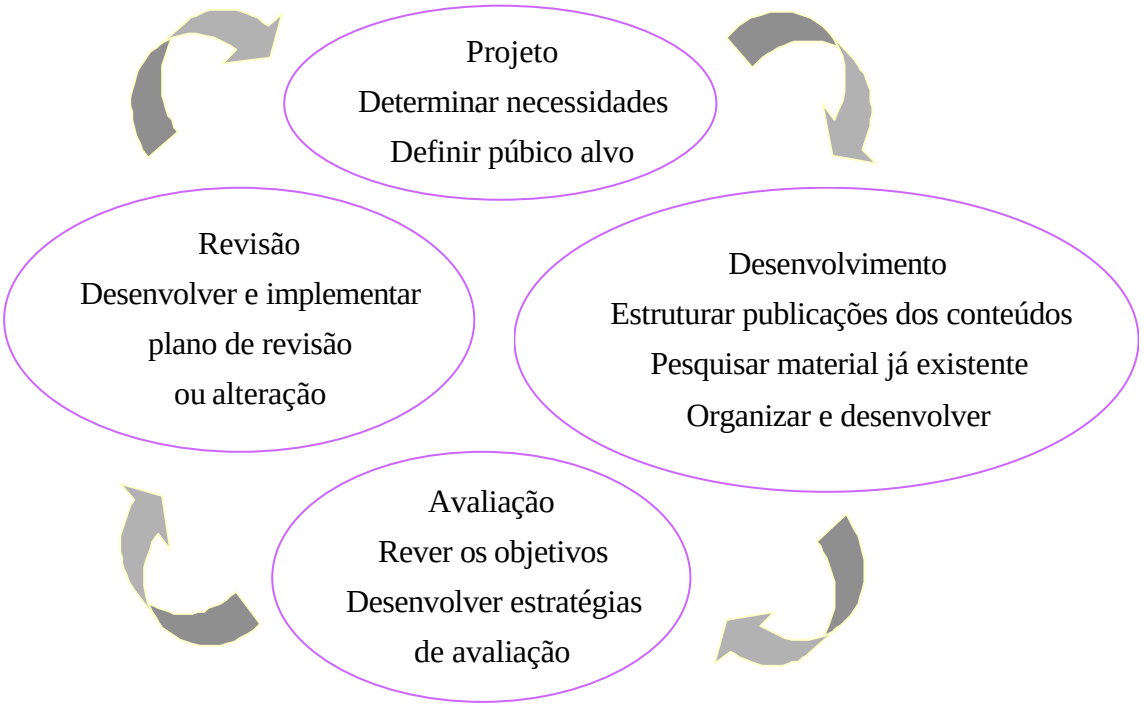

Figura 2 - Processo de desenvolvimento instrucional.

3.2. A *World-Wide Web* (WWW).

Em 1989, a WWW foi desenvolvido no European Particle Physics Laboratory (CERN) em Genebra pelo cientista da computação Tim Berners-Lee [23]. Segundo Berners-Lee, et all. [23] a meta da WWW era ser um monopólio do conhecimento humano tão fácil de atualizar quanto de ler. É descrita oficialmente como "uma iniciativa de recuperar informação hipermídia por uma vasta área, visando fornecer um acesso mundial a um grande universo de documentos" [24].

A WWW é um sistema de hipermídia executado na Internet [25], cujo veículo de informação padrão é o documento hipertexto, um arquivo texto elaborado em HTML (*HyperText Markup Language*) [26]. Estes documentos podem conter textos, sons, figuras, vídeos e podem ser interativos [27]. Através de palavras sublinhadas (*links*), pode-se recuperar outros documentos escritos por diferentes autores, em várias localizações, e até mesmo recuperar informações multimídia. O sucesso da WWW se deve à apresentação de informação de uma forma não linear [22] e somente nos últimos anos é que a WWW ganhou popularidade, merecendo atenção de fabricantes e pesquisadores interessados na expansão de aplicações sob o ambiente distribuído e multi-plataforma proporcionado por esta tecnologia [72].

3.2.1. Arquitetura *Web*.

A figura 3 mostra a arquitetura simplificada da *Web*, que é uma típica arquitetura cliente/servidor [73] [74]. De um lado fica o cliente, que é composto por *browsers Web* [100] [23] capazes de exibir e solicitar documentos da rede. Opcionalmente, o cliente pode estar acompanhado por aplicativos externos usados na apresentação do documento, ou parte destes, caso o *browser* sozinho não seja capaz de interpretar algum tipo de dado (por exemplo, sons ou imagens em movimento). Do outro lado da arquitetura *Web* fica o servidor, composto pelo servidor *Web* [75] [76], cuja principal função é atender os pedidos dos clientes *Web* por documentos armazenados no sistema de arquivos da plataforma onde se encontra instalado. Dependendo do pedido do cliente *Web*, o servidor *Web* pode disparar uma aplicação externa como, por exemplo, a execução de um programa via interface padrão *CGI (Common Gateway Interface*) [87] [88] [89].

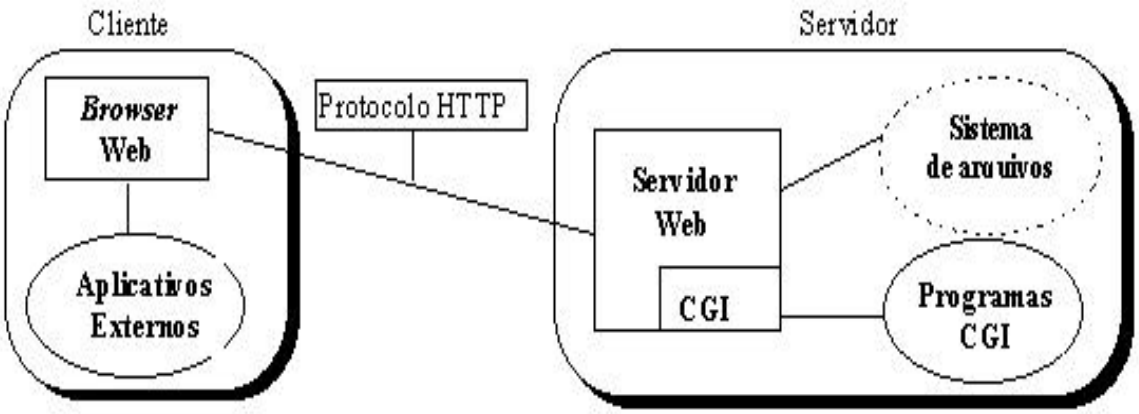

Figura 3 - Arquitetura *Web* simplificada.

Sua composição compreende múltiplos computadores "servidores", os quais podem enviar os documentos para os usuários da Internet que "navegam" de servidor para servidor, através de um *software browser* (navegador), permitindo o acesso a WWW [27]. Este *software* lê documentos e pode buscá-los de outras fontes; acessar arquivos por FTP, Gopher e uma variedade de outros métodos; permite pesquisas de documentos e bases de dados, bem como, interfaces para acessar qualquer tipo de programa externo executado em um servidor [25]. Alguns dos protocolos de comunicação que o *browser* WWW pode acessar, são [26]:

- HTTP (*HypeText Transfer Protocol*) – protocolo usado nas conexões a páginas disponíveis em *site* na WWW.
- FTP – (*File Transfer Protocol*) – protocolo utilizado para a transferência de arquivos de uma máquina para outra na Internet.
- Gopher – Facilita a localização de arquivos através de menus.
- Wais (*Wide Area Information Service*) – É um serviço que permite que o usuário tenha acesso à base de dados na Internet e efetue buscas por documentos através de palavras-chave.

Uma característica muito importante da *Word Wide Web* é que ela é baseado em três padrões abertos que permitem transferir a informação, descrever a formatação da informação e localizar a informação distribuída pela rede:

- Transferir a informação: Para o transporte da informação entre o servidor e o cliente Web foi proposto um protocolo de comunicação denominado HTTP [90] [75] [91]. Sua principal característica é ser um protocolo aberto e especializado na

transmissão de documentos *Web* sob a Internet. O protocolo HTTP é um protocolo *stateless*, ao contrário de diversos outros protocolos Internet que são *stateful*. Isto significa que o cliente *Web* envia o pedido de uma operação ao servidor *Web*, este atende o pedido e logo em seguida é encerrada a conexão. Nenhuma informação sobre a solicitação do usuário é mantida no servidor *Web*. A característica *stateless* do protocolo HTTP, proporciona eficiência e velocidade necessárias a sistemas de informação hipermídia distribuídos como a *Web*, mas origina vários problemas para o desenvolvimento de aplicações que exijam, por exemplo, a identificação do usuário.

- Descrever a formatação da informação: Para a apresentação e formatação de documentos na *Web* é utilizada a linguagem padrão HTML [92] [93] [76] [94], que permite que a estrutura dos documentos *Web*, bem como os vínculos (*links*) a outros documentos e recursos da Internet, sejam incorporados diretamente no formato texto. Documentos codificados em HTML podem ser interpretados pelos *browsers* e formatados segundo as características de cada plataforma em que são exibidos. A linguagem HTML é composta por um conjunto de elementos, denominados *tags*, que permitem ao usuário especificar a estrutura de um documento *Web*, como por exemplo, títulos, parágrafos, cabeçalhos, citações e vínculos hipertextos para outros documentos. Ela é baseada no padrão SGML (*Standard Generalized Markup Language*) [95] e, como tal, independe da plataforma em que é exibida, cabendo ao cliente *Web* interpretá-la de acordo com sua configuração. Outra importante característica da linguagem HTML é a possibilidade de geração de formulários (*forms*) contendo ícones e campos para preenchimento de dados pelos usuários. Os formulários proporcionam um maior dinamismo, pois permitem que as páginas *Web* funcionem como a "porta de entrada" para aplicações que necessitem de uma interação com o usuário. Para finalizar, é importante ressaltar que um documento *Web* não precisa estar no formato HTML. O cliente *Web* pode exibir documentos em diversos formatos, incluindo arquivos textos convencionais sem nenhuma formatação e arquivos com figuras no formato GIF ou JPEG, por exemplo.
- Localizar a informação: Para a identificação e localização de documentos WWW distribuídos pela Internet foi proposta a utilização de um formato para endereçamento de documentos denominado URL (*Uniform Resource Locator*) [97]. A sintaxe de uma URL é : <Protocolo>://<Host>/<Path>

/<Doc>[<#Location>], onde Protocolo indica o tipo de recurso Internet que deve ser usado para a conexão com os servidores, que pode ser, como já citado neste capítulo, HTTP, FTP, NEWS, WAIS e FILE; Host é o nome da máquina *host* na Internet segundo o DNS (*Domain Name Service*) [98], *Path* é uma lista de diretórios separado por barras; *Doc* o nome do documento ou de um programa a ser executado pelo servidor *Web* e *Location* é uma marca textual opcional de posição no documento. Por exemplo, a URL "http://www.icmc.sc.usp.br/staff/alunos/alunos.html" informa ao cliente *Web* para fazer uma conexão Internet usando o protocolo HTTP, cujo documento encontra-se na máquina *host* "www.icmc.sc.usp.br", no diretório "/staff/alunos/" e o arquivo é "alunos.html".

O servidor informa ao cliente qual tipo de arquivo é acessado. Quando o *browser* não entender o formato de arquivo encontrado (por exemplo: um arquivo no formato Adobe Acrobat), uma janela de ajuda é aberta para o usuário, com o objetivo de informá-lo. Como já citado antes, o cliente pode usar qualquer tipo de programa afim de exibir informações multimídia, como imagem, som ou vídeo [23].

A comunicação entre cliente e servidor *Web* é sempre feita na forma de pares requisição/resposta e é sempre iniciada pelo cliente. Inicialmente, o cliente envia uma mensagem de sincronização para o servidor pedindo abertura de conexão. O servidor responde aceitando ou não este pedido. Se aceitar, a conexão é estabelecida. Uma recusa pode indicar que o servidor está sobrecarregado e não consegue tratar novos pedidos de conexão que chegam. Normalmente, existe um número máximo de pedidos pendentes. Um pedido de conexão pode ser recusado se este limite tiver sido atingido.

Uma vez estabelecida a conexão, o cliente envia uma mensagem com a requisição feita pelo usuário, representada por uma URL. Esta mensagem é enviada através da rede para o servidor, que busca a informação no disco local ou dispara a execução de uma aplicação externa. Após o processamento do pedido, uma mensagem de resposta é enviada ao cliente que, ao recebê-la, retorna um pedido de fechamento de conexão.

Quando o cliente recebe a informação solicitada, é necessário verificar se os dados são válidos, através de informações contidas em campos do cabeçalho da mensagem. Somente após esta verificação o cliente pode mostrar o documento para o usuário. Para tanto, ele precisa saber qual o tipo e o formato do arquivo. Esta informação também é obtida no cabeçalho da mensagem. Todo este processamento impõe um atraso no cliente, que é refletido no tempo de resposta observado pelo usuário [49].

A figura 4 mostra a configuração cliente-servidor da WWW.

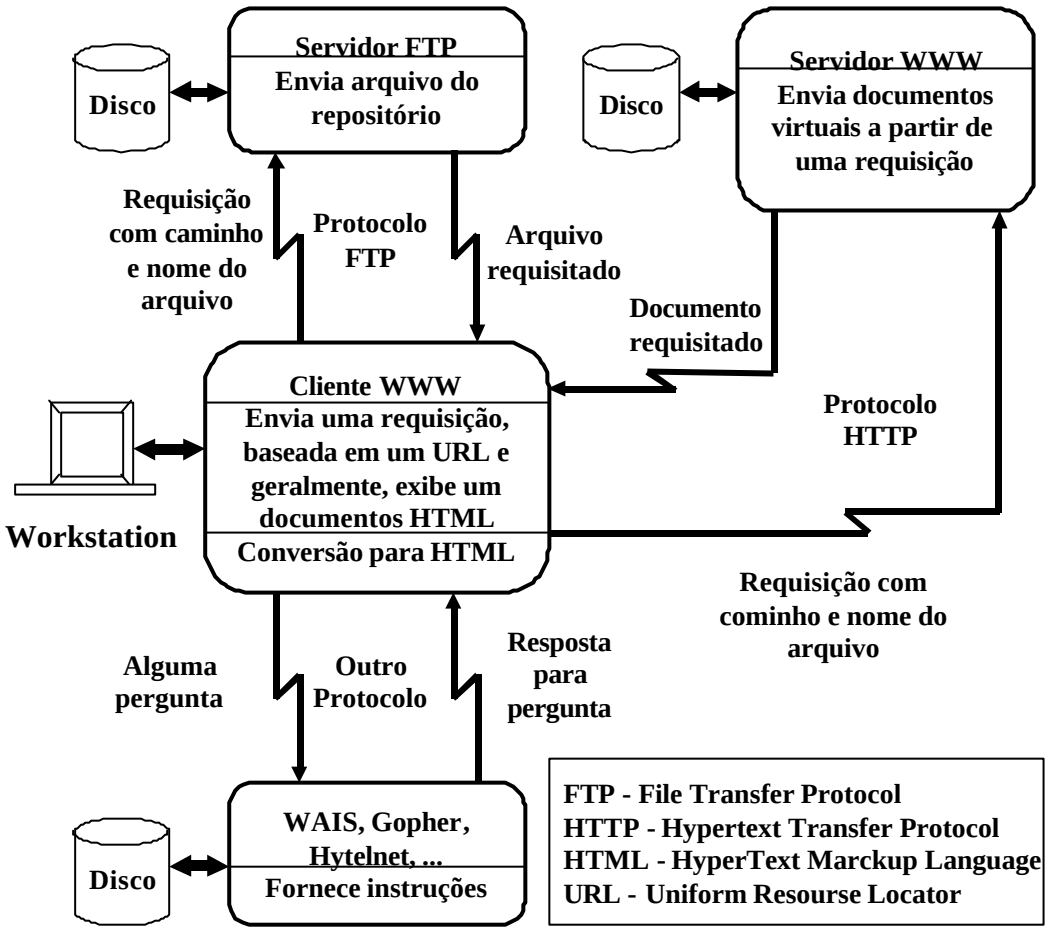

Figura 4 - Configuração Cliente/Servidor da WWW.

O *World Wide Web* foi talvez o grande responsável pelo sucesso do ensino via Internet. O ambiente agradável e de fácil navegação que oferece, conjugado com a possibilidade de integração de som e imagem, são os fatores que contribuíram para esse sucesso. A quantidade enorme de ferramentas disponíveis para a criação de *Web sites* e o fácil manuseamento das mesmas, permite também que a WWW esteja ao alcance de qualquer pessoa, sem necessidade de se ter grandes conhecimentos de informática para conceber uma *home page* com qualidade.

Alem dessas virtudes, a possibilidade de um professor poder expor claramente os temas que pretende lecionar, fazer exercícios sobre os mesmos e ainda ter ligações nessa mesma página para outras que já existam com material semelhante, torna este ambiente muito

atrativo.

Por sua vez, o aluno tem à sua disposição, uma quantidade enorme de informação sobre os temas que desejar investigar e estudar.

É muito comum incluir nestas páginas *Web* as chamadas listas de discussão, ou seja, um conjunto de perguntas e respostas que esclarecem as duvidas mais freqüentes e que fomentam a participação do aluno em debates sobre determinados assuntos de interesse comum.

Em suma, a WWW pode ser considerada um dos principais veículos de informação usado no mundo inteiro e a formação a distância, ganha novos contornos usando-o para se expandir [13].

4. Projetos de EAD em Andamento.

Em todo o mundo, em particular nos países mais desenvolvidos, a EAD está em franca expansão, sendo largamente implantada por meio de programas de grande porte. Isto tem ocorrido com mais intensidade nos países de grande extensão territorial, como Canadá, Estados Unidos e Austrália, que estão na fronteira avançada do uso extensivo dos processos de EAD, com inúmeros programas, a maioria promovido por suas melhores e maiores universidades e empresas. Alguns países da América Latina, como México e Venezuela, também possuem programas significativos de EAD. Alguns exemplos de programas dignos de nota por seu reconhecimento mundial de eficiência e qualidade [28] [29] [30] são:

- Master's Programme at a Distance for IBM – programa multinacional envolvendo Estados Unidos e Canadá que é operado pela IBM através de duas empresas denominadas Skill Dynamics - USA e Skill Dynamics - Canadá, desenvolvendo programas de pós-graduação em nível de Mestrado, com suporte acadêmico da Syracuse University.
- Universidad Nacional Experimental Simon Rodriguez (UNESR) – programa venezuelano que envolve toda uma instituição universitária de caráter privado e que opera em mais de 30 campi, contando também com o apoio acadêmico da Syracuse University.
- Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) – instituição mexicana de ensino superior de caráter privado que mantém um programa à base de EAD denominado "Sistema de Mejoramiento Continuo" envolvendo 26 campi com 44 programas de pós-graduação e 33 de graduação, com dois canais de satélite integralmente disponíveis e contando com o apoio técnico da Carnegie Mellon University.
- Open Learning Australia (OLA) – programa multi-institucional australiano, com alcance em toda a Oceania, envolvendo 40 instituições das quais 20 são universidades, com um poderoso sistema de EAD complementado por tecnologias de mídia convencional (rádio, televisão), oferecendo centenas de cursos de diversos níveis, inclusive de graduação e de pós-graduação.
- Electronic University Network (EUN) – instituição universitária totalmente dedicada à EAD e afins, situada na Califórnia, considerada a maior universidade *on-line* do mundo, já tem 14 anos de existência e já formou mais de 25.000 alunos através de cerca de 300 cursos.

No Brasil a EAD ainda apenas se inicia, não havendo atualmente, mais que alguns raros programas operando efetivamente. Outros poucos encontram-se em fase de projeto ou de implantação [31]. Dentre os programas implantados no Brasil, com a EAD aplicada formalmente destaca-se:

- Universidade Federal de Santa Catarina (UFSC) – notável programa de pós-graduação em Engenharia de Produção (mestrado e doutorado), envolvendo uma rede estadual de oito universidades oficiais e privadas, além de diversas empresas de porte tecnológico significativo;

Outros programas, sem a mesma formalidade programática do acima citado, e que podem ser considerados como mais importantes no Brasil são [1]:

- Faculdade Carioca – programa de graduação que está sendo implantado com base no Lotus Notes da IBM, envolvendo cerca de mil alunos em inúmeras disciplinas das áreas de Informática, Administração, Economia, Ciências Contábeis, Comunicação Social, Desenho Industria, Matemática e Letras.
- Escola do Futuro da Universidade de São Paulo (USP) – programa mantido pela Escola de Comunicações e Arte da USP (ECA-USP) que oferece gratuitamente uma série de cursos via BBS tais como Astronomia, Tratamento de Imagens e Atualização de Professores de 1º e 2º graus.
- Universidade Federal Paulista (antiga Escola Paulista de Medicina) – seu Centro de Informática na Saúde (CIS-EPM) disponibiliza na Internet, além de outros serviços, programas de educação em Biologia Molecular e Engenharia Genética.

Também a Universidade Católica de Brasília realizou estudos para a implantação de um programa formal de pós-graduação *Stricto Sensu* baseado em EAD.

Em 21 de junho de 1995 foi fundada a ABED - Associação Brasileira de Educação a Distância. A ABED é uma sociedade científica sem fins lucrativos que, entre outras coisas, promove congressos Internacionais de Educação a Distancia e também prêmios de excelência em Educação a Distância [32].

Os principais objetivos da ABED são:

- Estimular o estudo, a pesquisa e o desenvolvimento de projetos em educação a distância em todas as suas formas.

- Incentivar a prática da mais alta qualidade de serviços para alunos, professores, instituições e empresas que utilizam a educação a distância.
- Apoiar a "industria do conhecimento" do país procurando reduzir as desigualdades causadas pelo isolamento e pela distância dos grandes centros urbanos.
- Promover o aproveitamento de "mídias" diferentes na realização da educação a distância.
- Fomentar um espírito de abertura, de criatividade, inovação, de credibilidade e de experimentação na prática da educação a distância.

4.1. O Futuro da Educação a Distância.

O futuro da educação a distância passa obrigatoriamente pela Internet. O desenvolvimento de novas tecnologias como a videoconferência, o VRML, JAVA e o crescimento da largura de banda existente, vão permitir que daqui a poucos anos a integração de som e imagem sejam mais freqüentes, dando um aspecto mais dinâmico e atrativo aos *Web sites*, contribuindo para uma maior adesão de estudantes a este sistema. As capacidades multimídia oferecidas pela Internet estão ao alcance de milhões de pessoas que, cada vez mais se conscientizam que daqui a não muitos anos, os seus filhos não terão necessariamente que se deslocar à uma escola para aprender [13].

Spennemann [33] diz que a composição pedagógica da EAD não deve apenas resolver as questões das grandes distâncias. Deve também, e principalmente, buscar suprir as necessidades de interatividade do aluno com o tema de estudo, bem como valer-se do ferramental tecnológico disponível como forma de aperfeiçoar os aspectos pedagógicos do ensino, permitindo cumprir os principais fatores de uma educação centrada no aprendizado interativo, dinâmico e contextualizado.

O objetivo deste trabalho se enquadra perfeitamente na frase citada logo acima por Spennemann, ou seja, com um processo de acesso rápido e intuitivo o aluno realmente terá uma interatividade com o tema de estudo. Interatividade esta, fornecida por um modelo gráfico e portanto mais sugestivo de posicionamento em assuntos desejados e também pela possibilidade de acesso a um banco de dados referente ao conteúdo de um curso de EAD.

5. Integrando *Web* com Banco de Dados.

5.1. *Scripts Web* e a Interface CGI.

Como um dos objetivos deste projeto é o desenvolvimento de uma interface de pesquisa a um banco de dados via *Web*, para posterior geração de páginas HMTL dinâmicas, foi realizado um estudo um pouco mais detalhado de algumas dessas soluções. Estas soluções incluem o uso da interface CGI, APIs de servidores *Web*, SSIs ou através de linguagens de programação como Java.

Dependendo da aplicação, o documento consultado por meio de uma URL é estático, ou seja, um arquivo armazenado na plataforma do servidor *Web*. No entanto há situações, na qual este projeto é enquadrado, em que é necessário que os documentos sejam gerados em tempo de execução (*on line*) através da interação com o usuário. Antes mesmo do surgimento da *Web* já existiam muitos serviços de informações *on line*, como catálogos de bibliotecas, serviços de recuperação de informações e acesso a Banco de Dados. Para proporcionar serviços dinâmicos como estes no ambiente *Web*, pode-se usar programas auxiliares que são executados na plataforma do servidor quando solicitados pelo usuário. Estes programas são chamados *scripts Web*.

Um *script Web* é um programa que pode ser executado pelo servidor *Web* em resposta a um pedido de um cliente *Web*. Pode ser um código compilado a partir de um programa escrito em C, C++ ou Java, um código interpretado como Perl ou Php ou, por exemplo, um *shell script* do Unix. Um *script Web* pode chamar outros programas ou contactar outros servidores. Seu principal objetivo é gerar uma informação dinâmica (em tempo de execução) para o cliente *Web* que o solicitou. Por exemplo, um *script* pode usar os campos de entrada de um formulário HTML para gerar e enviar uma consulta apropriada a um servidor de Banco de Dados remoto. Cabe ao *script* retornar os dados num formato que o cliente *Web* consiga interpretar, como por exemplo, no formato HTML.

O servidor *Web* é responsável por realizar as seguintes operações quando inicia um programa *script*:

- Avaliar se o programa *script* pode ser executado. Por exemplo, o servidor NCSA HTTPd [102] para sistemas UNIX, requer que os programas *scripts* estejam localizados em um diretório específico (ex: "/cgi-bin"), com permissão de execução apropriados. Outros servidores UNIX podem ter regras diferentes. Em

servidores *Web* para Windows NT pode-se executar programas *scripts* em qualquer diretório desde que ele tenha extensão ".EXE", por exemplo.

- Iniciar o programa *script* e garantir que os dados de entrada do cliente *Web* serão passados para o programa *script*.
- Receber os dados retornados pelo programa *script* e os enviar de volta ao cliente *Web*.
- Enviar uma mensagem de volta ao cliente *Web* se algo de errado acontecer com o programa *script*.
- Encerrar a conexão de rede apropriadamente, quando o programa *script* finalizar sua execução.

Para a implementação destas ações foi especificada a interface CGI [87] [88] [89]. CGI é uma interface padrão, implementada pela maioria dos servidores *Web* existentes, para a execução de programas *scripts* externos ao servidor Web. A interface descreve como os programas *scripts* serão inicializados e como os dados serão passados entre estes e o servidor Web.

Os detalhes de implementação das regras CGI são dependentes do sistema operacional em que reside o servidor *Web*. Para sistemas UNIX, por exemplo, os programas *scripts* são executados como processos separados, os dados são enviados pelo servidor *Web* para a entrada padrão (*stdin*) e os resultados são retornados ao servidor *Web* através da saída padrão (*stdout*). Este exemplo mostra que a interface CGI é de fato um conjunto de regras padrão, uma para cada sistema operacional existente [75].

O processo de decodificação dos dados enviados pelo servidor *Web* para ser usado pelo programa *script* CGI é tedioso e exige um amplo conhecimento da linguagem do programa *script* que vai tratar os dados [76]. Felizmente, já existem várias bibliotecas de funções de domínio público escritas em diversas linguagens que realizam a tarefa de decodificação dos dados enviados pelo servidor *Web*, como por exemplo, a biblioteca cgihtml [101] desenvolvida para programas *scripts* que usam a linguagem C e também a do Perl CGI.

Existem algumas limitações no uso da interface CGI. A principal delas está relacionada com problemas de desempenho, especialmente em aplicações multi-usuário. Por exemplo, aplicações CGI não podem ser compartilhadas por vários usuários clientes. Mesmo se uma aplicação CGI ainda estiver sendo executada, quando da chegada de novos pedidos ao servidor *Web*, este inicia um novo processo da aplicação CGI. Quanto mais pedidos forem chegando, mais processos concorrentes serão criados na plataforma do servidor. O fato de se

criar um processo para cada pedido consome tempo e grande soma de recursos de memória, deteriorando o desempenho da aplicação.

5.1.1. Programas Executáveis CGI.

Esta arquitetura é composta por um ou mais programas CGI, geralmente implementados usando-se alguma linguagem de programação hospedeira do DBMS. Esta é a arquitetura mais imediata para se desenvolver aplicações *Web* Banco de Dados. De fato, para se usar a interface CGI implementada pelos servidores *Web* é necessário uma linguagem que possa ler da entrada padrão, obter as variáveis de ambientes e escrever na saída padrão. Como a grande maioria dos DBMSs existentes incorporam alguma linguagem de programação com estas características, praticamente todos eles podem se comunicar com a *Web* por meio de Programas Executáveis CGI sem grandes custos adicionais [49]. A figura 5 ilustra esta arquitetura.

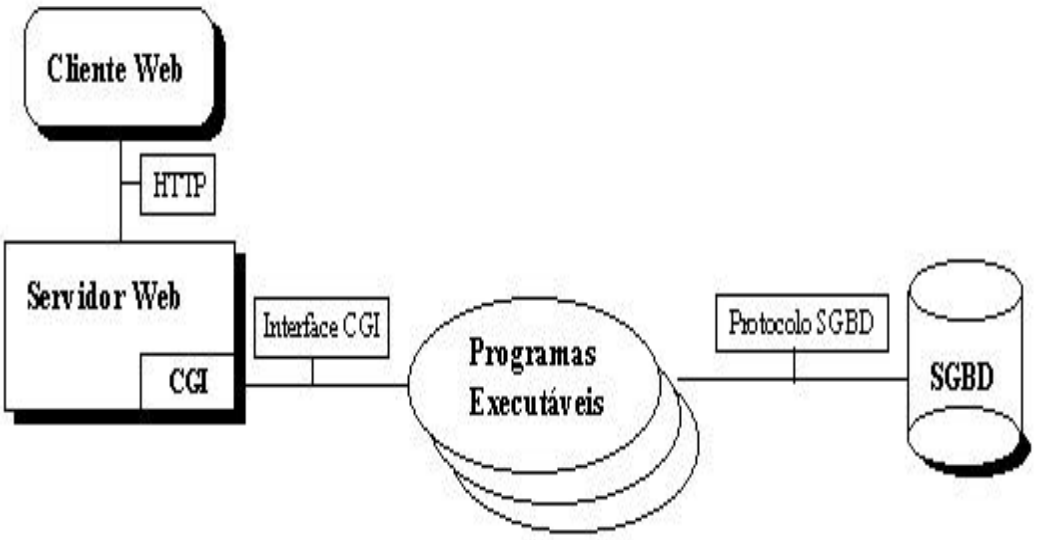

Figura 5 - Arquitetura de Programas Executáveis CGI.

O funcionamento da arquitetura é mostrado a seguir:

- Cliente *Web* solicita ao servidor *Web* a execução de uma aplicação externa, que neste caso é um programa executável na plataforma do servidor *Web*.
- Servidor *Web* dispara um processo para execução do programa através da interface padrão CGI, enviando os dados recebidos do cliente *Web* de acordo com a implementação do servidor *Web* para a interface CGI.
- O programa recebe os dados passados pelo servidor *Web* via interface CGI, decodifica-os, formula o comando SQL e abre uma conexão com o Banco de Dados para sua execução.
- O DBMS retorna os dados e a conexão é finalizada. O programa executável trata os dados recebidos e os repassa ao servidor *Web* via interface CGI, num formato que o cliente *Web* entenda, como por exemplo, no formato HTML. O processo iniciado pelo servidor *Web* para execução do programa é finalizado neste momento.
- O Servidor *Web* retorna os dados repassados pelo programa ao cliente *Web*.

5.1.2. Gerenciador de Aplicação CGI.

De forma a suplantar os diversos problemas da arquitetura CGI anterior como a dificuldade de gerenciar eficientemente o estado da aplicação, segurança no acesso ao Banco de Dados e otimização de consultas, pode-se implementar uma arquitetura de integração *Web* Banco de Dados dividido em dois módulos: vários despachantes (*dispatchers*), que são pequenos programas executáveis, e um (ou mais) gerenciador que coordena a aplicação. Esta arquitetura dá origem à arquitetura de integração denominada Gerenciador de Aplicação CGI. A figura 6 ilustra esta arquitetura.

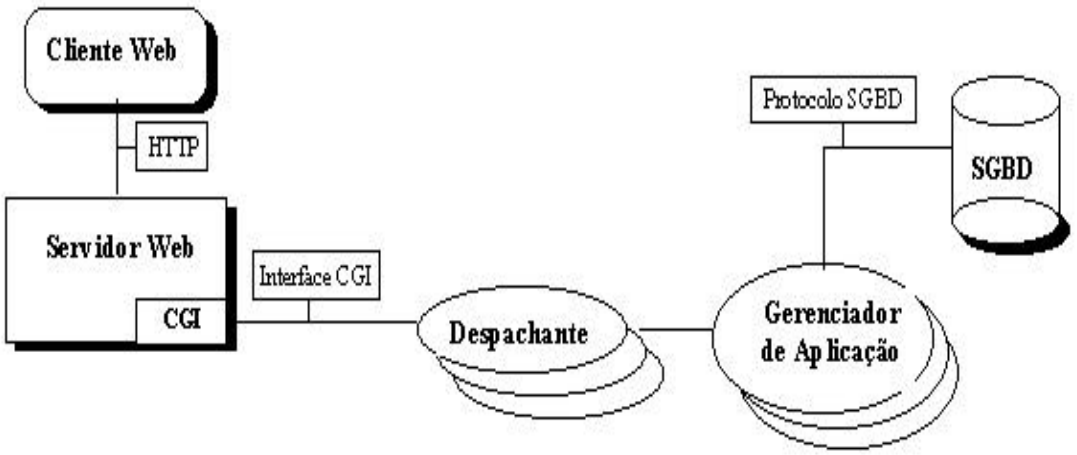

Figura 6 - Arquitetura Gerenciador de Aplicação CGI.

Um despachante CGI pode ser implementado sem nenhum conhecimento do SGBD. Analogamente, os Gerenciadores de aplicação são implementados sem conhecimento da interface CGI. Como os Gerenciadores de Aplicação mantêm a conexão com o Banco de Dados aberta após a execução de algum pedido do despachante, pode-se tomar vantagens integrais do esforço de otimização do SGBD, além de ser possível gerenciar o estado da aplicação eficientemente.

5.2. Servidor *Web* Estendido.

As arquiteturas que se encontram nesta categoria não usam a interface padrão CGI para execução das aplicações *Web* Banco de Dados. Ao contrário, as funcionalidades do servidor *Web* são estendidas de forma a proporcionar acesso mais otimizado e mais rápido a aplicações externas [103]. Ganha-se portanto em desempenho quando comparado às arquiteturas que usam a interface CGI, mas perde-se em portabilidade da aplicação, ficando o desenvolvedor limitado às características do servidor *Web* proprietário.

5.2.1. SSIs e APIs de Servidores *Web*.

A interface CGI não é o único meio de gerar documentos *Web*. Existem atualmente duas outras técnicas que também podem realizar esta tarefa, além de serem úteis para estender as funcionalidades dos servidores *Web*. São as SSIs (*Server Side Includes*) [104] [105] [106] e as APIs (*Application Programming Interface*) [105] [107] de servidores *Web* que serão descritas nas seções seguintes.

5.2.1.1. SSIs.

SSIs são trechos de códigos, também denominados *tags*, inseridos em um documento *Web* e que serão executados pelo servidor *Web* no momento em que um cliente solicitar o documento. É uma funcionalidade proprietária de servidores *Web* presentes, por exemplo, no servidor NCSA HTTPd [106] e no servidor WebQuest [104].

Por meio de códigos SSI é possível obter dados dinâmicos num documento *Web* de forma mais simples do que através de programas *scripts* CGI. Por exemplo, o servidor pode mostrar a última data de modificação de um documento solicitado pelo cliente através da inserção de uma *tag* do tipo "<!--#echo var = "LAST_MODIFIED" ->" que o servidor interpreta usando a variável ambiente LAST_MODIFIED definida na especificação SSI do servidor *Web*.

A sintaxe das *tags* SSIs segue, na maioria das vezes, o formato de um comentário HTML, como em:
<!--#"<tag><variável>"-->, onde <tag> é um dos comandos que serão executados e <variável> é um conjunto de variáveis de ambiente SSI. Implementações SSIs podem ter *tags* especiais como 'EXEC' (para execução de funções na linha de comando do sistema operacional), 'IF' (para desvio condicional) ou 'ODBC' (para execução de comandos SQL em um Banco de Dados relacional via driver ODBC). A *tag* 'EXEC', em particular, permite que sejam executados programas externos ao servidor *Web* em substituição à interface CGI.

Apesar da simplicidade, existem duas grandes desvantagens de se usar SSIs para o desenvolvimento de aplicações na *Web*. Em primeiro lugar o servidor *Web* deve verificar em cada documento *Web* solicitado a existência de códigos SSIs. Isto pode afetar o desempenho do servidor. Segundo, é um mecanismo não muito seguro sob o ponto de vista de acesso aos

dados. Algumas *tags* SSIs podem permitir que usuários clientes executem códigos na plataforma do servidor *Web* sem qualquer restrição, como por exemplo, por meio da *tag* 'EXEC'.

Para amenizar estes problemas os servidores *Web* podem ser configurados de forma a limitar o uso de algumas *tags* SSIs (como 'EXEC') a alguns diretórios, ou até mesmo a completa desativação de *tags* selecionadas. Pode-se também, configurar o servidor *Web* de forma a limitar o número de documentos *Web* para os quais o servidor deva verificar a existência de código SSI. Por exemplo, pode-se configurar o servidor *Web* para analisar a presença ou não de *tags* SSIs nos documentos que tenham somente a extensão ".shtml".

Figura 7 - Arquitetura de Aplicação SSI.

5.2.1.2. APIs.

Uma API de servidor *Web* é semelhante a uma API para desenvolvimento de aplicações em Banco de Dados cliente/servidor. Trata-se de um conjunto de funções que são incorporadas ao servidor *Web* podendo substituir completamente a interface CGI para a execução de programas *script*. As APIs, na realidade, podem realizar muito mais, permitindo, entre outras coisas, melhorar significativamente o desempenho de aplicações externas *Web*. São exemplos de servidores *Web* que incorporam APIs os servidores da Netscape [108], o servidor IIS da Microsoft [109], o servidor Apache [110] e o servidor WebSite [111].

As APIs resolvem os problemas de limitação e de eficiência comuns às aplicações que usam a funcionalidade CGI. Dentre estes destacam-se:

- Os programas escritos como APIs de servidores *Web* ficam residentes em memória uma vez iniciados e, portanto, executam muito mais rápido do que quando a interface CGI é usada. Uma solução integrada ao servidor *Web* como a proporcionada pelas APIs é bem mais eficiente, como ressaltado em [105] [112] [64]. De fato, mais de um cliente *Web* pode executar a mesma função API simultaneamente, e ao contrário de programas CGI, não é inicializado um novo processo para cada usuário distinto. Somente novas variáveis de memória são inicializadas, sendo usadas técnicas de sincronização como semáforos [74] para gerenciar a aplicação.
- As APIs permitem o compartilhamento de dados e recursos de comunicação com o servidor *Web*, ao contrário de programas CGI. Aplicações APIs executam no mesmo espaço de memória em que o servidor *Web* está executando e, em algumas implementações [108], pode-se ter acesso inclusive às estruturas de dados internas do servidor *Web*.
- A interface CGI é limitada ao envio e retorno de dados de/para o servidor *Web*. As APIs estendem esta característica permitindo, por exemplo, o desenvolvimento de funções para controlar o acesso a um Banco de Dados, lidar com erros específicos de execução de programas ou ainda o desenvolvimento de mecanismos auxiliares para a autenticação (verificação de senhas) de clientes *Web*.

As diferenças funcionais entre programas APIs de servidores *Web* e programas CGIs são:

- Uma aplicação API recebe os dados do servidor *Web* por meio de canais de comunicação próprios da implementação do servidor *Web* e não da entrada padrão (*stdin*) como nos programas que usam a interface CGI.
- O resultado da execução da aplicação API não é enviado para a saída padrão (*stdout*) como nos programas CGI, mas sim recuperados por meio de chamadas de funções APIs escritas especificamente para este fim.

Como já citado, uma vez executada pelo servidor *Web* a aplicação API fica residente no mesmo espaço de memória utilizado pelo servidor *Web*, esperando por novos pedidos do cliente. No entanto, este tempo não é ilimitado. De fato, a maioria das aplicações APIs são finalizadas, liberando recursos do sistema, quando ficam um determinado intervalo de tempo

sem atender pedidos de algum cliente *Web*. A figura 8 ilustra esta arquitetura.

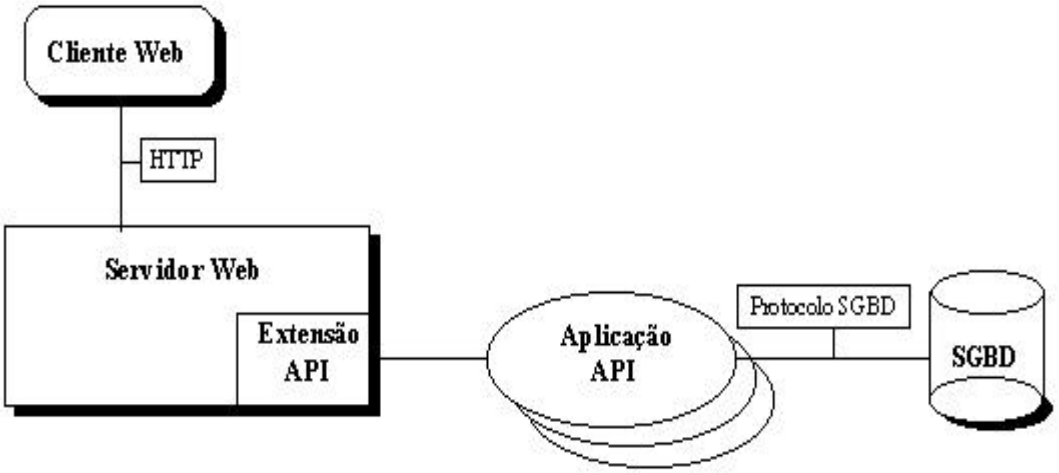

Figura 8 - Arquitetura de Aplicação API.

Apesar das várias vantagens de se usar APIs para desenvolvimento de aplicações, existem algumas desvantagens. São elas:

- São dependentes do servidor *Web* onde são implementadas.
- Podem desabilitar o servidor *Web* (*crash*) se não forem bem escritas e testadas, já que compartilham dos mesmos recursos de memória que o servidor *Web*.
- Podem afetar o desempenho do servidor *Web*, principalmente se os recursos de CPU forem limitados.
    - Programação por meio de APIs é uma tarefa difícil. Isso ocorre porque a compreensão das possíveis interações através de APIs exige um conhecimento profundo do projeto do servidor.

Concluindo, SSI é uma boa técnica para o desenvolvimento de aplicações *Web* dinâmicas, mas é um mecanismo proprietário, limitado e que pode afetar o desempenho do servidor *Web*. Já as APIs são mais flexíveis e permitem estender as funcionalidades dos servidores *Web* porém, também são proprietárias, requerem um profundo conhecimento do funcionamento do servidor *Web*, apresentam alto risco operacional de inabilitar o servidor *Web* e o desenvolvimento de aplicações APIs pode ser demorado se comparado a aplicações que usam a interface CGI.

6. Ferramentas de Programação.

6.1. A Linguagem de programação JAVA.

Este projeto será desenvolvido utilizando a linguagem de programação Java. Java foi escolhida, devido a sua engenharia de linguagem extremamente sólida, grande portabilidade em diferentes arquiteturas sem a necessidade de compilação, modelo de interface gráfica com o usuário, facilidades de programação em rede, e aplicações via *World Wide Web* (WWW) [34].

Uma grande vantagem óbvia é uma biblioteca em tempo de execução que lhe oferece independência de plataforma: pode-se usar o mesmo código em Windows 95, Solaris, Unix, Macintosh, etc. [35]. Isto certamente é necessário para programação na Internet.

Os principais componentes da linguagem Java são um compilador Java, que gera o código fonte Java para uma linguagem intermediária, independente da máquina, denominada *bytecodes* e um interpretador Java, que interpreta e executa os *bytecodes* Java (descarregados sob a rede) nos clientes *Web*. Daí sua principal característica de portabilidade em diferentes plataformas. Para que a linguagem Java execute em uma dada plataforma basta que o interpretador para aquela plataforma esteja presente.

Os programas Java, também denominados applets, permitem estender a linguagem HTML, exibir gráficos e imagens, além de poderem ser usados para validação de campos de entrada em formulários HTML.

Java também é totalmente orientado a objeto – ainda mais do que o C++. Tudo em Java é um objeto, exceto alguns tipos básicos como os números.

Segundo Cornell [35], as vantagens básicas em relação ao C++ são:

- É mais fácil gerar código sem *bugs* usando Java do que com C++.
  Os projetistas do Java pensaram muito sobre o que torna o código tão falível. Depois, incluíram recursos em Java que eliminam a possibilidade de criação de código com os tipos mais comuns de bugs. Segundo Cornell [35], *alguns calculam que aproximadamente cada 50 linhas de código em C++ possui pelo menos um bug.*
- Eliminam a alocação e a desalocação manual de memória.
  O "lixo" da memória em Java é coletado automaticamente. Não há necessidade em se preocupar com "restos" de memória.

- Introduziram verdadeiros *arrays* e eliminaram a aritmética de ponteiro.
  Não é necessário preocupar-se com a gravação sobre uma área importante da memória devido a um erro de cálculo com ponteiro.
- Eliminaram a possibilidade de confundir uma atribuição com um teste de igualdade, em uma expressão condicional.
- Eliminaram a herança múltipla, substituindo-a por uma nova noção de *interface* que derivou do C orientado a objeto.
  As interfaces lhe dão o que você deseja das heranças múltiplas, sem a complexidade inerente ao gerenciamento de diversas hierarquias de herança.

A sintaxe do Java, na realidade é uma versão mais limpa da sintaxe de C++. Não há necessidade de arquivos de cabeçalho, aritmética de ponteiro (nem mesmo uma sintaxe de ponteiro), estruturas, uniões, sobrecarga de operadores, classes de base virtual, etc. [35].

Segundo os autores[1] de Java, os objetivos e realizações do projeto, estão organizados em 11 tópicos [36] :

- **Simples** – *"A intenção era montar um sistema que pudesse ser programado facilmente, sem muito treinamento esotérico, e que influenciasse a prática comum nos dias de hoje. ... Assim, embora considerando que C++ era inadequado, projetamos Java o mais próximo de C++ possível, afim de torná-lo um sistema mais abrangente. Java omite muitos dos recursos de C++ que são raramente usados, mal-entendidos, confusos e que, de acordo com nossa experiência, causam mais problemas do que benefícios".*
- **Orientado a Objeto** – *"De forma simples, o projeto orientado a objeto é uma técnica que focaliza o projeto através dos dados (ou objetos) e das interfaces para eles. Fazendo uma analogia com a carpintaria, um carpinteiro "orientado a objeto" iria preocupar-se primeiro com a cadeira que ele estivesse montando e depois com as ferramentas usadas para criá-las; um carpinteiro "não orientado a objeto" estaria mais interessado nas ferramentas. ... As facilidades orientadas a objeto de Java são essencialmente as de C++".*

[1] Chuck McManis, Chris Warth, Herb Jellinek, Tim Lindholm, Arthur van Hoff, Michelle Huff, Jonathan Payne, Frank Yellin, Patrick Chan, Erik Gilbert, Eugene Kuerner, Mark Scott Johnson, Richard Tuck, Lisa Friendly, Sami Shaio, Bob Weisblat, James Gosling, Kim Polese, Kathy Walrath.

- **Distribuído** – *"Java possui uma ampla biblioteca de rotinas para lidar facilmente com protocolos TCP/IP (Transmission Control Protocol), como HTTP (HyperText Transfer Protocol) e FTP (File Transfer Protocol)"*.
- **Robusto –** *"Java tem por finalidade a criação de programas que sejam confiáveis sob diversos aspectos. Java dá bastante ênfase à verificação inicial em busca de possíveis problemas, verificação dinâmica (tempo de execução) posterior e eliminação de situações possíveis de erro. ... A maior diferença entre Java e C/C++ é que Java possui um modelo de ponteiro que elimina a possibilidade de gravar na memória e alterar seus dados"*.
- **Seguro –** *"Java foi elaborado para ser usado em ambientes de rede/distribuídos. Por causa disso, muita ênfase foi dada à segurança. Java permite a construção de sistemas livres de vírus e falsificação"*.
- **Arquitetura Neutra –** *"O compilador gera um formato de arquivo-objeto como arquitetura neutra – o código compilado pode ser executado em muitos processadores, desde que o sistema local tenha o módulo em tempo de execução de Java. ... O compilador de Java faz isso gerando instruções em bytecodes que nada têm a ver com uma arquitetura de computador em particular. Em vez disso, eles são projetados para serem fáceis de interpretar em qualquer máquina e facilmente traduzidos em código de máquina nativo no momento da execução"*.
- **Portátil –** *"Ao contrário de C e C++, não há aspectos da especificação "dependentes da implementação". Os tamanhos dos tipos de dados primitivos são especificados, assim como o comportamento da aritmética em relação a eles"*.
- **Interpretado –** *"O interpretador de Java pode executar os bytecodes do Java diretamente em qualquer máquina para a qual o interpretador tenha sido transportado. E como a "linkedição" é uma técnica mais incremental e leve, o processo de desenvolvimento pode ser muito mais rápido e exploratório"*.
- **Alto Desempenho –** *"Embora o desempenho dos bytecodes interpretados geralmente ultrapasse o esperado, há situações em que é necessário um desempenho melhor. Os bytecodes podem ser traduzidos (em tempo de execução), em código de máquina da CPU, na qual o aplicativo está rodando"*.
- **Multithreaded –** *"Os benefícios do multithreading são a melhor resposta interativa e o comportamento em tempo real"*.

- **Dinâmico –** *"De diversas maneiras, Java é uma linguagem mais dinâmica do que C ou C++. Ela foi projetada para adaptar-se a um ambiente em evolução. ... As bibliotecas podem adicionar livremente novos métodos e variáveis de instâncias sem qualquer efeito sobre seus clientes. ... Em Java, a descoberta do tipo de dado é em tempo de execução".*

Segundo Eckel [34], semelhante a qualquer linguagem humana, Java prove uma maneira para expressar conceitos. Se bem sucedido, este meio de expressão será significativamente mais fácil e mais flexível do que outras alternativas, quando os problemas se tornam maiores e mais complexos.

Eckel [34] diz ainda que a linguagem Java enfatiza precisão, confiança e conduta no dispêndio de performance. Isto é refletido em características tais como *garbage collection* automática, rigorosas verificações em tempo de execução, completa verificação dos *bytecodes* e um conservador sincronismo de tempo de execução. Sobre performance, McConnel [37] citou : *"Termine-o primeiramente, e aperfeiçoe-o então. A parte que necessita ser aperfeiçoada é geralmente pequena".*

6.2. Introdução ao Servlet JAVA.

Servlets são módulos baseados em solicitações/respostas que executam no lado do servidor, classificando-se como sendo uma API para servidores *web* estendido. Por exemplo, um servlet pode ser responsável por pegar os dados pela ordem de entrada em uma página HTML e aplicá-los em uma lógica de negociação usada em um banco de dados de uma companhia. Servlets são para servidores, assim como applets são para *browsers* [35].

A biblioteca API Servlet, usada para escrever os códigos fontes do servlets, não assume nada sobre como o servlet é carregado, ou o ambiente no qual o servlet é executado, ou o protocolo usado para transmitir dados para o usuário. Isto permite que os servlets sejam embutidos em diferentes servidores *Web*.

Servlets são um efetivo substituto para os *scripts* CGI: eles oferecem um modo para gerar documentos dinâmicos que é fácil de escrever e rápido para executar. Servlets são desenvolvidas com a API Servlet Java, que é uma extensão do padrão Java [38].

Alguns exemplos de utilização para os Servlets [39]:

- Processamento de dados *POSTed* acima do protocolo HTTP usando uma página HTML.
- Permitir cooperação entre pessoas. Um servlet pode manipular solicitações concorrentemente; eles podem sincronizar solicitações para suporte de sistemas tais como um conferencia *on-line*.
- Retransmitir solicitações. Servlets podem retransmitir solicitações para outros servidores e outros servlets. Isto permite balancear o carregamento entre vários servidores que espelham o mesmo conteúdo. Isto também permite o particionamento de um simples serviço lógico entre vários servidores, de acordo com o tipo de tarefa e limites organizacionais.
- Ser uma comunidade de agentes ativos. Um servlet escritor, poderia definir agentes ativos que compartilhassem trabalhos entre si. Os agentes poderiam passar dados entre si.

Um bom exemplo é um servlet HTTP que pode ser usado para gerar conteúdos de páginas dinâmicas no padrão HTML, recurso este que será bastante utilizado no desenvolvimento do sistema GAAHA.

Há várias vantagens em se desenvolver um servlet gerador de conteúdos dinâmicos, tais como [38]:

- servlet é mais rápido e que o *script* CGI.
- Eles usam uma API padrão (a servlet API).

Eles provem todas as vantagens do Java (executam em uma variedade de servidores sem à necessidade de serem reescritos).

Segundo Cornell & Horstmann [35], a vantagem de um servlet Java sobre um *script* CGI é obvia - *scripts* CGI podem fazer várias coisas que arriscam a segurança, e os administradores de sistema são naturalmente extremamente hesitantes sobre instalar *scripts* não testados.

6.3. Uma visão básica da Arquitetura de um Servlet.

Um servlet fornece mecanismos que gerenciam a comunicação com o cliente e o servidor. Quando um servlet aceita uma chamada de um cliente, ele recebe dois objetos: um objeto de solicitação que encapsula a comunicação entre o cliente e o servidor, e um objeto de resposta que encapsula a comunicação do servidor com o cliente.

O objeto de solicitação permite o servlet acessar informações tais como o nome do parâmetro passado pelo cliente, o protocolo que está sendo usado pelo cliente, o nome da máquina remota que fez a solicitação e o servidor que recebeu esta solicitação. O objeto de resposta fornece ao servlet a capacidade de resposta ao cliente.

Os servlets são carregados e executados pelo servidor, aceitando assim, zero ou mais solicitações do cliente e retornando dados para o mesmo. O servidor também tem o potencial de remoção dos servlets inicializados. Quanto a manipulação de solicitações realizadas pelo cliente, somente após o carregamento do servlet isto é possível [38].

6.4. JDBC.

JDBC (*Java DataBase Connectivity*) é uma API Java para execução de declarações SQL (*Structured Query Language*). JDBC fornece uma API padrão completa para desenvolvimento de ferramentas/banco de dados [40].

Usando JDBC, fica fácil transmitir declarações SQL para qualquer banco de dados relacional. Em outras palavras, com a API JDBC, não é necessário escrever vários programas para acessar gerenciadores de banco de dados de diferentes fabricantes. Um simples programa usando a API JDBC, será capaz de transmitir declarações para diferentes gerenciadores de banco de dados. E, com o desenvolvimento de uma aplicação utilizando a linguagem de programação Java, não será necessário se preocupar com a execução desta aplicação em diferentes plataformas. A combinação do Java e JDBC permite o programador escrever um programa apenas uma vez e executá-lo em diferentes plataforma [40].

A figura 9, exemplifica o modelo de acesso de uma aplicação Java à banco de dados.

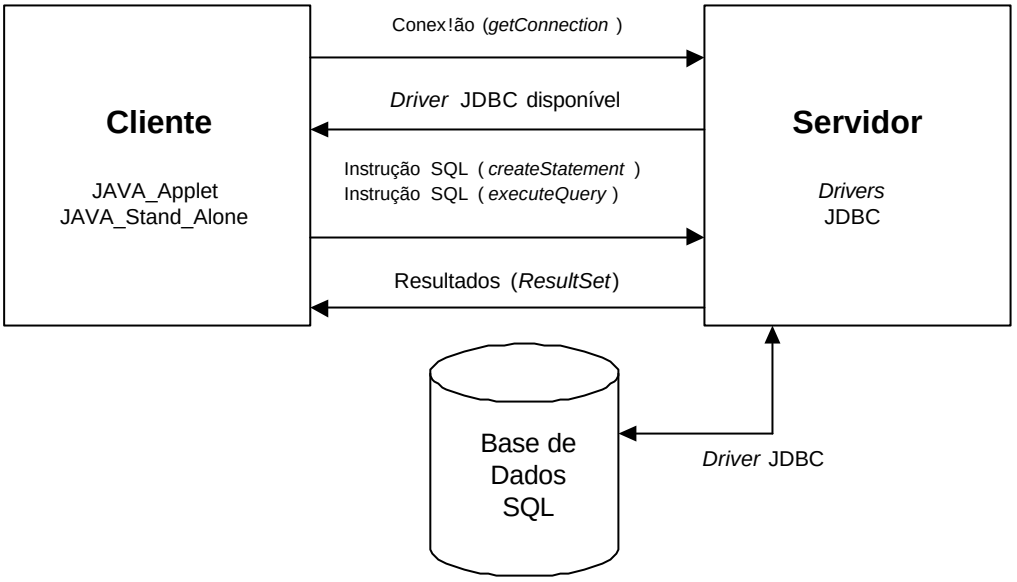

Figura 9 - Modelo de acesso a um DBMS.

6.5. MySQL.

MySQL é um servidor de banco de dados SQL pequeno, rápido, relacional, multi-usuário e *multi-thread*. A linguagem SQL foi criada por Michael Widenius e é a linguagem mais popular de banco de dados no mundo [41].

MySQL é uma implementação cliente/servidor como demostrado pela figura 10, que consiste em um *daemon* chamado mysqld e diferentes programas clientes. Sua utilização é livre para fins acadêmicos. A principal meta do MySQL é a velocidade, robustez e fácil utilização [41].

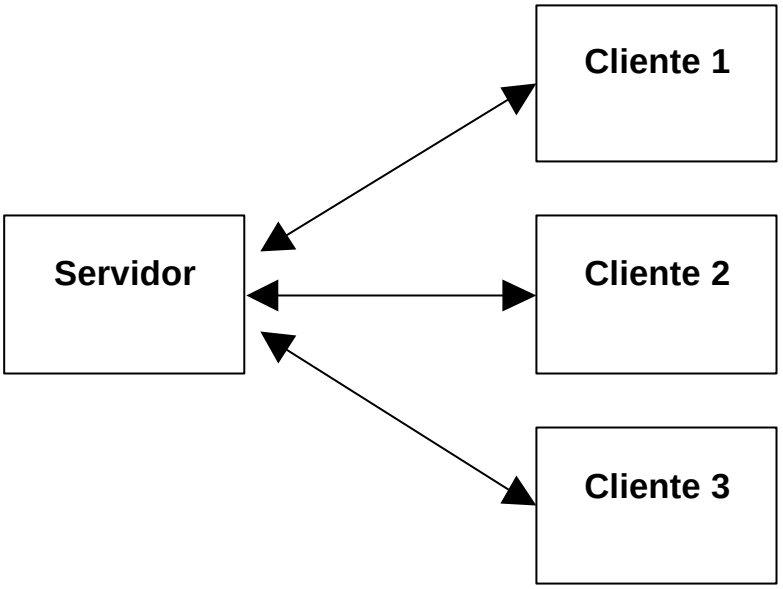

Figura 10 - Modelo Básico de um Cliente/Servidor.

7. Os Agentes de *software*.

Desde o início da história registrada, as pessoas têm-se fascinado com a idéia de agentes não humanos. A noção popular de andróides, humanoides, robôs, *cyborgs* e criaturas de ficção cientifica permeiam nossa cultura, formando um bloqueio inconsciente sobre a capacidade dos agentes de *software*. A palavra robô tem sua origem na palavra Tcheca para "aquele que trabalha com tarefas insalubres" e se tornou popular com a peça RUR: Rossum Universal Robots de Karel Capek de 1921. Enquanto os robôs de Capek eram trabalhadores de fábrica o público alimentava, de tempos em tempos, o sonho de ter robôs como empregados domésticos digitais que, como a empregada mecânica do desenho animado "Os Jetsons," poderia cuidar de tarefas mundanas da rotina de suas casas.

Talvez pelo fato de ter surgido em uma temática não propriamente científica, a imagem dominante perante o público em geral do que seria uma criatura corporificando a inteligência artificial tornou-se mais um pesadelo que um sonho. Seria possível que os robôs pudessem reverter a relação mestre-escravo com os humanos? Todos os dias os usuários de computadores se deparam com mistérios nos *softwares*, verdadeiras charadas ou *bugs*, incompreensíveis e perigosos, que reiteram o medo no poder que criaturas autônomas poderiam representar. Quanto mais inteligente o robô, maior a sua capacidade de identificar interesses próprios para sobrepor os interesses do mestre. Quanto mais parecido com o ser humano, mais próximo o robô estará de manifestar falhas e excentricidades humanas.

Apesar de autômatos de vários tipos terem existido por séculos, somente com o desenvolvimento dos computadores, em particular após a II Guerra Mundial, é que os agentes autônomos começaram a surgir. Para Norman [50] os predecessores mais relevantes dos agentes inteligentes de hoje são os servomecanismos e outros instrumentos de controle, incluindo controles de fábricas, de esteiras industriais, instrumento de vôo e aterrissagem em aeronaves. Significativamente, na atualidade, o foco atual parece ter-se deslocado do *hardware* para o *software*, dos átomos que compõem um robô mecânico para os bits que compõem um agente digital [51].

Alan Kay, um antigo proponente da tecnologia dos agentes, fornece um rascunho das raízes mais recentes dos agentes: “A idéia de um agente tem origem em meados da década de 1950 com John McCarthy. Mas o termo só foi concebido por Oliver G. Selfridge alguns anos mais tarde, quando os dois trabalhavam no Massachussetts Institute of Technology. Eles conceituaram um sistema que, quando a ele fosse dado um objetivo, poderia analisar os detalhes das operações apropriadas, fazer perguntas e solicitar orientações quando estivesse

sem alternativas. Um agente poderia ser um "*software* robô" vivente e operacional dentro do mundo da computação [52]".

Nwana [53] divide a pesquisa dos agentes em dois segmentos principais. O primeiro tem inicio aproximadamente em 1977 com raízes na inteligência artificial distribuída e concentrou-se principalmente em agentes deliberativos com modelos simbólicos internos. Este trabalho tem contribuído para a compreensão da interação e comunicação entre agentes, a decomposição e distribuição de tarefas, coordenação e cooperação, resolução de conflitos via negociação etc. O segundo, com início por volta de 1990, é um movimento recente e crescente que amplia a visão dos tipos de agentes até um nível de maior esperteza. A ênfase foi subitamente alterada da deliberação para a ação; do raciocínio para a ação remota. A grande diversidade de aplicações e abordagens é uma chave para o quanto os agentes estão se tornando um foco central.

Como a variedade de agentes tem proliferado, o termo tem sido largamente aplicado sem consenso sobre seu significado.

Abaixo estão algumas definições de agentes de *software*:

- Programas que permitem que suas tarefas sejam agendadas com antecedência em máquinas remotas (em maneira não muito diferente dos *jobs Batch*).
- Programas que têm o poder de processar algumas tarefas enquanto são instruídos em uma linguagem de programação de um nível mais alto ou através de *scripts* [54].
- Programas que podem abstrair ou encapsular os detalhes das diferenças entre fontes de informação e serviços de computação [55].
- Programas que são capazes de implementar uma "função cognitiva" primitiva ou agregada [56] [57].
- Programas que manifestam características da inteligência distribuída [58].
- Programas que são capazes de desempenhar o papel de mediador entre pessoas e programas [59] [60].
- Programas que desempenham o papel de um "assistente inteligente" [61] [62].
- Programas que podem migrar de computador para computador de acordo com decisão própria [63].
- Programas que podem apresentar-se para o usuário como uma personalidade confiável [65] [66] [67].

- Programas que são vistos pelos usuários com manifestação de intencionalidade e outros aspectos mentais [68].
- Programas que falam uma linguagem de comunicação própria dos agentes [69].

Apesar do espectro tão disperso de definições, duas abordagens distintas mas relacionadas para a definição de agente têm predominado: uma baseada na noção de um agenciamento como algo feito por algumas pessoas e outra baseada na descrição dos atributos que um agente deve possuir. Gilbert [70] em um documento da IBM, descreve os agentes inteligentes em termo do espaço definido por três dimensões de agenciamento, inteligência e mobilidade (Figura 11).

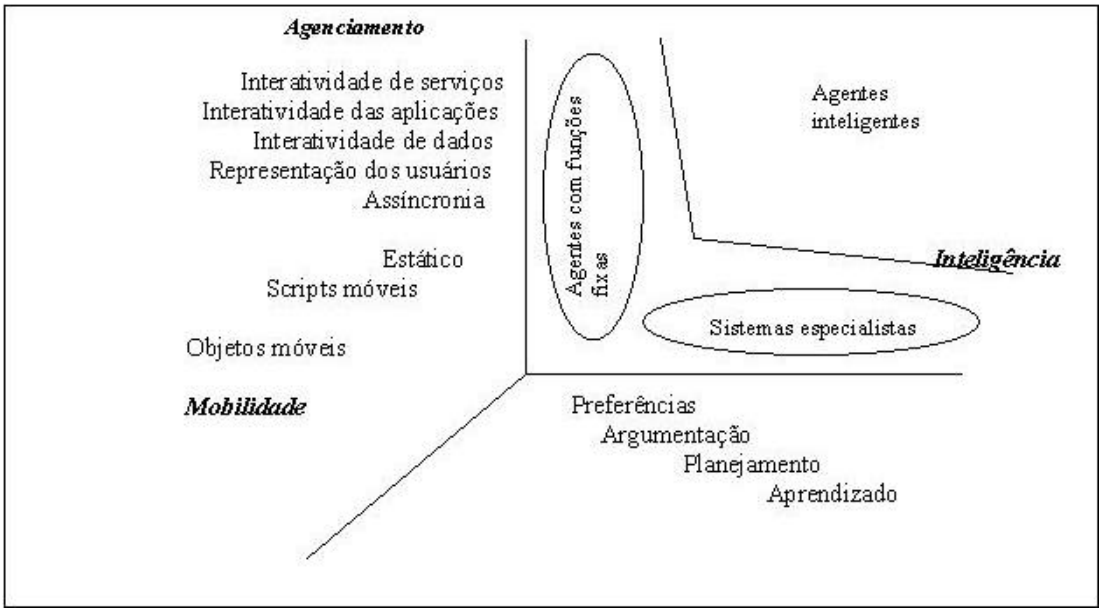

Figura 11 - Escopo dos agentes inteligentes.

"Agenciamento é um grau de autonomia e autoridade adequado ao agente e pode ser medido no mínimo qualitativamente pela natureza da interação entre o agente e outras entidades do sistema. No mínimo, um agente deve "rodar" assincronamente. O grau de agenciamento é aumentado se um agente representa um usuário sob alguma forma...Um agente mais avançado pode interagir com ... dados, aplicações, ... serviços ...(ou) outros agentes.

Inteligência é o grau de argumentação e aprendizado que o agente é capaz de desempenhar: a habilidade do agente em aceitar os comandos do usuário e executar tarefas a ele delegadas. Altos níveis de inteligência incluem uma modelagem do usuário... e argumentação... Em níveis ainda mais elevados na escala da inteligência estão os sistemas que podem aprender e se adaptar a seus ambientes, ambos em termos dos objetivos dos usuários e em termos dos recursos disponíveis para os agentes...

Mobilidade é o grau pelo qual os agentes viajam através da rede... *Scripts* móveis podem ser compostos em uma máquina e endereçados a outra, para execução... (Objetos móveis são) transportados de máquina para máquina durante a execução, além de carregar dados acumulados."

A aplicação de um agente de *software* recuperador de informação na máquina de pesquisa do sistema GAAHA, pode atuar diretamente na superação das dificuldades dos usuários com a obtenção de informações referente a um curso de EAD.

8. Agente Automático de Análise de Texto.

A área de Recuperação de Informações (*Information Retrieval*) procura coletar documentos e artigos com texto relevante [83]. As ferramentas de Recuperação de Informações, geralmente, trabalham com técnicas de indexação, capazes de indicar e acessar mais rapidamente documentos de um Banco de Dados textual [77].

Existem três tipos de indexação [77]:

- Indexação tradicional: é aquela onde uma pessoa determina os termos descritivos ou caracterizadores dos documentos, os quais farão parte do índice de busca.
- Indexação *full-text* (ou indexação do texto todo): onde todos os termos que compõem o documento fazem parte do índice.
- Indexação por *tags* (por partes do texto): onde apenas algumas partes do texto são escolhidas automaticamente, para gerar as entradas no índice (somente aquelas consideradas mais importantes ou mais caracterizadas).

O primeiro tipo pode ser encontrado na maioria das Bibliotecas, sendo objeto de estudo da área de Ciência da Informação (parte da Biblioteconomia). A criação de Thesaurus (índice hierárquico de assuntos) é uma das preocupações desta área. Entretanto, o maior problema deste tipo de indexação é que a intervenção humana pode gerar falhas no índice, além de ser uma atividade difícil e que consome muito tempo.

Tais problemas ocorrem porque é necessário estabelecer hierarquias pré-definidas e nelas encaixar os documentos. Também será necessário analisar os documentos para determinar em que categoria se encaixam. Erros de classificação podem ocorrer quando o conteúdo dos documentos não é bem analisado (ou apenas seu título é considerado) ou quando não existem (não foram definidas) categorias intermediárias (ou seja, quando um documento não se encaixa bem em nenhuma das categorias pré-definidas, dando a impressão de que deveria haver outra categoria).

Outros problemas podem ocorrer quando o usuário do índice for recuperar documentos. Se o assunto de busca não for bem determinado ou se não existir uma categoria adequada para o que se deseja, talvez sejam recuperados documentos irrelevantes ou então nenhum documento seja recuperado.

Já o segundo tipo de indexação (*full-text*) procura indexar todos os termos, sem estruturas hierárquicas entre estes. As ferramentas indexam automaticamente todas as palavras do

documento, gerando índices volumosos.

Para diminuir o problema do tamanho dos índices, mas principalmente para evitar a indexação de termos indesejados ou inconsistências no índice, podem ser aplicados filtros neste processo [77] [78] [79].

Por exemplo, as chamadas *stop-words* (palavras que não fazem diferença no conteúdo, tais como preposições e artigos) podem ser eliminadas. Datas e números podem ser normalizados (utilizando-se um padrão). Pronomes podem ser substituídos pelos nomes correspondentes. Sinônimos podem ser substituídos por um termo-padrão, e palavras compostas podem ser substituídas por um termo sinônimo simples (composto de uma palavra só).

O terceiro tipo de indexação (por *tags*) procura indexar somente as partes relevantes do documento. Para tanto, as ferramentas automatizadas deverão analisar cada documento, procurando por marcas (*tags*) que identificam estas partes mais importantes.

Chakravarthy [80] gera descrições dos documentos, usando generalizações (resumos em uma frase) do seu conteúdo. Entretanto, tais descrições precisam ser feitas manualmente por pessoas, o que aumenta o trabalho inicial, apesar de diminuir o esforço para recuperação.

Moulin [81] esclarece como a identificação das partes significativas (aquelas a serem indexadas) pode ser feita automaticamente, declarando que os documentos possuem uma macroestrutura (composta por cabeçalhos, títulos, capítulos, etc), uma microestrutura (a qual contém o conteúdo lógico do documento e que pode ser identificada por palavras-chave que introduzem condições, exceções, referências, etc) e uma camada de domínio (contendo as demais informações).

Nesta mesma linha, as ferramentas descritas em Hardy [82] extraem palavras-chave com base em marcas (*tags*) da macroestrutura dos documentos. Há uma tabela de *tags* específica para cada tipo de arquivo (por exemplo: programas contém *procedures*, funções, comentários, etc; mensagens de correio eletrônico contém destinatário, remetente, assunto).

Entretanto, estas ferramentas encontram problemas quando o documento não possui uma macroestrutura pré-concebida (tal como acontece nos textos livres armazenados em arquivos ".TXT"). Neste caso, somente as palavras das cem primeiras linhas serão indexadas, para evitar uma indexação *full-text*. A premissa de Hardy [82] é a de que no início do documento é que se encontra sua descrição.

Para a indexação por *tags*, a maioria dos trabalhos adota o uso de gramáticas, *parsers* e expressões regulares para a definição e identificação das marcas.

O maior problema das técnicas de indexação por *tags* é que os tipos de documentos

devem ter sido analisados previamente, ou seja, já se deve conhecer que tipo de conteúdo compõe os documentos e quais as marcas que identificam as partes relevantes de um documento. Então as ferramentas acabam sendo muito específicas para determinados tipos de informação e documentos.

Outro grave problema que ocorre nas técnicas tradicionais de recuperação de informações é que muitas vezes documentos importantes não são recuperados, enquanto que outros irrelevantes são apresentados ao usuário [96].

Isto se dá porque estas técnicas estão baseadas na presença ou não de palavras nos documentos. Entretanto, pode haver documentos relevantes que não contém as palavras especificadas para a busca e pode haver documentos que contém as palavras de entrada mas que não tratam do assunto desejado.

Este problema é denominado de indexação imprecisa e ocorre porque a pessoa ou técnica que descreve e indexa os documentos pode utilizar termos diferentes de quem procura pelos documentos. Alguns trabalhos procuram solucionar tal problema com técnicas de indexação semântica ou por contexto.

Cowie [83] sugere que as técnicas para indexação devem gerar índices sensitivos mais intimamente relacionados ao real significado de um texto em particular e não baseados na presença de termos sem identificação do contexto.

Chen [78] define contexto ou espaço conceitual como sendo um conjunto de palavras que definem um assunto ou área do conhecimento. Algumas técnicas então procuram recuperar documentos baseadas no contexto dos documentos. Também discute técnicas baseadas na freqüência de termos em documentos. Para determinar a importância de um termo em um contexto (o quanto ele ajuda a definir um contexto), é necessário primeiro identificar os termos que compõem os documentos, bem como quantas vezes cada termo aparece nos documentos.

Também para tratar o problema de busca contextual, há a técnica de Chakravarthy [80], a qual se utiliza de expansões semânticas de palavras. Expandir semanticamente uma palavra nada mais é do que encontrar outras palavras relacionadas com ela, utilizando então este conjunto para busca de documentos. Chakravarthy [80] utiliza as definições de um dicionário para achar as palavras que se relacionam, eliminando *stop-words* e modela estas relações através de redes semânticas, criadas manualmente.

Entretanto, os problemas desta técnica são saber que palavras expandir para fazer a busca e se as novas palavras acrescentadas realmente fazem parte do contexto. Segundo os experimentos de Chakravarthy [80], algumas das novas palavras não fazem parte do contexto,

o que pode fazer com que documentos irrelevantes sejam recuperados.

Quando houver mais de um contexto possível para uma dada situação, Wiebe [84] cita técnicas que utilizam modelagem de contextos alternativos, permitindo que contextos diferentes possam ser explorados em paralelo.

A raiz do problema de contextos diferentes está na imprecisão dos termos (termos com significados diferentes). Este problema pode ser notado tanto no momento da criação do índice, como na hora da recuperação. Isto se dá porque as pessoas utilizam vocabulários diferentes para exprimir suas intenções [85].

Para tratar este problema, Souza [86] definiu técnicas para geração de linguagens (escolha de termos, expressões, representações, signos e símbolos) e para a interpretação de termos e linguagens. Esta técnicas sistemáticas fazem parte da área conhecida como Engenharia Semiótica.

Neste sistema (GAAHA), a técnica utilizada como base para a seleção de palavras significativas de um texto é a técnica baseada na freqüência de ocorrências de palavras [46]. A ênfase neste método é mais da área estatística do que de uma aproximação lingüistica para análise de texto automatizado [78]. As razões para isto são várias. A primeira é que a análise lingüística é muito dispendiosa para ser implementada [42]. Parte do problema é devido ao pouco progresso que tem sido realizado na teoria da semântica formal. Entretanto, existe alguma razão para otimismo neste campo, apresentado no trabalho de Keeman [43] [44]. Sem dúvida alguma, a teoria da linguagem será de extrema importância para o desenvolvimento de sistemas IR. Mas, atualmente, tais teorias não tem sido suficientemente desenvolvidas para serem aplicadas eficazmente em sistemas IR. A segunda razão é que a aproximação estatística tem sido examinada e testada, com resultados satisfatórios [45].

Luhn [46] em um dos seus primeiros artigos publicados, propõe que a freqüência da ocorrência de uma palavra em um texto forneça uma medida útil para a significancia da palavra. Isto favorece que a posição relativa das palavras dentro de uma sentença tenha valores significativos, fornecendo assim, uma medida útil para determinar sentenças significativas. O fator significativo de uma sentença, consequentemente, será baseado na combinação destas duas medidas. Luhn assume que a freqüência do dado pode ser usada para extrair palavras e sentenças para representar um documento.

Seja f a freqüência de ocorrência de várias palavras em um texto e r sua ordem de classificação (*rank order*), isto é, a ordem de sua freqüência de ocorrência como demonstrado na tabela 1. O resultado da interseção do relacionamento f e r, gera uma curva similar a uma hiperbólica, como exemplificado na figura 12. Esta curva demostra a lei de Zipf [47] , ou seja,

o produto da freqüência do uso das palavras com a classificação das classes (*rank order*) é aproximadamente uma constante. Zipf testou sua lei analisando as palavras de um jornal (*American Newspaper English*).

Luhn usou esta lei como uma hipótese nula, definindo assim os limites de *cut-offs*, um superior (*upper*) e um inferior (*lower*), excluindo assim as palavras não significantes, ou seja, as palavras que estão fora destes limites [45]. As palavras que excedem os limites impostos pelo *upper cut-off* são consideradas comuns e aquelas abaixo do limites do *lower cut-off* são consideradas raras, e portanto, não contribuem significativamente no conteúdo de um texto. Com isso, Luhn chegou a uma técnica capaz de encontrar palavras significativas em um texto, e também assumiu que o poder de resolução das palavras significantes, no que diz respeito a habilidade de distinguir conteúdos de palavras, atinge um pico no meio do caminho entre os dois extremos do *cut-off*, caindo em direção aos extremos. Uma certa arbitrariedade é envolvida na determinação do *cut-off*. Não existe um oráculo que dá os valores do *cut-off*, o estabelecimento deles se dá via um julgamento de tentativas e erros.

Palavra do texto	Freqüência que ocorrem (f)	Rank Order (r)	(Zipf) Peso da Palavra
E	73	1	73
No	56	2	112
Matemática	13	3	33
Scanner	5	4	20
Oráculo	1	5	5

Tabela 1 - Representação das palavras de um texto.

É interessante notar que estas idéias foram base para muitos trabalhos posteriores, referente ao desenvolvimento de sistemas IR. Luhn, por si só, utilizou esta técnica para projetar um método de extração de resumos automaticamente. Ele foi o desenvolvedor de uma medida numérica de significancia para sentenças baseada no número de palavras significantes e não significantes em cada parte de uma sentença. As sentenças foram classificadas de acordo com a sua contagem numérica, e a mais alta classificação foi incluída no resumo (realmente extraído) [42]. Edmundson e Wyllys [48] realizaram algumas generalizações dos

trabalhos desenvolvidos por Luhn, normalizando suas medidas com relação a freqüência de ocorrência das palavras em um texto.

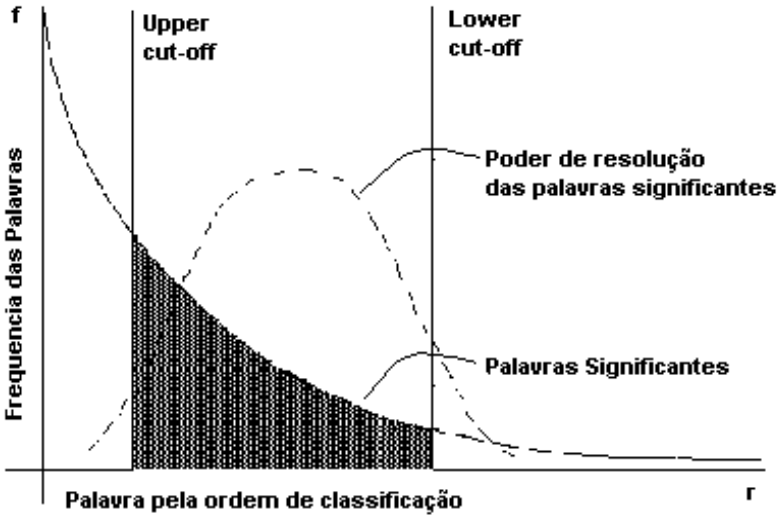

Figura 12 - Gráfico relacionando a freqüência de ocorrência *f* e a seqüência de posição *r*.

Rijsbergen [42] resumiu a proposta de Luhn sobre a análise automática de um texto, dizendo que a freqüência dos dados pode ser usada para extrair palavras e sentenças que representam um documento.

Não existe nenhuma razão para que tais análises fiquem restritas somente a palavras. Elas podem ser igualmente aplicadas para frases e de fato isto tem ocorrido [42].

O desenvolvimento e vantagens no processo de representação de um documento, tem sido muito abordada pela publicação *"Annual Review of Information Science and Technology"*.

As Figuras 13, 14 e 15, representam os resultados da aplicação da teoria de Luhn e Zipf em um texto técnico em Inglês, em um obra da literatura Brasileira e da qualificação para Pós-Graduação em nível de mestrado para este sistema (GAAHA).

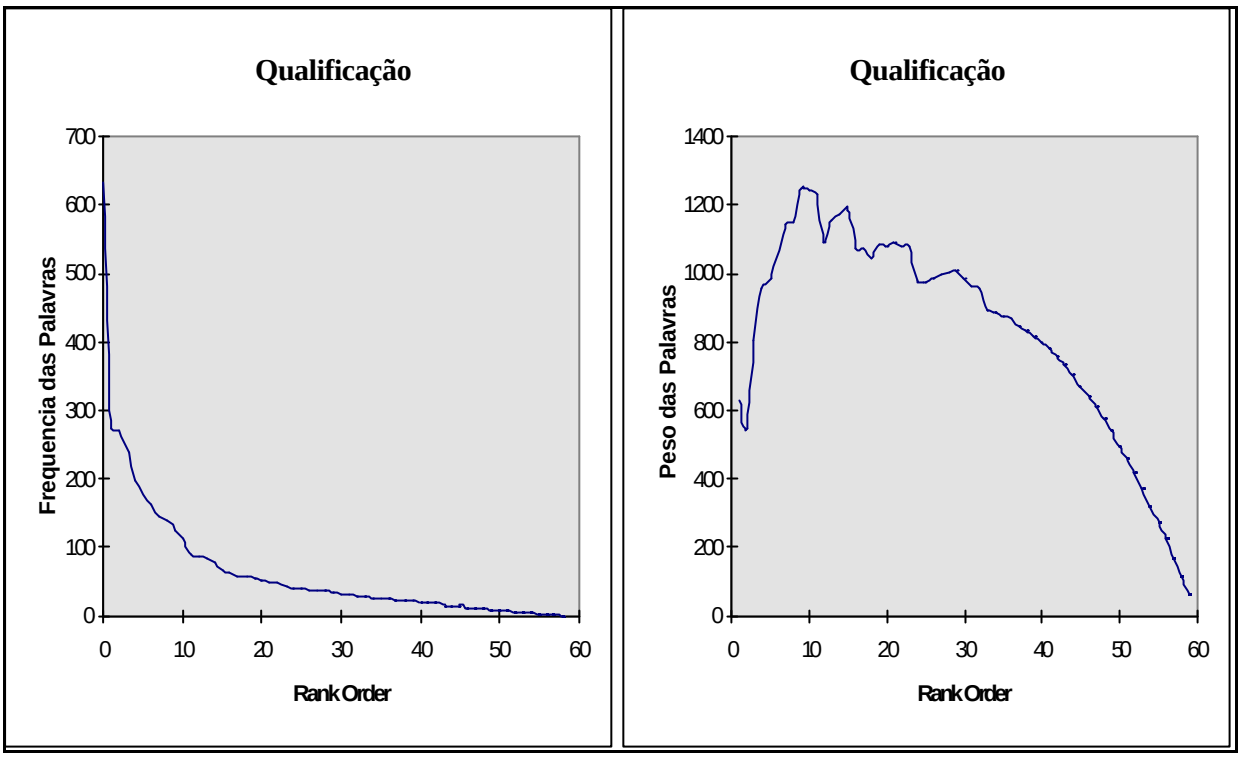

Figura 13 - Gráfico da teoria de Luhn e Zipf no texto da Qualificação.

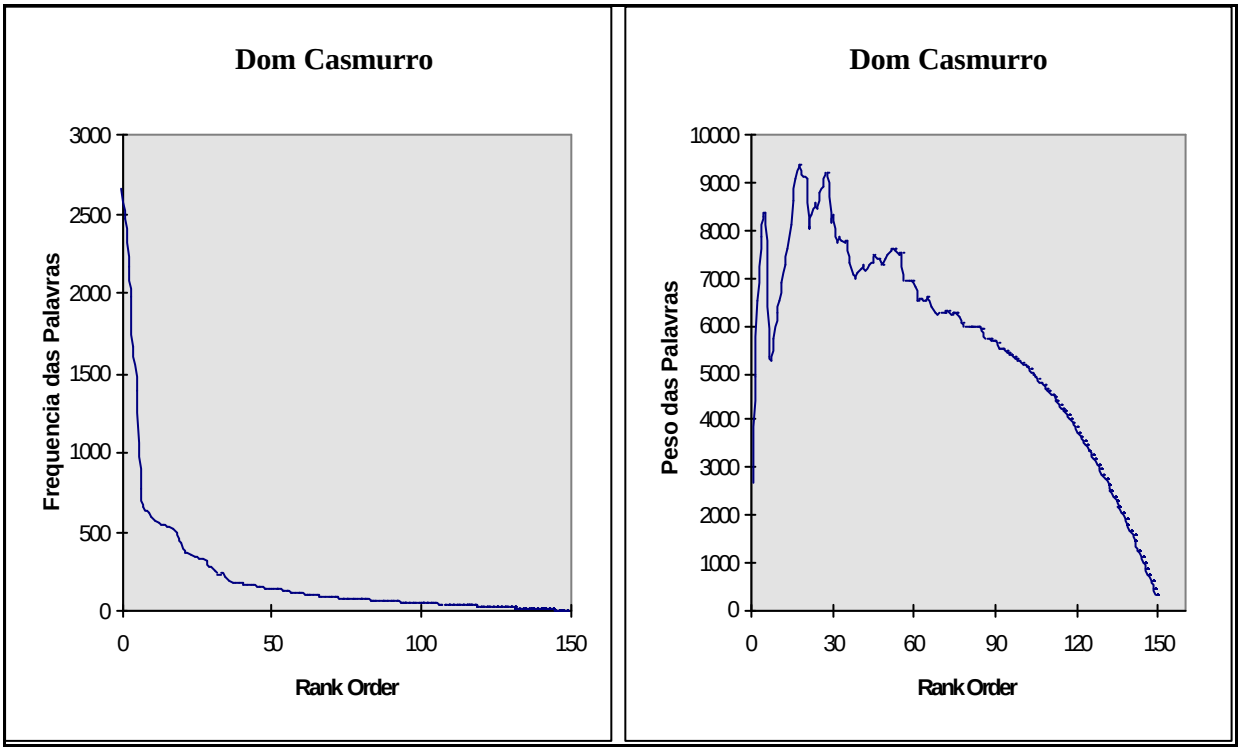

Figura 14 - Gráfico da Obra de Machado de Assis - Dom Casmurro.

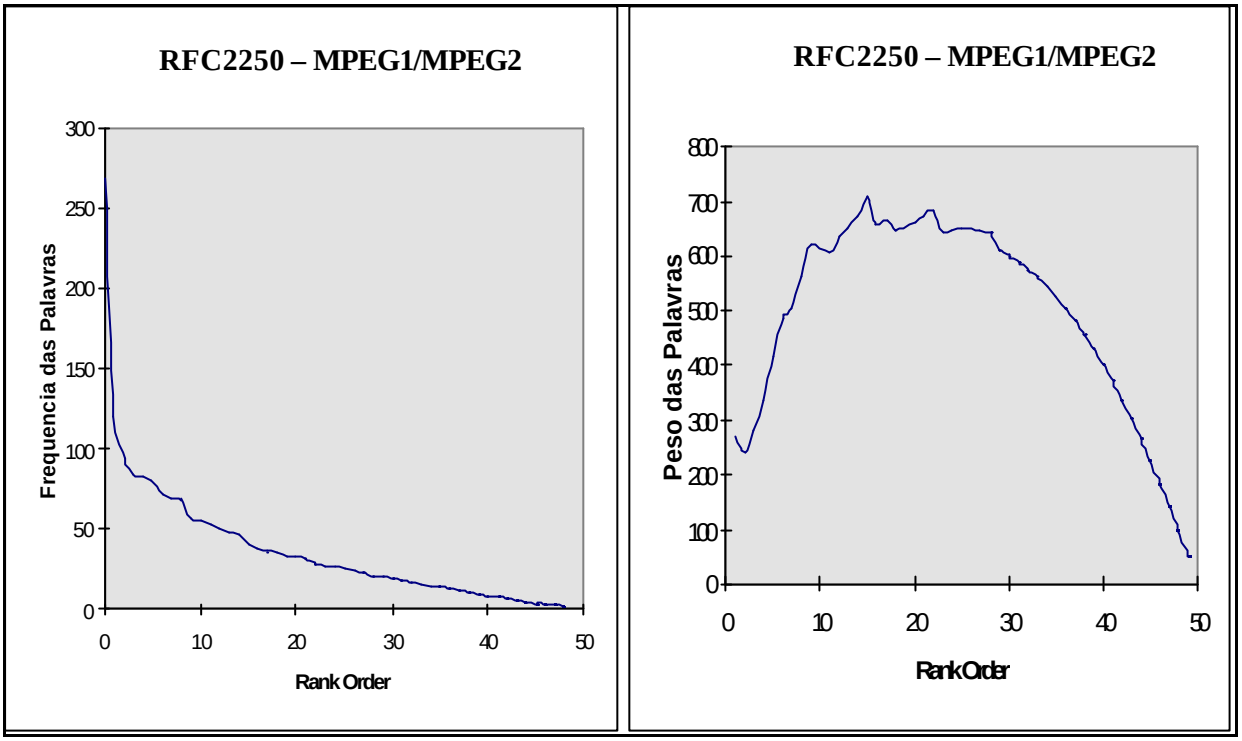

Figura 15 - Gráfico da RFC 2250 MPEG1/MPGE2.

9. O sistema GAAHA.

9.1. Descrição do Problema.

"A Internet é agora um dos maiores repositórios mundiais de informação", segundo Chakravarthy [80]. Esta frase define bem o potencial da Rede Mundial Internet no mundo "globalizado" em que vivemos atualmente. Pessoas das diferentes nacionalidades, falando as mais diversas línguas, com interesses e conhecimentos variados podem trocar e compartilhar informações, disponibilizando publicamente suas informações e, ao mesmo tempo, procurando por conhecimento e experiências que outros adquiriram e vivenciaram.

Na sua forma mais conhecida e utilizada, a WWW pode ser vista como um "grande e autônomo sistema de informações distribuído e heterogêneo" [99], utilizado como fonte de informações nas mais diversas áreas do conhecimento, seja por motivos profissionais, educacionais ou simplesmente por lazer.

Entretanto, a maioria das informações disponíveis na Internet não estão na forma mais adequada para serem tratadas por meios computacionais. Estas informações, conhecidas como dados não-estruturados, estão na forma de textos, imagens, gráficos, vídeos ou sons.

Os chamados dados estruturados (aqueles convencionais, armazenados na maioria dos Sistemas de Gerência de Bancos de Dados) são mais fáceis de serem tratados por meios computacionais, porque existem linguagens formais (como SQL) que permitem sua manipulação e consulta de forma mais concisa e precisa. Entretanto, há o trabalho inicial de se definir a estrutura destes dados e depois coletá-los e incluí-los nos Bancos de Dados.

Já os dados não-estruturados necessitam de mecanismos computacionais diferentes dos tradicionalmente usados, para que possam ser coletados, armazenados, manipulados e consultados.

Chen [79] cita a frustração dos usuários com o problema da "sobrecarga de informações". Ela ocorre quando o usuário tem muita informação ao seu alcance, mas não tem condições de tratá-la ou de encontrar o que realmente deseja ou lhe interessa.

A grande maioria das aplicações que trafegam pela Internet atualmente, são construídas utilizando-se como base o padrão HTML. Esses documentos escritos em HTML, tem uma evolução muito dinâmica, *softwares* tem novas versões e cursos (de educação a distância ou treinamento) mudam seus conteúdos freqüentemente. Este dinamismo muito rápido de mudança de conteúdo, cria o problema de como manter esses documentos estruturados (indexados) corretamente, visando um esquema de ajuda para os usuários.

Existem duas soluções para este problema. A primeira seria a criação manual de um índice de ajuda, e a segunda, seria ter um mecanismo automático de criação de índice.

Focalizando no tópico de EAD, seria muito proveitoso para um aluno ter um mecanismo de pesquisa estruturado, rápido e intuitivo de posicionamento em tópicos desejados de um determinado curso.

Analisando as soluções para o problema em questão, facilmente chegamos a conclusão que obviamente um índice de ajuda, construído por uma pessoa, seria muito mais informativo que um índice gerado por um mecanismo automático. Contudo, por ser muito mais fácil a geração automática, seria possível gerar novos índices de ajuda toda vez que qualquer mudança, por menor que seja, fosse efetuada na documentação. Portanto, uma página menos informativa, porém gerada automaticamente, seria muito mais útil para os usuários.

9.2. Apresentação.

Visando a solução para o problema descrito no sub-item 9.1., foi desenvolvido um sistema de ajuda chamado de GAAHA. O GAAHA é uma solução para a análise do conteúdo de um curso residente em um *Web Site* e posterior criação de um mapa gráfico preciso, intuitivo e interativo de posicionamento em tópicos do curso. O GAAHA tem um robô *parser* responsável pela geração de uma base de dados indexada que é utilizada para pesquisas de palavras-chave entradas pelo usuário e também para geração de uma vista hierárquica de todos os *links* de um curso, aqui chamado de mapa do curso.

Este sistema é composto por três subsistemas: um agente *crawler* para a indexação das informações; um agente *mentor* que é uma rotina de pesquisa executada no servidor, responsável pela pesquisa no banco de dados; e um cliente WWW para a entrada dos dados de pesquisa.

O agente *crawler* e o agente *mentor* são os dois módulos principais desenvolvidos neste projeto. O cliente consiste de um *Web Browser* (tal como o Netscape ou Explorer) e um documento HTML criado dinamicamente pelo agente *mentor*. A figura 16 representa uma vista gráfica em alto nível do sistema.

Os módulos *crawler* e *mentor* são caracterizados como agentes devido a algumas características próprias de funcionamento que os integram, tais como, a manifestação de características de inteligência distribuída, por serem capazes de desempenhar o papel de

mediador entre pessoas e programas, e também por desempenharem o papel de um "assistente inteligente".

O agente *crawler* se caracteriza principalmente pela autonomia do seu funcionamento, ou seja, através de um pedido de indexação de um curso de EAD, todos os documentos que compõem este curso são indexados sem qualquer interação externa do administrador. Quanto ao agente *mentor*, sua característica é mais voltada para um mordomo que fica a disposição do usuário para responder as dúvidas solicitadas.

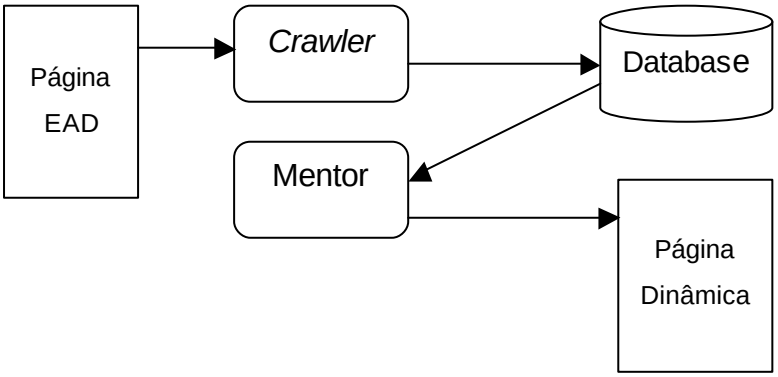

Figura 16 - Vista estrutural do sistema GAAHA.

O GAAHA usa duas técnicas principais:

- Análise de texto usando ferramenta estatística.
- Programação distribuída usando agentes de *software*.

Basicamente o programa gera um banco de dados relacional com as palavras mais significativas, os *links* de referências, os *headers* e os títulos do texto. Cada informação útil gerada, é relacionada com os locais onde elas aparecem. Obviamente o texto analisado deverá estar formatado no padrão HTML.

O tipo de indexação aplicada neste projeto para a análise da estrutura de um texto, é a indexação automática conhecida como *full-text* [77]. Como descrito no capítulo 8, este tipo de indexação, indexa automaticamente todas as palavras do documento. O grande problema deste tipo de indexação é o tamanho dos índices gerados, pois, como todas as palavras são armazenadas, os índices ficam volumosos e com muitas palavras insignificantes.

Para diminuir o problema do tamanho dos índices e também para armazenar apenas palavras significativas, um filtro foi implementado. Para a implementação deste filtro, foi utilizada uma técnica estatística de indexação automática, baseando-se na freqüência de ocorrência de palavras nos documentos [46] [78], ou seja, a palavra só é armazenada no banco de dados se estiver dentro de uma faixa de limite mínimo (*lower cut-off*) ou limite máximo (*upper cut-off*) de freqüência de ocorrência. Esta técnica foi descrita mais detalhadamente no capítulo 8.

A escolha do tipo de indexação (*full text*) e do filtro (por freqüência) foi pelo fato de que os mesmos não exigem qualquer tipo de formatação especial ou *tags* de reconhecimento em um texto, facilitando e até incentivando a utilização deste sistema (GAAHA). Quando existem exigências quanto a inserções de *tags* ou formatações especiais, o desenvolvedor tem que conhece-las para então aplica-las no curso a ser desenvolvido. Como atualmente o desenvolvedor não tem que se preocupar com os comandos HTML para o desenvolvimento de um curso, já que, existem diversas ferramentas que facilitam o desenvolvimento para este padrão, seria muito inconveniente e até mesmo desestimulante a adoção de uma técnica que exigisse o uso de *tags* ou formatações especiais nos textos que compõem um curso de EAD.

Outro grande motivo da adoção destas técnicas é a possibilidade imediata de indexação de cursos já existentes, utilizando este sistema (GAAHA).

Depois do agente *crawler* gerar um banco de dados, aplicando os mecanismos de indexação e filtragem das palavras correspondente a um determinado curso, é possível via uma URL, executar um servlet Java passando como parâmetro o nome do curso a ser pesquisado. O resultado desta execução, é a construção dinâmica pelo agente *mentor* de uma página HTML de ajuda, correspondente ao curso em questão. Esta página contém basicamente um botão que possibilita a construção dinamicamente do mapa do curso, um campo para digitação das palavras-chave de pesquisa ao banco de dados e um *link* para uma página informativa contendo informações de utilização da máquina de pesquisa.

O mapa do curso é uma estrutura em árvore que representa a hierarquia de todos os documentos e seus respectivos *links* contidos, possibilitando ao usuário uma visão mais ampla de todo o curso e um rápido posicionamento do tópico desejado.

Após a interpretação das palavras-chave, o agente *mentor* constrói uma página HTML com os resultados da pesquisa, retornando-a ao usuário.

O paradigma de programação utilizado é o OOP (*Object Oriented Programming*), o que não poderia ser de outra maneira, visto que, a linguagem de programação JAVA, a qual foi utilizada para o desenvolvimento deste sistema, trabalha sob este modelo [35].

O modelo de desenvolvimento deste sistema é baseado em um *thread* inicial de rastreamento do arquivo HTML, e, a cada *link* de referencia encontrado e ainda não visitado, é gerado um novo *thread* de busca, executando a mesma operação subseqüentemente. O número máximo de *threads* executados concorrentemente é entrado via parâmetro. Isto é desejável já que, se um grande número de *threads* forem executados simultaneamente, poderiam causar problemas de performance ao sistema. Estes problemas são relativos a plataforma e ao sistema operacional onde os *threads* são executados.

Quanto ao agente *mentor*, cuja responsabilidade é a interpretação das palavras-chave entradas pelo usuário e geração das páginas dinâmicas de resposta, é utilizado um Servlet Java para pesquisa no banco de dados. Também o agente *mentor* é responsável pela construção dinamicamente de um mapa gráfico hierárquico do curso .

9.3. Agente *Crawler*.

O agente *Crawler* consiste em algumas rotinas de análise de páginas e armazenamento em um banco de dados. O principal propósito deste agente é prover conteúdo para um banco de dados, através de um driver JDBC.

Sua máquina de pesquisa consiste em um *crawler Web* que é inicializado através de uma página principal, cujo os *links* são perseguidos e analisados, produzindo um banco de dados de palavras, títulos, *headers* e *hyperlinks*. A figura 17 exemplifica o modelo de funcionamento do agente *crawler*.

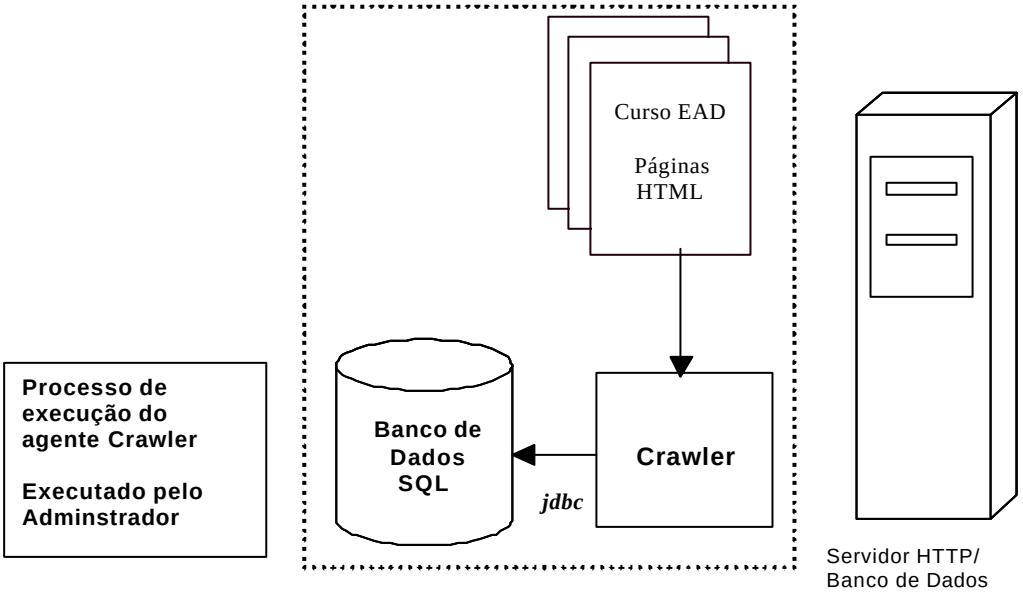

Figura 17 – Estrutura do agente *crawler*.

O agente *Crawler* é inicializado através de um objeto chamado *WebCrawler*. Através de uma URL inicial entrada pelo usuário, um objeto *PageSeeker* do tipo *thread* é criado. Este objeto é responsável principalmente pela análise e armazenamento dos dados em tabelas temporárias. Um objeto do tipo *HtmlStreamtokenizer* recebe o conteúdo da página e analisa-o, classificando-o em função do *tag* HTML encontrado. A cada *link* encontrado e que ainda não tenha sido visitado, um novo objeto *thread PageSeeker* é criado, ver figura 18 e 19. O controle das páginas já indexadas é realizado através das tabelas temporárias chamadas *visitadas* e *a visitar*. Este tipo de controle de páginas já visitadas tem que existir, visto que, um curso de Educação a Distância, assim como qualquer *Web site*, pode ser constituído por um grafo cíclico. Sendo assim, um controle de páginas já visitadas é extremamente desejável, caso contrário, o sistema entraria em um *loop* "eterno", porque dois *links* poderiam referenciar-se entre si, causando assim um "*ping-pong*" de conexões.

```
//Analisa o conteúdo das páginas
loadPage(_pageToFetch, this.getName() );
command = "SELECT url FROM visitar WHERE thread=\""+this.getName()+"\"";
rs = (ResultSet) stmt.executeQuery(command);
//Percorre todas as URLs a serem visitadas
while (rs.next()) {
   if ( (rs.getString(1)).indexOf("#") != -1)
        page = (rs.getString(1)).substring(0, (rs.getString(1)).indexOf("#"));
    else
        page = rs.getString(1);

   command = "SELECT count(url) FROM visitado WHERE url=\""+rs.getString(1)+"\"";
   rs1 = (ResultSet) stmt_aux.executeQuery(command);
   rs1.next();

   //Se a pagina não foi consultada
   if ( rs1.getInt(1) == 0 )
        //Cria um novo objeto do tipo Thread
        new PageSeeker( _base_URL, _parent, page,_threadLimiter,
                                              con_banco_dados, exclui_url );
}
```

Figura 18 – Código de criação dos *threads*.

Como cada página é analisada em um *thread* separado, várias páginas podem ser indexadas simultaneamente, dependendo apenas do número de *threads* máximo configurado pelo administrador no início da indexação. A figura 19 é uma representação gráfica do principal módulo do agente *crawler*, descrevendo a criação dos vários objetos do tipo *PageSeeker*.

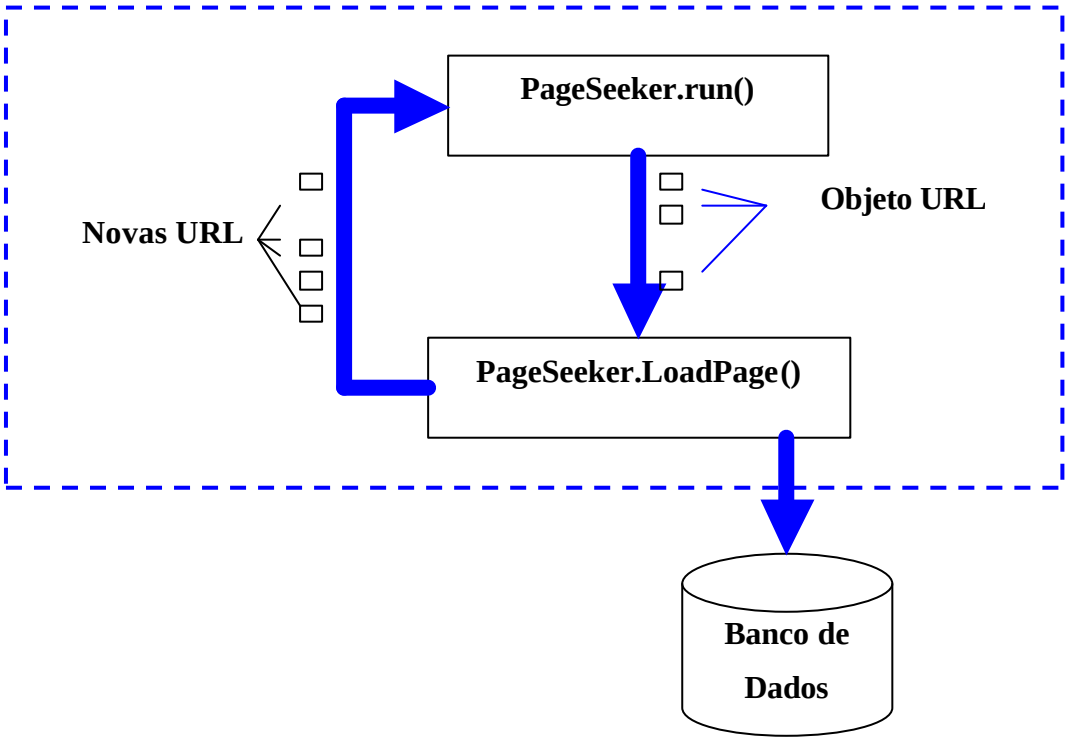

Figura 19 - Representação Gráfica do principal módulo do agente *crawler*.

A máquina de análise do agente *crawler* é composta por algumas funções principais básicas:

- no objeto *PageSeeker*:
  - método *loadpage()*
    - Este método abre uma conexão com uma URL recebida como parâmetro. É criado um objeto do tipo *HtmlStreamTokenizer* que recebe o conteúdo desta página. Este conteúdo é analisado, e armazenado em tabelas temporárias.
- no objeto *DBSmanager*:
  - método *run()*
    - lê o conteúdo do banco de dados temporário, analisando-o para posterior armazenamento no banco de dados permanente.

Como já citado no sub-item 9.2., neste sistema foi aplicado a técnica de indexação automática conhecida como *full-text* [77], com um filtro que utiliza uma técnica estatística de indexação automática baseando-se na freqüência de ocorrência de palavras nos documentos [46] [78].

Os valores de *cut-offs* são entrados como parâmetros pelo administrador na forma de valores percentuais. Estes valores são convertidos em valores absolutos em relação a quantidade total de palavras do banco de dados e comparados com as freqüências das palavras. O banco de dados é composto por todas as palavras das páginas que compõem o curso de EAD, cuja freqüência esteja dentro dos limites de *cut-off*.

No capítulo 8 foi descrito que uma certa arbitrariedade é envolvida na determinação dos valores de *cut-offs*, não existindo um oráculo que estabeleça estes valores com precisão, e que o estabelecimento deles se dá via um julgamento de tentativas e erros. Também foi descrito que Luhn [46] usou esta lei como uma hipótese nula para definição destes valores. Neste projeto para determinação dos valores percentuais iniciais, alguns cursos de EAD dos mais variados tamanhos foram analisados, aplicando-se a teoria de Luhn e Zipf [46] [47]. Esta análise teve como principal objetivo apenas estabelecer um valor inicial.

Estes valores foram prefixados como *default* nos argumentos de entrada dos limites de *cut-offs* do agente *crawler*, evidentemente com a possibilidade de alteração pelo administrador, caso seja necessário.

Em algumas situações, é extremamente desejável que haja restrições quanto ao acesso em algumas páginas, por se tratar de informações sigilosas ou que necessitem de algum controle de acesso. Para estes casos, o sistema prevê a leitura de um arquivo texto com o mesmo nome do curso e extensão .conf, cujo conteúdo pode ser URLs, diretórios ou simplesmente nomes de arquivos. Toda página que se enquadrar com algum padrão existente no arquivo de restrições, não será indexada.

A figura 20 mostra a execução da interface gráfica do agente *crawler*.

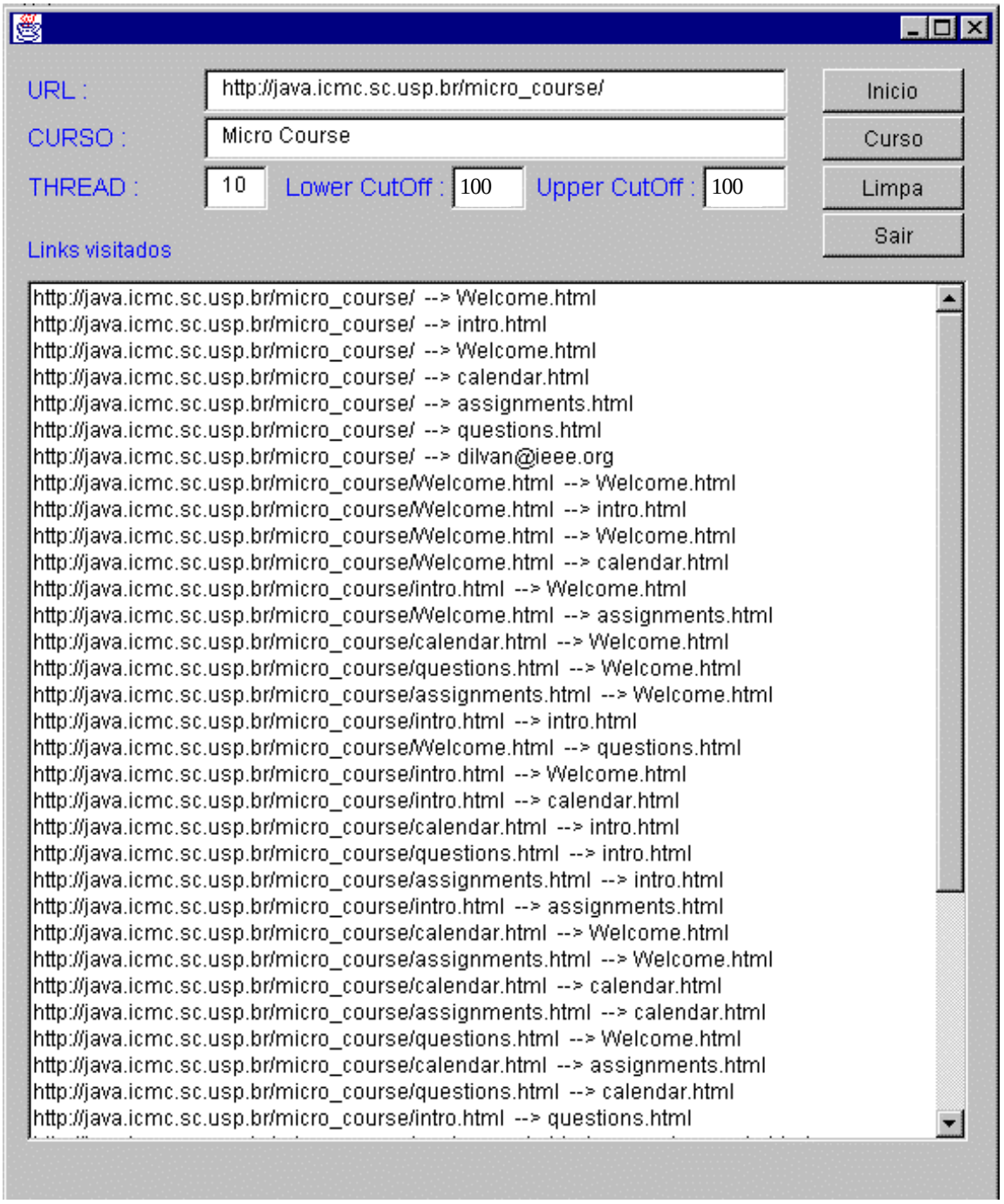

Figura 20 - Execução do agente *crawler*.

9.5. Agente *Mentor*.

O agente *mentor* consiste basicamente em uma rotina de pesquisa e acesso a um banco de dados. Este módulo reside no servidor *Web*, onde é executado através de uma solicitação contida em uma página *Web*, carregada por um *browser* cliente. Como este agente

provê dados como necessidade básica, é desejável que ele seja pequeno e rápido, agilizando assim o seu processamento. Este agente é responsável pela pesquisa em um banco de dados por comparação de chaves, apresentando então uma resposta a consulta realizada. Esta resposta, é uma página HTML construída dinamicamente pelo agente *mentor*. A figura 21 exemplifica o modelo de funcionamento do agente *mentor*.

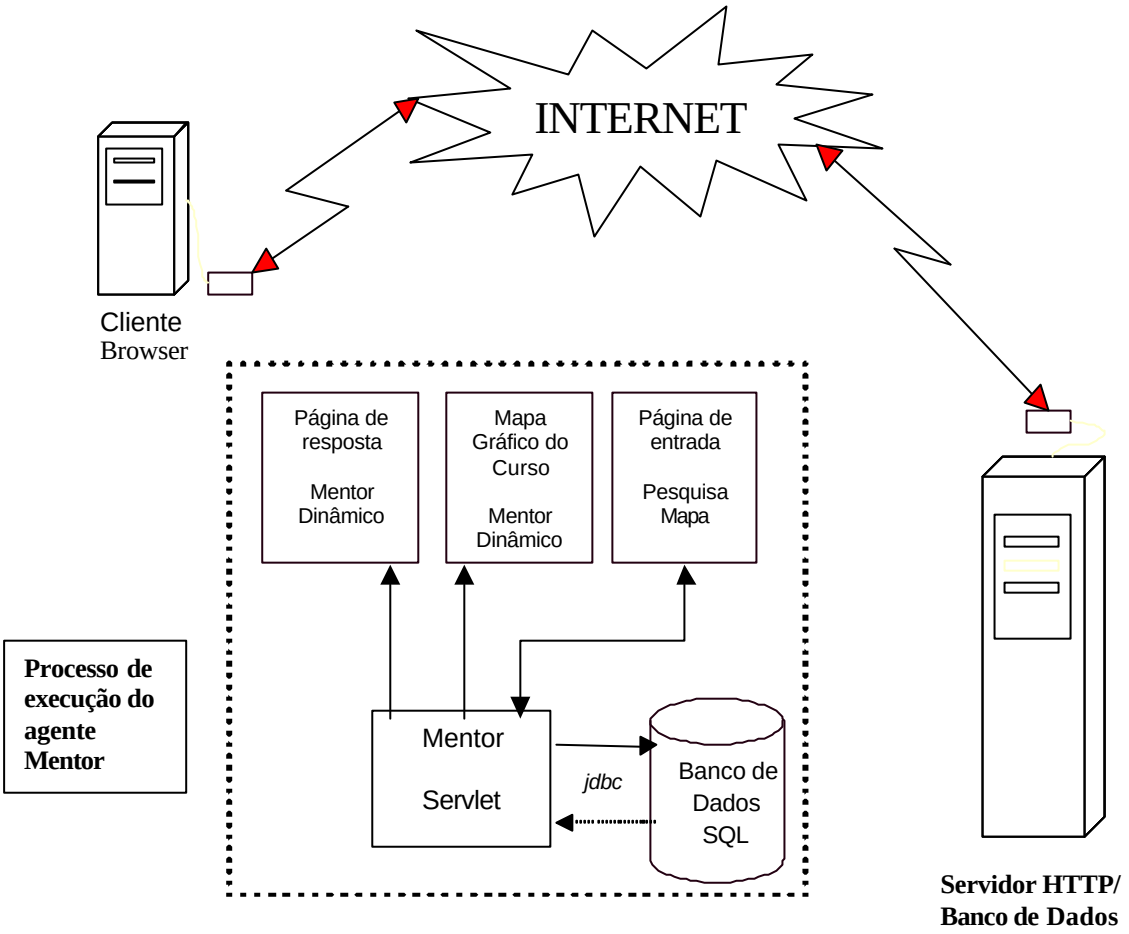

Figura 21 - Estrutura do agente *mentor*.

A rotina de pesquisa do agente *mentor* é um programa de interface com o usuário, que provê um *gateway* para o usuário final acessar o banco de dados criado pelo agente *crawler*. Isto é complementado pela capacidade de uso dos recursos de *forms* existente no *Web browser*.

A interface foi projetada para ser amigável para o usuário, utilizando recursos GUI como seu principal método de comunicação. O programa interage com o usuário através de

uma representação gráfica de uma página de pesquisa, onde o usuário deverá entrar com informações as quais deseja pesquisar.

Após a submissão pelo usuário, a rotina de pesquisa recebe as entradas dos dados e constrói uma apropriada expressão de pesquisa no padrão SQL. Esta expressão é utilizada para consultar o banco de dados por registros que contenham relevantes chaves. O resultado é coletado e traduzido para o formato gráfico, que, em seguida, é enviado de volta para o *browser* cliente. Se a informação não se enquadrar dentro do espaço de uma página no vídeo, o programa também permite que o usuário navegue para frente ou para trás entre as páginas, com cada página contendo 10 registros. A figura 22 mostra o fluxo de dados na máquina de pesquisa.

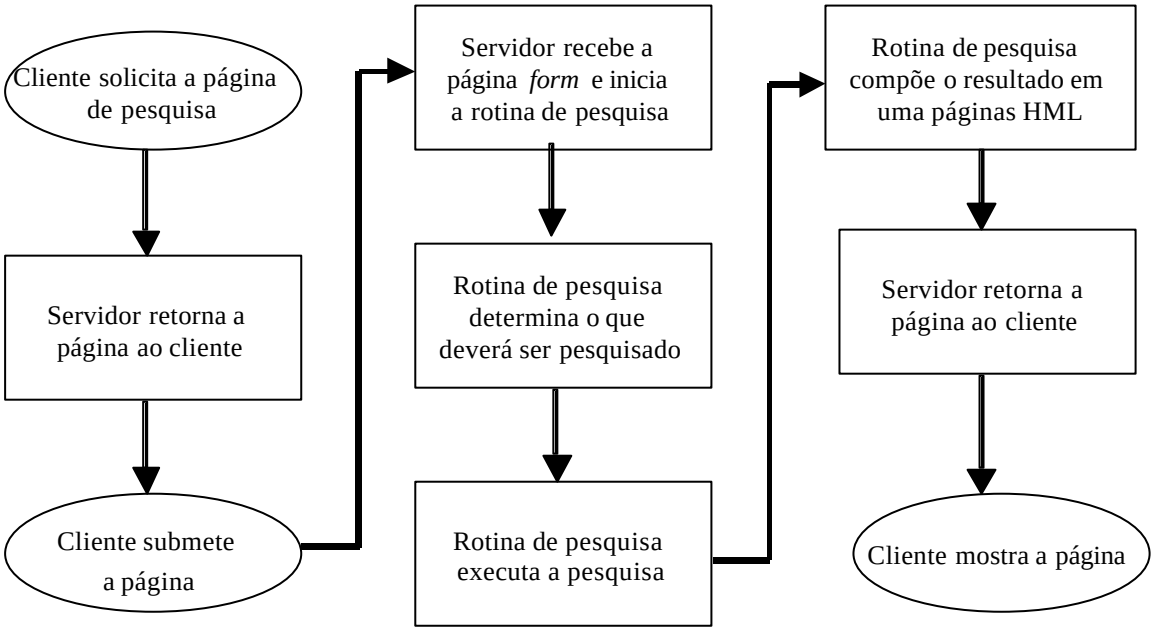

Figura 22 - Fluxo de dados no sistema de pesquisa.

Algumas funcionalidades básicas quanto aos critérios de pesquisa foram incorporadas neste agente, permitindo que uma pesquisa tenha suporte a lógica *booleana*, tais como (*or*, *not*) e *wildcards* (*,?). Estes critérios poderão ser aplicados em uma base de dados restrita por títulos, *headers*, ou geral, sendo esta última opção, o conteúdo de todo o banco de dados, independente do *tag* HTML, cuja palavra pertença. A seleção de uma destas opções é realizada pelo usuário através de uma lista, apresentada na página de pesquisa, sendo que a

opção geral é selecionado por *default*, caso não haja interação por parte do usuário.

A máquina de pesquisa foi desenvolvida permitindo também que uma expressão chave de pesquisa seja construída com o formato de linguagem natural. Neste caso, não é aplicado nenhum método de aproximação lingüistica para análise de texto usando alguma teoria da semântica formal, e sim um método estatístico, cuja descrição foi apresentada no capítulo 8.

O acesso ao banco de dados também usa o mesmo driver JDBC que o agente *crawler*. A máquina de pesquisa é uma aplicação muito simples se comparada com o agente *crawler*. Enquanto o agente *crawler* tem de comparar textos e perseguir *links* codificados em documentos HTML, a principal missão da máquina de pesquisa do agente *mentor* é tentar encontrar o que o usuário está pesquisando, retornando-o a resposta. A máquina de pesquisa é composta por algumas funções principais básicas:

- no objeto *Search*:
  - método *init()*
    - Este método é chamado apenas uma vez. Ele é responsável pela conexão com o banco de dados através do driver JDBC.
- método *Services*()
  - Manipula o processamento do *form*, quando é executado através de uma página *Web* via o método *get*.
- Método *recResult()*
  - através das palavras-chave entradas pelo usuário, o método *recResult* montará uma expressão no padrão SQL, submetendo-a ao banco de dados para receber a resposta da solicitação.
- Método *RecMap()*
  - Constrói um mapa gráfico dinâmico do curso, através de uma consulta ao banco de dados, usando como critério o nome do curso recebido como parâmetro *hidden* do *form* HTML.

A figura 23 mostra a execução da interface gráfica do agente *mentor*.

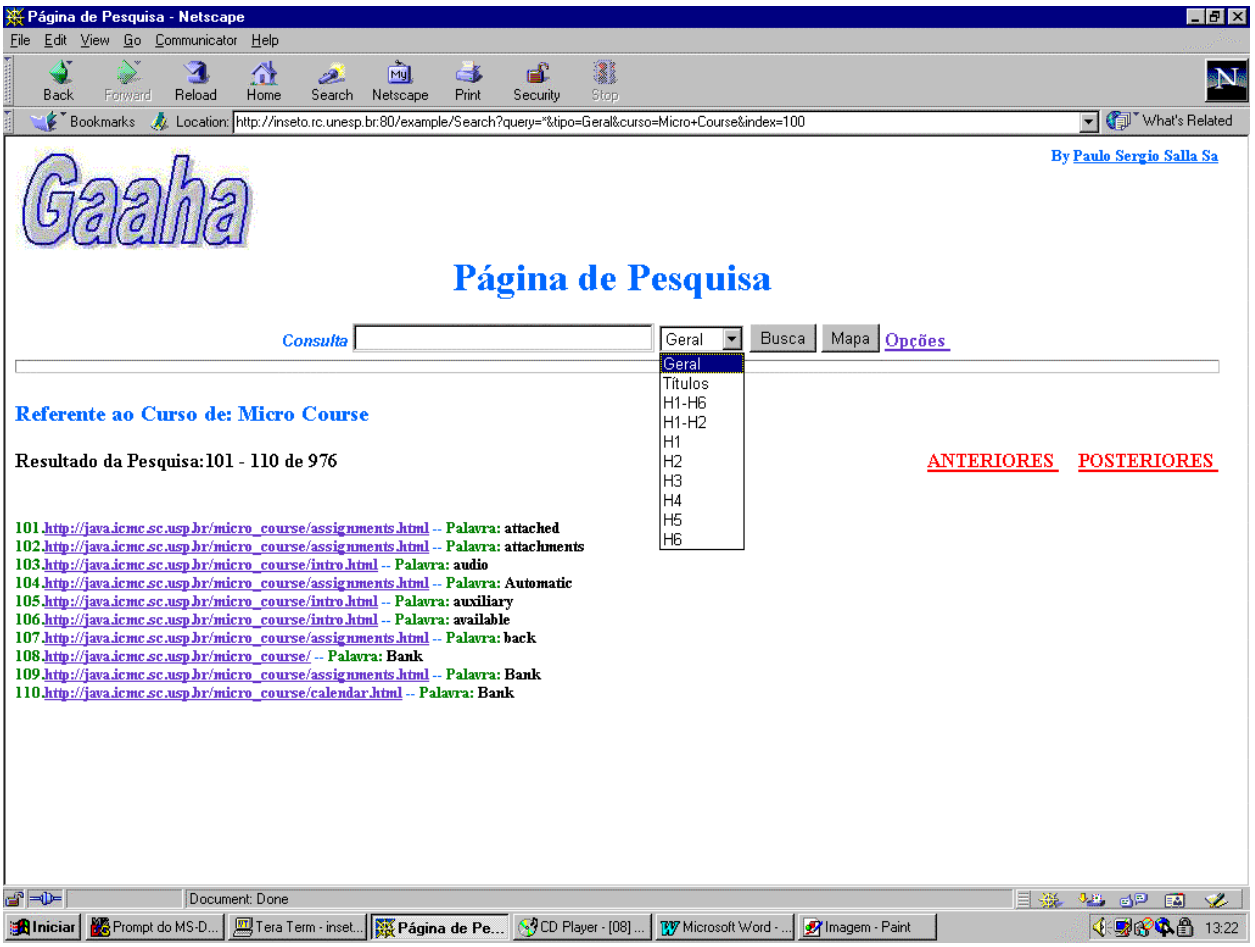

Figura 23 - Execução do agente *mentor*.

9.6. Cliente *Browser.*

O *browser* é uma ferramenta de fundamental importância no sistema. Ele pode ser qualquer *Web Browser* capaz de aceitar entrada de texto e mostrar a ação de um *button*. Existem duas páginas *Web* primárias através das quais o usuário interage. A primeira é um simples *form* de entrada de texto onde o usuário especifica a consulta. A segunda página é a resposta da consulta produzida pelo servidor.

9.7. Banco de Dados.

O servidor de banco de dados tem dois propósitos para este projeto.

- Armazenar e indexar o conteúdo das páginas de um curso de EAD.
- Ser um comum *link* entre os três principais subsistemas (*crawler, search* e o cliente).

O DBMS utilizado neste projeto é o MySQL, sendo utilizado o utilitário mysqladm que é baseado em texto para a criação do banco de dados.

O banco de dados nada mais é do que um repositório de palavras-chave produzidas pelo agente *crawler*. Ele é percorrido pelo agente *mentor* para produzir saídas dinâmicas de documentos no padrão HTML. Devido a utilização de um *driver* JDBC para o acesso ao banco de dados, qualquer DBMS (ex: Oracle, Sybase) poderia ser utilizado.

9.7.1. Acesso ao Banco de dados.

O acesso ao banco de dados é realmente o coração da rotina de pesquisa do agente *mentor*. Toda a transação é realizada através de um driver JDBC do tipo 4, chamado *twz1jdbcForMysql*. Este driver tem um protocolo nativo com código puro em Java, permitindo uma chamada direta de um cliente (o servlet, no caso) para o servidor DBMS. É um driver proprietário escrito especificamente para o servidor de banco de dados MySQL.

Os dados são originados com o subsistema *crawler*, o qual produz um banco de dados temporário como mostrado na figura 24. O propósito do banco de dados temporário é armazenar os dados lidos através de uma conexão HTTP com a URL do curso desejado, sem se preocupar com qualquer tratamento. Após o armazenamento das páginas do curso nas tabelas temporárias, é iniciado um *thread* responsável pela leitura destas tabelas. Antes do efetivo armazenamento nas tabelas permanentes, os dados são filtrados baseando-se na freqüência de ocorrência dos mesmos, e os que satisfizerem os critérios exigidos, serão armazenados definitivamente, obedecendo os relacionamentos entre as tabelas. O agente *crawler* tem permissão de leitura e escrita nas tabelas, já o agente *mentor*, tem apenas permissão de leitura.

Neste projeto, o servidor de DBMS está residente na mesma máquina que o servidor HTTP. Portanto, ficando como canal de comunicação entre os dois, apenas o I/O do disco. Entretanto, nada impede que os dois servidores fiquem em máquinas distintas, tendo assim como canal de comunicação a rede de computadores.

No início do desenvolvimento deste projeto, foi desenvolvido um protótipo que no lugar de armazenar os dados em tabelas temporárias, eram armazenados em tabelas *hash*, e posteriormente depois de analisados, eram armazenados em um banco de dados. Mas, como as tabelas *hash* usam a memória RAM para armazenamento, alguns problemas com estouro de memória ocorriam dependendo do tamanho do curso analisado. Para resolver este problema optou-se pelo uso de tabelas em disco usando o DBMS.

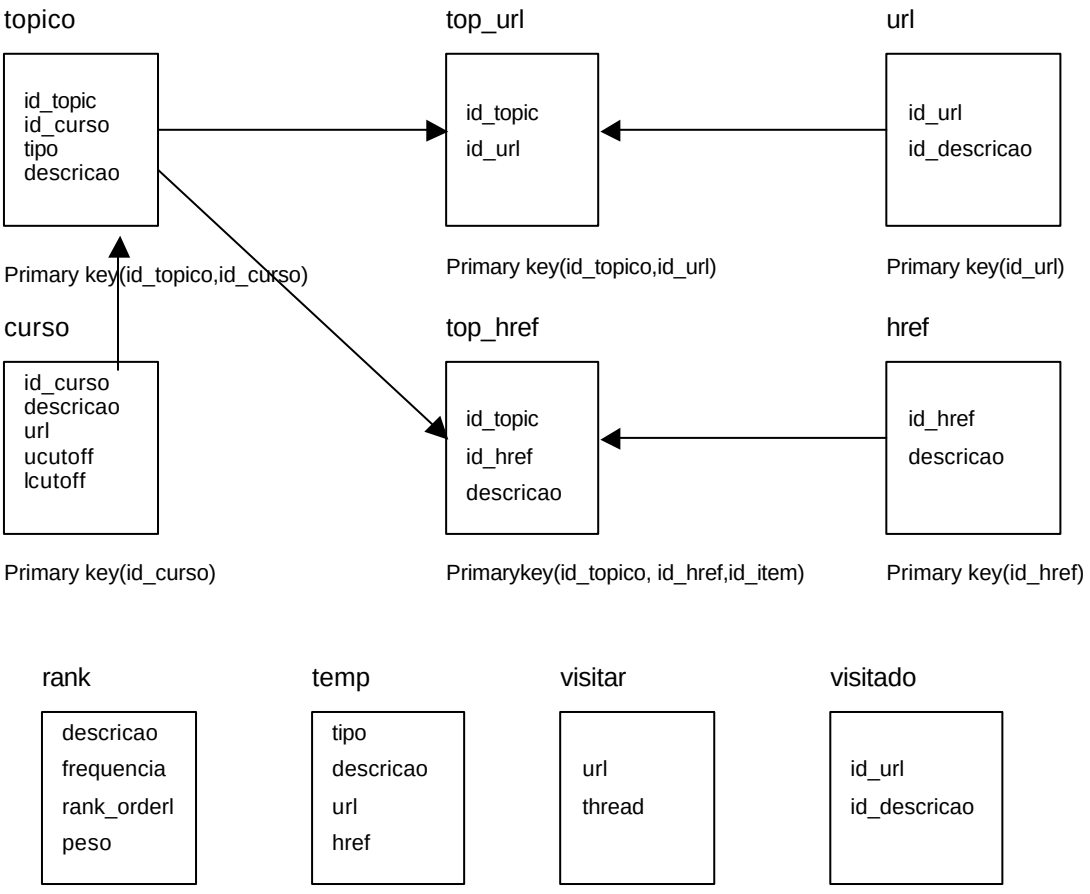

Figura 24 - Modelo Relacional do banco de dados.

9.8. Mapa Gráfico Dinâmico.

Como já citado no sub-item 9.2., o mapa gráfico do curso é uma estrutura em árvore que representa a hierárquica de todos os documentos e seus respectivos *links* contidos, possibilitando ao usuário uma visão mais ampla de todo o curso e um rápido posicionamento no tópico desejado.

Este mapa é criado dinamicamente através da análise do banco de dados usando como chave o nome do curso. Esta chave é passada como campo *hidden* na página HTML, cuja ativação é através do método *recMapa* da classe *Search*, esta última sendo estendida da classe *HttpServlet*. O critério de busca de informações de um determinado curso no banco de dados é baseado nos títulos e *links* das páginas. Com o resultado obtido através do acesso ao banco de dados e através do recurso de utilização de tabelas em páginas HTML é montada uma estrutura em árvore que representa o mapa gráfico do curso.

A grande vantagem deste mapa é a visão total do curso em um esquema hierárquico, permitindo assim um rápido posicionamento em qualquer página desejada pelo aluno.

A figura 25 mostra a interface gráfica do mapa do curso gerado a partir do agente *mentor*.

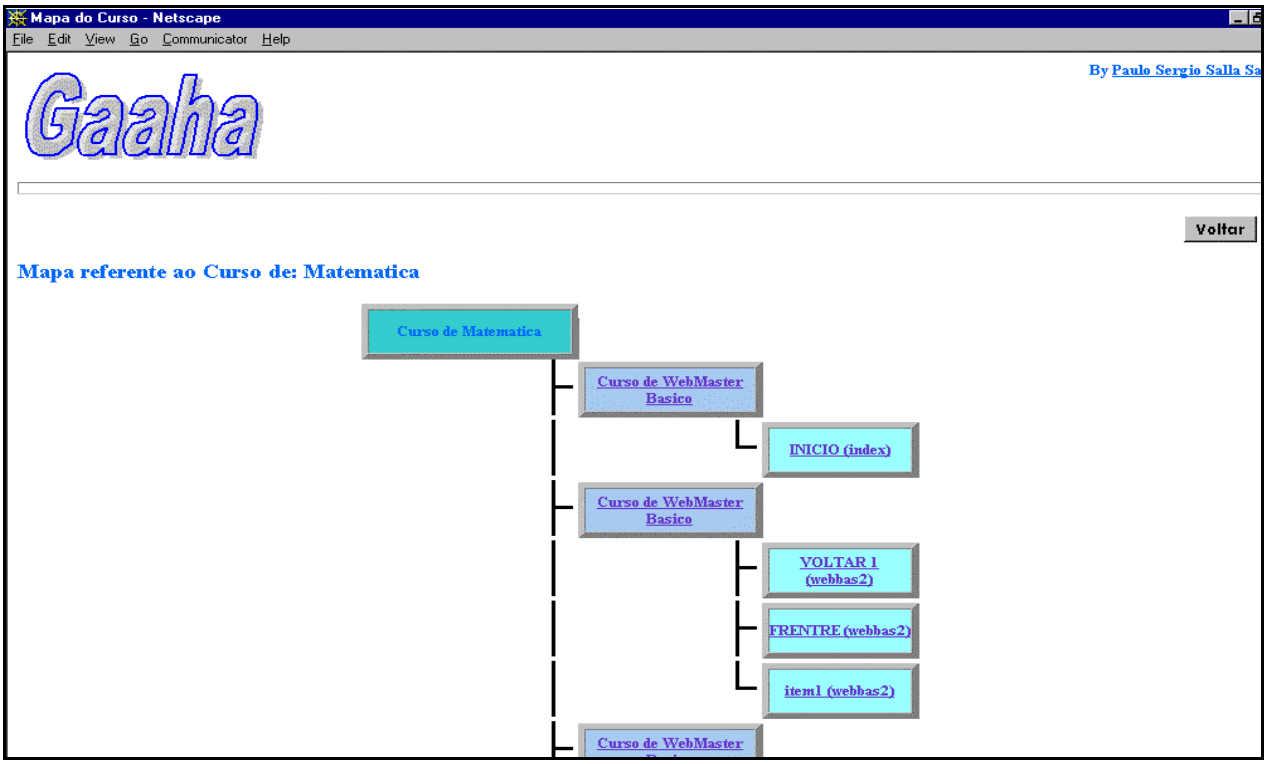

Figura 25 - Mapa gráfico do curso.

10. Conclusão.

O objetivo deste capítulo é fazer uma análise geral do sistema Gerador Automático de Arquivos HTML de Ajuda para Aplicação em Educação a Distância (GAAHA), e também analisar algumas oportunidades úteis no desenvolvimento deste projeto, oferecendo algumas sugestões para realizações de futuras pesquisas sobre tópicos relacionados a este projeto.

O resultado da execução do sistema GAAHA é realmente a geração de uma base de dados indexada contendo as palavras que compõem um curso, sendo que, a totalidade de armazenamento das palavras só é alcançada quando os valores de *lower cut-off* e *upper cut-off* são iguais a 100 %. Quanto maior for o valor de *lower cut-off* e *upper cut-off,* maior a chance de ser armazenada palavra sem muita significancia, segundo a teoria de Luhn [46]. Caso estes valores não sejam bem definidos, corre-se o risco de, ou armazenar palavras insignificantes, ou não armazenar palavras significantes. A conseqüência do não armazenamento de todas as palavras que compõem um curso é a construção de um banco de dados mais enxuto e eficiente. Portanto, a construção dinâmica do arquivo de ajuda contendo as respostas a uma consulta feita por um usuário, torna-se mais "limpa" e rápida.

A execução do sistema GAAHA, também tem como resultado a geração dinâmica de um mecanismo de posicionamento gráfico hierárquico em árvore, contento todos os *links* de referencias de um curso de EAD.

10.1. Visão do Projeto.

O projeto GAAHA tem como principal objetivo prover um significante método de pesquisa a uma coleção de páginas que compõem um curso de EAD. As informações são analisadas e armazenadas em um banco de dados através da atuação de um agente *crawler*. Utilizando uma chave de pesquisa entrada pelo usuário, um agente *mentor* pesquisará um banco de dados, e com as respostas desta pesquisa, montará páginas HTML dinâmicas. Estas páginas são retornadas ao *browser* cliente.

Três fatores importantes foram considerados neste projeto:

- O desempenho.
- A flexibilidade.
- A inovação.

10.1.1. Desempenho.

A tabela 2 mostra alguns valores obtidos para a análise e indexação de alguns cursos de EAD pelo agente *crawler* do sistema GAAHA. Os dados destes cursos foram obtidos diretamente via Internet dos seus *sites* oficiais, ou seja, o sistema foi executado em uma rede física diferente em relação aos cursos analisados.

Portanto, não há qualquer restrição quanto ao local de execução do agente *crawler*, desde que haja uma conexão TCP/IP, este poderá ser executado em qualquer máquina ou rede. É obvio que seria muito mais conveniente se tratando de performance, que ele fosse executado na máquina em que residem os servidores HTTP e DBMS. Deste modo, ficaria apenas como canal de comunicação entre os dois, o I/O do disco.

CURSOS	LC %	UC %	TP	FPUC	FPLC	TPA
Sist. Operac.	100	100	47922	3034	1	10719
Sist. Operac.	70	70	47922	257	25	4018
Mat. Financ.	100	100	3506	222	1	534
Mat. Financ.	70	70	3506	76	8	122
Mic. Course	100	100	1472	93	1	727
Mic. Course	70	70	1472	23	5	295
Dir. Tribut.	100	100	3173	159	1	804
Dir. Tribut.	70	70	3173	49	7	98
Ling. Portug.	100	100	3038	159	1	1159
Ling. Portug.	70	70	3038	51	6	64
Rac. Lógico	100	100	2042	143	1	463
Rac. Lógico	70	70	2042	67	6	64
Cont. Geral	100	100	2855	184	1	656
Cont. Geral	70	70	2855	55	7	85
Dir. Constit.	100	100	2461	156	1	610
Dir. Constit.	70	70	2461	52	6	96

Tabela 2 – Análise de indexação em cursos de EAD pelo agente *Crawler*.

Onde:

LC : Limite do *lower cut-off* em porcentagem.

UC : Limite do *upper cut-off* em porcentagem.

TP : Total de palavras que o texto contém, incluindo as repetidas.

FPUC : Freqüência referente ao peso do limite *upper cut-off*.

FPLC : Freqüência referente ao peso do limite *lower cut-off*.

TPA : Total de palavras analisadas dentro dos limites, sem incluir as repetidas.

A Figura 26 demonstra uma análise em um curso, cuja frequência da palavra de maior ocorrência é 263 e de menor ocorrênica é 1. Neste exemplo, tanto o *lower cutoff* como o *upper cutoff* foram definidos com o percentual de 50%, assim sendo, serão armazenadas as palavras que estão entre os limites de 36 ocorrências (*upper cutoff*) e 12 ocorrências (*lower cutoff*). A palavra de maior poder de resolução se encontra no pico da curva, cuja frequência de ocorrência é 23.

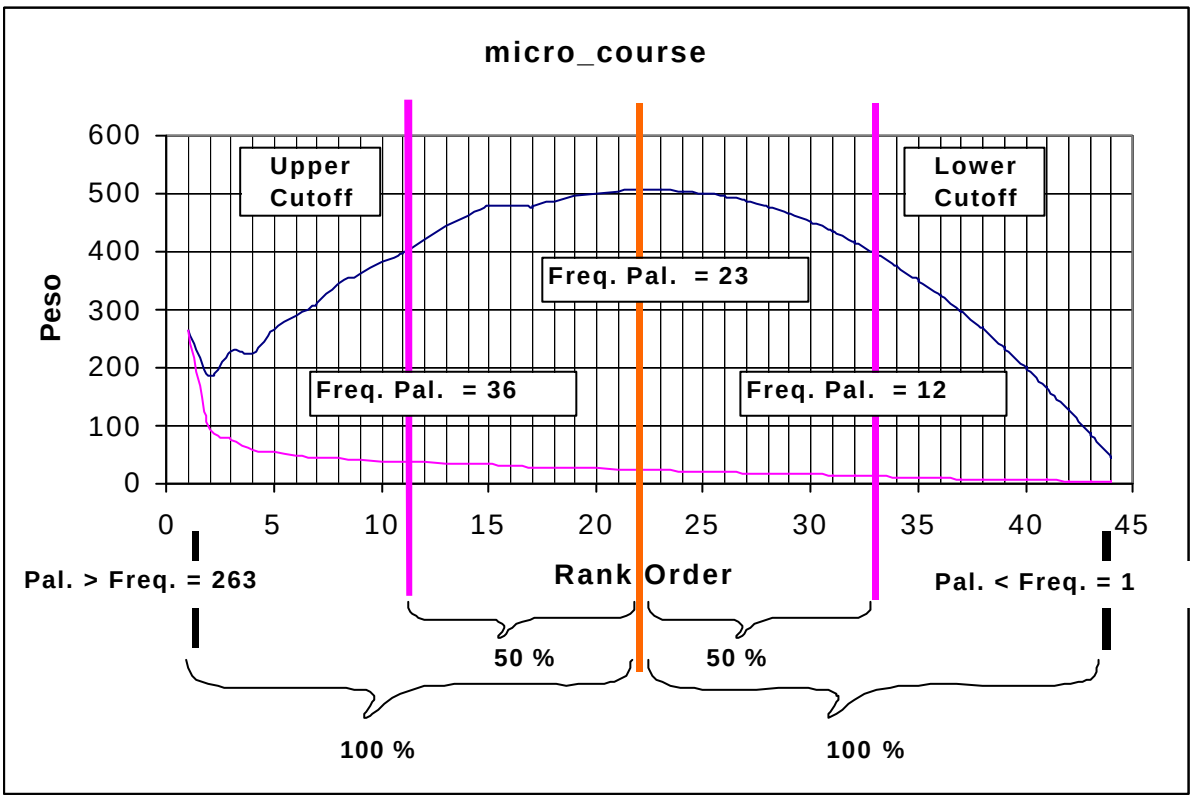

Figura 26. Análise de um curso aplicando a teoria de Luhn e Zipf

Quanto ao agente *mentor* do sistema GAAHA, um ponto muito importante referente ao desempenho deste agente, relacionado ao ambiente *Web*, refere-se ao tempo de resposta a uma requisição do usuário, o que geralmente depende do desempenho dos vários componentes da *Web* [71]. Três desses componentes afetam diretamente uma transação *Web* Banco de Dados:

- Desempenho do cliente *Web*.
- O tráfego na rede.
- O desempenho do servidor *Web*.

Além destes fatores pode-se identificar outros dois que afetam particularmente o desempenho de aplicações *Web* Banco de Dados. São eles:

- número de usuários: aplicações *Web* podem atingir um grande número de usuários simultâneos, o que pode degradar o desempenho do servidor *Web*, do servelet ou até mesmo do próprio DBMS.
- Balanceamento de carga: mecanismos mais sofisticados devem usar arquiteturas de processamento distribuído que podem distribuir os pedidos de usuários entre várias máquinas disponíveis.

A tabela 3 mostra alguns valores referente ao acesso a um banco de dados utilizando o agente *mentor*, via *Web*. Como já descrito acima, o tempo de resposta pode variar dependendo do tráfego da rede e outros motivos, também já relatados. No caso exemplificado na tabela 3, o acesso a pesquisa foi realizado em uma máquina cliente fisicamente localizada na mesma rede física do servidor HTTP e DBMS.

Curso	*String* de Pesquisa	Quantidade de Resposta	Tempo (seg)
Micro Course	*	863	4
Os Course	*	14317	35
Os Course	Browser	21	2
Racioc. Lógico	*	480	3
Racioc. Lógico	Algébricas	1	1
Direito Tributário	*	863	4

Tabela 3 – Análise de indexação em cursos pelo agente *Mentor*.

10.1.2. Flexibilidade.

O sistema é bastante flexível, permitindo que vários cursos possam ser indexados em um mesmo conjunto de tabelas. O número de *threads* e os limites de *cut-offs*, podem ser configurados pelo administrador no momento da indexação. É possível fazer a indexação de vários cursos sem que seja necessário a reexecução do aplicativo. Há também um controle visual de páginas visitadas, como mostrado na figura 20.

Quanto a página de pesquisa, cujo agente *mentor* é o responsável, existe uma boa flexibilidade na entrada de padrões de consultas, podendo o usuário fazer uma simples consulta com apenas uma palavra, ou até mesmo compor uma frase e submetê-la para pesquisa.

O programa tem uma grande portabilidade, podendo ser executado em diferentes *hardwares* e diferentes sistemas operacionais. Tanto o agente *crawler* como o agente *mentor* podem ser executados em um computador pessoal, assim como em uma poderosa estação de trabalho. Esta flexibilidade na execução se deve ao fato de que o sistema foi desenvolvido utilizando como linguagem básica o compilador Java.

10.1.3. Inovação.

Quanto a inovação, apesar de existir uma semelhança deste programa com os *searchs engines* disponíveis na Internet, como o Alta Vista (http://www.altavista.com), o GAAHA tem uma interface com o usuário mais trabalhada. Além disso, esse programa é capaz de usar técnicas e heurísticas muito mais particulares ao problema em questão (geração de arquivos de ajuda), pois o seu universo de aplicação é muito mais restrito que o dos *searchs engines*, tanto quanto ao seu escopo de aplicação, quanto a quantidade de documentos a serem pesquisados.

Os resultados desse trabalho demonstram que os objetivos do projeto foram alcançados e que o GAAHA é uma aplicação inovadora que deverá se mostrar muito útil para professores e alunos de cursos de EAD.

10.2. Oportunidades do Desenvolvimento.

Este projeto ainda não pode ser considerado como totalmente pronto. Ele ainda está em um estagio de versão *alpha*. Existem ainda alguns trabalhos que deverão ser realizados para que ele possa ser considerado uma aplicação comercial. Os itens abaixo relacionam alguns pontos que oferecem oportunidades para um desenvolvimento adicional:

- Algumas otimizações podem ser feitas sem grandes problemas, visando o aumento de performance. A substituição do gerenciador de banco de dados MySQL pelo PostGres já aumentaria a performance, pois o PostGres é muito mais robusto e consistente. O DBMS Oracle e Sybase, ou qualquer outro que tenha um *driver* JDBC, também poderia ser usado. Outro fator que melhoraria a performance é a otimização das expressões SQL, elaborando expressões mais eficientes.
- O processamento do agente *crawler*, quando o volume de informação é muito grande, fica muito moroso. Estudos terão de ser desenvolvidos visando melhorar a performance deste agente. O foco desta baixa performance, é justamente a quantidade excessiva de *selects*, *inserts*,*resultSet* e *wheres* que são realizados no banco de dados.

- O *layout* de apresentação do resultado da pesquisa, não é dos mais intuitivos, podendo ser necessário a sua modificação.
- O agente *crawler* tem alguns *bugs* que terão de ser resolvidos. Estes *bugs* ocorrem quando o volume de dados analisado é muito grande. Na fase em que os dados deveriam ser lidos das tabelas temporárias, analisados e armazenados nas tabelas permanentes, um problema ocorre aparentemente causado pelo *driver* JDBC. Este problema impede a inserção dos dados nas tabelas permanentes. Para resolver este problema, outros *drivers* deveriam ser estudados e testados.

Mesmo ainda sem estes melhoramentos, o sistema deve ser utilizado. Esta é a melhor forma de se encontrar *bugs* e receber um *feedback* dos usuários. Estas mudanças não afetam a estrutura básica do sistema.

10.3. Trabalhos Futuros.

Abaixo estão alguns tópicos interessantes para futuros trabalhos e desenvolvimentos adicionais:

- Desenvolver um controle de acesso de indexação de cursos, possibilitando que dois cursos com nomes iguais mas com conteúdos diferentes, possam ser indexados. Deste modo, o nome do curso ficaria vinculado com o seu desenvolvedor. Alem disso, os dados ficariam mais seguros, já que, para a reindexação de um determinado curso, seria exigido uma autenticação via senha.
- Marcação de caminho possibilitando que a trilha de pesquisa do usuário seja guardada como histórico para futuras consultas e portanto facilitando a memorização.
- Indexação utilizando páginas com o padrão XML com *tags* pré-fixados pelo desenvolvedor do curso. Estes *tags* poderiam ser descritos em um arquivo de configuração na execução do agente *crawler*.
- Gerador de resumos automáticos de cursos de EAD, utilizando a teoria de Luhn e Zipf [46] [47]. Alem do índice de pesquisa já existente, seria bastante útil se o usuário tivesse acesso a um resumo do curso em questão.

- Inserir agentes que analisem os caminhos de busca e ofereçam sugestões de como encontrar a informação para o usuário. Para isso, é necessário que os agentes estudem o comportamento do recolhimento das informações e o armazenamento destas informações. Como resultado deste estudo, o agente geraria uma lista de sugestões de consultas ao usuário.
- Utilizar um DBMS que de suporte ao uso de procedimentos armazenados (*stored procedures*). Eles são usados para otimizar o desempenho de operações e consultas mais comuns. Se uma interação da aplicação *Web* Banco de Dados for utilizada com muita freqüência, procedimentos armazenados no DBMS deveriam ser utilizados sistematicamente.

Referências Bibliográficas.

[1] LOYOLLA, W. P. D. C., Prates, M. *Educação à Distância Mediada por Computador (EAD)*: Uma Proposta Pedagógica. (http://www.puccamp.br/~prates/edmc.html, 1999).

[2] KEEGAN, D. *The foundations of distance education*. London: Croom Helm, 1986.

[3] PERRATON, H. A theory for distance education. In: SEWART, D., KEEGAN, D., HOLMBERG, B. (ed.) *Distance education*: International perspectives. New York: Routledge, 1988. p.34-35.

[4] MCISAAC, M. S., GUNAWARDENA, C. N. Distance education. In: JONASSEN, D. (ed.) *Handbook for Research on Educational Communications and Technology*. New York: MacMillan, 1997.

[5] JONASSEM, D. H. Applications and limitations of hypertext technology for distance learning. In: *Distance Learning Workshop*. San Antonio, Texas: Armstrong Laboratory, 1992.

[6] GARRISON, D. R., SHALE, D. An analysis and evaluation of audio teleconferencing to facilitate education at a distance. *International Journal of Distance Education*, v.1, n.1, p. 7-13, 1990.

[7] KEARSLEY, G. *The nature and value of interaction in distance learning*. In: DISTANCE EDUCATION RESEARCH SYMPOSIUM, 3., Washington: George Washington University, May p.18-21, 1995.

[8] MOORE, M. G., KEARSLEY, G. *Distance education:* a systems view. Belmont, California: Wadsworth, 1996.

[9] ROBERTS, J.M. The Story of Distance Education: A Practitioner's Perspective, *Journal of the American Society for Information Science*, v.47, n.11, p.811-816, 1996.

[10] SABA, F. *Introduction to Distance Education*. (http://www.distance-educator.com/intro.htm, 1998).

[11] SPODIK, E.F. *The Evolution of Distance Learning* – 4. Tools Available for Distance Education. (http://sqzm14.ust.hk/distance/evolution-distance-learning.htm, 1998).

[12] WILSON, J.M. Distance Learning for Continuous Education. *Learning, Communications and Information Technology*, v.32, n.2, 5 p., 1997. (http://www.educause.edu/pub/er/review/reviewArticles/32212.html, 1999).

[13] MATOS, H. A. B. *Sistemas de Formação*. Universidade de Coimbra. Coimbra, Portugal. (http://student.dei.uc.pt/~kikas/DLIndex.html, 1998).

[14] SALVADOR, V. L. G. Hipermídia interativa – a educação do futuro, no presente. *Tecnologia Educacional*. v.22, n.123/124, p.22-23, 1995.

[15] MAGALHÃES, M. G. M. *Estudo e avaliação da Educação à Distância utilizando a tecnologia WWW*. Dissertação (Mestrado) – Instituto de Física de São Carlos. São Carlos, 1997.

[16] RAVET, S., LAYTE, M. *Techonogy-based training*. London: Kogan Page, 1997.

[17] LAASER, W. et al. *Manual de criação e elaboração de materiais para educação a distância*. Brasília: CEAD, Editora Universidade de Brasília, 1997.

[18] LAASER, W. Virtual colloquy on the Internet. *Journal of Reserach in Educacional Media*, v. 4, n.1, p. 43-49, 1997.

[19] MAKI, W. S., MAKI, R. Learning without lectures: a case study. *The Computer*, p.107-108, 1997.

[20] AGOSTINHO, S., LEFOE, G., HEDBERG, J. *Online Collaboration for Effective Learning*: A Case Studies of the a Post Graduate University Course. (http://cedir.uow.edu.au/CEDIR/flexible/resources/lefoe.html, 1999).

[21] WILLIS, B. *Instructional Development for Distance Education*, ERIC Document Reproduction Service. (http://ericae.net/edo/ED351007.htm, 1999).

[22] CARVIN, A. More than just hype: the World Wide Web as a tool for education. *High School Journal*, v.79, n.2, p.76-86, 1995.

[23] BERNERS-LEE, T., et al. The World-Wide Web. *Communications of the ACM*. v.37, n.8, p.76-82, 1994.

[24] HUGHES, K. *Entering the Word-Wide Web*: A guide to cyberspace. 5.ed. Honolulu Community College, Hawaii, 1993. (http://www.hcc.hawaii.edu/guide/www.guide.html, 1999).

[25] DILLENBOURG, P., SCHINEIDER, D. Collaborative learning and the Internet. In: INTERNATIONAL CONFERENCE ON COMPUTER ASSISTED INSTRUCTION. Hsinchu, Taiwan, March, 7-10 1995.

[26] SCHNEIDER, D., BLOCK, K. The World-Wide Web in education. *Andrea Newsletter*, v.2, n.5, 1995.

[27] BARRIE, J. M., PRESTI, D. E. The World Wide Web as na instructional tool. *Science*, v.274, p.371-372, 1996.

[28] GOMES, M.T. *Seja um Alunauta e Garanta seu Futuro.* (http://www.2.uol.com.br/exame, 1997).

[29] KLEMM, W.R., UTSUMI, T. *Affordable and Accessible Distance Education:* A Consortium Iniciative. (http://www.usq.edu.au, 1997).

[30] LEWIS, J.H., ROMISZOWSKI, A. *Networking and the Learning Organization:* Issues and Scenarios for the 21st. Century. (http://www.usq.edu.au, 1997).

[31] GOMIDE, S. Volta às Aulas Virtual no Brasil. *Internet World*, p.62-67, 1996.

[32] ABED Associação Brasileira de Educação à Distância. (http://www.abed.org.br, 1999).

[33] SPENNEMANN, D.H.R. *On-Line Study Packages for Distance Education.* (http://www.csu.edu.au, 1998).

[34] ECKEL, B. *Thinking in Java.* New Jersey: Prentice Hall, 1998.

[35] CORNELL, G., HORSTMANN, C. *Core Java.* São Paulo: Makron Books,1997.

[36] *Java White Papers.* (http://java.sun.com/whitePaper/java-whitepaper-1.html, 1998).

[37] MCCONNELL, S. *Code Complete:* A Practical Handbook of Software Construction. Washington: Microsoft Press, 1993. Cap. 9.

[38] *Servlet Tutorial*, JavaServer Products. (http://jserv.javasoft.com/products/java-server/documentation/toolkit/servlets/servlet_tutorial.html 1999).

[39] WILLIAM, C. *Developing Java Servlets.* (http://webreview.com/wr/pub/97/10/10/feature/main.html, 1999).

[40] HWANG, Y. *Database Application Development in Client/Server Environment.* (http://bpa239023.bpa.arizona.edu/mis696g/default.htm, 1999).

[41] MILLER,M.J. *MySQLSQLManual.* (http://www.atd.ucar.edu/softwaree/ MySql/ MySQL_SQL_Manual0.41.html, 1999).

[42] RIJSBERGEN, B. S. *Information Retrieval*. London: Butterworths, 1970.

[43] KEENAN, E. L. On semantically based grammar. *Linguistic Inquiry*, v.3, p. 413-461, 1972.

[44] KEENAN, E. L. *Formal Semantics of Natural Language*. Cambridge: University Press, 1975.

[45] MONTEGOMERY, C. A. Linguistics and Information Science. *Journal of the Americam Society for Information Science*, v.23, p.195-219, 1972.

[46] LUHN, H. P. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, v.2, p.159-165, 1958.

[47] ZIPF, H. P. *Human Behavior and the Principle of Least Effort.* Cambridge, Massachusetts: Addison-Wesley, 1949.

[48] EDMONDSON, H. P., WYLLYS, R. E. Automatic Abstracting and IndexingSurvey and Recommendations. *Communications of the ACM*, v.4, p.226-234, 1961.

[49] LIMA, I. N.. *O Ambiente Web Banco de Dados:* Funcionalidades e Arquiteturas de Integração. Dissertação (Mestrado) – Departamento de Informática da PUC do Rio de Janeiro, 1997.

[50] NORMAN, D.A. How might People Interact with Agents? In: BRADSHAW, J.M.. Ed. *Software Agents*. Menlo Park, Calif. : AAAI Press ; Cambridge, Mass. : MIT Press, 1997.

[51] NEGROPONTE, N. Agents*:* From Direct Manipulation to Delegation. In: BRADSHAW, J.M.. Ed. *Software Agents.* Menlo Park, California: AAAI Press ; Cambridge, Mass. : MIT Press, 1997.

[52] KAY, A. Computer Software. *Scientific American*, v.25, n.3, p.53-59, 1984.

[53] NWANA, H.S. Software Agents: An Overview. *Knowledge Engineering Review*. v.11, n.3, p.205-244, 1996.

[54] APPLE COMPUTER. *AppleScript Language Guide*. Reading, Mass.: Assison-Wesley, 1993.

[55] KNOBLOCK, CRAIG A.; AMBITE JOSE L. *Agents for Information Gathering Software Agents*. Menlo Park, California : AAAI Press, MIT Press., 1997.

[56] MINSKY, M. *The society of Mind*. New York: Simon & Schuster, 1986.

[57] MINSKY, M. and RIECKEN, D. A Conversation with Marvin Minsky about Agents. *Communications of the ACM*, v.37, n.7, p.23-29, 1994.

[58] MOULIN, B., CHAIB-DRAA, B. An Overview of Distributed Artificial Intelligence. In: O'HARE, G.M.P., JENNINGS, N.R. Eds. *Foundations of Distributed Artificial Intelligence*. New York: Wiley, 1996. p.3-55.

[59] COUTAZ, J. *Interfaces Homme Ordinateur*: Conception et Réalisation. Paris: Editions Bordas,1990.

[60] WIEDERHOLD, GIO. Mediators in the Architecture of Future Information Systems. Computer, p.38-49, 1992.

[61] BOY, G.A. *Intelligent Assistant Systems.* San Diego, California: Academic Press, 1991.

[62] MAES, P. Agents that Reduce Work and Information Overload. In: BRADSHAW, J.M. Ed. *Software Agents.* Menlo Park, California: AAAI Press ; Cambridge, Mass. : MIT Press, 1997.

[63] WHITE, J. Mobile Agents. In: BRADSHAW, J.M. Ed. *Software Agents.* Menlo Park, California: AAAI Press ; Cambridge, Mass. : MIT Press, 1997

[64] *Netscape Communications Corporation*: The NSAPI Versus the CGI Interface, (http://home.netscape.com/newsref/std/nsapi_vs_cgi.html, 1998).

[65] BALL, G. et al. Lifelike Computer Characters: The Persona Project at Microsoft Research. In BRADSHAW, J.M. Ed. *Software Agents.* Menlo Park, California: AAAI Press ; Cambridge, Mass. : MIT Press, 1997.

[66] BATES, J. The role of Emotion in Believable Agents. *Communications of the ACM* v.37, n.7, p.122-125, 1994.

[67] HAYES-ROTH, B et al. Multiagent Collaborative in Direct Improvisation. In: LESSER, V., GASSER, Les Eds. *Proceedings Of The First Internaional Conference On Multi-Agent Systems (ICMAS-95)*. Menlo-Park, California: AAAI Press, 1995. p.148-154.

[68] SHOHAM, YOAV. An Overview of Agent-Oriented Programming. In : BRADSHAW, J.M., Ed. *Software Agents*. Menlo Park, California: AAAI Press ; Cambridge, Mass. : MIT Press, 1997.

[69] GENESERETH, M.R. An Agent-Based Framework for Interoperability. In : BRADSHAW, J.M., Ed. *Software Agents*. Menlo Park, California: AAAI Press ; Cambridge, Mass. : MIT Press, 1997.

[70] GILBERT, D. et al. *IBM Intelligent Agent Strategy*. IBM Corporation, 1995.

[71] MURTA, C. D., ALMEIDA, J. M,. ALMEIDA,V. A. Análise de Desempenho de um Servidor WWW. In: XXII SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 22.1996, Recife. *Anais...* Recife: SBC, 1996. p.391-402.

[72] *W3C Consortium. W3C* . (http://www.w3.org/pub/WWW/, 1998)

[73] BERSON, A. *Client/Server Architectur*. New York : McGraw-Hill, 1992.

[74] COMER, D.E., STTEVENS, D. L.. *Internetworking with TCP/IP . Volume III. Client-server programming and applications*. Upper Saddle River, N.J. : Prentice-Hall, 1994.

[75] YEAGER, N.J., McGRATH, R.E. *Web Server Technology* - The advanced guide for World Wide Web Information Providers. San Francisco : Morgan Kaufmann : 1996.

[76] MAGID, J., MATTHEWS, R. D., JONES, P. *The Web server book*: tools & techiniques for building your own internet information site, Ventana Press, 1995.

[77] YATES, R. B.. An extended model for full text databases. *Journal of the Brazilian Computer Society,* v.2, n.3, abril, 1996.

[78] CHEN, H. et all. *A concept space approach to addressing the vocabulary problem in scientific information retrieval*: na experiment on the worm community system. MIS Department, University of Arizona, 2 de Julho de 1996. (http://ai.bpa.arizona.edu/ papers/ ,1998).

[79] CHEN, H. *A textual database/knowledge-base coupling approach to creating computer-supported organizational memory*. MIS Department, University of Arizona, 5 de Julho de 1994. (http:// ai.bpa.arizona.edu/papers/ , 1997).

[80] CHAKRAVARTHY, A. S., HAASE, Kenneth B. *NetSerf*: using semantic knowledge to find Internet information archives. IN: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 18, 1995, Seattle. *Proceedings*... Seattle : ACM, 1995.

[81] MOULIN, B., ROUSSEAU, D. Automated knowledge acquisition from regulatory texts. *IEEE Expert*. v.7, n.5, p.27-35, 1992.

[82] HARDY, D. R., SCHWARTZ, M. F. Essence: a resource discovery system based on semantic file indexing. In: USENIX, San Diego, CA, 1993. *Proceedings*... http://citeseer.nj.nec.com/papers/cs/2109/ftp%3A%23@S@%23%23@S@%23ftp.cs.colorado.edu%23@S@%23pub%23@S@%23techreports%23@S@%23schwartz%23@S@%23Essence.Conf.pdf

[83] COWIE, J., LEHNERT, W. Information extraction. *Communications of the ACM,* v.39, n.1, p. 80-91, jan 1996.

[84] WIEBE, J.M., HIRST, G., HORTON, D. Language use in context. *Communications of the ACM,* v.39, n.1, p.102-111, jan 1996.

[85] FURNAS, G. W. et all. The vocabulary problem in human-system communication. *Communications of the ACM,* v.11, n.30, p. 964-971, nov 1987.

[86] SOUZA, C. S. The semiotic engineering of user interface languages. *International Journal of Man-Machine Studies*. New York : Academic Press, 1993.

[87] *National Center for Supercomputing Applications*: Common Gateway Interface Overview, University of Illinois at Urbana-Champaign, (http://hoohoo.ncsa.uiuc.edu/cgi/overview.html , 1997).

[88] *National Center for Supercomputing Applications*: The CGI Specification - version 1.1, University of Illinois at Urbana-Champaign, (http://hoohoo.ncsa.uiuc.edu/cgi/interface.html , 1997).

[89] *W3C Consortium*: CGI-Common Gateway Interface, Work in progress, (http://www.w3.org/pub/WWW/CGI/ , 1998).

[90] *W3C Consortium*: The Hypertext Transfer Protocol, (http://www.w3.org/hypertext/WWW/Protocols/Overview.html , 1999).

[91] STEIN, D. L. *How to set up and mantain a world wide web site*: the guide for information provides. Reading : Addison-Wesley, 1995.

[92] BERNERS-LEE, T., CONNOLLY, D. *Hypertext markup language specification - 2.0.*, IETF HTML Working Group, (http://www.ics.uci.edu/pub/ietf/html/rfc1866.txt , 1998).

[93] McARTHUR, D. C. World Wide Web : Preparing documents for online presentation, *Dr. Dobb's journal*, p.18-26, 1994.

[94] *W3C Consortium*: The HyperText Markup Language, (http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html , 1999).

[95] *W3C Consortium*: Overview of SGML and XML Resources, (http://www.w3.org/MarkUp/SGML/ , 1999).

[96] LOH, STANLEY. Descoberta De Conhecimento Em Bases De Dados Textuais. (http://mozart.ulbra.tche.br/~loh/apostilas/dc-texto.htm, 1999).

[97] BERNERS-LEE, T., MASINTER, L., McCAHIEL, M. *Uniform Resource Locators (URL)*, RFC 1738. Mindesota : University of Minnesota, 1994.

[98] LIU, C., ALBITZ, P. *DNS and BIND*, Sebastopol, CA: O'Reilly, 1992.

[99] SILBERSCHATZ, A., STONEBRAKER, M., ULLMAN, J. Database research: achievements and opportunities into the 21st century. *Journal of the Brazilian Computer Society*, v.2, n.3, p.1996.

[100] BERGHEL,H. The client's side of the World-Wide Web. *Communications of the ACM*, v.39, n.2, p.86-98, 1996.

[101] KIM, E. E. *CGIhtml version 1.66*. (http://www.eekim.com/software/cgihtml/ ,1998).

[102] *National Center for Supercomputing Applications*: NCSA HTTPd. A WWW Server, University of Illinois at Urbana-Champaign, (http://hoohoo.ncsa.uiuc.edu/docs/Overview.html, 1999).

[103] REICHARD, K. *Web servers for database applications*, Internet Systems, (http://www.dbmsmag.com/9610i08.html, 1999).

[104] *WebQuest Support and Information Center*: SSI+2.0 Reference, (http://webquest.questar.com/reference/ssi/ssi+20ref.sht, 1997).

[105] DENNY, B. R. SSI, CGI, API, or SSS, Choosing the Right Tool for the Job, (http://solo.dc3.com/white/extending.html, 1999).

[106] *National Center for Supercomputing Applications*: Server Side Includes (SSI) - NCSA HTTPd, University of Illinois at Urbana-Champaign, (http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html, 1999).

[107] BENETT, G. *Intranets*: Como implantar com sucesso na sua empresa. São Paulo : Editora Campus, 1996.

[108] *Netscape Communications Corporation*: The NSAPI - Netscape Server API, (http://www.netscape.com/newsref/std/server_api.html, 1998).

[109] *Microsoft Corporation*: Internet Server API Reference, (http://microsoft.com/win32dev/apiext/isapiref.htm, 1998).

[110] *Apache Corporation*: Apache API Notes, (http://apache.org/docs/misc/API.html, 1999).

[111] *WebSite Professional*: WebSite API 1.1 SDK Introduction and Overview, (http://solo.dc3.com/wsapi/ , 1999).

[112] *Microsoft Corporation*: Internet Server API Overview, (http://microsoft.com/win32dev/apiext/isapimrg.htm, 1998).

Glossário.

Âncora – tipo de ligação hipertexto em documento da WWW. Ao "clicar" sobre uma âncora, o usuário "salta" para outro ponto do mesmo documento ou para qualquer outro documento que esteja disponível na WWW.

Applet – são pequenos programas escritos em Java que são embutidos em páginas Web para produzir efeitos especiais.

API – *Application programmer's Interface*, é simplesmente um conjunto de funções disponíveis para a programador utilizar .

Bugs – Erros internos de lógica de programação nos programas.

CGI - *Common Gateway Interface* é um padrão que permite outros programas interagir com o servidor Web e com o usuário.

Crawler – Um programa que percorre páginas dos documentos HTML, como uma espécie de "robo vasculhador".

Browser (navegador) – programa utilizado para visualizar as páginas na WWW. Como o endereço de um *site*, o *browser* recebe as informações disponíveis no *site,* o *browser* recebe as informações disponíveis no *site* e as interpreta, exibindo na tela do computador do usuário imagens, textos, sons, animações, etc.

Cliente/Servidor – arquitetura de interconexão de computadores, na qual existe um computador central (o servidor) que libera (ou não) acesso a computadores a ele ligados (os clientes).

Documento – na Internet, é um arquivo HTML ou página da WWW

E-mail *(eletronic mail – correio eletrônico)* – é um serviço da Internet que permite a troca de mensagens escritas, às quais podem ser anexadas imagens, vídeos, documentos ou qualquer outro tipo de arquivo.

FTP *(File Transfer Protocol)* – é um dos protocolos mais populares, pois funciona como padrão para a transferência de arquivos na Internet.

Gopher – sistema não gráfico de navegação na Internet, no qual o usuário vê menus que indicam arquivos disponíveis em diferentes computadores da rede.

Hipertexto – é uma estrutura não linear de ligação de várias informações (textos, imagens), possibilitando o usuário seguir diferentes caminhos de leitura de documentos. Os documentos são apresentados com palavras em destaque; estas palavras representam as ligações com outros documentos.

Home page – página de apresentação. A primeira página que o usuário vê ao entrar em determinado endereço na WWW.

IE – Internet Explorer (Browser da Microsoft).

Interface – elo de comunicação e interação entre computador e o usuário.

Java – Linguagem de programação utilizada para criar interatividade em páginas disponíveis na **WWW**. Permite a criação de programas independentes de plataformas, ou seja, podem ser executados em qualquer computador ou sistema operacional.

Link (elo de ligação) – ponto de ligação entre partes diferentes de um hipertexto ou entre hipertextos. Ponto de um texto ou imagem, através do qual o usuário "salta" para outra fonte de informação relacionada (texto, imagem, animação). É a ferramenta essencial de navegação entre documentos.

Monitor - Um monitor é uma coleção de procedimentos, variáveis e estruturas de dados que são todas agrupadas juntas em um tipo especial de módulo ou pacote.

On line – possibilidade de interação via computador permitindo uma flexibilização quanto ao tempo e localização.

Password – é uma proteção à um sistema via uma senha.

Plataforma – é a combinação de um sistema operacional mais o equipamento, ou o *hardware* no qual esse sistema é executado.

Protocolo – conjunto de regras e procedimentos para transmissão de dados entre dispositivos ligados em rede.

Scripts – é uma coleção de procedimentos e funções não muito extensos para uma execução rápida.

Server, Servidor – termo genérico que designa o computador considerado central em algum processo. Portanto, existem os servidores de impressão (computadores que gerenciam a impressão em uma rede), servidores de mensagens (que gerenciam as mensagens enviadas e recebidas por componentes da rede), etc.

Site (*localidade*) – é qualquer endereço na Internet.

SQL – padrão para uma Linguagem de Consulta Estruturada (Structured Query Language). SQL é uma linguagem projetada para interação com um banco de dados relacional.

Tags (marcações ou marcadores) – códigos de formatação de texto usados em documentos HTML da WWW.

Thread – Um thread é um simples fluxo seqüencial de controle dentro de um programa.

URL (*Uniform Resource Locator*) – padronização da localização ou dos detalhes de endereçamentos dos recursos da Internet.

WAIS (*Wide Area Information Service*) – sistema utilizado para localização de informações em bancos de dados disponíveis na Internet.