UNIVERSIDADE FEDERAL DE PELOTAS
Instituto de Física e Matemática
Curso de Informática

Tecnologia de Hiperdocumentos Dinâmicos

por

Isabel Florinda Bernardo de Kerlan

Dissertação submetida à avaliação, como requisito parcial para a obtenção do título de Bacharel em Informática

Prof. Maurício Nunes
Orientador

Pelotas, Dezembro de 2000

Agradecimentos

Agradeço ao meu mestre e orientador Mauricio Porto pela dedicação ao nosso trabalho, pelos conselhos, pela paciência e pelos ensinamentos transmitidos durante este curto espaço de tempo, que na verdade parece bem maior, pelo exemplo de vida. Obrigada sempre.

Agradeço aos meus pais, que encaminharam a pessoa que eu sou hoje. Sem os seus exemplos de luta, perseverança, ambição, amor ao trabalho e, acima de tudo, de qualidade, eu provavelmente não estaria escrevendo esta página, pelo menos não com esta emoção.

Agradeço aos meus irmãos pelo incentivo, e embora distantes têm dado todo apoío e compreensão com a minha dedicação.

Agradeço ao Luís, meu namorado, que me apoiou dando o seu incentivo, me aconselhando, embora durante esses esses anos distantes um do outro, vibrou comigo a cada conquista e esperou com muita paciência para que esse trabalho estivesse pronto.

Aos meus colegas do curso, em especial ao meu colega Josê Wilson pelo coleguismo e por sempre se colocar a disposição para o que eu precisasse com toda boa vontade, além da paciência que teve a me ajudar na organização e formatação do meu trabalho.

Aos meus professores com quem convivi durante essa fase de vida, passando bons e não tão bons momentos juntos em nome de um objetivo.

Finalmente, agradeço a todos os meus amigos. Sem a amizade de vocês todo e qualquer trabalho seria inútil.

Sumário

Lista de Abreviaturas

API (*Application Programming Interface*)-Interface de Programação de Aplicações.

DTD (*Document Type Definitions*)- Definições de Tipo de Documento)

DSSSL (*Document Style and Semantics Specification Language*)- Linguagem de Estilo e de Especificação Semântica do Documento

CERN Centro Europeu de Pesquisas Nucleares

DNS Sistema de Nome de Dominio

CGI (*Common Gateway Interface*) -Interface de Gateway Comum.

CSS (*Cascading Style Sheets*)- Folha de Estilo em Cascata

DOM (*Document Object Model*)- Modelo de Documentos de Objetos

FTP *(File Transfer Protocol*)-Protocolo de Transferencia de Arquivo

GCA (*Graphic communications association*)-Associação de Comunicação Grafica

HTTP (*HyperText Transfer Protocol*) -Protocolo de Transferência de Hipertexto

HTML (*HyperText Markup Language*) -Linguagem de Marcação de Hipertexto

IIS (*Internet Information Server*) -Servidor de Informação de Internet.

IP Protocolo de Internet

ISO (*International Standards Organization*)- Organização de Padrão Internacional

JDBC (*Java DataBase Connectivity*) -Conexão de Banco de Dados em Java

LANs (*Local Area Network*)- Area de Redes Locais

NNTP (*News Network Transfer Protocol*)

NLS (*On-line Sytem*)- Sistema *On-line*

SGBD Sistema de Gerenciamento de Banco de Dados

SMTP (Simple Mail Transfer Protocol)-Protocolo de Transferencia de e-mail Simples

TCP Protocolo de Controle de Transmição

URL (*Universal Resource Locator*) -Localizador de Recurso Universal

WAP (*Wireless Application Protoco*l) Protocolo de Aplicação de Rádio

Web (v. WWW)

WWW (*World Wide Web*)

XSL (*XML Style Language*)- Linguagem de estilo XML

Lista de Figuras

Lista de Tabelas e Listagens

Resumo

Este trabalho propõe um estudo sobre as tecnologias usadas na geração de hiperdocumentos dinâmicos, caracterizados por serem construídos ou completados imediatamente antes do seu envio ao browser. Através deste trabalho pretende-se avaliar as diferentes técnicas de construção desses documentos e analisar suas aplicabilidades. Serão construídos pequenos exemplos dos principais conceitos e técnicas estudados, a fim de consolidar o conhecimento adquirido e servir de orientação para os novos usuários dessas técnicas.

1 INTRODUÇÃO

Até alguns anos atrás, a área de informática era restrita a apenas alguns poucos grupos empresariais e instituições governamentais, e o conhecimento nessa área era guardado a sete chaves por seus programadores e desenvolvedores. Com o avanço da micro-eletrônica, computadores cada vez menores e mais baratos começaram a ser produzidos, o que causou uma explosão de crescimento na base de computadores instalados em todo mundo.

Com o aumento de micros, novos programas e sistemas, cada vez mais desenvolvidos, começaram a surgir para atender a demanda de novos sistemas em áreas cada vez mais distintas e com características cada vez mais complexas. Paralelamente a esse crescimento uma outra área estava se desenvolvendo: a área de redes, e juntamente com ela a aflorar, pelo mundo afora, a idéia de uma rede mundial de computadores.

Por volta de 1990, um professor do CERN chamado Tim Berners-Lee, elaborou uma forma de acesso a documentos via uma rede de computadores interna ou remota. A intenção de Bernes-Lee era a de poder trocar informações com outros cientistas em todo mundo, utilizando recursos de texto, imagens e multimídia. No final de 1990 os protocolos e métodos criados por ele já começavam a ser utilizados pela comunidade científica, e assim surgiu a *World Wide Web* (WWW), ou apenas *Web*.

Baseada em uma arquitetura com padrões abertos e independentes, a *Web* foi criada no topo da Internet, incorporando assim as características de ser um ambiente distribuído e multiplataforma [LIM 97].

Originalmente, os documentos apresentados na Web eram estáticos, isto é, os documentos ficavam armazenados no servidor Web prontos para serem enviados ao cliente *Web*. Desta forma, os documentos eram acessados instantaneamente, mais não era possível o envio de dados ou conteúdo dinâmico.

Com a crescente popularização do uso da Web, novas tecnologias foram surgindo para permitir a construção e recuperação de documentos contendo não apenas conteúdo estático mas também conteúdo dinâmico atualizado a cada recuperação.

Assim, a proposta deste trabalho é o estudo sobre as tecnologias usadas para construção dos hiperdocumentos dinâmicos

Inicialmente no cápitulo 2 será dada a definição, conceituação de WWW, Internet, hipertexto, hiperdocumentos dinâmicos e o modelo cliente-servidor. Seguidamente no capítulo 3 é apresentado um estudo sobre as arquiteturas das aplicações *Web*, ainda neste capítulo é mostrado um diagrama da taxonomia das arquiteturas Web. No capitulo 4 será apresentado um estudo sobre as linguagens descritoras de geração de hiperdocumentos dinâmicos. Será demostrado também o funcionamento das tecnologias através da

implementação de exemplos no capítulo 5. E finalmente no capítulo 6 apresentará as conclusões e contribuições.

2 CONCEITOS BÁSICOS

Este capítulo apresenta algumas definições com objetivo de termos alguma ideia rápida e sintética de alguns termos que abordaremos neste trabalho.

2.1.1 Internet

A Internet é uma rede de computadores que estão interligados em escala mundial (longa distância). A Internet é baseada no protocolo TCP/IP (*Transfer Control Protocol / Internet protocol*), que é a linguagem de mais baixo nível através da qual transitam as informações: pacotes com conjuntos de *bytes* que são trocados entre computadores. Para realizar esta troca de informações, cada computador é identificado por um número, o seu número IP, como por exemplo, a máquina responsável pelo servidor *Web* principal da UFPEL tem um número IP 200.17.161.33.

O protocolo TCP/IP trabalha diretamente com os endereços numéricos. Como para o ser humano convém trabalhar com nomes, existe servidores responsaveis (DNS) pela tradução dos nomes textuais para o correspondente numero IP.

Alguns programas do UNIX (Linux) permitem determinar se uma máquina está ativa, qual é a rota de acesso á uma outra máquina, e os endereços IP de máquinas ligadas a Internet. Estes programas são, respectivamente: *ping, traceroute e nslookup*. Existem também programas que permitem realizar uma conexão entre duas máquinas para serviços especiais, tais como o FTP (transferência de arquivos) e o TELNET (emulador de terminal).

Assim como existem redes de longa distância como a Internet, as redes locais (*LANs - Local Area Networks*) conectam computadores localizados num mesmo prédio ou conjunto próximo de prédios. Uma rede local é a candidata perfeita para o uso de uma Intranet, ou seja, uma pequena rede Internet, mas só que ao nível de corporação. A Intranet segue os mesmos conceitos da Internet, e pode inclusive usar ferramentas similares, mas o seu objetivo não é a troca de informações com o mundo todo, mas apenas ao nível de uma empresa, ou mais simplesmente, dentro de uma rede local que costuma também ter uma conexão a Internet [GRA 96].

2.1.2 WWW (*World Wide Web*)

A sigla WWW significa *World Wide Web* e é também conhecida como W3. Definido formalmente como sendo um sistema de informações hipermídia baseado na Internet, ou seja uma rede que permite o acesso e a transmissão de informações através de computadores que interligam o mundo inteiro.

O projeto W3 deu aos usuários da informática, e da Internet, uma poderosa ferramenta que permite o acesso à uma grande variedade de documentos, realizado de uma maneira muito simples. Atualmente o W3C *(WWW Consortium)* é o organismo responsável pela padronização das ferramentas, protocolos e linguagens ligados ao WWW [TEI 95].

Em resumo, o WWW é uma ferramenta hipermídia (resultado da mistura de paginas de internet a outros meios de comunicação como sons, video etc.) baseada na Internet e no uso de uma arquitetura cliente-servidor (*browser/server*), cujo protocolo de base é o HTTP (*Hypertext Transfer Protocol*), que transporta os hiperdocumentos.

2.1.3 Hipertexto e Hiperdocumento

A idéia de um sistema on-line capaz de manter e gerenciar uma vasta quantidade de informações interconectadas e acessíveis através de uma interface simples e consistente foi descrita pela primeira vez por Bush em 1945. Em um artigo entitulado "*As we may think*", este autor afirma que a maior parte dos sistemas de indexação e organização de informações em uso na comunidade científica são artificiais, já que levam em conta uma ordenação puramente hierárquica.

Segundo [BUSH 45], a mente humana funciona através de associações, seguindo uma intrincada rede de representações. Inspirado no processo associativo que embasa o exercício do pensamento, ele imaginou um dispositivo capaz de suportar uma vasta biblioteca, bem como notas pessoais, fotografias, rascunhos, etc. Este sistema, denominado *Mamex*, tinha como característica essencial a habilidade de unir dois ítens quaisquer de sua base de dados através de *links* (elos) rotulados.

As idéias de Bush permaneceram no papel até a década de 60. Em 1968, Doug Engelbart desenvolveu o *On-line System* (NLS) no Instituto de Pesquisa de *Stanford*. Este sistema incorporava várias idéias de Bush e tinha como objetivo proporcionar ao grupo de pesquisadores um ambiente de trabalho *online.*

Bush e Engelbart definiram diversas idéias e conceitos utilizados no desenvolvimento de sistemas de hipertexto. Entretanto, o termo hipertexto foi criado somente em 1965 por Ted Nelson, exprimindo a noção de escritura/leitura não linear em um sistema de informática denominado Xanadu.

Nelson foi um pioneiro em hipertexto com Xanadu, um sistema de edição e gerenciamento de textos que foi projetado para instalação em um ambiente distribuído. Através dele os usuários tinham a possibilidade de interagir, criar novos textos, introduzir novos documentos ou modificar os já existentes. Já o termo hiperdocumento corresponde a um documento que incorpora os conceitos de hipertexto. [PIM 99].

Um hipertexto é basicamente como um outro texto qualquer de um computador, que pode ser armazenado, editado ou lido; entretanto ele possui uma importante diferença: o hipertexto possui ligações dentro do texto para outras partes deste texto, ou mesmo, para outros documentos separados. Sendo assim, um hipertexto pode ser lido de uma maneira diferente da convencional, visto que ele não possui apenas uma única seqüência lógica a ser seguida. Pode-se "avançar" dentro de um assunto seguindo diferentes caminhos (passando de uma ligação à outra, atraves de *hiperlinks* segundo interesse do usuário). Os *hyperlinks* são os fios de conexão que formam *a World Wide Web* [STA 97].

2.1.4 Hiperdocumentos Dinâmicos

São documentos cujo o conteúdo é gerado automaticamente dependendo de vários fatores como por exemplo a data ou hora; ou que podem ser reconstruídos instantaneamente a partir de uma solicitação específica, tornando o documento mais interativo com o usuário. O que caracteriza um documento HTML dinâmico é a existência de código de programação que o complete de alguma forma ou permita que o documento interaja com o usuário reagindo às suas ações e/ou informações fornecidas.

No caso de o documento ser gerado ou completado imediatamente antes de seu envio ao *browser* (navegador), esse código é executado no lado do servidor Web (vide 1.1.5), tendo como resultado o hiperdocumento enviado ou seja, este é construído dinamicamente.

Quando o código faz parte do hiperdocumento enviado ao *browser*, na forma de *scripts* (em geral, Javascript), é executado pelo próprio *browser*, acrescentando caracteristicas de dinamismo (interação, por exemplo) ao hiperdocumento.

O código no lado servidor pode ser executado pelo próprio servidor *Web* ou ser invocado por este a um outro programa, que lhe devolve como resultado o hiperdocumento, resposta que é repassada ao cliente (*browser*). O primeiro caso caracteriza um servidor estendido e o segundo a delegação dessa tarefa através de uma interface denominada CGI [vide 2.2.1].

2.1.5 Modelo cliente-servidor

A *Web* funciona segundo o popular modelo cliente-servidor que é um modelo conceitual, adotado para disciplinar e orientar o projeto e a implementação de aplicações distribuídas formadas a partir de diversos processos que interoperam para resolver um problema computacional comum. Esses processos podem executar no mesmo processador ou então podem ser distribuídos por meio de vários processadores em uma rede. Na arquitetura cliente/ servidor uma aplicação distribuida é conceitualmente modelada para ser composta somente por dois processos cooperantes: o processo cliente e o processo servidor na *Web*. O cliente e servidor se comunicam através de um protocolo de alto nível, conhecido como HTTP (*HyperText Transfer Protoco*l). Um servidor *Web* é um programa cujo único propósito é fornecer documentos para os clientes quando estes forem requeridos. Um cliente *Web* é um programa que faz a interface com o usuário e realiza pedidos de documentos a um servidor especificado pelo usuário. O servidor fica usualmente em um estado de espera, aguardando o pedido de uma página que vai ativa-lo e fazer com que este forneça as informações solicitadas [TAN 97].

Clientes Web

O cliente *Web* (*browser*) é um programa que interage com o usuário, permitindo a vizualização desses as paginas de Internet.

Os *browsers* mais utilizados são *browsers* para ambientes graficos como o *Netscape*, o *Internet Explorer*, existindo tambem *browsers* para uso em modo texto, como o *Lynx*, por exemplo. Eles permitem a realização de tarefas como:

- Consulta a páginas locais de documentos em HTML e documentos de texto;
- Consulta a páginas remotas de documentos em HTML (através de uma conexão a uma máquina remota, utilizando-se do protocolo HTTP);
- Exibição de documentos que incluem: textos, formulários, gráficos, sons, animações, e até mesmos programas Java ou ActiveX;
- Interação com o usuário através da navegação de um documento;
- Acesso a facilidades de envio e consulta de e-mails (protocolo SMTP);
- Acesso a facilidades de envio e consulta do sistema de *News* (protocolo NNTP);
- Acesso aos recursos de ftp: busca e listagem de arquivos (protocolo FTP);
- Acesso a serviços de rede como: *gopher*, *wais*, *telnet*.
- Execução de programas externos (*helper applications*) para tratar informações transmitidas pela rede para as quais o *browser* não está habilitado à exibir diretamente na tela. Exemplo: arquivos compactados do tipo Zip, formatos gráficos específicos (CDR, DXF, TIFF), documentos em formato *Adobe Acrobat* (PDF) ou *PostScript* (PS), etc.
- Execução de um módulo *plug-In* que será responsável da integração de novas potencialidades de tratamento de informações junto ao *browser*.
- Controle de acesso a documentos protegidos (interação com o usuário);

Todos estes recursos do *browser*, descritos acima, são acessados através de um URL - *Uniform Resource Locator*, que é uma forma padrão de especificar o documento ou recurso desejado e, opcionalmente passar parâmetros [TEI 95].

Servidor Web

Servidores são programas que fornecem serviços especializados através de uma rede de computadores. Eles aceitam os pedidos vindos do cliente (*browser*), executam, e retornam os resultados para os mesmos.

As tarefas de um servidor Web são:

- Estabelecer a conexão com o cliente, através do protocolo HTTP;
- Fornecer os documentos solicitados pelos clientes, onde se destacam os seguintes tipos de documentos e arquivos à serem enviados pela rede Internet do servidor para o cliente:
  - Documentos hipermídia em formato HTML (htm, html - Os mais usados);
  - Documentos contendo textos e imagens: txt, doc, rtf, pdf, ps etc.
  - Arquivos de Imagens: gif, jpg;
  - Arquivos de sons: wav, mid, mod, au, mp3;
  - Arquivos de animações: avi, mov, gif;
  - Arquivos com elementos 3D: vrml;
  - Programas em Java ou ActiveX;
  - Arquivos em geral (zip, tar, gz, etc.)
- Executar programas e *scripts*, conhecidos como programas CGIs - *Common Gateway Interface* (Podem ser programas escritos em "C", *scripts* Unix, perl, etc);
- Receber chamadas à programas e *scripts* com os respectivos parâmetros, e realizar a passagem destes parâmetros aos programas que vão ser executados, assim como, recuperar os resultados da execução dos programas e repassá-los ao cliente;
- Receber os dados de um formulário HTML e repassá-los à um programa pelo protocolo CGI.
- Controlar o acesso a certos arquivos e áreas do disco (controle de usuários, de grupos, da origem do acesso por domínio ou IP);
- Mapear os endereços lógicos referenciados nas URLs (pseudo-diretórios) para os endereços físicos do disco onde estes estão armazenados (diretórios reais). Exemplo: /~osorio/... pode ser convertido para /home/osorio/public_html/
- Atualizar as informações nos arquivos de log de acessos e log de erros;
- Aceitar e responder aos comandos do protocolo HTTP do tipo GET, PUT, POST e HEAD;
- Gerenciar recursos adicionais como: os "*server side image maps*", os "*server side include documents", "local search engines"*

3 ARQUITETURA DE APLICAÇÕES WEB

Inicialmente, a computação distribuída era associada a redes de computadores que utilizavam algum sistema operacional de rede como, por exemplo, o *Netware* e caracterizavam-se fundamentalmente como servidores de arquivos. Nesse modelo há necessidade de transferir integralmente todos os arquivos, sejam programas ou dados, para execução pelo cliente, devido a ter sido projetado para atender clientes sem disco local.

Logo a seguir, a computação distribuida foi além dos servidores de arquivos e passou então a envolver serviços baseados em aplicativos cliente/servidor que caracterizam a arquitetura de aplicação *Web*. O cliente envia uma solicitação, o processamento é geralmente feito na máquina servidora remota e, nesse caso, somente os resultados são devolvidos ao cliente. Desta forma, minimiza-se o tráfego de informações pela rede.

3.1 Características das Aplicações Web

A arquitetura das aplicações *Web* apresenta as seguintes características:

- A interface com o usuário é padronizada e definida por um *browser Web*, que geralmente recebe, trata e apresenta um arquivo recebido no formato HTML;
- O servidor *Web* pode atender a diversos clientes simultaneamente;
- A comunicação é baseada no protocolo HTTP;
- Todos os serviços *Web* baseiam-se no modelo cliente/servidor;
- Pode haver distribuição de carga entre o cliente e o servidor.

3.1.1 Aplicações típicas da Web

Nem todo tipo de sistema de aplicação é adequado para o uso na *Web*. Entre estes, figuram por exemplo, sistemas que exigem tempo de resposta previamente conhecido, como os sistemas de tempo real, ou sistemas que envolvem o manuseio de quantidades muito grandes de dados.

Entre os que têm sido mais freqüentemente encontrados, estão as aplicações de Banco de Dados e as que envolvem trabalho colaborativo (*groupware*) em equipes fisicamente distribuídas.

Aplicações de Bancos de Dados na Web

Para realizar o acesso a um banco de dados via Web é necessário um *browser*, um Servidor *Web* e uma interface de acesso ao servidor de banco de dados. Quando a interface de acesso é ativada, ela recebe as informações fornecidas pelo *browser*, envia a consulta ao servidor de banco de dados e devolve o resultado em um formato específico, por exemplo HTML, ao servidor *Web* para este repassar ao *browser* que gerou a solicitação.Com o aumento de aplicações *Web* SGBD - aplicações de acesso a banco de dados via Web foram desenvolvidas diversas tecnologias para essa integração.

Uma utilização comum do ambiente cliente/servidor é em ambientes de bancos de dados que trabalham com SQL. A SQL (*Structured Query Language*) é uma linguagem declarativa que inicialmente destinava-se apenas a gerência de dados com o uso de poucos comandos simples. A medida que aplicativos SQL migraram para ambientes cliente/servidor mais exigentes tornou-se clara a necessidade de também gerenciar as funções que manipulam os dados. Para isso, surgiram os procedimentos armazenados, um procedimento armazenado corresponde a uma coleção de declarações SQL e de lógica procedimental que é compilada, verificada e armazenada em um banco de dados. Com estes procedimentos em vez de enviar vários comandos SQL separados, o cliente envia uma única mensagem que ativa um procedimento armazenado. Os procedimentos armazenados são normalmente ativados por comandos SQL, porém, existe um tipo que possui um disparador (*trigger*), que é um tipo especial de procedimento armazenado automaticamente ativado pelo banco de dados sempre que ocorre um evento específico. Estes recursos ajudam a economizar tráfego de rede, tornando mais viável sua utilização através da *Web*.

Groupware

O *groupware* inclui cinco tecnologias básicas para dar suporte ao trabalho colaborativo: gerenciamento de documentos multimídia, fluxo de trabalho, e-mail, conferências e agendamento. O *groupware* permite aos usuários coletar dados não estruturados e organizá-los como um conjunto de documentos, possibilitando a visualização, armazenamento, replicação e roteamento. [CAR 99]

3.2 Taxonomia das aplicações Web

A classificação das arquiteturas de aplicações *Web* pode ter por princípio o local em que são executados os códigos de instruções (*scripts*). A fig.1 apresenta um diagrama de taxonomia das arquiteturas Web mais comumente encontrada

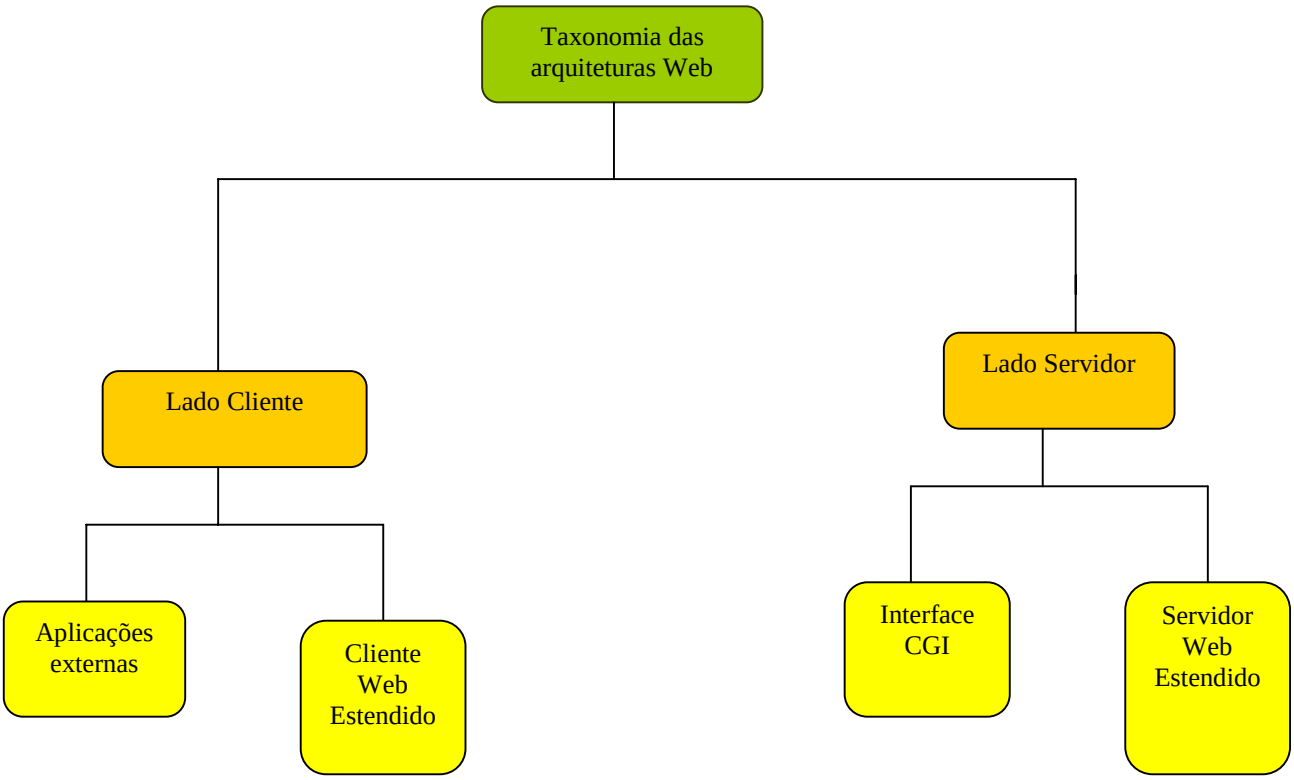

Figura 1 - Taxonomia das Arquiteturas Web

3.2.1 Lado cliente - Aplicações externas

O *browser Web* pode ser configurado para utilizar outros aplicativos para apresentação de dados que ele não seja capaz de lidar diretamente, entre elas, execução de aplicações *Web* SGBD. Ou seja, a arquitetura de integração pode consistir de uma aplicação externa ao cliente *Web* instalado na máquina do cliente *Web* para acesso ao banco de dados. Esta arquitetura pode ser implementada usando-se mecanismos como os tradicionais *plug-ins* disponibilizados para os principais *browsers* [YEA96].

3.2.2 Lado cliente - Extensões do cliente

Através da extensão das linguagens ditas de código móvel, como Java - transportadas pela rede e interpretadas pelo *browser* - é possível fazer com que estas possam suportar transações *Web* SGBD. Os *applets* Java podem receber entrada de dados de usuários na *Web*, processar a informação, e interagir com servidores de bancos de dados utilizando APIs nativas ou JDBC (*Java DataBase Connectivity*) [LIF98, CAM98].

3.2.3 Lado Servidor - CGI

CGI (*Common Gateway Interface*) é uma interface padrão que descreve regras padrão para execução de *scripts* ou programas externos ao servidor *Web*. Esta é uma das arquiteturas mais utilizadas e considerada uma das mais portáveis atualmente, visto que a maioria dos servidores *Web* implementa algum tipo de interface CGI [LIF98].

3.2.4 Lado servidor - Extensões do servidor

O servidor pode ser estendido para ele mesmo executar o código de geração de páginas dinâmicas, e então, mandá-las para o *browser*. Essa extensão apresenta-se de duas formas:

Aplicação API

API (*Application Programming Interface*) é uma extensão das funções dos servidores *Web* que possibilitam, entre outras coisas, o acesso a banco de dados [LIM 97]. Esta arquitetura pode ou não implementar uma sessão contínua com o SGBD, isto depende das funcionalidades da API utilizada.

Aplicação SSI

As SSIs (*Server Side Includes*) são trechos de códigos (marcas) especiais inseridos em um documento *Web*, por exemplo uma transação sobre um SGBD, para serem executados pelo servidor *Web* no momento em que um cliente solicitar o documento. O servidor executa esses trechos de códigos como subprocessos e em seguida o documento *Web* exibe as informações fornecidas por eles [LIF98].

3.3 Modelo Arquitetônico em Estágios (N-Tiers)

Tal arquitetura costuma ser referenciada como "arquitetura N-Tiers". Surgiu para facilitar a distribuição da carga de serviço para geração das aplicações, evitando assim que todo serviço seja concentrado em uma máquina servidora só. A maioria das aplicações estão dispostas em três estágios (*three tiers*): estágio de interação (interface) com o usuário,

representado pelo *browser* cliente, estágio onde reside a "lógica da aplicação", no servidor *Web* e estágio onde residem os dados, mantidos pelo SGBD

Segundo [REN 94], no que tange a equilíbrio de processamento, pode-se dizer que o mesmo corresponde a quantidade de trabalho realizado em cada lado cliente e servidor. A maior parte do processamento da aplicação pode ser organizada em três segmentos: interativo, aplicação e banco de dados. Em uma aplicação centralizada no servidor o cliente realiza apenas parte do processamento interativo. Se o servidor realizar apenas parte do processamento do banco de dados a aplicação é centralizada na estação de trabalho. O ponto de equilíbrio em um processamento cliente/servidor ocorre entre o segmento de processamento interativo e processamento do banco de dados.

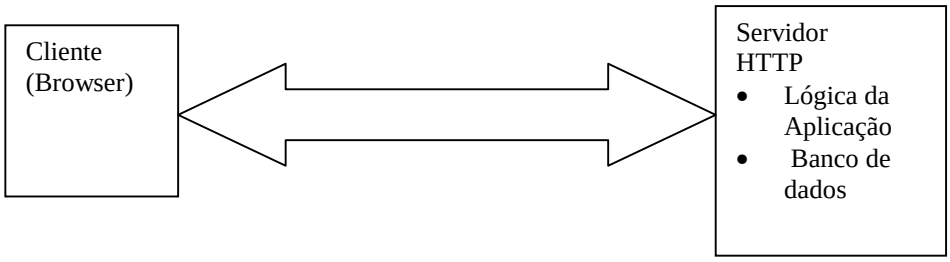

Figura 2 - Modelo N-Tiers em dois Estagios

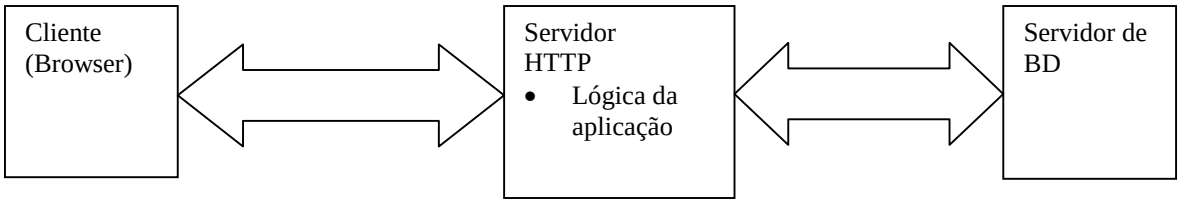

Figura 3 - Modelo N-Tiers em três estagios

3.4 Distribuição do código

Quando desenvolvendo aplicações *Web*, seja para uso na Internet ou apenas numa Intranet, é necessário um bom entendimento sobre como tirar o melhor proveito da distribuição do código, que pode estar no lado cliente (executado pelo *browser*), no lado servidor (executado pelo servidor *Web*) e ainda no próprio SGBD (Sistema Gerente de Banco de Dados). A distribuição desse código é que determina a arquitetura da aplicação *Web*.

Os códigos no lado cliente e no lado servidor têm propósitos diferentes. O código no lado servidor é usado pelo servidor *Web* para gerar o documento HTML (ou parte dele) que será enviado ao *browser*. Pode-se, por exemplo, usar código *script* no servidor para efetuar consultas a um banco de dados e formatar suas respostas para apresentação numa página HTML e então enviá-la ao usuário como se fosse uma página estática. Toda a execução do código no servidor ocorre antes de o documento HTML ser enviado ao *browser* do usuário.

Código no lado cliente, por sua vez, é usado para aumentar o grau de interação dos documentos com o usuário. O exemplo mais freqüente é o uso de código *script* (geralmente em Javascript) no documento HTML para validar campos de entrada de um formulário, fornecendo uma realimentação imediata quando o usuário comete um erro no preenchimento.

Estatísticas recentes mostram que 90% da Internet utiliza *browsers* com recursos de interpretação de *scripts*. Entretanto existem diferenças de versões e forma de referenciar os elementos que compõe o documento. Isso geralmente leva à necessidade de escrever-se o código *script* (Javascript) com capacidade de ser executado pelos dois principais *browsers*, *Netscape* e *Internet Explorer*. Mas, via de regra, pode-se adotar os princípios ilustrados na tabela:

Tabela 1 - Resumo da comparação do uso do codigo script

PARA ATINGIR	Lado Servidor Script	Lado Cliente Script
100% Intranet	SIM	SIM
80% Internet	SIM	SIM
100 Internet	SIM EXCLUSIVO	NÃO

4 LINGUAGENS DESCRITORAS DE DOCUMENTOS

4.1 Definição e origem histórica

Uma linguagem descritora é uma linguagem que utiliza o processo de marcação que corresponde a inclusão de códigos aos documentos para identificar a sua estrutura e/ou o formato no qual deve ser apresentado Essas linguagens permitem que o autor se concentre no conteúdo do que quer expressar e não na forma com irá ser apresentada ao leitor. Elas descrevem como o documento será apresentado de uma forma relativamente independente da ferramenta e da plataforma que será utilizada pelo leitor, aumentando dessa forma a portabilidade reduzindo assim os custos de conversão dos documentos [GAI 96].

Muito antes do uso intenso de computadores, a marcação era usada por editores para passar instruções para os tipógrafos sobre como determinados elementos deveriam ser preparados para impressão. No final da década de 60 *Graphic communications association*, GCA, criou o comitê *GenCode* para especificar códigos genéricos de identificação de elementos nos documentos. Pouco tempo depois, a IBM propôs a linguagem GML (*Generalized Markup Language*), criada por Charles Goldfarb, Ed Mosher e Raymond Lorie, cujas premissas básicas eram: (1) a marcação deveria descrever a estrutura do documento ao invés das suas características físicas, e (2) a marcação deveria ser legível por um programa ou ser humano A demanda por novos tipos de documentos gerou a necessidade de uma forma padronizada de escrevê-los. Em 1978 foi criado o comitê de Linguagens de Computadores para processamento de textos do *American National Standards Institute* (ANSI), compostos por membros das comunidades GML e *GenCode.* Como resultado, foi criada a meta-linguagem SGML, que se tornou um padrão recomendado pela *International Organization Standardization* (ISO) em 1986 [PIM 99]

4.1.1 SGML

A SGML (*Standard Generalized Markup Language*m) é a linguagem padrão de marcações genéricas e oferece um esquema de marcação simples, independente de plataforma e extremamente flexível, ela é simplesmente uma maneira de representar documentos ou seja é uma meta-linguagem: linguagem para definir linguagens.

A SGML possui cinco características básicas:

- Permite a criação de linguagens de marcação descritas.
- É hierárquica com elementos e componentes interligados.
- Suporta conjuntos flexíveis de etiquetas.
- Possui especificação formal completa.
- É legível tanto para humanos como para máquinas.

O SGML descreve um padrão para o uso de marcações descritivas mescladas ao documento. O SGML também fornece um método padrão para nomear as estruturas de um texto, definindo modelos hierárquicos para cada tipo de documento produzido.

SGML não se dita quais tipos de elementos devem ser criados ou como eles se relacionam, deixando essa responsabilidade com o autor da aplicação, podendo este criar quantos tipos de documentos quantos forem necessários. Por outro lado aplicações podem dar visões distintas para um mesmo tipo de documento. A estrutura e o tipo de elementos contidos em um documento SGML são formalmente definidos por uma gramática.

A aplicação que utilizar um documento SGML processa-o em conjunto com sua definição, o que permite validar o documento. Tanto os documentos como a sua definição podem ser utilizados por diversas aplicações. Um documento SGML tem conteúdo textual e pode ser editado em qualquer editor e ser lido e compreendido com facilidade pelo usuário.

Estrutura

Um documento SGML pode ser dividido em três camadas: estrutura, conteúdo e estilo. Apesar de separar os três, o SGML lida principalmente com as inter-relações entre estrutura e conteúdo num documento.

Uma aplicação SGML possui um arquivo chamado de DTD- *Document Type Definition* o qual define as regras de formatação para dada classe de documentos e é bem parecido com um banco de dados que descreve os tipos de informação que guarda e a relação entre os campos que possui. Um DTD disponibiliza uma base para os elementos (capítulos, títulos de capítulos, seções e tópicos) que constituem um documento.

Um DTD também especifica regras para a relação entre os diversos elementos, por exemplo: um título de capítulo deve ser o primeiro elemento ao se iniciar um novo capítulo; ou cada lista deve conter um mínimo de dois itens. Essas regras, definidas pelo DTD, asseguram que o documento tenha uma estrutura consistente e lógica. Um DTD acompanha um documento sempre. Um documento cujo conteúdo foi marcado de acordo com um DTD específico dá-se o nome de instância do documento. Para cada elemento do documento, a DTD especifica seu nome e o modelo de dados do seu conteúdo.

Conteúdo

O conteúdo de um documento é a própria informação contida pelo mesmo. O conteúdo inclui títulos, parágrafos, listas, tabelas, gráficos e áudio. O método de identificação da posição do conteúdo em um DTD é chamado de *tagging*.

A criação de um documento em SGML envolve a inserção de tags ao redor do conteúdo. Essas marcações indicam o início e o fim de uma determinada parte da estrutura. No exemplo a seguir, <par> indica o início de um parágrafo e </par> indica o seu final:

```
<par>
   Conteúdo é a informação por si mesma.
</par>
```

Estilo

O padrão SGML não se preocupa com a criação de estilos. Essa preocupação deu origem a diversos outros sistemas como o CALS, DSSSL ou o FOSI. Apenas, a ISO aprovou em 1996 o DSSSL como padrão a ser usado junto com o SGML.

Por ser tão poderosa, SGML é bastante complexa e o processamento de documentos pode se tornar complexo e custoso. Por esse motivo, seu uso ficou limitado a uma classe de usuarios como as grandes editoras, os departamentos de defesa e de processamento de impostos dos EUA, empresa de aviação, etc [PIM 99].

SGML, entretanto não é poderosa o suficiente para suportar a especificação de documentos hipermidia contendo relações temporais ou ligações hipertexto complexas. Esse fato levou a ISO a montar um comitê, presidido por charles Goldfarb, para desenvolver a meta-linguagem *Hytime* (*Hypermedia/Time-based Document Structuring Language)* (Vide 3.1.2 em HTML) como uma extensão de SGML.

4.1.2 Principais Linguagens

A seguir são apresentadas as principais e mais utilizadas linguagem descritoras de documentos baseados em SGML:

HTML

HTML *significa HyperText Markup Language* - Linguagem de Marcação de Hipertexto. É fruto do "casamento" dos padrões *HyTime* e SGML. Ela é simplesmente uma linguagem de marcação e serve para indicarmos formatações para textos, inserir imagens e ligações de hipertexto.

Os *browsers* são os responsáveis por identificar as marcações em HTML e apresentar os documentos conforme o que foi especificado por essas marcações.

HyTime (*Hypermedia/Time-based Document Structuring Language*)- é um padrão para representação estruturada de hipermídia e informação baseada em tempo.

O padrão *HyTime* é independente dos padrões de processamento de texto em geral. Ele fornece a base para a construção de sistemas hipertexto padronizados, consistindo de

documentos que aplicam os padrões de maneira particular. Um documento HTML é definido segundo um DTD de SGML. Os documentos em HTML são como arquivos ASCII comuns, que podem ser editados em vi, emacs, textedit, ou qualquer editor simples.

Um programa HTML possui três partes básicas, a estrutura principal, o cabeçalho e o corpo do programa. Todo programa HTML deve iniciar com o comando `<HTML>` e ser encerrado com o comando `</HTML>`. A área de cabeçalho é opcional e é delimitada pelo par de comandos `<HEAD> </HEAD>`. Esses comandos servem para criar titulos.

O corpo do programa é representado `por <BODY> e <BODY>`, é um comando obrigatório, pois a maioria dos outro comandos HTML são colocados dentro dessa área.

Alguns elementos HTML podem ter um ou mais atributos, que definem alguma característica especial..

Exemplos de elementos HTML com atributos:

```
<BODY BACKGROUND="/gifs/backgrounds/papel-de-parede.gif"> corpo do
documento </BODY>
```

define uma imagem de fundo, disposta em mosaico, para a página, 10 pixels de espessura e alinhada pela esquerda.

Os *links*, ou ligações, entre documentos ou elementos de documentos acompanham invariavelmente, a estrutura dos documentos HTML. Uma interconexão ou *link* é uma relação unária entre dois elementos de documentos. Esse dois elementos são chamados de âncoras de *links.*

A estrutura básica (mínima) de uma página HTML é a seguinte:

```
<!DOCTYPE HTML Public "-//IETF//DTD HTML//EN" -->
<HTML>
      <HEAD>
                <TITLE> Descrição do documento </TITLE>
            [elementos opcionais]
      </HEAD>
      <BODY>
            [texto e elementos HTML]
      </BODY>
</HTML>
```

A primeira linha: `<!DOCTYPE HTML Public "-//IETF//DTD HTML//EN" -->` é um rótulo (ou tag) SGML que informa ao visualizador que ele deve interpretar o documento de acordo com a definição de documento (DTD), aceitando os rótulos que ele conhece e ignorando os rótulos que ele não conhece. A maioria dos *browsers* atuais assume essa definição por default, na ausência da declaração acima.

O elemento <HTML> e </HTML> marcam o início e o final do documento HTML. No documento deverá conter duas sub-estruturas distintas: o cabeçalho, delimitado por <HEAD> e </HEAD>, e o corpo do documento, entre os rótulos <BODY> e </BODY>.

O bloco do cabeçalho, marcado por <HEAD> e </HEAD> pode conter informações sobre o conteúdo do documento utilizada para fins de indexação e organização. Não contém informação que será exibida na página.

<TITLE> é o único elemento obrigatório do bloco do cabeçalho. Deve conter o título do documento que aparece geralmente, fora da página, na barra de título do browser. O título deve conter informações que descrevam o documento [STA 97].

No documento HTML, um comentário é apresentado da seguinte forma:

```
<! -- this is a coment -- >
```

A linguagem HTML é o padrão usado por todos os navegadores da *Web*. Ela passou por uma fase de inovações que incluem: Suporte a *style sheets* usando folha de estilo, suporte a internacionalização (como por exemplo a especificação da linguagem e do conjunto de caracteres de um documento), suporte a acessibilidade, novos recursos para especificação de tabelas e formulários, suporte a linguagem de *script*, definição do elemento <OBJECT> para uso em aplicações multimídia.

Todas as inovações apresentadas acima deram origem a tecnologia DHTML que nada mais é do que uma extensão de HTML, que será apresentada a seguir [PIM 99].

DHTML

DHTML não é uma linguagem, mas sim um conjunto de tecnologias que juntas disponibilizam as ferramentas necessárias para tornar dinâmica a linguagem HTML. Essas tecnologias são:

- HTML (*HyperText Markup Language*): A conhecida linguagem baseada em *tags* para a construção de páginas *Web* estáticas.
- CSS (*Cascading Style Sheets*): Conhecida como folhas de estilo, essa tecnologia permite controlar a formatação dos diversos elementos que compõem uma página *Web*.
- CSS *Positioning:* Permite a alteração do posicionamento de um elemento da página como textos e imagens dinamicamente, através de uma linguagem *client scripting*.
- *Client scripting*: Trata-se de um pequeno programa, que será interpretado pelo *browser* do cliente e não no servidor. Algumas linguagens de *scripting* que podem ser utilizadas são JavaScript e VBScript.
- DOM (*Document Object Model*): Trata-se do modelo de objetos (com suas propriedades e métodos) que são expostos ao programador DHTML. Através do envio

de mensagens a estes objetos, o programador pode explorar a interatividade com o usuário.

DHTML apresenta as seguintes características:

- Performance: o processamento é realizado localmente, ou seja, no *browser* do usuário, o que garante boa performance já que não exige o tráfego de informações pela rede durante a interação.
- Compatibilidade: O DHTML não apresenta boa compatibilidade entre os *browsers*. Na verdade não existe um padrão para o DOM, que é o centro dessa tecnologia. Tanto a *Microsoft* como a *Netscape* já suportam esse padrão a partir das versões 4.0 de seus *browsers*, mas cada uma com seu modelo de objetos. Logo, o código *client scripting* deve ser escrito de acordo com o *browser* destino, a menos que se faça uso das propriedades protegidas, que são um subconjunto das funcionalidades comuns a ambos os *browsers.*
- Orientado a objetos: Cada elemento de uma página HTML é visto como um objeto, que pode ser acessado e ter suas propriedades, como cor e posicionamento, alteradas dinamicamente.

A tecnologia DHTML nos abre caminho para uma nova geração de aplicações *Web*. Através de DHTML é possível ter o total controle sobre os elementos que compõem uma página HTML, o que nos fornece grande flexibilidade para interagir com o usuário[RAM 99].

Os documentos HTML foram projetados para uso por aplicações tipo *browser*, restringindo à especificação da forma de apresentação de documentos Web, dificultando assim o uso de HTML por outras aplicações. Mas essa limitação é superada com o uso de SGML. Entretanto para suportar aplicações na *Web* SGML é complexa demais, enquanto que HTML é simples demais. Verifica-se assim a necessidade do uso de uma versão mais simplificada de SGML. Com esse objetivo surgiu a meta-linguagem XML como um subconjunto de SGML [PIM 99].

XML

A XML (*Extensible Markup Language*) foi desenvolvida em 1996 por um grupo de peritos que se organizaram junto com a W3C (*World Wide Web Consortium*) com o objetivo de criar uma versão simplificada de SGML que fosse apropriada para WWW.

Para elaboração dessa linguagem, começou-se com todo o padrão SGML e decidiu-se o que deixar de lado, ou seja, a SGML foi simplificada, removendo opções e recursos que não são essenciais para o bom funcionamento da XML. Muitos dos recursos SGML tiveram de ser retirados para que a XML fosse leve e pequena o suficiente para se tornar eficaz, tornando-se assim, um perfil da SGML, menos complexa.

XML é um sistema de codificação que permite que qualquer tipo de informação seja distribuído através da WWW. Ao contrário do HTML, a XML é verdadeiramente para todos os propósitos e oferece o panorama de uma ampla variedade de aplicações, cada uma servindo a uma função em particular e usando a *Web* como um mecanismo de distribuição.

A Linguagem XML é baseada em SGML, os links são baseados em *HyTime* e no mecanismo do *Extended Pointer* e os estilos são baseados em DSSSL. [PIM 99].

A XML possui uma estrutura padrão que permite criar sua própria estrutura ou usar aquelas já definidas por terceiros que melhor se encaixe com as suas necessidades, tal estrutura permite que se crie a sua própria linguagem de marcação.

A vantagem de se poder definir sua própria linguagem de marcação é que ela oferece a liberdade de capturar e publicar informações uteis sobre como seus dados e suas estruturas ficarão, ao invés de ter que se acostumar com o formato criado por outras pessoas.

A linguagem de marcação XML foi projetada com o objetivo de possuir as características de dar suporte à marcação generalizada na Web, produzir documentos que idealmente fossem válidos de acordo com o livro de regras da SGML, fornecer suporte para *hiperlinks* que fossem altamente compatíveis com a abordagem URL e fornecer um mecanismo de folha de estilo genérico e poderoso.

Os documentos XML são legíveis e claros. É possível criar ou atualizar documentos XML com qualquer editor e processador de textos simples que possa lidar com documentos ASCII.

Todo documento XML começa com uma informação de cabeçalho que descreve as regras estruturais, lista recursos externos que podem formar parte do documento, declara recursos internos que podem ser requeridos nos documentos, lista tipos de recursos que não são XML (notações) que podem ser encontrados nos documentos e lista verdadeiros recursos que não são XML (entidades de dados binários) que podem ser encontrados nos documentos.

Um documento XML se parece muito com um documento HTML em relação a sua sintaxe da marcação, como é mostrado no exemplo a seguir:

```
<Spice>
<Name>Shichuan Peppercorns</Name>
<Country0f0rigin Country = "China"/>
<Description>Pungent, distinctive. Excellent with slow cooked, earthy
dishes
</Description>
<Exemple> Shichuan Braised Chicken</Exemple>
</Spice>
```

Com XML tem-se a liberdade de usar qualquer nome que se desejar para implementar tags em seus dados, como ilustrado no exemplo acima.

Os documento XML são compostos de marcas e conteúdos. Existem sete tipos de marcações que podem ocorrer em um documento XML:

1. tags de início e término (servem para denotar os elemento)
2. designações de atributos (permitem informações associadas aos elementos)
3. referências a entidades (permitem incluir uma unidade de texto no documento XML)
4. comentários (os comentários assumem a mesma forma de comentários da HTML)
5. instruções de processamento (também conhecida como a declaração da XML serve para armazenar informações espécificas ao aplicativo
6. seções CDATA (permite isolar um bloco de texto do interpretador, pelo fato de as vezes num documento conter grandes quantidades de caracteres considerados especiais por um interpretador XML)
7. declarações de tipos de documento (permite associar o documento XML a uma DTD)

XML é muito simples e muito rigorosa a respeito da marcação. Todos os marcadores precisam estar explícitos[MCG 99].

Vantagens:

A XML apresenta vantagens de fornecer uma informação mais simples de modelar, uma maior precisão em pesquisas e documentos válidos e bem formatados

A XML foi projetada para ser utilizada em conjunto com especificações complementares, principalmente XSL, *Xlink* e *Xpointer* bem como outras já existentes, como é o caso de CSS, que permitem a especificação de documentos estruturados hierarquicamente. Entretanto XML possui a desvantagem de não tratar dos aspectos de apresentação e reutilização do conteúdo do documento, nem do suporte a estruturas hipertexto [PIM 99]. A seguir apresentaremos a linguagem WML que uma extensão de XML.

WML

WML é uma linguagem de programação baseada no XML (*Extensible Markup Language*). A especificação oficial do WML foi desenvolvida e é mantida pelo *Wap forum* um consórcio industrial fundado pela *Nokia*, *phone.com*, Motorola, e *Ericsson*. As especificações definem a sintaxe, variáveis e elementos usados no WML.

Um dispositivo de comunicação é tido como WAP, significa que este possui um software, conhecido como microbrowser, e este software tem a capacidade de entender tudo que é especificado como sendo WML.

A primeira informação dentro de um XML é conhecida como um *prolog*. Quando o *prolog* for opcional, ele consistirá em duas linhas de código: A declaração XML (que

define a versão do XML) e a declaração de documentação (um ponteiro indicando o documento DTD do WAP forum).

Inserimos então a *tag* <wml>, que inicia o documento:

```
<wml>
```

Depois do *prolog*, cada XML contém um único elemento que contém todos outros sub-elementos e entidades. Como no HTML todos elementos são delimitados pelos caracteres:

```
<> e </>
```

Ficaria então: <elemento> Minha Informação </elemento>. Pode haver somente um elemento "documento" por documento. Com o WML, o elemento "documento" é <wml>. Todos outros elementos estarão contidos nele.

O WML foi desenvolvido para uma pequena banda, e *displays* muitos pequenos. Uma parte deste desenvolvimento é o conceito utilizado como *deck/cards*. Um único documento WML é conhecido como *Deck.* Uma única interação com o usuário é conhecida como *Card.* A vantagem do projeto é que múltiplas telas podem ser baixadas para o cliente numa única requisição.

Com a habilidade de se conectar dinâmicamente a um servidor remoto, se abre aos dispositivos WAP, a possibilidade de transação de mensagens, base de dados das empresas e até mesmo o *e-commerce*. Os dispositivos WAP interagem com estas origens de dados através da *Gateway* WAP.

5 TECNOLOGIAS, FERRAMENTAS E EXEMPLOS

A Web está mudando como as pessoas se comunicam no mundo todo. Essa mídia global vem aumentando de popularidade mais rapidamente que qualquer outro meio de comunicação em toda historia. Com todo esse crescimento têm surgido novas tecnologias e ferramentes diferentes para o desenvolvimento de aplicativos na Web

Desse modo neste capitulo será abordado as tecnologias mais frequentemente usadas para geração de hiperdocumentos dinâmicos.

Um hiperdocumento dinâmico é construido no todo ou em parte, no lado servidor antes de ser enviado ao cliente. Esse processo de construção é baseado em diversas tecnologias que utilizam vários recursos para promover a interação do usuário com as informações apresentadas em uma página. Esses recursos são baseados em pequenos programas, encontrados na maioria das vezes na forma de *scripts* que são interpretados e executados por um interpretador adequado de comandos ou *shell* do sistema.

A seguir, são apresentados os recursos para geração dinâmica de hiperdocumentos tanto no lado servidor como no lado cliente:

Lado Servidor

Os recursos utilizados no lado servidor baseiam-se no acesso a bases de dados, tratamento de dados recebidos por formulários, realização de contagem de acessos, oferecimento de diversas informações personalizadas etc, e se apresentam de duas formas: A primeira é baseada na delegação de tarefas pelo servidor *Web* através do protocolo CGI, a segunda é baseada na assunção dessas tarefas pelo próprio servidor através da extensão do servidor *Web*.

Lado Cliente

O dinamismo no lado cliente ocorre através da interação do *browser* com o usuário através de programas como JavaScript e Vbscript.

5.1 Protocolo CGI

É um protocolo de comunicação (um código) através do qual o servidor HTTP (ou servidor *Web*) intermédia a transferência de informações entre um programa *(script*) que se encontra no mesmo computador que o servidor e um cliente (*browser*). Os programas (*scripts*) são então chamados de programas ou *scripts* CGI pelo fato de utilizarem o protocolo (código) CGI para receber os dados do servidor, executar e enviar o resultado de volta ao cliente.

Como funciona a CGI

O formulário HTML é um dos métodos pelo qual o usuário pode interagir com um *script* CGI. Tal *script* pode ser escrito em diversas linguagens. Para entender melhor como isso acontece, é preciso analisar como é feita a construção de um formulário HTML. Os marcadores HTML identificam e definem a estrutura e os vários componentes de um formulário. A list. 1 mostra a construção de um formulário HTML e a fig. 4 mostra o aspecto final deste exemplo.

Os marcadores <FORM> e </FORM> especificam que o conteúdo entre elas faz parte do formulário. Dentro do marcador <FORM> pode-se usar os atributos ACTION, METHOD e ENCTYPE. Apenas os dois primeiros serão abordados.

Quando o cliente faz um pedido ao servidor, este identifica a URL que está sendo informada através do atributo ACTION do marcador FORM. O atributo ACTION indica ao servidor qual ação deve ser tomada quando o formulário for submetido. Uma das ações possíveis é a chamada de um *script* CGI.

Listagem 1-Exemplo de formulario HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Exemplo de FORM</TITLE>
</HEAD>
<BODY>
  <FORM ACTION="/cgi-bin/exemplo.pl" METHOD="POST">
    Digite um texto: <INPUT TYPE="Text" NAME="texto"><p>
    <INPUT TYPE="SUBMIT" VALUE="Feito!">
  </FORM>
</BODY>
</HTML>
```

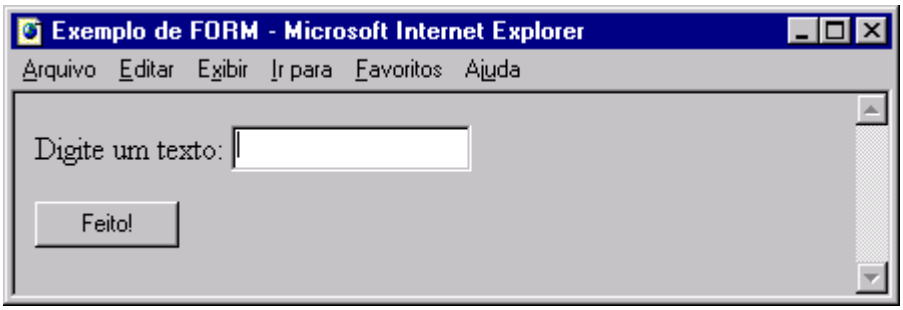

Figura 4 - Aparência do Exemplo de FORM

Os passos executados para o tratamento de uma requisição apartir de uma URL feito pelo *browser* Web ao servidor Web são mostrados na fig. 5. O cliente, ao fazer uma requisição ao servidor (passo 1), pode chamar um *script* CGI usando os métodos GET ou POST, através do atributo METHOD do marcador FORM. O modo de transmissão das informações a partir do servidor ao *script* (passo 2) depende do método utilizado.

Ao receber as informações, o *script* analisa e executa os dados enviando o resultado para o servidor (passo 3), para que este repasse ao *browser* do cliente (passo 4).

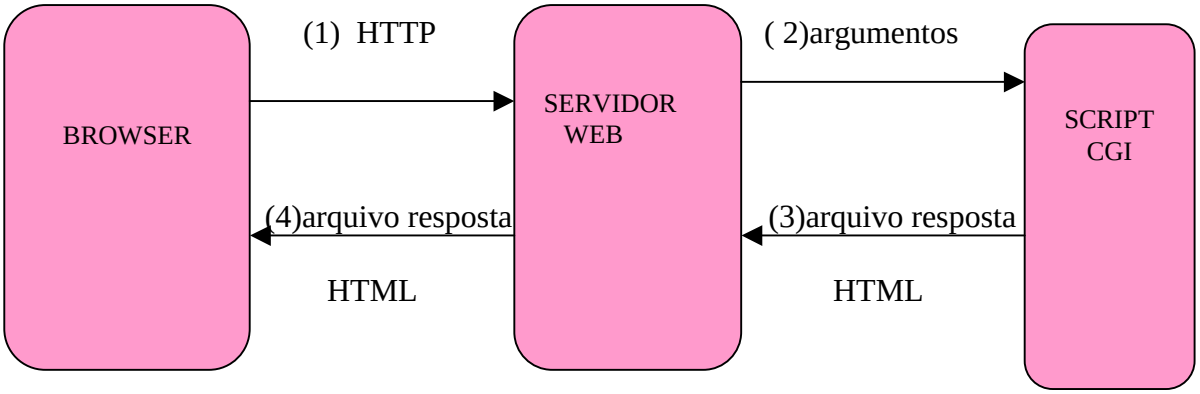

Figura 5 - Tratamento passo a passo de uma requisição do browser ao servidor

Como utilizar a CGI

Para utilizar a CGI é necessário escrever um programa (*script*) em qualquer linguagem de programação, desde que esta possa ser utilizada no servidor em que a aplicação será disponibilizada. O *script* ou programa é colocado então num diretório chamado /cgi-bin. Logo após, esse programa (*script*) é referenciado a um URL.

Por exemplo um programa chamado simples.sh instalado no diretório cgi-bin, seu URL seria:

http://meu.servidor.com/cgi-bin/simples.sh

Ao ser selecionado um URL para o programa (*script*) CGI ocorrerá a execução desse programa no sistema servidor, podendo então o programa estar pronto para fazer qualquer tarefa que o usuário desejar.

Quando um *script* CGI é executado pelo servidor, são definidas diversas variáveis de ambiente, cada uma delas contendo informações sobre o software servidor, sobre o navegador de onde veio a solicitação e sobre o próprio *script*. Desse modo os *scripts* CGI usam essas variáveis de ambiente passadas do servidor para o *script* para receber as entradas. A maneira como as informações são passadas do servidor ao *script* depende do método utilizado. Nas próximas seções são explicados os métodos e as variáveis de ambiente utilizadas para o desenvolvimento de *scripts* CGI.

Método GET

Método utilizado em pesquisas simples nas quais a *string* procurada contém menos de 255 caracteres. O servidor sabe que tudo o que segue o ponto de interrogação é a string de pesquisa. Assim, ele coloca o conteúdo a ser pesquisado dentro da variável de ambiente QUERY_STRING.

Método POST

Este método também pode usar as variáveis de ambiente, mas ele envia a maioria das informações para um *script* CGI por meio da entrada padrão (*stdin*).

Ao receber as informações do *browser* do cliente, o servidor as envia para a sua saída padrão (*stdout*), que é a entrada padrão (*stdin)* do *script* CGI. Tal *script* precisa dividir as informações em pares de chave e valor, processá-las e enviar o resultado para a sua saída, que é a entrada do servidor.

Variáveis de ambiente

Determinadas informações são armazenadas em variáveis de ambiente (v. tab.1, tab. 2 e tab. 3) pelo cliente e pelo servidor ao fazer um pedido ou chamar um *script* CGI.

Tabela 2 - Algumas variáveis de informações do servidor

Variável	Significado
SERVER_SOFTWARE	O nome e versão do servidor *Web*.
SERVER_NAME	O nome do host do servidor (ou endereço IP).
GATEWAY_INTERFACE	O nome e a versão da interface CGI.
SERVER_PROTOCOL	O nome e versão do protocolo do servidor.
SERVER_PORT	A porta TCP/IP na qual a solicitação foi recebida.

Tabela 3 - Algumas variáveis de informações do cliente

Variável	Significado
REMOTE_HOST	O nome do host do cliente.
REMOTE_ADDR	O endereço IP do cliente.
HTTP_USER_AGENT	O nome e a versão do *browser* do cliente.

Tabela 4 - Algumas variáveis de informações do script

Variável	Significado
REQUEST_METHOD	O método de solicitação do HTTP.
SCRIPT_NAME	O nome do programa CGI que o cliente chama dentro do URI.
QUERY_STRING	As informações que seguem um ponto de interrogação (?) no URL que mencionou o *script*.
CONTENT_TYPE	O tipo de conteúdo dos dados que serão retornados.
CONTENT_LENGTH	O tamanho do conteúdo que o *script* pode esperar receber de um cliente se o método POST for usado.

Para o desenvolvimento de aplicações CGI para Web são utilizadas as seguintes linguagens de programação: *Shell-Script*, Perl (*Practical Extraction and Report Language*) e C. A mais popular é a chamada linguagem interpretada Perl pois são mais fáceis de serem arrumadas, modificadas e mantidas do que as compiladas como é caso da linguagem C já que para alterar qualquer aplicação desenvolvida em C é preciso possuir o código fonte e compilar novamente a cada alteração realizada.

Serão mostradas as formas de apresentação de cada linguagem descritas acima nas subseções seguintes:

5.1.1 Shell

Scripts do *shell* do Unix são uma boa opção para programas pequenos e temporários, pois são fáceis de escrever e pode-se ver o resultado de imediato.

A seguir será apresentado dois exemplos em *shell script:*

Exemplo 1 - showEnv

Esse exemplo é mostrado pela list. 2 que apresenta um *script* em *shell* que devolve ao *browser* do usuário uma série de variáveis do ambiente CGI, precedidas pelo número de argumentos de chamada recebidos (argc) e tais argumentos (argv), na íntegra.

Listagem 2- Exemplo ShowEnv

```
#!/bin/sh

echo Content-type: text/plain
echo

echo CGI/1.0 test script report:
echo

echo argc is $#. argv is "$*".
echo

echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = "$PATH_INFO"
echo PATH_TRANSLATED = "$PATH_TRANSLATED"
echo SCRIPT_NAME = "$SCRIPT_NAME"
echo QUERY_STRING = "$QUERY_STRING"
```

```
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo AUTH_TYPE = $AUTH_TYPE
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

Exemplo 2 - fortune

Neste exemplo, apresentdo pela list. 3, a cada requisição do usuário, é devolvido um texto com alguma citação ou máxima popular, obtido randomicamente do próprio utilitário fortune.

Listagem 3- Exemplo Furtune

```
#!/bin/sh

FORTUNE=/usr/games/fortune

echo Content-type: text/plain
echo

if [ -x $FORTUNE ];  then
      $FORTUNE portugues
else
      echo Cannot find fortune command on this system.
fi
```

A ferramenta para utilização do shell script é qualquer editor de texto. O Bloco de Notas do Windows ou o Edit do DOS são adequados para essa tarefa.

5.1.2 Perl

Perl é uma linguagem desenvolvida especificamente como uma ferramenta de gerenciamento de sistemas para UNIX. Perl possui fortes capacidades de processamento de textos e uma sintaxe flexível tornando-a uma ótima escolha como linguagem para CGI.

O exemplo 1 é apresentado na list. 4 e tem como finalidade escrever na tela a frase "ola mundo".

Listagem 4- Exemplo 1 do Perl

```
#!/usr/local/bin/perl
#
#linhas de comentário
#
print 'ola mundo':# exibe o texto olá mundo
```

Exemplo 2

Este exemplo é mostrado na list. 5, é um programa contador.pl que tem como função ler um arquivo de texto chamado conta.txt que contêm um número gravado nele, em seguida atribui o conteúdo desse arquivo a uma variavel chamada $count e aumenta o valor dessa variavel em 1.

Listagem 5- Exemplo 2 do Perl

```
#programa contador.pl
open(counter,"+< conta.txt"); #abre o arquivo conta.txt
   # usando o file handler counter
   #para se referir a ele

$count=<counter>;# atribui o conteudo do arquivo
   # para a variavel &count
$count++; # soma 1 na variavel
seek(counter,0,0);# posiciona o cursor no inicio do arquivo conta.txt
print counter $count;#grava através do comando print o conteudo de
&counter
   # no arquivo conta.txt
close counter; #Fecha o arquivo conta.txt
print &count;#Imprimena na tela o valor de $cont
```

Para utilizar o perl o programador pode utilizar como ferramenta, além de um editor de texto, outros editores orientados a linguagem , com recursos de ajuda on-line e depuração.

5.1.3 Linguagem C

A linguagem C é uma linguagem de nível relativamente baixo. Ela cuida das estruturas de dados na memória sem muita proteção. O programas em C rodam muito mais rápido do que os programas escritos em uma linguagem interpretada, como o Perl.

Os programas em C podem fazer praticamente tudo que um script shell ou em perl podem fazer, comunicando-se com outros programas inclusive

Seguidamente veremos um exemplo mostrado pela list. 6, que apresenta os argumentos recebidos em um formato HTML:

Listagem 6-Exemplo em C

```
Main (int argc,char *argv[ ] ) {
   Printf ("Content-Type: text/html\n\n");
  print ("<HTML>\n<HEAD>\<TITLE>Exemlo CGI compilado</TITLE>\n");
  printf ("<BODY>\n");

  if (argc > 1) {
     printf ("<BR>Argumentos fornecidos:\n");
     while (--argc)
          printf ("<BR>%s\n" , *++argv);
} else
   printf ("<BR>Nenhum argumento fornecido.\n");
printf ("</BODY>\n</HTML>\n");
return 0;
}
```

Para fazer uso dessa linguagem o programador utiliza editores e compiladores existentes, como o compilador C ou C++.

5.2 Servidores estendidos

Servidores são programas que fornecem serviços especializados através de uma rede de computadores. Eles aceitam os pedidos vindos do cliente *(browser)*, executam, e enviam o resultado para o *browser*. Quando a execução do código é feita através do próprio servidor, o servidor passa então a ser denominado de servidor estendido.

Os servidores estendidos são definidos pela extensão do servidor *Web* que permite a disponibilização pelo próprio servidor de uma interface para o acesso ao servidor de banco de dados.

O servidor *Web* é construído de forma a permitir esse tipo de aplicação. Quando o servidor estendido atende um pedido feito por uma pagina, se encarrega de processar o código *(script)* dessa pagina e devolve a resposta em HTML para o cliente. Pode-se obter esse recurso através das seguintes formas:

Servidor Web da Microsoft (*o Internet Information Server (IIS*)) que é capaz de executar sozinho o código *script* em uma página ASP. Se o caso da execução do código *script* for em uma página PHP, o servidor dependerá de uma interface de intercâmbio entre aplicações como o API que serve para delegar tarefas ao outro programa no caso o PHP.

Servidor Apache - É o servidor mais utilizado no mundo, pelo fato do código fonte ser aberto (público), o mesmo pode ser estendido para que ele próprio execute o código *script*, como faz com PHP.

Até agora, a programação de CGI (*Common Gateway Interface*) tem sido usada normalmente para fornecer inteligência baseada em servidor dentro de aplicativos da Web. Entretanto, os programas CGI normalmente são complexos e inflexíveis. Com a possibilidade do *script* ser executado no lado servidor, um servidor poderá atender a um grande número de pedidos de usuários de forma mais rápida e usando menos memória, e também, será bem mais fácil criar conexões entre o *browser* e os dados em formatos normalmente incompatíveis com HTML, como bancos de dados.

Existem diversas formas de utilizar os servidores estendidos sendo que nesse trabalho abordaremos apenas duas formas: ASP e PHP.

5.2.1 ASP

ASP (*Active Server Pages* - Páginas de Servidor Ativas) são um ambiente para programação por *scripts* no servidor, é usado para criar páginas dinâmicas, interativas e de alta performance. Como as páginas ASP, os *scripts* rodam no servidor e não no cliente. É o próprio servidor que transforma os *scripts* em HTML padrão, fazendo com que qualquer *browser* do mercado seja capaz de acessar um *site* que usa ASP.

Uma página da Web ASP consiste em um documento HTML com lógica de *script* no lado do servidor incorporada. A lógica de *script*, é executada dinamicamente e depois retirada do documento, para que apenas o código HTML seja enviado para o navegador.

Entre os recursos que podem ser implementados via ASP, podemos citar:

- Programação em VBScript ou JavaScript
- Acesso a banco de dados
- Sessões (persistência de informações no servidor)
- ASP surgiu juntamente com o lançamento do *Internet Information Server*

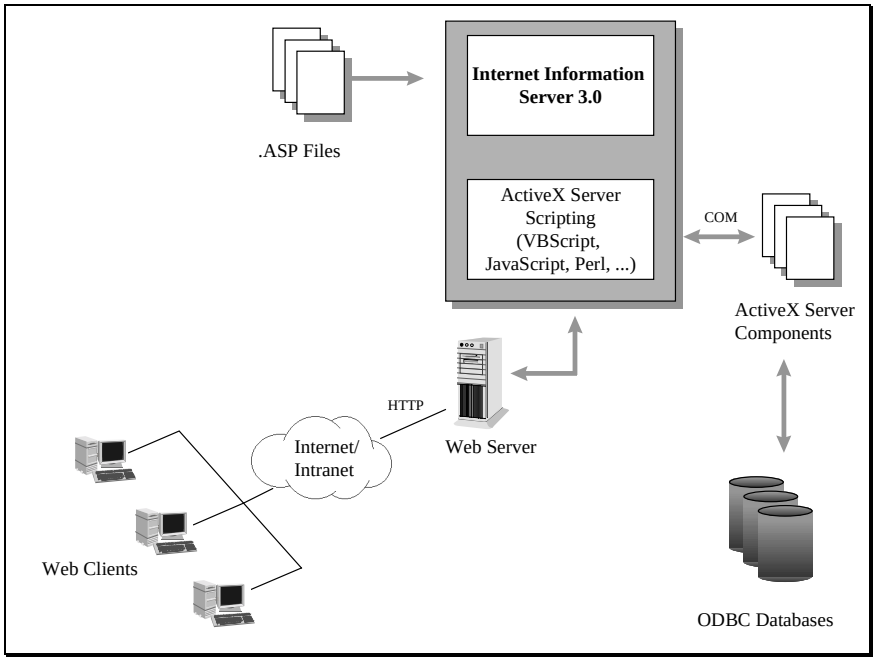

Figura 6 - Servidor IIS usando o ASP

Como uma página da *Web* ASP é um documento HTML, todas as *tags* HTML são permitidas no arquivo ASP e podem ser usadas normalmente.

A seguir é apresentado dois exemplos utilizando ASP através das figuras

```
<HTML>

<p>
<% FOR i = 3 TO 7 %>
<FONT SIZE=<%= i %>>
Hello World!<BR>
</FONT>
<% Next %>

</HTML>
```

Figura 7 - Um exemplo simples em ASP usando server-side scripting**

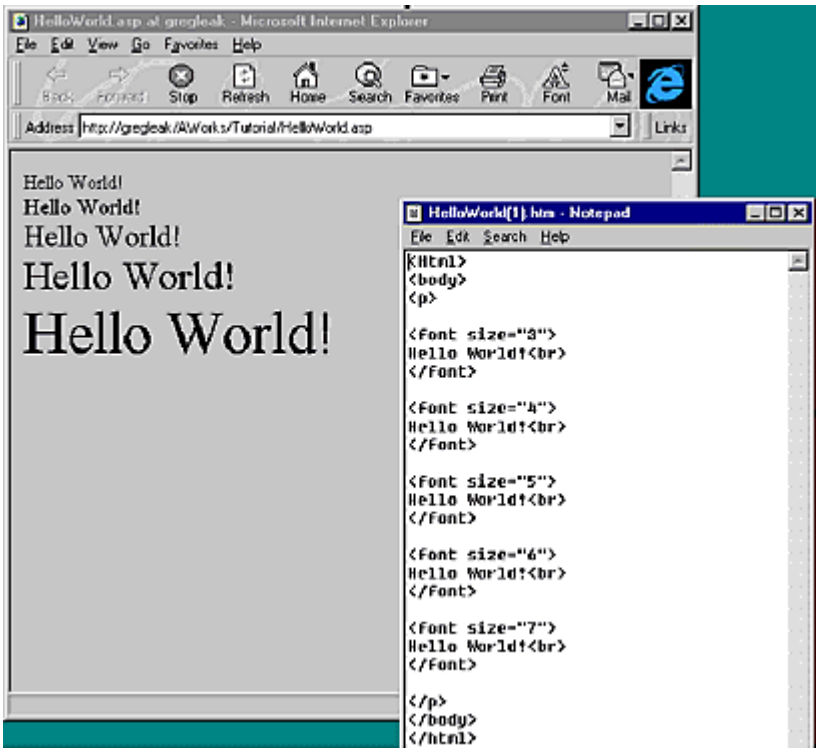

Figura 8 - Exemplo do programa Hello World em ASP

Como os arquivos ASP são arquivos do tipo texto(ASCII), eles podem ser escritos em qualquer editor de textos comum como por exemplo o *edit* ou o *notepad*. Existe também o *Ms-Visual Interdev* que proporciona um ambiente mais agradável de desenvolvimento.

5.2.2 PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. O código PHP é executado no servidor, sendo enviado para o cliente apenas HTML puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

O que diferencia PHP de um *script* CGI escrito em C ou Perl é que o código PHP fica embutido no próprio HTML, enquanto no outro caso é necessário que o *script* CGI gere todo o código HTML, ou leia de um outro arquivo.

PHP tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como *dBase, Interbase, mSQL, mySQL, Oracle, Sybase,*

PostgreSQL e vários outros. Construir uma página baseada em um banco de dados torna-se uma tarefa extremamente simples com PHP.

A seguir são mostrado dois exemplos de PHP:

Exemplo 1-Formulario

Ao clicar num botão "*Submit"* em um formulário HTML as informações dos campos serão enviadas ao servidor especificado para que possa ser produzida uma resposta. O PHP trata esses valores como variáveis, cujo nome é o nome do campo definido no formulário. List. 7 a seguir mostra isso, e mostra também como o código PHP pode ser inserido em qualquer parte do código HTML:

Listagem 7- Exemplo de tratamento de informações em um formulario em PHP

```
<html>
<head><title>Aprendendo PHP</title></head>
<body>

<?php
if ($texto != "")
  echo "Você digitou \"$texto\"<br><br>";
?>

<form method=post action="<? echo $PATH_INFO; ?>">
<input type="text" name="texto" value="" size=10>
<br>
<input type="submit" name="sub" value="Enviar!">
</form>

</body>
</html>
```

Exemplo 2

No exemplo que é apresentado pela list. 8 seguinte, criaremos um *script* com uma saída simples, que servirá para testar se a instalação foi feita corretamente:

Listagem 8- Exemplo de teste de instalação em PHP

```
<html>
<head><title>Aprendendo PHP</title></head>
<body>

<?php
echo "Primeiro Script";
?>

</body>
</html>
```

Uma das ferramentas utilizada para trabalhar com PHP é o PHPEd, para Windows.

5.3 *Scripting* no lado Cliente

Os codigos (*scripts)* que são processados no lado cliente geralmente tratam apenas de pequenas consistências de telas e validação de entrada de dados. Esses códigos podem ser escritos em JavaScript ou VBScript. Porém o Vbscript que é baseado na linguagem Visual Basic só funciona no *browser Internet Explorer*, sendo por isso caído em desuso. Nesse caso a mais indicada é a linguagem JavaScript.

5.3.1 JavaScript

JavaScript é uma linguagem de *script*, interpretada baseada em objetos desenvolvida pela *Netscape*. Com JavaScript é possível introduzir recursos de interatividade em uma página *Web*, fazer animações, imagens clicáveis independentes de CGI e muita coisa que se faz com *applets.*

JavaScript não é compilada em *bytecodes*, mas interpretada bloco-a-bloco pelo *browser* que lê a página HTML. Cada *script* em JavaScript consiste de um único módulo contínuo de código onde ficam todas as funções e variáveis utilizadas pelo programa. O código é posicionado no arquivo HTML entre descritores <script> e </script>. Como um programa JavaScript só existe dentro de um arquivo HTML, não existem aplicações JavaScript independentes de *browser*.

A seguir serão mostrados dois exemplos:

Exemplo 1

O exemplo mostrado na list. 9 usa o JavaScript chamando a função alerta.

Listagem 9- Exemplo de JavaScript chamando uma função

```
<html>
<head>
<script language="JavaScript">
<!-- Hiding function hello() { alert("Alo!"); }
// -->
</script>
 </head>
<body><a href="" onMouseOver="hello()">link</a><
/body>
</html>
```

Exemplo 2

Criar janelas é uma das grandes características do JavaScript. Pode-se construir novas janelas, carregar documentos HTML, navegar pela Internet, tudo isso através do JavaScript. Nesse exemplo será mostrado através da list. 10 como se pode abrir uma janela e escrever alguma coisa nela.

Listagem 10- Exemplo de como abrir uma janela em JavaScript

```
<html>
<head>
<script language="JavaScript">
function WinOpen() {
   msg=open("","DisplayWindow","toolbar=no,directories=no,menubar=no");
   msg.document.write("<HEAD><TITLE>E a&iacute;?</TITLE></HEAD>");
    msg.document.write("<CENTER><h1><B>Isso         &eacute;         muito
legal!</B></h1></CENTER>
");
}
</script>
</head>
<body>
<form>
<input        type="button"        name="Button1"        value="Aperte-me"
onclick="WinOpen()">
</form>
</body>
</html>
```

Para incluir o JavaScript no documento HTML utiliza-se como ferramenta qualquer editor de texto.

6 CONCLUSÕES

Este trabalho teve como finalidade oferecer uma visão sobre as principais tecnologias empregues para construção de hiperdocumentos dinâmicos.

As Aplicações Web evoluiram muito em função do desenvolvimento de varias ferramentas e tecnologias de geração de hiperdocumentos dinamicos.

Essas tecnologias estão baseadas em ferramentas para construção de hiperdocumentos com certas propriedades, metodos e eventos que são manipulados por linguagens de *scripts*, possibilitando assim uma maior interação com o usuário e uma maior vantagem no uso de aplicações em Banco de Dados.

A combinação dos *scripts* construídos do lado servidor com *scripts* do lado cliente, possibilita um maior aproveitamento dos recursos disponíveis para criar páginas dinâmicas, e no processo de criação deve-se ponderar bastante para decidir se utiliza *script* apenas em um dos lados (e qual) ou em ambos

A *Web* atrai hoje em dia a maior parte das atenções de desenvolvemento de aplicativos, por esse motivo têm surgido um número crescente de tecnologias e ferramentas, entre as quais foi abordado neste trabalho. Os recursos que cada tecnologia e ferramanta oferece variam bastante de nível desde as mais "pobres/desprovidas" como as baseadas em CGI, até as mais sofisticadas como sevidores estendidos, que atingem performances bem maiores, e as ferramentas mais "produtivas" como o Visual Interdev que é uma ferramenta "sofisticada"de desenvolvimento de aplicações Web, sendo costituída por um ambiente de programação visual, que gera o código (em VBScript e ou Javascript) automaticamente.

Por esses e outros motivos, é possível afirmar que o estudo sobre as tecnologias de hiperdocumentos dinâmicos foi bastante enriquecedor, por ter produzido uma documentação que será útil para trabalhos de pesquisa na referida área.

Glossário

Applets São programas Java designados para rodar dentro de um navegador.

Gopher Programa que faz pesquisa por palavra chave.

Groupware É uma aplicação de banco de dados que inclui cinco tecnologias básicas para dar suporte ao trabalho colaborativo: gerenciamento de documentos multimídia, fluxo de trabalho, e-mail, conferências e agendamento. Permite aos usuários coletar dados não estruturados e organizá-los como um conjunto de documentos, possibilitando a visualização, armazenamento, replicação e roteamento.

Hipertexto É um texto especial que se pode vincular a outra página na Web

Hiperdocumento É um texto que incorpora os conceitos de hipertexo

Hipermídia É uma tecnologia que permite criar em uma página um *link* que carrega um documento hipermídea, como um arquivo de vídeo ou de áudio.

Internet É uma rede de computadores que estão interligados em escala mundial (longa distância), baseada no protocolo TCP/IP (*Transfer Control Protocol / Internet protocol*).

Intranet É uma rede privada corporativa ou educacinal que usa os protocolos TCP/IP da Internet para seu transporte fundamental.

link Elo de ligação entre referências.

Middleware Camada de software intermediária que faz a interface entre outras duas camadas de software

Plug-in Tecnologia que permite a reprodução de arquivos de áudio e vídeo a medida que a pagina Web é exibida no navegador.

Script Trecho de código interpretado por um interpretador de comando.

Telnet Emulador de terminal.

7 Referências Bibliográficas

[BUSH 45] VANNEVAR, Bush. **As We May Think**. Atlantic Monthly. july 1995 Disponível por WWW em http://www.w3.org/History/1945/vbush/ (05/09/2000).

[CAM 98] CAMPOS, M. L.; RESENDE, L. M e LESER, S. M. **Laboratório de Banco de Dados** - Parte de um Projeto Final de Curso da UFRJ, 1998. Disponível por WWW em http://www.lci.ufrj.br/~labbd/principa.htm (10/09/2000).

[CAR 99] CARDOSO, Adriane de Q. **Avaliação Comparativa entre Formas de Acesso a Banco de Dados Via Web**. Departamento de Informática da UFPEL, 1998 (monografia de conclusão do curso)

[INF 96] JOSE A. Ramalho. A Caminho da Linguagem Distribuída. **Informática Exame**. Junho de 1996 p. 94-98.

[GRA 96] GRALLA, Preston. **Como Funcionam as Intanets**. Editora Quark, São Paulo, 1996

[LIF98] LIFSCHITZ, S. & Lima, I**. Arquiteturas de Integração Web SGBD: um Estudo do Ponto de Vista de Sistemas de Banco de Dados**. Departamento de Informatica da Pontífica Universidade Cátolica do Rio de Janeiro, 1998.

[LIM 97] LIMA, Iremar N. O. **Ambiente Web Banco de Dados: Funcionalidades e Arquiteturas de Integração**. Departamento de Informática da Pontifica Universidade Católica do Rio de Janeiro, 1997, 81 p (Dissertação, Mestrado em Ciência da Computação).

[MCG 99] MCGRATH, Sean. **XML: Aplicações Práticas**. Editora Campus, Rio de Janeiro, 1999

[GAI 1996] GAITHER, Mark; Hassing, S. & Herwin, M. **World Wide Web Com HTML & CGI: biblia do programador.** Editora Berkeley Brasil, São Paulo, 1996

[OLI 2000] OLIVEIRA, Adelize. **Desenvolvendo Sites Wap**. Editora Advance Books Florianopolis-SC, 2000

[PIM] PIMENTEL, Maria da Graça. **Hiperdocumentos Estruturados na WWW: Teoria e Prática.** Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação, p

[RAM 99] RAMALHO, José Antônio Alves. **HTML Dinâmico**. Editora Berkeley Brasil, São Paulo, 1999

[REN 94] RENAUD, Paul E. **Introdução aos Sistemas Cliente/Servidor**. Editora Infobook, Rio de Janeiro, 1994

[STA 97] STAUFFER, Todd. **Dominando o Essencial**: HTML 3.2. Editora Campus, Rio de Janeiro,1997

[TAN 97] TANENBAUM, Andrew. **Sistemas de Redes de Computadores**. Editora Campus, Rio de Janeiro 1997

[TEI 99] TEIXEIRA, J. et Alli. **Redes de Computadores**: **Serviço de Administraçào e Segurança.** Editora Makron Books, são Paulo, 1999

[WEI 97] WEINMAN, W. **Manual de CGI**. Editora Makron Books, São Paulo, 1997.

[YEA96] YEAGER, N.J. & McGrath, R.E.. Web Server Technology - **The advanced guide for World Wide Web Information Providers**, Morgan Kaufmann Publishers, Inc., 1996.

UNIVERSIDADE FEDERAL DE PELOTAS
Instituto de Física e Matemática
Curso de Informática

Tecnologia de Hiperdocumentos Dinâmicos

por

Isabel Florinda Bernardo de Kerlan

Dissertação apresentada aos Senhores:

_____
Prof. M. Rosane da Silveira

_____
Prof. Marcia Baltar

Vista e permitida a impressão.
Pelotas, ___/___/___.

_____
Prof. Mauricio Porto
Orientador.