UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE CIÊNCIA

DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Trabalho de Licenciatura

Tema

Active Server Pages
Sua aplicação ao Sistema de Gestão de Fichas de Obra

Carla Denise Sigava Abreu de Jesus Xavier

Julho/ 2002

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE CIÊNCIA

DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Trabalho de Licenciatura

Tema

Active Server Pages
Sua aplicação ao Sistema de Gestão da Ficha de Obra

Elaborado por: *Carla Denise Sigava Abreu de Jesus Xavier*

Supervisores: *Eng. Venâncio Massingue e Dra. Marisa Balas*

Julho/ 2002

Dedicatória

Em memória da minha avó, Adelaide Chongo,

Aos meus pais e irmãs.

Carla

Agradecimentos

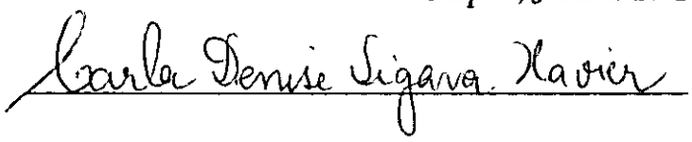
Os meus agradecimentos vão para as pessoas que me prestaram apoio durante todo o meu percurso estudantil. Dentre essas pessoas não posso deixar de mencionar as seguintes:

- *Os professores da escola primária e secundária que me transmitiram os conhecimentos básicos;*
- *Os professores universitários que complementaram os conhecimentos anteriormente adquiridos;*
- *Aos meus supervisores, Dra Marisa Balas e Eng Venâncio Massingue, pelo esforço despendido e sabedoria transmitida, durante o trabalho;*
- *Aos meus pais e irmãs, que sempre me apoiaram e acarinharam;*
- *À direcção e pessoal da Exi, pelo apoio prestado durante o trabalho;*
- *A meus amigos e colegas, pelos momentos difíceis e alegres partilhados;*
- *Aos funcionários do Departamento de Matemática e Informática, pelo excelente trabalho que tem feito.*

Declaração de Honra

"Declaro que este trabalho é resultado das minhas próprias investigações e que o mesmo foi realizado apenas para ser submetido como Trabalho de Licenciatura em Informática na Universidade Eduardo Mondlane"

Maputo, Julho de 2002



Carla Denise Sigava Abreu de Jesus Xavier

Resumo

A crescente concorrência com que as empresas se deparam tem feito com que os gestores de negócios pensem em modelos de suporte à tomada de decisão baseados em novas tecnologias, com objectivo de melhorar a prestação dos seus serviços.

A Internet oferece uma grande capacidade de distribuição da informação. Para as empresas, isto significa que alguns serviços podem tornar-se disponíveis, independentemente da localização geográfica, facilitando deste modo, a relação entre os clientes e as empresas.

Neste trabalho, é feita uma definição sobre a Internet e os conceitos a ela relacionados, desde as páginas estáticas às tecnologias utilizadas para o desenvolvimento de páginas dinâmicas.

Das tecnologias definidas, seleccionou-se uma, efectuou-se o seu estudo e posterior aplicação ao *Sistema de Gestão de Fichas de Obra,* da Exi, Empresa de Engenharia e Comercialização de Sistemas Informáticos. A aplicação do modelo, na Exi, tem como objectivo criar uma maior aproximação entre o cliente e a empresa, melhorando deste modo a prestação dos seus serviços.

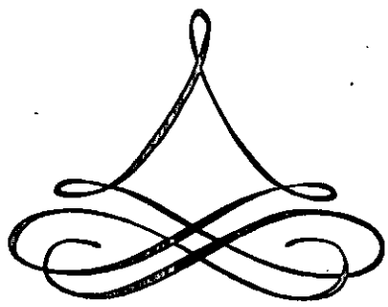
Sendo os sistemas baseados em redes, volúveis à ameaças, foi feita uma investigação sobre os mecanismos usados para garantir a segurança na transmissão da informação pela Internet.

Índice

Índice de figuras

Capítulo I

Introdução



1. Introdução

As empresas executam um determinado negócio, e o mundo dos negócios tem como objectivo o retorno do capital investido[Silva,1999]. Num mercado, onde a concorrência é crescente, para que uma empresa esteja apta a competir, é necessário que as suas decisões estejam bem direccionadas através de informações sobre o mercado em que esta deseja actuar. Face à enorme quantidade de informação que é diariamente produzida, a informática presta uma grande contribuição na selecção e organização de dados de modo a transformá-los em informação útil.

Em Moçambique, com a evolução das tecnologias de informação e o aumento do uso de computadores conectados em rede, as empresas vão procurando soluções baseadas em sistemas de informação automatizados para controlar os seus processos mais importantes, cujo objectivo principal é a obtenção de lucros.

O desafio que se coloca às empresas fornecedoras de tecnologias de informação e comunicação, não é apenas fornecer e implementar soluções adequadas, como também prestar suporte técnico aos seus clientes. As actividades de suporte técnico passam pelo acompanhamento da evolução dos sistemas informáticos, e da resolução dos problemas apresentados pelos clientes.

Para a empresa de Engenharia e Comercialização de Sistemas Informáticos, EXI, a criação de um sistema on-line de atendimento ao cliente baseado na tecnologia *Web*[1] contribuirá para a melhoria dos níveis de serviço prestados pelos seus departamentos aos clientes, diminuindo desta forma o tempo de resposta, permitindo o controlo do cumprimento dos planos de manutenção assinados entre a companhia e os seus clientes, e uma maior informação sobre o andamento dos serviços solicitados pelos clientes.

[1] A principal página do site *www* e suas páginas associadas, imagens, documentos, multimedia, e outros artigos armazenados em um servidor *www* ou no disco duro de um computador.

1.1 Definição do problema

Nos dias de hoje, a concorrência e a evolução da Internet tem feito com que a preocupação das empresas se torne cada vez mais concentrada no cliente e na gestão das suas relações.

A tecnologia fornece, actualmente, novas dimensões na gestão das relações com o cliente. No entanto, as empresas enfrentam vários desafios na execução da sua estratégia de se tornarem mais orientadas ao cliente. Elas necessitam de soluções que integram o *front-end* com o *back-office,* abertas para a Internet.

A Exi, Empresa de Engenharia e Comercialização de Sistemas Informáticos, tem em funcionamento um sistema de atendimento ao cliente: o Sistema de Gestão de Fichas de Obra. Este sistema foi desenvolvido em *Visual Basic* e tem como objectivo aumentar o nível de serviços prestados pelos seus departamentos aos seus clientes, solucionando deste modo alguns inconvenientes, tais como, demora excessiva na resposta aos clientes, falta de controlo do cumprimento dos planos de manutenção assinados entre a companhia e os seus clientes, etc.

Um dos módulos que a Exi deseja desenvolver, e que permitiria colocar à disposição do cliente informação do seu interesse em tempo útil, é o interface para à Internet. Este objectivo pode ser alcançado através da criação de uma aplicação *Web*, que permitirá que o cliente aceda à informação a partir de qualquer ponto do mundo.

1.2. Objectivos

1.2.1 Objectivo Geral

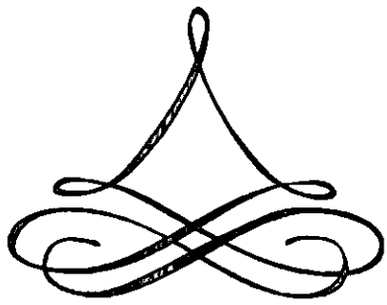
- Desenvolver um modelo de suporte ao cliente, baseado numa tecnologia de desenvolvimento de páginas dinâmicas para Internet, interligado ao Sistema de Gestão de Fichas de Obra, da Exi.

1.2.2 Objectivos Específicos

- Estudar a tecnologia de desenvolvimento de páginas dinâmicas para Internet;
- Avaliar o sistema de informação existente (Sistema de Fichas de Obra), de modo a seleccionar o subsistema a ser modelado;
- Propor o modelo baseado na tecnologia de desenvolvimento de páginas dinâmicas para Internet, a ser implementado;
- Testar o modelo.

Capítulo II

Metodologia



2. Metodologia

Para alcançar os objectivos propostos foram realizadas as seguintes actividades:

- Uso da metodologia descritiva, de modo a permitir uma abordagem detalhada dos elementos que serviram de base teórica, e da metodologia analítica, que consistiu na análise do sistema actual, onde foi implementado o modelo.

- Recolha de dados que consistiu na:

  - Pesquisa e consulta bibliográfica como forma de identificar a melhor bibliografia a ser usada;

  - Utilização de informação disponível;

  - Realização de entrevistas não estruturadas.

- Estudo da tecnologia de desenvolvimento de páginas dinâmicas para Internet;

  - Definição de conceitos relacionados à Internet;

- Analise do caso de estudo (Sistema de Gestão de Fichas de Obra), onde o modelo será implementado;

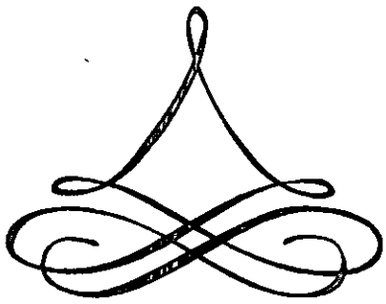
  - Construção do modelo;

  - Documentação do modelo durante a execução do trabalho;

- Implementação do modelo.

Capítulo III

Internet e alguns conceitos



3. Internet e alguns Conceitos

Neste capítulo, é feita uma descrição sumária sobre os conceitos ligados à Internet.

3.1. Internet

A Internet teve origem na rede *Advanced Research Projects Network* (*Arpanet*), criada pelo Departamento de Defesa dos Estados Unidos no inicio dos anos 70. Esta rede interligava vários centros militares e de pesquisa com objectivos de defesa na época da Guerra Fria [Branski,1998].

Nos meados dos anos 80, a *National Science Foundation (NSF),* utilizando a tecnologia desenvolvida pela *Arpanet*, criou uma rede de alta velocidade para permitir que centros de pesquisa e universidades tivessem acesso aos seus supercomputadores. Esta interconexão entre diferentes redes de computadores veio a tornar-se Internet [Branski,1998].

Segundo Almeida (1997), a Internet pode ser definida como uma colecção de milhares de computadores interligados, utilizados por milhões de utilizadores que compartilham um meio comum, permitindo a interacção entre eles para a troca de informações digitalizadas.

Baseado no protocolo *TCP/IP*[2] a Internet disponibiliza ao utilizador inúmeros serviços tais como: *e-mail*(*correio electrónico*), *ftp*[3] (*file transfer protocol*), *Telnet* (*Terminal Remoto),* etc. Contudo, a Internet só se tornou verdadeiramente popular depois do aparecimento da *WWW* (*World Wide Web*) [Lúcio,1999].

3.1.1 World Wide Web (WWW)

Criada em 1989 pelo *European Laboratory*, a *WWW* é uma colecção de sistemas de computadores, que apresenta a informação em formatos multimédia, tais como: gráficos, som, animação e vídeo [Lúcio,1999].

Neste meio, a linguagem utilizada é simples. Pois, na *WWW* começa-se por dar a cada documento um endereço e um nome, que indica onde está situado (em que servidor), de modo a que se possa ter acesso a ele. A esse endereço chama-se *URL* (*Uniform Resource Locator*).

[2] Veja Anexo A para mais detalhes sobre o protocolo TCP/IP
[3] Protocolo para a transferência de ficheiros

O protocolo que permite que, ao escrever um *URL* no *browser*[4]do utilizador, o pedido seja atendido pelo servidor, é o *HTTP*[5] (*Hypertext Transport Protocol*). Após receber o pedido, o servidor responde, enviando um documento em hipertexto, composto por códigos que definem a sua formatação e as ligações a outros documentos, sendo este formato denominado *HTML (Hypertext Markup Language)*. Este documento, em html, é então exibido ao utilizador, sob a forma gráfica. A linguagem *HTML* permite criar páginas simples, objectos variados e *links*[6] para outras páginas. No entanto, estas páginas não tem capacidade de se auto-modificarem ou interagirem com o utilizador ou com outras aplicações. Isto levanta sérios problemas quando se pretende disponibilizar na Internet aplicações que fornecem informações de modo dinâmico em *HTML*.

As primeiras aplicações que disponibilizavam informações de modo dinâmico em *HTML*, utilizavam uma tecnologia chamada *Common Gateway Interface (CGI)*. Contudo, os servidores baseados em *CGIs* eram lentos e pouco seguros. Esta tecnologia evoluiu dando lugar às Linguagens *Scripts* [Lúcio,1999].

3. 2. Linguagens Scripts

Linguagens *Scripts* são usadas para escrever instruções que permitem criar páginas dinâmicas[7]. Existem várias linguagens *Scripts* já bastante evoluídas, sendo as mais conhecidas o *Visual Basic Script* e o *JavaScript* [Lúcio,1999].

As linguagens *Scripts* situam-se entre o *HTML* e as linguagens convencionais. O *HTML* tem como função mostrar e interligar texto. As linguagens convencionais utilizam os recursos do computador de forma a realizar operações de manipulação de informação[Lúcio,1999].

3.2.1. Scrips no Servidor e no Cliente

Scripts são pedaços de código de uma linguagem *script*, integrados directamente no código *HTML* da página, e que podem ser executados pela própria máquina ou como resposta a algum evento desencadeado pelo utilizador. Os *scripts* podem ser executados pelo cliente ou pelo servidor de uma aplicação em rede, conforme seja mais útil para essa aplicação.

Antes de se entrar em pormenores sobre os *scripts* no servidor e no cliente, urge a necessidade de definir em breves palavras os conceitos cliente e servidor.

---

[4] Software que interpreta os ficheiros da WWW, os formata em páginas da Internet, e os exibe ao utilizador.
[5] Para mais detalhes sobre o protocolo de comunicação entre o cliente e o servidor, veja anexo B
[6] Conexão entre dois documentos.

Servidores são máquinas ou aplicações que providenciam um determinado serviço a outras máquinas ou aplicações chamadas *Clientes*.

As aplicações Internet possuem sempre dois lados: o do cliente e o do servidor. Ao cliente é-lhe apresentada uma página onde ele pode realizar pedidos ao servidor. O servidor é quem efectua o processamento final e envia os resultados desse processamento ao cliente.

3.2.1.1 Scripts no Servidor

Um *script* que é interpretado pelo servidor é denominado *Server Side Script*. Um *Server Side Script* é uma instrução processada pelo servidor e que gera *HTML*. O *HTML* resultante é enviado, como parte da resposta *HTTP*, para o *browser*. Como exemplo de tecnologias *Server Side* temos: *Active Server Pages, Common Gateway Interface, ColdFusion, Java Server Pages* e *Personal Home Pages*. A *figura 3.1* ilustra o processamento do *Server Side Script* (Busel et al, 2000) .

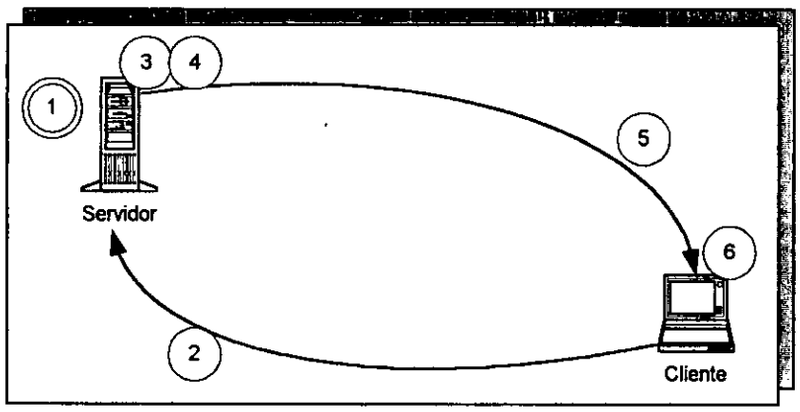


Figura 3.1 - Funcionamento do Server Side Script

Legenda da figura 3.1

1. Autor escreve instruções, que ficam armazenadas no servidor;
2. Cliente requisita a página;
3. Servidor localiza ficheiro de instruções;
4. Servidor processa instruções para criar *HTML*;

[7] Para mais detalhes sobre a diferença entre páginas dinâmica e estáticas, veja anexo C

5. Segmento *HTML* é enviado ao browser;

6. Browser processa *HTML* e edita a pagina.

3.2.1.2 Scripts no Cliente

Um *script* que é interpretado pelo cliente é denominado *Client Side Script*. Um *Client Side Script* é também um conjunto de instruções, mas não é processado pelo servidor. Em vez disso, é enviado ao *browser*, onde é processado. O resultado é depois emitido pelo *browser* no monitor. Como exemplo de tecnologia *Client Side* temos: *Java, ActiveX Controls e Dynamic HTML*. A *figura.3.2* ilustra o funcionamento da tecnologia *Client Side*.

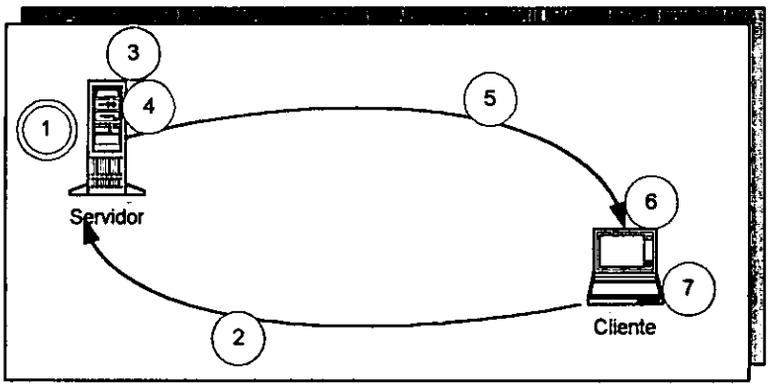


Figura 3.2 - Funcionamento do Client Side Script

Legenda da figura 3.2

1. Autor escreve o código da página;

2. Cliente requisita página;

3. Servidor localiza ficheiro da página;

4. Servidor solicita o *script engine*[8] para processar o *script* e gerar *HTML*;

5. Segmento *HTML* é enviado ao *browser*;

6. *Browser* processa o *Client Side Script* ;

7. *Browser* processa *HTML* e edita a pagina.

[8] Interpretador de script

3.2.1.3 Scripts no Servidor Versus *Scripts no Cliente*

A maior vantagem do *Client Side Script* em relação ao *Server Side Script,* é o tempo de resposta, pois o facto do *script* ser interpretado no *browser* diminui o tempo de resposta.

No entanto, a maior desvantagem do *Client Side Script*, é a dependência que este possui em relação às funcionalidades do *browser* no suporte de um determinado *script*. Por exemplo, se se pretende visualizar uma página que contém o *Client Side Script*, em dois *browsers* diferentes, o resultado da visualização poderá ser diferente, isto porque alguns *browsers* não possuem capacidades para interpretar certas linguagens *scripts*.

Outra desvantagem do *Client Side Script* é a visibilidade do código fonte, para o utilizador, não fornecendo, segurança da informação.

Pelas vantagens aqui apresentadas, no modelo a desenvolver será usada uma tecnologia *Server Side*.

3.2.2. Tecnologias Server Side[9]

3.2.2.1 Common Gateway Interface (CGI)

A principal vantagem ao escrever aplicações *Web* através da interface *CGI* é a portabilidade da aplicação. Apesar disso, 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 utilizadores. Quando chegam novos pedidos ao servidor enquanto uma aplicação *CGI* ainda está a ser executada, este inicia um novo processo da aplicação *CGI*. Quanto mais pedidos forem chegando, mais processos concorrentes serão criados na plataforma do servidor. O facto de se criar um processo para cada pedido consome tempo e grande parte de recursos de memória, deteriorando, deste modo, o desempenho da aplicação.

[9] Este ponto é fruto de investigação efectuada por vários sites da Internet.

3.2.2.2 Active Server Page (ASP)

ASP é uma tecnologia da Microsoft usada para a construção de páginas dinâmicas. O código ASP é processado no momento em que é feito o pedido por um *software* denominado *Servido Web*[10]. Este processamento gera HTML que é depois enviado ao *browser* e usado para criar a página.

A tecnologia *ASP* oferece todos os recursos de aplicações *CGI*. O *ASP* é mais robusto por não criar um processo no servidor para cada pedido do utilizador, como acontece com *CGI*. Um servidor que usa *ASP* ao invés do *CGI* pode atender maior número de pedidos de utilizadores de forma mais rápida e usando menos memória[Albino, 2000].

A tecnologia ASP possui uma característica particular: o seu código (que se encontra entre os delimitadores <%%>), deve ser processado num computador que possui o sistema operativo Windows, porém as páginas geradas podem ser visualizadas a partir de qualquer computador e através de qualquer *browser*.

ASP pode ser usado para personalizar páginas de acordo com as necessidades específicas do utilizador, isto significa que o texto, imagens, tabelas, etc, da página podem ser automaticamente seleccionados no momento em que o utilizador requisita a página. A tecnologia ASP utiliza como linguagem *script* o *VBScript* e o *Jscript*.

3.2.2.3 ColdFusion

Desenvolvido pela *Allaire*, o *ColdFusion* deve o seu sucesso à facilidade com que os dados podem ser extraídos de bases de dados para a Internet. Não se pode dizer que o *ColdFusion* é uma linguagem de programação; mais do que uma linguagem, é uma plataforma de desenvolvimento. Assim como o *HTML*, utiliza elementos *CFML (ColdFusion Markup Language)*.

---

[10] É um software, conectado à *World Wide Web*, que fornece, ao browser, recursos pedidos pelos utilizadores. O recurso pedido normalmente é identificado por um *URL*.

Para que se possa usufruir destes recursos, é preciso ter o *ColdFusion Server* instalado na mesma máquina do Servidor Web. O *ColdFusion* funciona em servidores *Windows NT, Linux, HP-UX (Unix da HP)* e *Solaris (Unix da SUN)*.

Quando o utilizador acede uma determinada página de um *site*[11], o seu *browser* faz o pedido dessa página ao servidor onde ela está hospedada, e o servidor verifica a extensão da página pedida. Caso seja um arquivo com extensão *.HTML*, ele procede normalmente, devolvendo a página ao utilizador. Caso a extensão seja *.CFM* ou *.CFML*, que são as extensões do *Cold Fusion*, esse servidor envia o pedido ao servidor *Cold Fusion*, que processa a página pedida.

A maior diferença entre o *ColdFusion* e o *ASP* é que as soluções *ASP* são construídas usando *VBScript (ou JScript)* e objectos, o *ColdFusion* utiliza *tags*[12] que encapsulam funcionalidades. Apesar do *ColdFusion* não possuir objectos internos, ele contém um conjunto de soluções para os problemas comuns, incluindo o acesso às funcionalidades *ADO (Active Data Object)*.

3.2.2.4 Java Server Pages (JSP)

É uma tecnologia que permite combinar *HTML* e *Java* para dinamicamente gerar páginas. Assim como as páginas *ASP*, as páginas *Java Server Pages (JSP)* contêm pequenas porções de código (*scripts*) junto ao código *HTML* que são processadas pelo servidor e depois disto enviadas para o cliente.

3.2.2.5 Personal Home Page (PHP)

PHP (Personal Home Page) é uma linguagem *script*, usada para a criação de páginas dinâmicas. É executada em servidores *Windows* e em todas as variantes do *UNIX*. A sintaxe da *PHP* é uma mistura de *Perl, C*, e *Java*, o que a torna mais fácil para quem já utilizou uma destas linguagens, proporcionando o rápido desenvolvimento de páginas dinâmicas. De forma parecida com *ASP*, os trechos de programação ficam entre os delimitadores <? e ?>.

[11] É um endereço cuja porta de entrada é sempre á sua página inicial.
[12] são etiquetas, cercadas pelos parênteses de ângulo (< >), usadas na construção de páginas html para formatar instruções embutidas no texto

Basicamente, em termos de funcionalidades, *PHP* é similar ao *CGI*, pois permite colher dados de um formulário, gerar páginas de conteúdo dinâmico, enviar e receber *cookies*[13].

PHP suporta uma grande variedade de bases de dados e possui acesso nativo a *Adabas, dBase, Informix, Interbase, mSQL, MySQL, Oracle, Sybase, SQL Server, Unix dbm*, além de suportar *ODBC (Open Database Connectivity)*, fazendo com que o *PHP* possa trabalhar praticamente com todos os Sistemas de Gestão de Base de Dados existentes.

3.2.2.6 Tecnologia usada

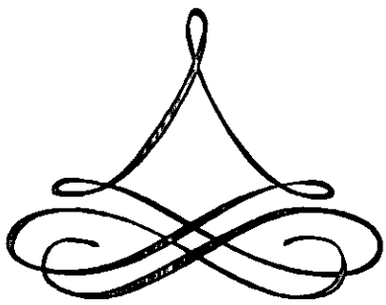
Nos últimos anos, o domínio do sistema operativo Windows conduziu muitas organizações/instituições para um compromisso com as tecnologia da Microsoft e a plataforma Windows. Uma estratégia comum para tais empreendimentos foi o desenvolvimento de *front-ends* em *Visual Basic* para as suas aplicações empresariais[Reiff,2001].

O Sistema de Gestão de Fichas de Obra não foge à regra. Por essa razão, das tecnologias aqui mencionadas, a tecnologia ideal seria *Active Server Pages* (*ASP*). Pois, possui a vantagem de utilizar uma linguagem *script* similar a linguagem de programação usada na construção do sistema actual, o *VBScript*. Este aspecto irá reduzir o tempo de construção do modelo e os custos de formação dos técnicos. Irá também facilitar a manutenção numa fase posterior.

[13] Pequenos pedaços de dados armazenados num arquivo do cliente e controlados por um browser.

Capítulo IV

Active Server Pages



4. Active Server Pages (ASP)

4.1 Definição

ASP é uma tecnologia da Microsoft que disponibiliza um conjunto de componentes para o desenvolvimento de páginas dinâmicas, interactivas e de alta performance. Nas páginas ASP, os *scripts* são processados no servidor e não no cliente, sendo, deste modo, um *Server Side Script*. É o próprio servidor que transforma os *scripts* em *HTML* padrão, fazendo com que qualquer *browser* do mercado seja capaz de aceder um site que usa *ASP*.

Estas páginas consistem em arquivos de extensão **.asp* no formato texto que contém combinações de *scripts* e *tags HTML* [Santos,2000].

Para que um *script* seja identificado como *ASP*, é necessário que use:

- os delimitadores <% %> ou;
- o *tag* ***<script>*** do *HTML*, especificando o atributo "*RUNAT= Server*", dentro dele.

O ASP surgiu com o lançamento do *Internet Information Server 3.0* (IIS 3.0). Esta é uma solução *Microsoft*, que exige que o servidor web esteja instalado num computador que possui o sistema operativo da *Microsoft (Windows)*.

O *ASP* possui uma série de funcionalidades tais como, acesso a bases de dados, persistência de informação no servidor através de sessões, programação em *Visual BasicScript* e *JScript*. Estas funcionalidades são alcançadas, através de um conjunto de objectos internos e de componentes que o *ASP* possui.

4.2. Modelo de Objectos ASP

O ASP providencia um conjunto de objectos[14] relacionados, denominado *Modelo de objectos* que gere a interacção entre o servidor e o *browser*. A *figura. 4.1* ilustra o relacionamento entre os objectos, e o seu papel na interacção entre o cliente e o servidor .

[14] Para maiores detalhes veja anexo D

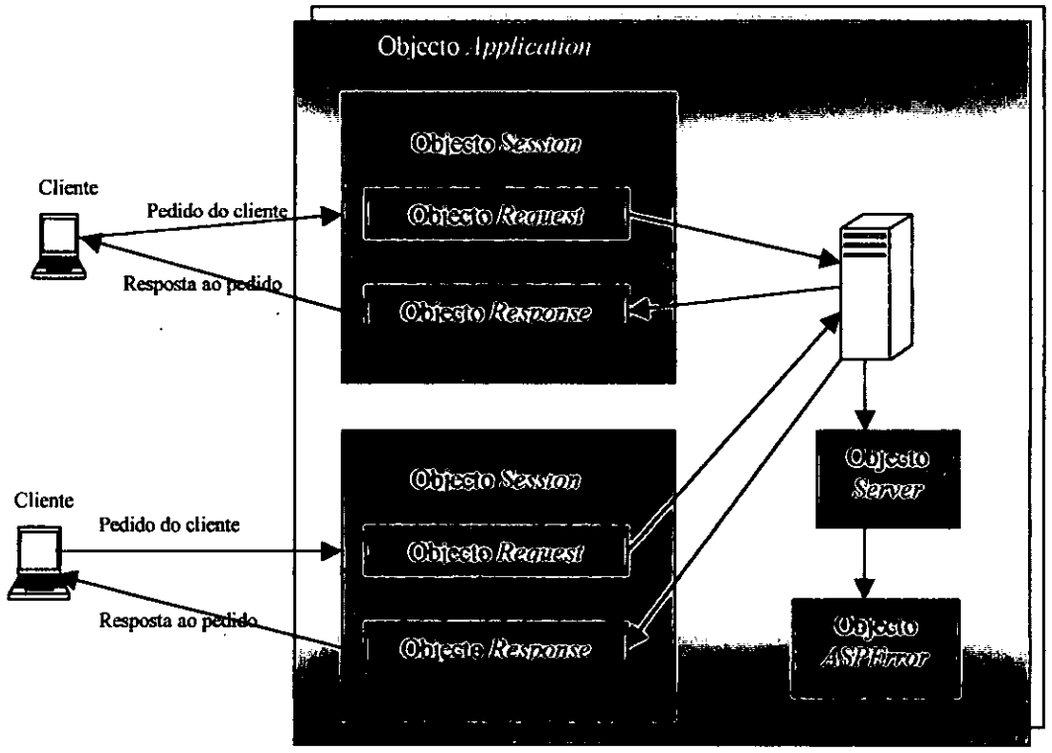


Figura 4.1- Relacionamento entre os objectos, segundo Busel et al, 2000

4.2.1 Objecto Request

Serve para possibilitar a captura, em páginas ASP, de dados enviados pelos formulários HTML ao servidor(Santos, 2000).

O cliente pede ao servidor para criar uma página HTML, requisitando um *script* ASP. O servidor interpreta este tipo de página como ASP. Toda informação enviada com o pedido é empacotada num objecto *request*. Esta informação é depois acedida pelo *script* ASP, que a usa para a construção da página (Busel et al,2000).

Segundo Busel et al (2000), a informação está categorizada em conjuntos, e cada conjunto de informação é armazenado como uma colecção. As colecções guardam valores sobre:

- Os valores enviados pelo cliente e que se encontram no URL. Esta colecção é armazenada numa colecção chamada *QueryString*.

- Se o cliente estiver a enviar um pedido de formulário, então os valores dos elementos do formulário serão armazenados na colecção *Form*.

- O *Servidor Web* contém informação sobre o pedido, sobre a resposta e sobre o próprio servidor. Estas informações são denominadas *HTTP Server Variables* e estão disponíveis como uma colecção.

- Se o cliente enviar alguns *cookies* com o pedido, estes serão inclusos nas suas próprias colecções.

- Por último, se o cliente enviar para o servidor qualquer certificado de segurança, este deverá ser incluso na sua própria colecção.

Através da informação que é inclusa no pedido, e do código *script*, o servidor pode dinamicamente gerar páginas para o cliente. Para que o cliente possa exibir informação, o servidor precisa de um mecanismo para enviar os dados de volta para o cliente. Este trabalho é da responsabilidade do objecto *response* (Busel et al, 2000).

4.2.2 Objecto Response

Antes de uma página *ASP* ser enviada ao cliente, o servidor lê o seu conteúdo, e se encontrar *tags HTML* ou texto, ele envia a página ao cliente. Caso contrário, se encontrar *scripts*, executa-os e repassa o resultado *HTML* para o *browser*. Para representar essas respostas *HTML*, o *ASP* possui o objecto *Response* (Santos,2000).

Segundo Busel et al(2000), com este objecto pode-se:

- Inserir informação numa página a ser enviada ao cliente;

- Enviar instruções ao *browser* para criar *cookies* no cliente;

- Redireccionar o cliente para outra página;

- Controlar se a página será enviada como foi criada ou se será completamente construída e depois enviada.

4.2.3 Objecto ASPError

Contém detalhes sobre qualquer erro gerado por um *script ASP*. No caso de ocorrência de erros, o Objecto *ASPError* direcciona o utilizador para uma página de erros mais comuns (Busel et al, 2000).

4.2.4 Objecto Server

Representa o servidor(Santos, 2000), permite o acesso a algumas das suas propriedades e possibilita a instanciação, em páginas ASP, de componentes escritos pelos programadores.

Estes componentes estendem bastante o poder da programação, permitindo:

- Acesso à Bases de Dados;
- Criação/ leitura/ alteração de arquivos;
- Envio de e-mail.

4.2.5 Objecto Application

Ao conjunto de páginas ASP de um mesmo directório virtual[15] dá-se o nome de aplicação ASP. Esta aplicação é iniciada a primeira vez que um utilizador tenta aceder a qualquer página desse directório, e é finalizada quando o servidor é desligado (Santos,2000).

Com este objecto, podem-se criar variáveis , cujo valor pode ser acedido ou modificado por qualquer utilizador conectado ao directório virtual.

4.2.6 Objecto Session

Sempre que um utilizador se conecta a um aplicativo ASP, inicia-se uma sessão para o mesmo no servidor *Web*. Para representar esta sessão; ASP possui um objecto interno chamado *Session* (Santos,2000).

Este objecto é muito parecido com o objecto *application*, a diferença reside no facto do objecto *Session* poder armazenar valores ligados a um único visitante do site (dono da sessão). Com este objecto podem-se criar variáveis cujo valor pode ser acedido ou modificado somente pelo dono da sessão(Santos,2000).

As sessões criadas por aplicações *ASP*, são mantidas por elementos chamados *cookies*. O servidor envia esses dados (*cookies*) para o *browser* os armazenar na máquina do utilizador.

[15] Para maiores detalhes sobre directórios virtuais, veja anexo E.

4.2.7 Objecto ObjectContext

É usado para controlar as transações dentro da página [Busel et al, 2000]. Por exemplo se duas pessoas estiverem a fazer uma alteração num mesmo instante sobre a mesma base de dados, através da página Web, o sistema, através deste objecto, deve ser capaz de aceitar uma alteração enquanto cancela ou adia a outra.

4.3. Componentes Active Server

Muitos desenhadores de páginas possuem objectivos semelhantes aquando da construção de páginas. Estes objectivos incluem:

- Técnicas de navegação através do site;
- Gestão de anúncios no site;
- Molde das páginas de modo a acomodar as capacidades do *browser* do utilizador.

Seria dispendioso para os desenhadores escrever código que os permitisse alcançar os objectivos acima. A Microsoft resolveu este dilema com um conjunto de ferramentas, que automatiza estas tarefas, chamadas componentes ASP. [Busel et al, 2000]

4.3.1 Componente Ad Rotator

O Componente *Ad Rotator* cria um objecto *Ad Rotator* que automatiza a rotação de imagens de anúncios em uma página. Sempre que um utilizador abre ou recarrega a página, o componente *Ad Rotator* exibe um anúncio novo baseado na informação especificada em um *Rotator Schedule File*[16].

Pode-se registrar quantos utilizadores fazem clique em cada anúncio fixando o parâmetro *URL* no *Rotator Schedule File* para direccionar os utilizadores ao *Redirection File*[17].

[16] Um ficheiro opcional que implementa redireccionamento e permite ao Componente *Ad Rotator* registrar quantos utilizadores fazem clique em cada anúncio.

[17] Um ficheiro de texto que contém um ficheiro de informação e o horário de exibição para os anúncios. Este arquivo deve estar disponível em um *path* virtual do servidor.

4.3.2 Componente Browser Capabilities

Referencia um ficheiro denominado *Browscap.ini*, que detalha todas as versões de todos os *browsers* alguma vez criados. Ele usa esta informação para determinar se o presente *browser* suporta *frames*, tabelas, etc.

O componente *Browser Capabilities* cria um objecto *BrowserType* que proporciona aos *scripts* uma descrição sobre as capacidades do *browser* do cliente.

Quando um *browser* se conecta ao servidor, ele envia automaticamente um "*User Agent HTTP header*". Este *header* é um *string* ASCII que identifica o *browser* e seu número de versão. O objecto *BrowserType* compara o *header* com as entradas do ficheiro *Browscap.ini*.

Se encontrar uma coincidência, o objeto *BrowserType* assume as propriedades do *browser*, que estão listadas no *User Agent Header*.

Se o objecto não encontrar nenhuma coincidência, para o *header* no ficheiro browscap.ini, ele assume as propriedades *default* do *browser* especificadas no ficheiro *browscap.ini*. Se o objecto não encontrar uma coincidência, e as propriedades, *default*, do *browser* não tiverem sido especificadas no ficheiro *browscap.ini*, ele fixa toda a propriedade para "Desconhecido".

4.3.3 Componente Content Linking

Usa um ficheiro do tipo texto para gerir um conjunto sequencial de páginas. Permite que o administrador providencie informação extra sobre cada página da sequência, e mantém os *links* numa lista ordenada, de modo que possam ser facilmente mantidas.

O componente *Content Linking* cria um objecto *Nextlink* que administra uma lista de URLs de forma que se possa tratar as páginas num site, tal e qual as páginas de um livro. O componente *Content Linking* pode ser usado para automaticamente gerar e actualizar tabelas de conteúdos e links de navegação. O Componente *Content Linking* referencia um ficheiro *Content Linking List* que contém a lista das páginas. Esta lista é armazenada no servidor.

4.3.4 Componente Content rotator

O componente *Content Rotator* cria um objecto *ContentRotator* que automaticamente faz a rotação dos conteúdos, em string HTML, numa página. Sempre que um utilizador requisita uma página, o

objecto exibe um novo conteúdo, em HTML, baseado em informação previamente especificada em um ficheiro *Content Schedule*.

Porque os conteúdos em HTML contêm tags, pode-se exibir qualquer tipo de conteúdo que o HTML possa representar: texto, imagens, ou *hyperlinks*. Por exemplo, pode-se usar este componente para girar por uma lista de cotações diárias ou *hyperlinks*, ou para mudar a cor do texto e de fundo sempre que a página for aberta.

4.3.5 Componente Counters

Cria um objecto que permanece activo durante o tempo de vida de uma aplicação, e pode ser usado para armazenar, incrementar e receber um valor.

4.3.6 Componente Loggins Utility

Permite que uma aplicação esteja habilitada a ler um Ficheiro *log* do *Internet Information Servers* (*IIS*).

4.3.7 Componente MyInfo

É usado para armazenar informações pessoais do administrador do servidor.

4.3.8 Componente Page Counter

Providencia um contador de páginas, que incrementa em um cada vez que uma página é acedida.

O componente *Page Counter* cria um objecto *PageCounter* que conta e exibe a quantidade de tempo no qual uma página permanece aberta. Em intervalos regulares, o objecto escreve o número de acessos a um arquivo de texto de forma que no caso de um paralisação do servidor, o dados não se percam. O componente *Page Counter* usa um objecto chamado *Central Management* para registrar quanto tempo cada página permaneceu aberta na aplicação.

Quando uma instância do objecto *PageCounter* é criada, em uma página, usando o método *Server.CreateObject*, o objecto recupera a conta de acesso actual, para a página especificada, a partir do objecto de *Central Management*.

4.3.9 Componente Permission Checker

É usado para monitorar se um certo utilizador tem permissão para ler ou executar um determinado ficheiro.

O componente *Permission Checker* cria um objecto *PermissionChecker* que usa os protocolos de autenticação de *password* providenciado pela *Microsoft® Internet Informação Servidor* (IIS) para determinar se um utilizador tem permissão para ler um ficheiro.

Pode ser usado para ajustar uma página baseada em *ASP* para vários utilizadores. Por exemplo, se uma página contém *hyperlinks*, o componente *Permission Checker* pode ser usado para testar se o utilizador tem permissão para as páginas designadas. Se o utilizador não possuir as permissões apropriadas, pode-se omitir ou alterar o *hyperlink* para essas páginas, de modo a que o utilizador não tenha acesso.

4.3.10 Componente Tools

Inclui um controlo para ver se um determinado ficheiro existe ou se um determinado utilizador é o dono do site.

4.4. O ASP e o acesso à base de dados

Um sistema de base de dados é um sistema de armazenamento de dados baseado em computadores, isto é, um sistema cujo objectivo global é registar e manter informação. Esta informação tem que estar disponível, pois é necessária para o processo de decisão de uma organização [Date, 1989].

Microsoft tem desenvolvido tecnologias de acesso aos dados , independentemente do formato, tais como as tecnologias *Open Data Base Connectivity (ODBC)* e *Object Linking and Embedding Database (OLE-DB)* [Busel et al, 2000]. E como não podia deixar de ser, estas tecnologias afectam *ASP*, pois o *OLE-DB* é usado em ASP, através de um interface de programação conhecido por *ActiveX Data Objects (ADO)*.

4.4.1 ODBC (Open Data Base Connectivity)

É um padrão de acesso aos dados. Foi desenhado para permitir o uso, pelo programador, de um conjunto de rotinas de acesso aos dados armazenados na base de dados. Isto significa que, assim que o programador esteja conectado à base de dados, usando *ODBC*, os dados podem ser manipulados sem que haja preocupação sobre o local ou o tipo de base de dados onde os dados se

encontram. Isto permite ao programador aceder a uma base de dados *Oracle* do mesmo modo que acede a uma base de dados *SQL Server*. A figura 4.4.1 ilustra a construção de uma aplicação de acesso aos dados usando *ODBC*.

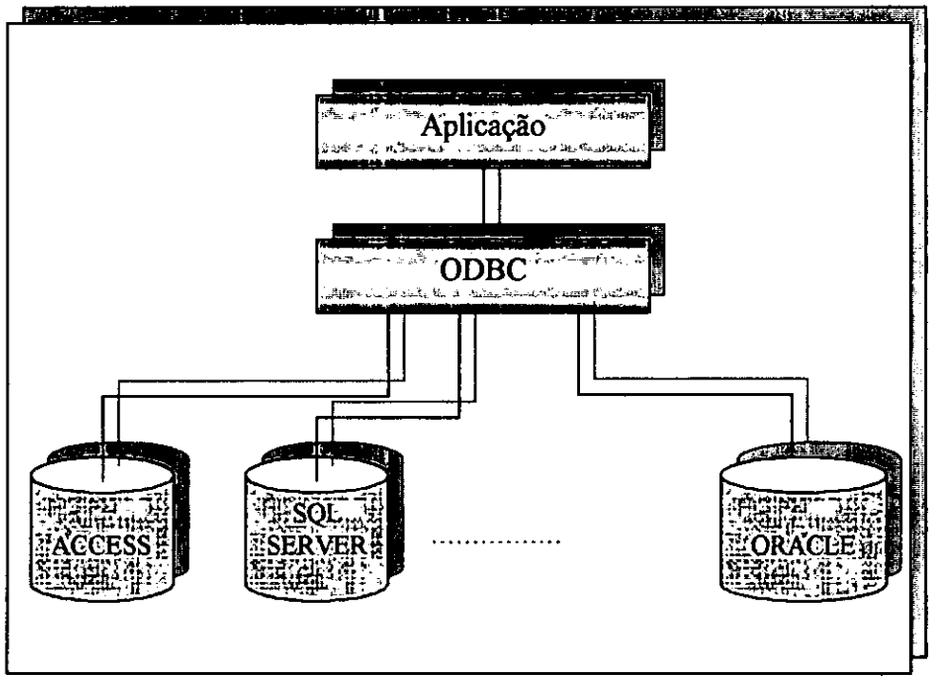


Figura 4.2 - Acesso aos dados usando ODBC

4.4.2 OLE-DB (Object Linking and Embedding Database)

Bases de dados armazenam informação de tal forma que possam ser uniformemente acessíveis pelo *ODBC*. Mas o que acontece quando se pretende aceder aos dados armazenados nos formatos *word, excell, outlook*, etc? Geralmente a informação contida em cada um dos documentos acima mencionados não é compatível com o formato de base de dados, e muitas das vezes *ODBC* não consegue ter acesso a esse tipo de dados.

Para colmatar esta situação, a *Microsoft Universal Data Access* desenvolveu uma tecnologia que tem o potencial para aceder aos dados contidos em qualquer tipo de *armazém de dados*. Essa tecnologia é denominada por *OLE-DB*. A figura 4.4.2 ilustra a construção de uma aplicação de acesso aos dados usando *OLE-DB*.

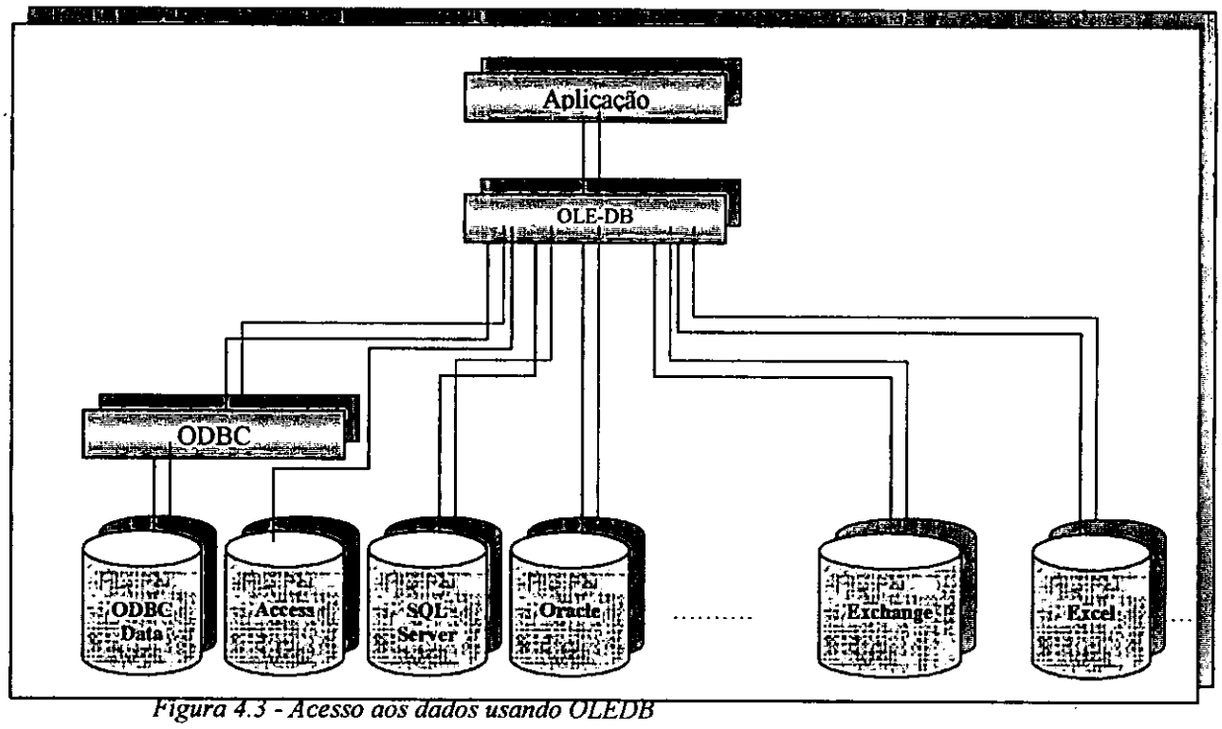


Figura 4.3 - Acesso aos dados usando OLEDB

Como pode observar, apesar de *OLE-DB* ser semelhante ao *ODBC*, ele, além de suportar conexões à bases de dados através de *ODBC*, permite o acesso a uma maior variedades de formatos de dados.

4.4.3. Active Data Objects (ADO)

Um provedor de dados (*Data provider*) é uma unidade de código escrita numa linguagem de programação, que usa objectos *OLE-DB* para providenciar as instruções necessárias de comunicação e passagem de dados entre as base de dados e o consumidor de dados (*Data consumer*).

O consumidor de dados aparece na forma de um conjunto de objectos conhecidos por *ActiveX Data Objects (or **ADO**)*. *ADO* é um interface que permite a comunicação entre as páginas *ASP* e o *OLE-DB*. A figura seguinte ilustra o modelo de objectos *ADO*.

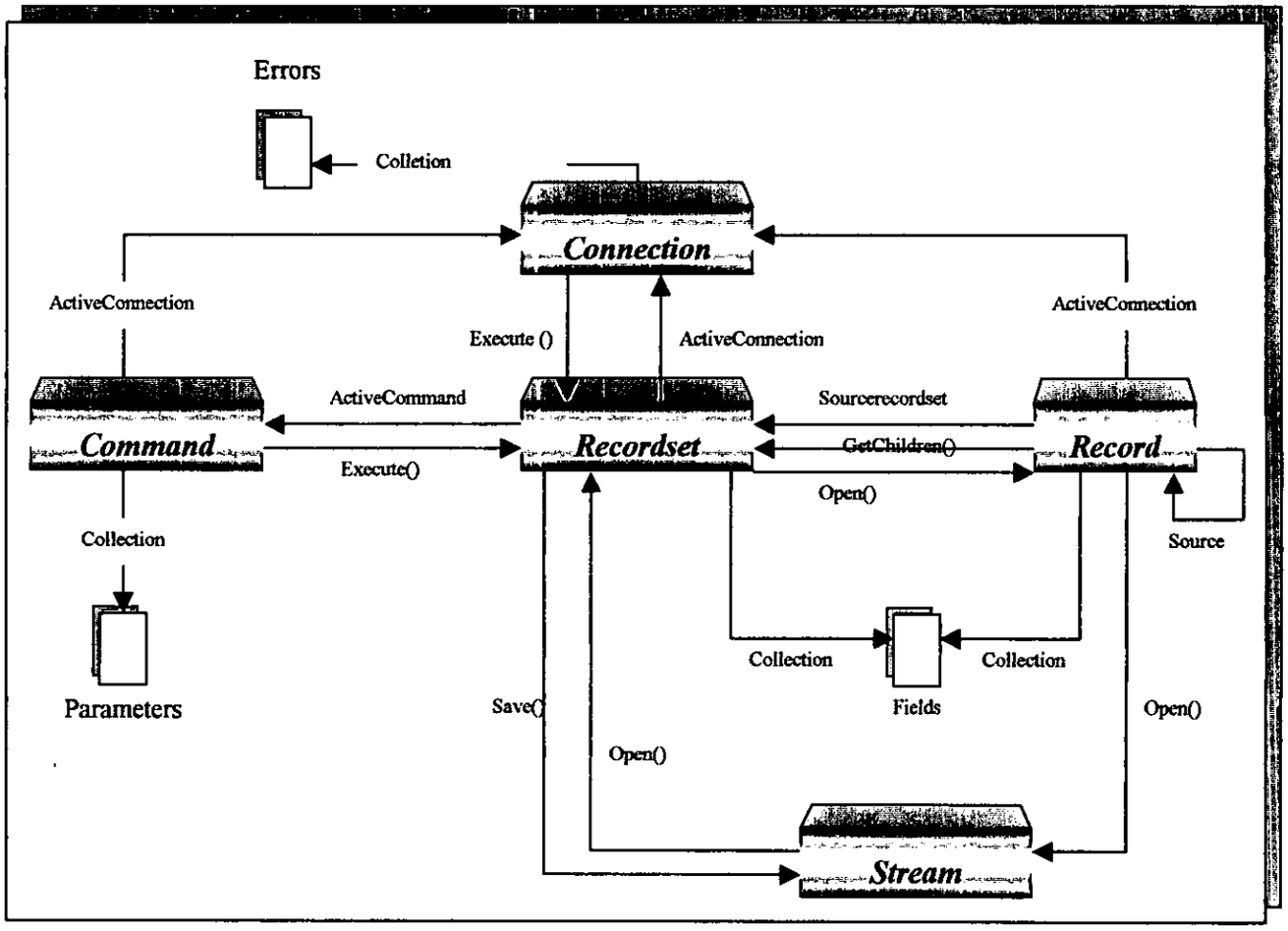


Figura 4.4 - Modelo de Objectos ADO, segundo Busel et al, 2000

Como pode observar na figura 4.4.3, o modelo *ADO* possui cinco objectos principais:

- ***Connection***- É um objecto usado para representar a conexão física entre as página ASP e a base de dados. A conexão pode ser feita, usando *Connection Strings*[18], *Data Link Files* [19], *Data Source Name*[20]

- ***Command***- permite a execução de comandos na base de dados.

- ***Recordset***- contém todos os dados obtidos através de uma acção específica na base de dados.

- ***Record***- permite lidar com dados em armazéns semi-estruturados (tal como ficheiros numa estrutura de directório) como se fossem registos numa base de dados.

[18] Um conjunto de caracteres que contém toda informação necessária para conectar à base de dados.
[19] método de conexão à base de dados, usando um ficheiro que contém o *connection string.*
[20] Outra maneira de estabelecer a conexão sem que seja necessário digitar o *connection string.*

➧ ***Stream***- permite a manipulação de dados contidos em recursos web, tal como ficheiros *HTML*.

4.4.3.1 Objecto Connection

É o que o *ADO* usa para armazenar informação sobre a conexão à base de dados. Para criar um objecto *Connection*, usa-se a seguinte sintaxe:

```
<%

dim objconn

 set objconn= server.createObject("ADODB.Connection")

%>
```

Onde o *ADODB.Connection* é o identificador do objecto *Connection.*

Após a criação do objecto *Connection*, tem que se fazer a conexão à base de dados. Para tal, recorre-se ao método *Open*. A sintaxe usada para o método *Open* é:

ObjConn.Open ConnectionString, UserId, Password, Options

O *connection string* pode ser escrito da seguinte forma:

```
ObjConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; " & _
             "Data Source=d:\FichaObra\FichaObra.mdb;" & _
             "Persist Security Info=false"
```

Pode-se também gravar o *connection string* num ficheiro *Server Side Include(SSI)*. Quando o ASP encontra um comando # *INCLUDE,* ele procura pelo argumento *file*, que especifica o ficheiro a ser incluso, depois coloca o conteúdo do ficheiro encontrado no ficheiro corrente[Busel et al, 2000]. Por exemplo, pode-se criar um ficheiro *SSI,* neste caso usando *Data Source Name*, que contém os detalhes de conexão, chamado *conexao.asp*:

```
<%

dim objconn

 set objconn= server.createObject("ADODB.Connection")
 datasource="ficha1"
 ObjConn.Open datasource,"SA",""


%>
```

Pode-se incluir o código acima, em cada página que usa a conexão, escrevendo a seguinte linha :

```
<!-#INCLUDE FILE="conexao.asp" - - >
```

Assim que a conexão terminar, tem que se libertar os recursos do sistema associados à conexão. Para tal usa-se o comando:

ObjConn.Close

Isso não remove o objecto da memória. Para o remover da memória faz-se o seguinte:

Set ObjConn=Nothing

4.4.3.2 Objecto Command

Permite a execução de comandos *Structured Query Language (SQL)* sobre a base de dados. A criação do Objecto *Command* é feita usando a seguinte sintaxe:

```
dim objCommand

Set ObjCommand= server.createObject("ADODB.Command")
```

O Objecto *Command* permite inserir, actualizar e apagar dados de uma determinada base de dados . Estas operações são executadas usando os seguintes comandos:

- *Select:* retorna um *recordset*, contendo dados da base de dados;
- *Insert:* adiciona registos à base de dados;
- *Update:* actualiza porções de informações existentes num registo da base de dados;
- *Delete:* remove registos da base de dados.

4.4.3.3 Objecto Recordset

Segundo *Busel et al*, 2000, um *recordset* é definido como um conjunto de dados resultante de uma consulta à base de dados e pode ser representado usando o Objecto *Recordset*.

Um objecto *Recorset* é responsável pelo acesso aos campos e dados de uma ou mais tabelas em uma base de bados[Pereira, 2001]. Possui funcionalidades que permitem manipular os dados, adicionar, remover ou esconder registos e procurar dados dentro de um determinado registo. A seguir está um exemplo da criação de um *recordset*:

```
Dim objRS

Set objRS = Server.CreateObject("ADODB.Recordset")
```

4.4.3.4 Objecto Record

O objecto *Record* representa um registo num *recordset*. Este objecto é usado em sistemas de ficheiro e de correio para mover e copiar ficheiros ou mensagens. A criação de uma instância do objecto *Record* é feita usando a sintaxe:

```
Dim objRecord

Set objRecord = Server.CreateObject("ADODB.Record")
```

4.4.3.5 Objecto Stream

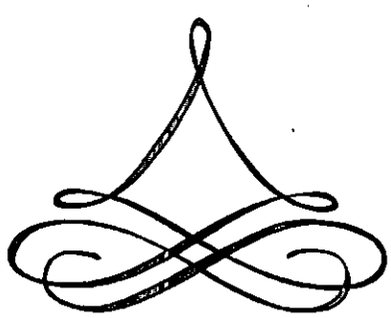
Este objecto é usado para representar os dados binários, mais concretamente os dados do tipo texto. Para criar uma instância do objecto *stream* usa-se a seguinte sintaxe:

```
Dim objStream

Set objStream = Server.CreateObject("ADODB.Stream")
```

Capítulo V

Segurança



5. Segurança

Com a rápida evolução e vulgarização do modelo cliente/servidor e o crescimento do comércio na Internet, existe a necessidade de ter uma comunicação segura.

A segurança na Internet consiste essencialmente em dois aspectos distintos: a segurança das transacções e a integridade das redes privadas. A segurança das transacções refere-se à possibilidade de duas entidades poderem conduzir uma transacção privadamente sem influência de outros, com autenticação através de assinaturas digitais, se necessário. No que se refere à integridade das redes, este aspecto visa essencialmente a protecção dos recursos informáticos, ligados à Internet, de uso ou acesso sem autorização.

Uma enorme agitação tem ocorrido devido ao potencial da Internet no que diz respeito ao comércio electrónico, no entanto, surge uma das maiores preocupações das empresas que fazem transacções na Internet: como as fazer com segurança ?

Neste sentido, para conseguir acabar com o uso fraudulento de informações confidenciais, como por exemplo senhas de acesso aos dados, foram desenvolvidos mecanismos baseados em criptografia na tentativa de tornar as transacções mais seguras, tal como: a autenticação mútua das entidades envolvidas, mecanismos de certificação, etç.

Este capítulo tem como objectivo, sensibilizar os gestores de negócios bem como utilizadores para o problema de segurança, e apresentar de uma forma resumida e acessível formas e mecanismos que permitem a prevenção de possíveis perigos inerentes a Internet.

Para a compreensão do funcionamento de sistemas seguros, são apresentados conceitos básicos de segurança.

Conceitos básicos sobre a segurança

- ***Autenticidade***

  O controle de autenticidade está associado à identificação correcta de um utilizador ou computador. O serviço de autenticação em um sistema deve assegurar ao receptor que a mensagem é realmente procedente da origem informada em seu conteúdo. Normalmente, isso é implementado a partir de um mecanismo senhas ou de

assinatura digital. A verificação de autenticidade é necessária após todo processo de identificação, seja de um utilizador para um sistema, de um sistema para o utilizador ou de um sistema para outro sistema. [Puttini [3]].

- ***Confidencialidade:***

  Confidencialidade significa proteger informações contra sua revelação para alguém não autorizado - interna ou externamente. Consiste em proteger a informação contra leitura e/ou cópia por alguém que não tenha sido explicitamente autorizado pelo proprietário daquela informação. A informação deve ser protegida independentemente do mídia que a contenha, como por exemplo, mídia impressa ou mídia digital. Deve-se considerar não apenas a protecção da informação como um todo, mas também de partes da informação que possam ser utilizadas para interferir sobre o todo. No caso de uma rede, isto significa que os dados, enquanto em trânsito, não serão vistos, alterados, ou extraídos da rede por pessoas não autorizadas ou capturados por dispositivos ilícitos[Puttini [3]].

- ***Integridade:***

  A integridade consiste em proteger a informação contra modificação sem a permissão explicita do proprietário daquela informação. A modificação inclui acções como escrita, alteração de conteúdo, remoção e criação de informações. Deve-se considerar a protecção da informação nas suas mais variadas formas, como por exemplo, armazenada em discos ou fitas de *backup*. Integridade significa garantir que se o dado está lá, então não foi corrompido, encontra-se íntegro. Isto significa que aos dados originais nada foi acrescentado, retirado ou modificado.

O protocolo *Secure Socket Layers (SSL)* e o *firewall*, fornecem funções que permitem que se alcancem os três requisitos citados anteriormente (autenticidade, confidencialidade e integridade).

O protocolo SSL utiliza diferentes algoritmos criptográficos para implementar segurança utilizando autenticação com certificados, e codificação de dados com chaves. É um protocolo utilizado para garantir que a comunicação entre um servidor e um cliente seja segura e cifrada.[Puttini [1]]

O *firewall* é uma barreira, entre a rede local e a Internet, formada por software e hardware, através da qual só passa tráfego autorizado.

5.1. Protocolo SSL

Foi inicialmente desenvolvido pela *Netscape*, tornando-se o padrão para comunicações seguras e autenticadas entre clientes e servidores na Internet. Este protocolo é baseado nos algoritmos de criptografia simétricos[21] e assimétricos[22] [Gonzalo,2000]. O algoritmo assimétrico é versátil, pois permite a passagem de chaves através da Internet. O simétrico, por outro lado, é mais rápido e é usado durante toda a conversa após a passagem da chave secreta.

5.1.1 Criptografia de chave pública e secreta

É uma técnica baseada no uso de duas chaves, uma pública conhecida por todos, e outra privada, somente conhecida pelo dono das chaves (normalmente o servidor *web*). Essas chaves são utilizadas para cifrar[23] dados como ilustra a *figura 5.1*:

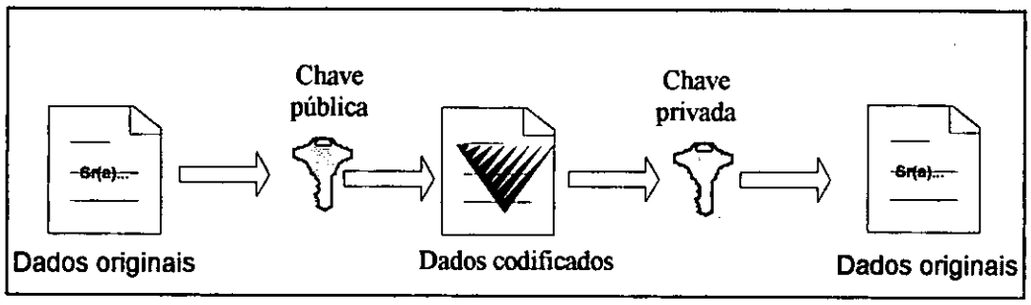


Figura 5.1 - Ilustração da criptografia baseada em chave pública

O processo de cifrar e decifrar[24] uma mensagem com um algoritmo de chave pública é computacionalmente pesado e seria inviável utilizá-lo para cifrar toda a comunicação entre, por exemplo, o *browser* e o servidor *web*. De facto, o *SSL* só utiliza esse método para a construção de chaves secretas, só conhecidas pelo cliente e servidor. Esse processo, conhecido como *SSL handshake* pode ser resumido da seguinte maneira[Gonzalo,2000]:

1. O servidor envia sua chave pública ao cliente;
2. O cliente cria uma chave secreta aleatória, cifra-a com a chave pública e envia ao servidor;

[21] Baseados em chave secreta
[22] baseados em chave pública
[23] Codificar os dados com chave
[24] Descodificar os dados com chave

3. O servidor, usando a chave privada, obtém a chave secreta;

4. Cliente e servidor iniciam a comunicação cifrando/decifrando as mensagens com a chave secreta. Eles utilizam um algoritmo de criptografia simétrico, veja a *figura 5.2*:

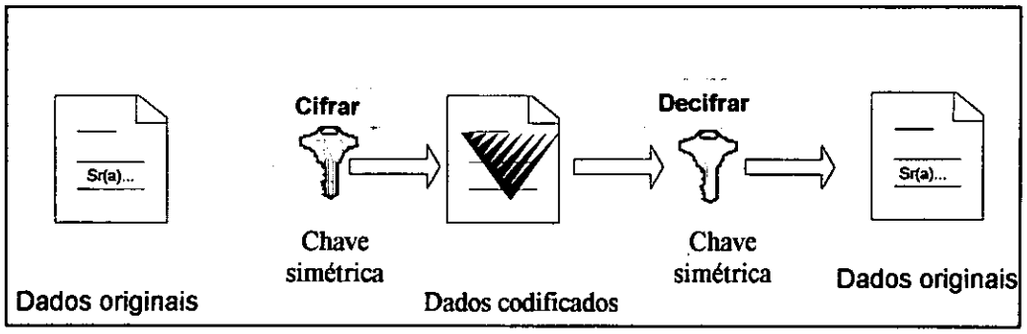


Figura 5.2 - Ilustração do algoritmo de criptografia simétrica

Como pode observar da figura *5.1.1.b*, os algoritmos utilizados são chamados de simétricos porque é utilizada a mesma chave para cifrar e decifrar as mensagens [Gonzalo,2000].

Para garantir que as mensagens não sejam alteradas no trajecto, o protocolo *SSL* adiciona a cada mensagem um *MAC (Message Authentication Code)*, que é verificado na recepção. O *MAC* é uma sequência de bits resultante da aplicação de uma função de *hashing*[25] à mensagem, combinada com uma chave secreta. Essas funções de *hashing* têm a propriedade de que qualquer alteração na mensagem implica um novo *MAC*. Mas como só o cliente e o servidor conhecem a chave secreta, um estranho não poderá alterar a mensagem e criar o *MAC* correcto. Para maior segurança, o *SSL* cifra o *MAC* com a mensagem[Gonzalo,2000].

5.1.2 Certificados

O mecanismo descrito acima garante a segurança da comunicação entre cliente e servidor, mas não garante que o cliente esteja a comunicar com o servidor correcto. O protocolo *SSL* tenta resolver esse problema da seguinte maneira: durante o *SSL handshake*, o servidor envia para o cliente, em lugar de uma simples chave púbica, um certificado para testar se a chave pública contida no certificado é realmente do servidor em questão.

[25] Um conjunto de funções usadas para criar e destruir objectos hash. Objecto hash é um objecto usado para baralhar mensagens. Um hash é um resultado de tamanho fixo obtido através da aplicação de uma função matemática em uma quantidade arbitrária de dados.

O certificado do servidor pode ser encarado como o seu bilhete de identidade. Normalmente, para um bilhete de identidade ser aceite como válido, duas condições têm que ser satisfeitas: o bilhete tem que ter sido emitido por um órgão reconhecido; e este tem que ser genuíno, isto é, ter as marcas de água, selos ou carimbos apropriados[Gonzalo,2000].. Nos próximos parágrafos descrevemos como esses conceitos são implementados no protocolo *SSL*

Para implementar a primeira ideia, ou seja a emissão do certificado por um órgão reconhecido, o *SSL* usa o conceito de ***Certification Authority*** **(CA)**, que é uma entidade capaz de emitir certificados.

Para garantir que um certificado foi realmente emitido por uma *CA*, estes são construídos utilizando uma outra propriedade de criptografia de chave pública: o par de chaves pública e privada pode ser utilizado no sentido inverso ao descrito inicialmente. Ou seja, o dono da chave pode cifrar uma mensagem com a sua chave privada, para posteriormente ser decifrada com a chave pública. Só que agora, qualquer um que tenha uma cópia da chave pública poderá decifrar a mensagem. Logo, cifrar com a chave privada não garante a inviolabilidade da mensagem, mas garante que a mensagem foi realmente enviada pelo detentor da chave privada e não por um outro[Gonzalo,2000].

Para construir o certificado de um servidor, uma *CA* adiciona aos dados do mesmo uma assinatura digital, testando a sua validade, tal como ilustrado na *figura 5.3*. Essa assinatura é criada ao aplicar uma função de *hash* aos dados do certificado e ao cifrar o resultado com a chave privada da *CA*. Para verificar a integridade e validade do certificado, o cliente reaplica a função de *hash* e a compara-a com a enviada na assinatura, decifrada com a chave pública da *CA*. Este processo é ilustrado na figura *5.1.2a)*.

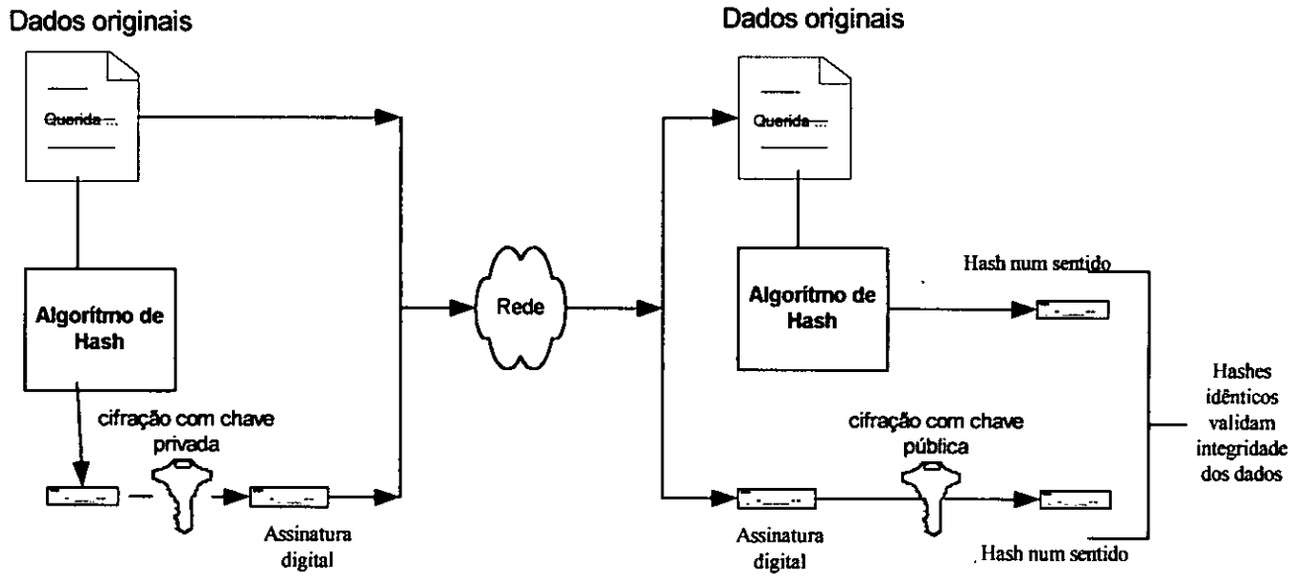


Figura 5.3 - Processo de criação de certificados, segundo Gonzalo, 2000

Logo, o cliente deve manter uma lista das chaves públicas das *CA*s em que confia. A interacção entre o certificado do servidor e o certificado da CA é ilustrada na *figura 5.4*.

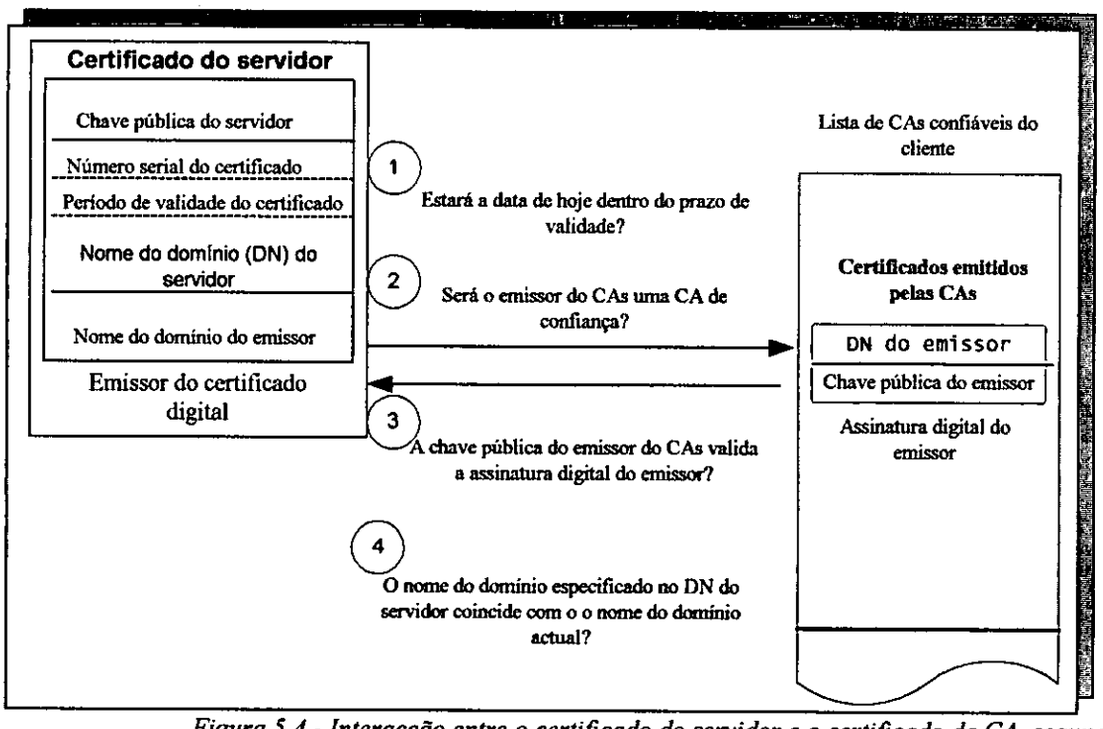


Figura 5.4 - Interacção entre o certificado do servidor e o certificado da CA, segundo Ginzalo,2000

Como pode observar na figura *5.4*, ao receber o certificado do servidor, o cliente:

- verifica o período de validade do certificado;

- verifica se a *CA* que assinou o certificado está na lista de *CA*s de confiança, ou seja, se o *software* tem uma cópia do certificado da *CA*;

- utiliza a chave pública no certificado da *CA* para validar o certificado do servidor;

- verifica se o nome no certificado é o nome do servidor especificado pelo usuário;

- Caso todos os testes passem, a conexão *SSL* é estabelecida. Caso contrário, o *software* informa ao utilizador do problema, e, normalmente, indaga se a conexão pode ser estabelecida mesmo assim.

5.2 Firewall

A tradução da palavra firewall é parede de fogo, isto é, um *firewall* é uma barreira entre a rede local e a Internet, através da qual só passa tráfego autorizado. Este tráfego é examinado pelo *firewall* em tempo real e a selecção é feita de acordo com regras estipuladas.

Como um porteiro na entrada de um edifício, o *firewall* é colocado na entrada da rede de computadores (ou na entrada da *intranet* corporativa). Para fazer correctamente o seu trabalho, todo o tráfego deve passar por ele. A *figura 5.2* ilustra um *firewall* colocado à entrada de uma rede.

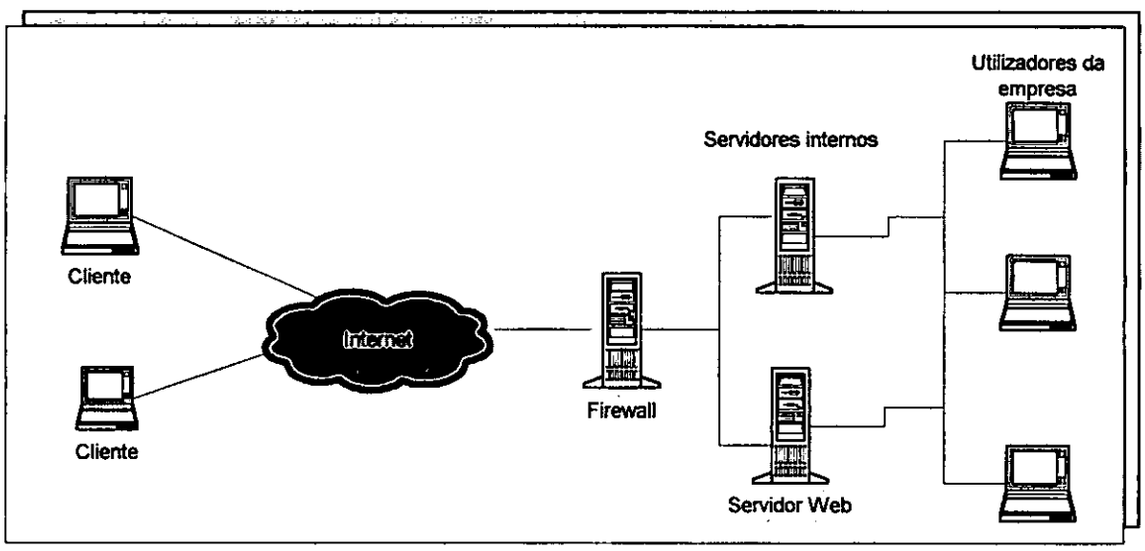
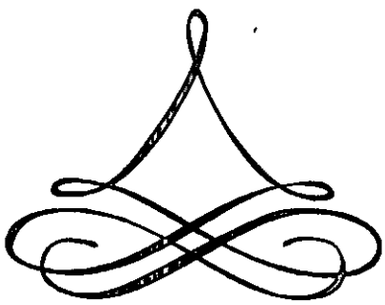


Figura 5.5 - Posicionamento do firewall numa rede

Capítulo VI

Caso de Estudo



6. Caso de Estudo – Sistema de Gestão de Fichas de Obra

O objectivo deste capítulo é apresentar um modelo de aplicação da tecnologia ASP no Sistema de Gestão de Fichas de Obra, da Exi. Este modelo centrar-se-á no atendimento ao cliente, permitindo o acompanhamento, pelo cliente, da resolução dos problemas por ele apresentados.

Para tal, teve-se em mente aproveitar ao máximo a estrutura e recursos já disponíveis, não apenas por questões económicas, mas também para melhorar estrategicamente a possibilidade da sua implementação.

6.1 Descrição do Sistema de Gestão de Fichas de Obra

Face à crescente redução do nível de serviço prestado pelos seus departamentos aos seus clientes, e a constantes reclamações destes no que concerne a:

- Demora excessiva na resposta aos clientes;
- Falta de controlo do cumprimento dos planos de manutenção assinados entre a companhia e os seus clientes;
- Falta de informação sobre o andamento dos pedidos de serviço solicitados pelos clientes;
- As fichas de obra resultantes dos pedidos de serviço passam por vários técnicos e em algumas ocasiões perdem-se ou não são registadas todas as intervenções efectuadas;
- Falta de controlo do desempenho dos técnicos;
- Demora na entrega de dados para a facturação;

A direcção da Exi decidiu desenvolver uma aplicação (*Sistema de gestão de Fichas de Obra*) que pudesse atender aos requisitos do cliente e garantir a redução dos custos de acesso à informação disponível.

Esta aplicação foi desenvolvida em *Visual Basic*, e possui como sistema de gestão de base de dados o *SQL Server*. A base de dados da aplicação contém informação sobre: clientes da empresa, funcionário, fichas de obra, contratos assinados entre os clientes e a empresa, etc. Veja o esquema de tabelas do sistema no *anexoH figura H1*.

Para manipular a informação da base de dados, o sistema possui quatro processos, que são:

- Registar pedido de serviço;
- Alocar técnico;
- Executar serviço;
- Tratar material;

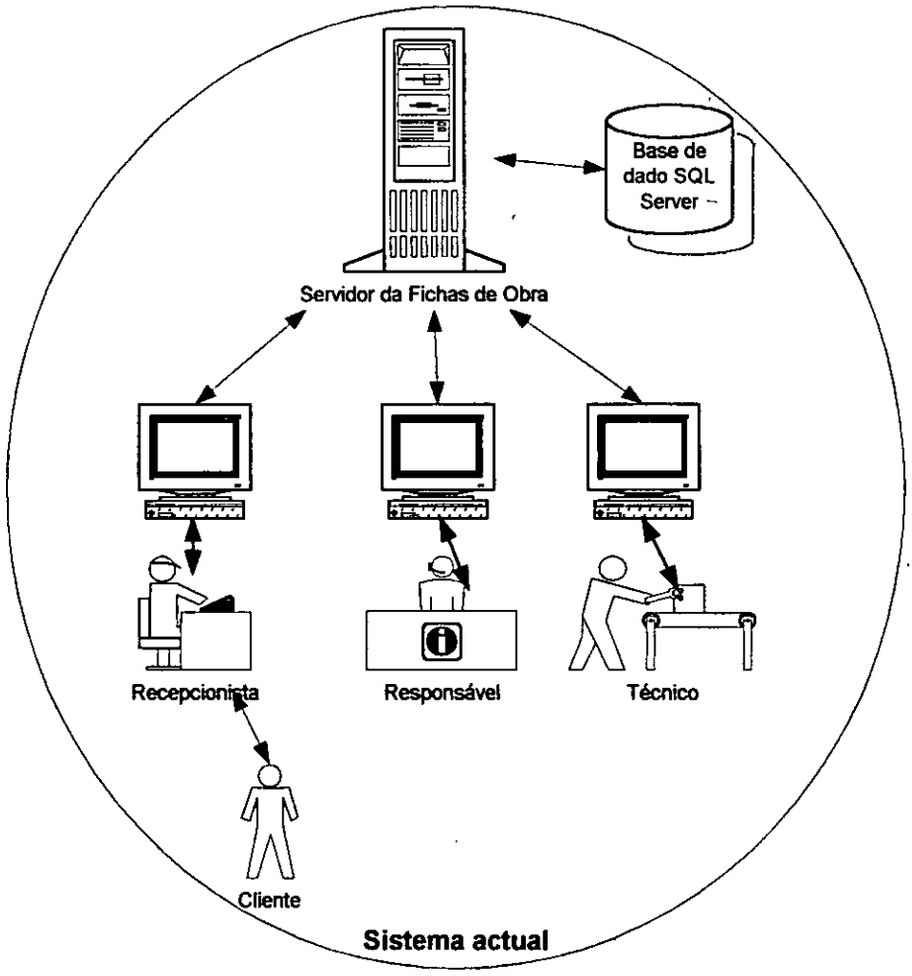


Figura 6.1 - Interacção entre os vários intervenientes do processo de gestão da ficha de obra.

A interacção entre os vários intervenientes do sistema actual está ilustrada na *figura 6.1*, e é descrita detalhadamente a seguir:

a) Registar pedido de serviço

Em conformidade com a participação do cliente, o recepcionista:

- Verifica a identidade do cliente, se o cliente não constar na base de dados, ele cria um novo;
- Verifica se o cliente possui contrato a para intervenção requerida;
- Regista a participação do cliente, incluindo o material a ser reparado;
- Verifica para que área se enquadra essa participação e envia ao respectivo responsável.

b) Alocar técnico

Com base na ficha de obra recebida, o responsável da área:

- Recebe a participação da recepção;
- Verifica a disponibilidade do técnico;
- Atribui a tarefa ao técnico;

c) Executar serviço

Em conformidade com a ficha de obra recebida, o técnico:

- Examina a participação;
- Consulta o *stock*, para o caso de ter que adquirir novo material;
- Emite uma requisição proforma;
- Avisa ao responsável da área;
- Aguarda a resposta da requisição, mudando o *status* da ficha de obra;
- Recebe material e actualiza o *status* da ficha de obra;
- Realiza a tarefa incumbida;
- Actualiza o *status* da ficha de obra.

d) Requisitar material

Em conformidade com a requisição proforma, o responsável da área:

- faz a requisição do material necessário para a solução do problema diagnosticado e envia ao stock;
- Faz actualização do estado da ficha de obra.

Concluído o serviço, a ficha de obra é enviada para a Direcção Financeira onde é elaborada a factura, de acordo com o descrito na ficha de obra.

A factura depois de emitida, é enviada para o cliente, ou é solicitada por este para efectuar o respectivo pagamento. Efectuado o pagamento, o cliente dirige-se à recepção para levantar o equipamento, onde lhe é passada uma nota de entrega para ser assinada depois de se realizarem todos os testes de operacionalidade do equipamento.

Este sistema , responde às necessidades internas da empresa. Porém, a crescente concorrência que se tem verificado, actualmente, tem forçado a direcção da Exi a: criar mecanismos de aproximação entre o cliente e a empresa e desenvolver estratégias de melhoria dos seus serviços.

Para tal, decidiu-se fazer uso da Internet para possibilitar o acesso, pelo cliente, ao Sistema de Gestão de Fichas de obra, de modo a torná-lo parte integrante do sistema. Este objectivo pode ser alcançado através de tecnologias de desenvolvimento de páginas dinâmicas, que permite com que o utilizador interaja com o sistema, e tenha acesso a informação personalizada (*Veja capítulo III para mais detalhes*).

6.2 Modelo proposto

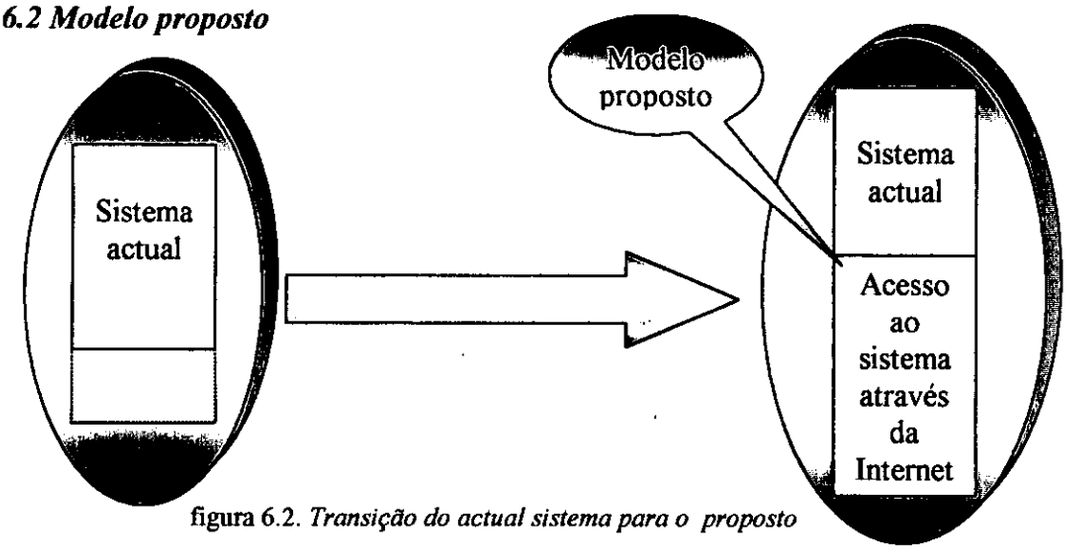


figura 6.2. *Transição do actual sistema para o proposto*

O modelo, ilustrado na *figura 6.2*, foi elaborado de modo que a sua integração, na Exi, não implique mudanças significativas no funcionamento do sistema actual, e permite consultar informação da empresa remotamente. A aplicação do modelo na Exi está ilustrada na *figura 6.3*.

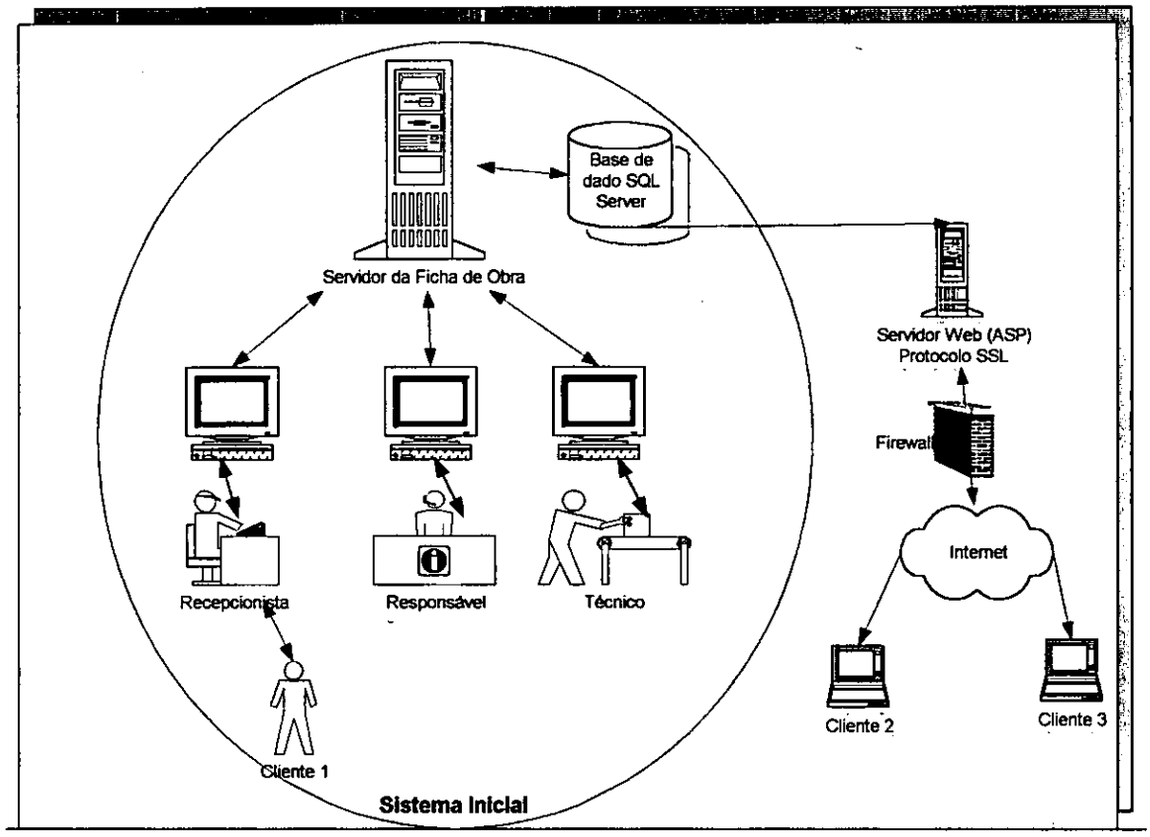


Figura 6.3 - Modelo de acesso à informação

Segundo o modelo, as alterações a efectuar no sistema serão:

- Inclusão do módulo que permite a consulta, da informação do sistema, a partir da Internet ;

- Durante o processo " *Registar pedido de serviço*", descrito anteriormente, o recepcionista para além das actividades que tem executado, irá registar e fornecer ao cliente, caso não a tenha, a senha com a qual terá acesso à informação.

Neste modelo, é necessário ter em conta o aspecto, segurança. Pois, permitir acesso a um sistema através da Internet acarreta riscos, a partir do momento que isso acontece, as seguintes situações são passíveis de ocorrer:

- roubo de informação confidencial, porque a Internet facilita a monitoração do tráfego por parte de estranhos;

- alteração da informação em tráfego, possibilitando a entrada de dados fraudulentos na base de dados;

- um computador pode assumir o nome do servidor da empresa, sem conhecimento do utilizador, fazendo com que estes enviem informação confidencial ao servidor pirata.

Para contornar os problemas acima mencionados, propõe-se o uso do protocolo *Secure Socket Layer (SSL)*, descrito no *capítulo V*.

No modelo propõe-se o uso da tecnologia *Active Server Page (ASP)*, que como foi visto no *capítulo IV*, possui objectos e componentes que permitem com que o utilizador interaja com o sistema, remotamente.

ASP, é uma tecnologia *Server Side*, o que significa que o código fonte não é visível ao utilizador. Isto trás alguma vantagem em termos de segurança da informação.

6.2.1 Requisitos do modelo

Como foi visto no ponto anterior, será usada como tecnologia de desenvolvimento de páginas dinâmicas a tecnologia *Active Server Page*. Para que tal aconteça, é necessário que os seguintes requisitos sejam preenchidos:

- ferramenta de desenvolvimento de páginas para Internet para criar e editar *scripts* ASP;

- servidor *Web* que suporta ASP, para publicar as páginas;

- base de dados, que contém a informação a consultar;

- *browser*, para testar as páginas.

a) Ferramenta de desenvolvimento de páginas para Internet

No mercado, existe um certo número de editores de texto e aplicações com os quais se podem construir páginas dinâmicas para Internet, entre os quais encontram-se os seguintes:

- *Microsoft Visual Interdev* 6.0;

- *Microsoft FrontPage*;

- *Allaire's Homesite*;
- *NotePad.*

De referir que qualquer ferramenta acima mencionada pode ser usada para o efeito.

b) Servidor Web

A selecção do servidor *Web* depende do sistema operativo. Por exemplo, se o sistema operativo for:

- Windows 95/98, o servidor *Web* a será o *Personal Web Server* (PWS). O PWS está disponível em várias fontes, que são:
  - *Microsoft Visual Interdev* 6.0;
  - Disco do *Windows* 98;
  - *Front Page.*
- Windows NT Server 4.0, o servidor Web a usar será o IIS 4.0, que está disponível como parte do Windows NT 4.0 Option Pack.
- Windows 2000, o servidor Web será o IIS 5.0.

c) Browser

Qualquer *browser* do mercado pode ser usado para visualizar as páginas. A seguir são dados alguns exemplos de *browsers*:

- Internet Explorer;
- Netscape.

6.3 Desenvolvimento do modelo

Durante o desenvolvimento de um sistema de informação, é necessário ter-se em conta a metodologia de desenvolvimento de sistemas a ser usado. Laudon e Laudon, 1998, definem metodologia de desenvolvimento de sistemas como uma colecção de métodos, para cada actividade em cada fase de desenvolvimento.

Segundo Laudon e Laudon, o ciclo de vida tradicional de desenvolvimento de sistemas é caracterizado por seis fases, nomeadamente: definição do problema, análise, desenho, desenvolvimento, teste e implementação. Cada fase inicia após o término da fase anterior e um relatório técnico deve ser elaborado no final de cada fase, descrevendo os resultados obtidos na fase em causa .

Por outro lado, o ciclo de vida da prototipificação consiste na construção de um sistema experimental de forma rápida e barata, para avaliação do utilizador. Pois, porque muitos utilizadores não conseguem descrever os seus requisitos, no papel, a interacção com o protótipo permite ao utilizador, determinar exactamente o que ele precisa. A prototipificação, antecipa as mudanças de ideia que possam ocorrer, e possibilita que estas mudanças sejam facilmente incorporadas durante o processo de desenvolvimento.

A escolha do método/técnica de desenvolvimento, depende da natureza do sistema a ser construído e tem que se ter em conta o recurso temporal voltado ao negócio e à tecnologia de informação.

Pela sua característica, o modelo proposto exige uma grande participação dos intervenientes, pois lida com novas tecnologias tanto do ponto de vista do negócio quanto da própria informática.

No caso do recurso temporal, com a prototipificação, verifica-se um enorme ganho do ponto de vista da materialização e da obtenção de algo "palpável" e funcional, mesmo que de forma precária. Esse aspecto é muito importante pois muitos intervenientes no desenvolvimento de sistemas só se dão conta do que realmente está a ser objecto de discussão, através de exemplos práticos.

No modelo em causa, o método/técnica usada foi a prototipificação, pelas razões acima apresentadas. A *figura 6.4* ilustra as fases do ciclo de vida da prototipificação.

O processo de prototipificação descrito na *figura 6.4* possui quatro fases que são:

- Levantamento dos requisitos básicos
- Desenvolvimento do protótipo
- Uso do protótipo
- Revisão e melhoria do protótipo

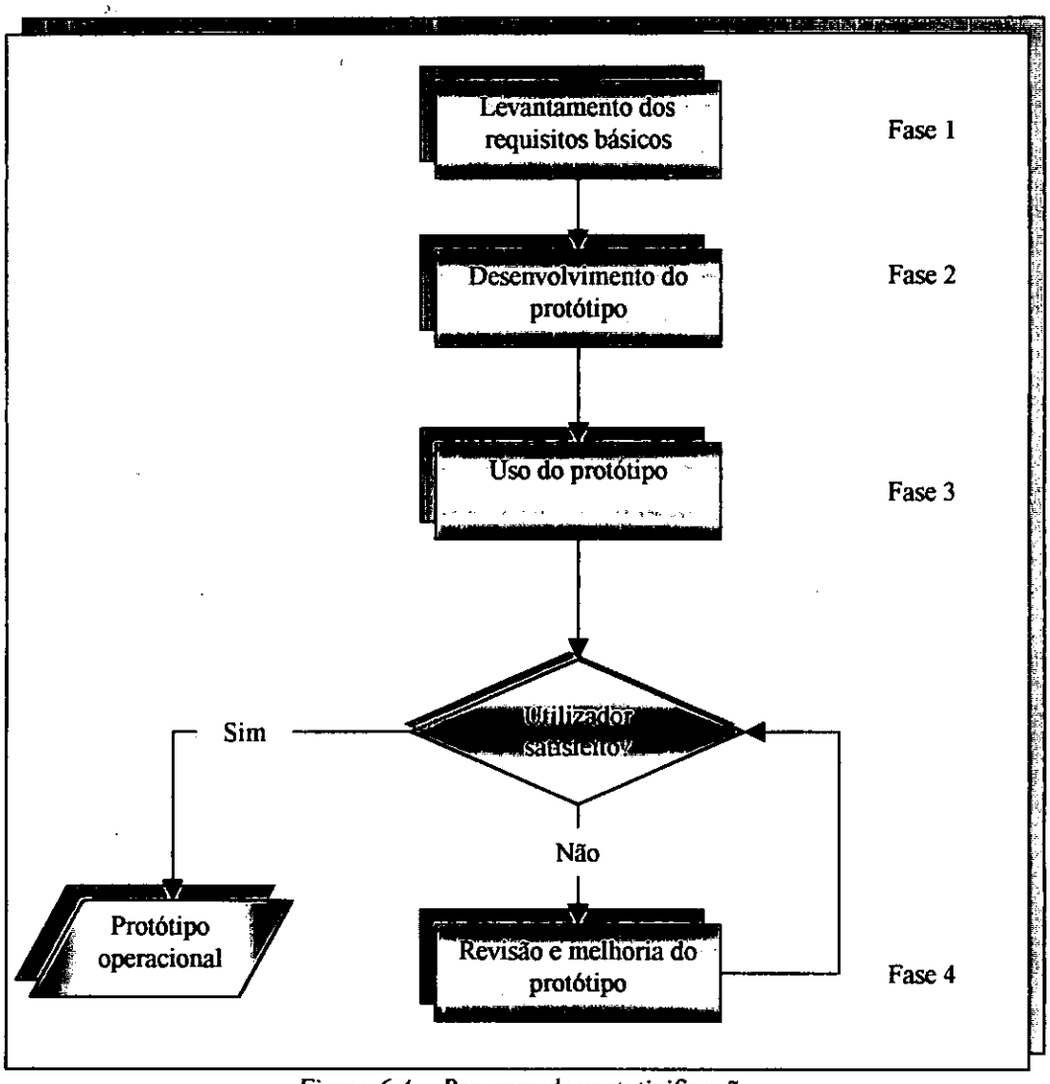


Figura 6.4 - Processo de prototipificação

1ª Identificação de requisitos básicos

Nesta fase, trabalha-se com o utilizador o suficiente para capturar a informação básica sobre os seus requisitos. Estes requisitos podem ser obtidos através das técnicas de recolha de dados. No caso vertente, a técnica usada foi a entrevista não estruturada com os intervenientes do actual sistema. Dessas entrevistas apuraram-se como requisitos os seguintes:

- Disponibilizar ao cliente informação apenas do seu interesse, ou seja, informação sobre as fichas de obras abertas por ele e não por outros clientes;

- Possibilitar ao cliente o acompanhamento do desenvolvimento das fichas de obra, facultando-lhe informação detalhada sobre esta;

2ª Desenvolvimento do protótipo

O desenvolvimento do protótipo foi feito com base na tecnologia *Active Server Page*, que, como foi visto no capítulo IV, possui objectos e componentes que permitem manipular a informação da base de dados. Para o efeito, foram utilizadas as seguintes ferramentas:

- *Internet Information Server* 5.0(IIS 5.0), que vem incorporado no sistema operativo Windows 2000;
- *Microsoft Visual InterDev*;
- Base de dados usada pelo Sistema de Gestão de Fichas de Obra;
- *Internet Explorer 5.0.*

O protótipo construído, contém apenas as funções importantes do modelo proposto. É composto por um conjuntos de páginas *Web* que permitem ao cliente consultar informação disponível na base de dados..

O primeiro interface do protótipo, faz uma descrição geral do sistema, e fornece orientação sobre como aceder à informação disponível, ou obter permissão para tal. Em seguida, é solicitado o *username(*nome de utilizador*)* e a *password*(senha de entrada*)* do utilizador. Estes dados são fornecidos na recepção da empresa. Se o cliente tiver introduzido correctamente os dados acima, terá acesso aos restantes interfaces (páginas) do protótipo. Estes interfaces permitem, ao cliente, consultar informação sobre: historial das suas fichas de obra, as fichas de obra concluídas, por concluir e os detalhes de cada uma delas. Estes detalhes incluem: estado, relatório do técnico, intervenções efectuadas sobre a ficha entre outros. De referir, que para garantir que o cliente tenha acesso a sua informação, fez-se uso do *objecto Session* (um dos objectos internos da tecnologia ASP). A comunicação(ligação) entre as páginas no protótipo é ilustrada na *figura 6.5.*

A validação do utilizador e consulta de informação é feita com base nos dados armazenados na base de dados da empresa. No *anexo H figura H2*, está ilustrado o esquema de tabelas usado no protótipo.

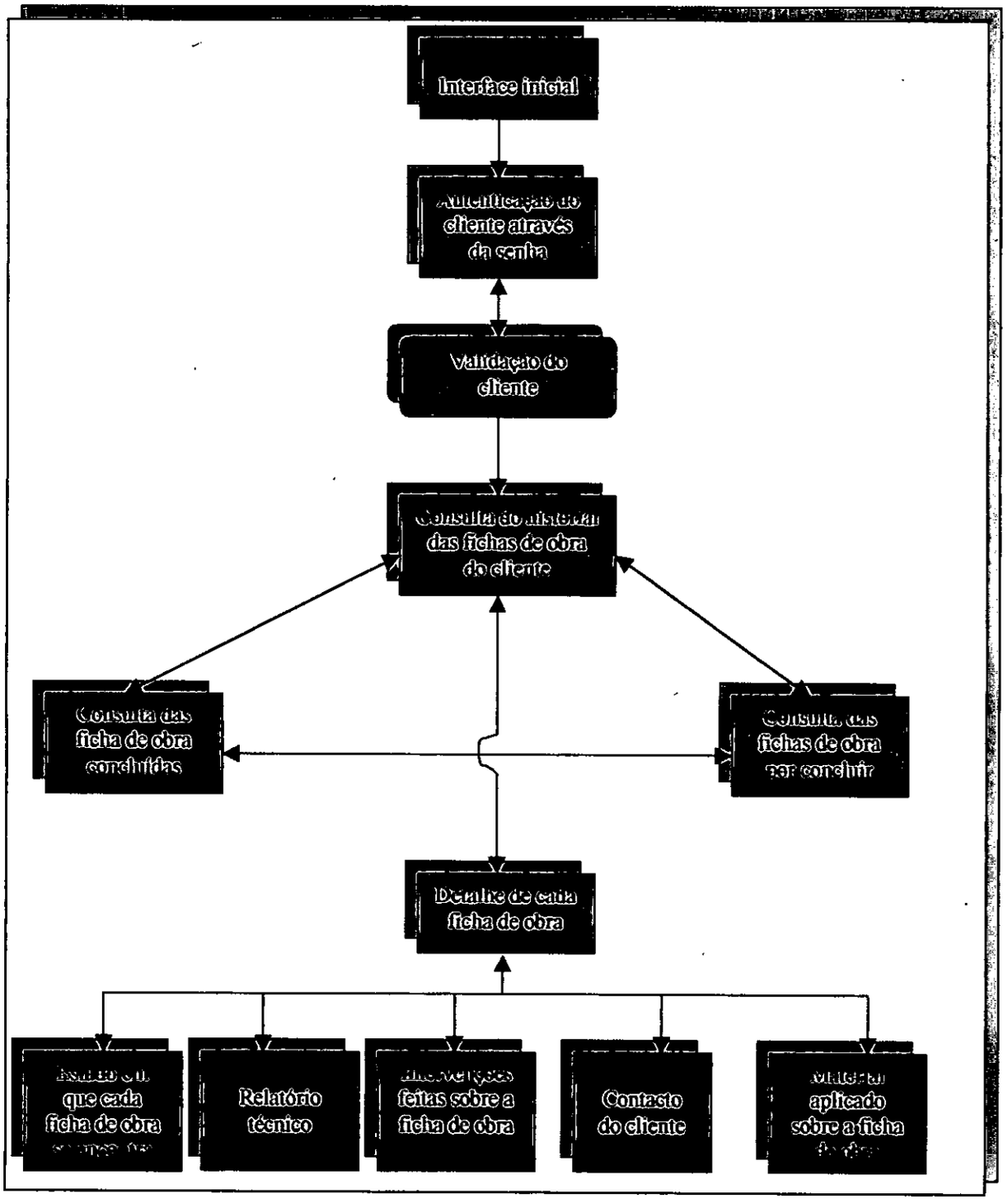


Figura 6.5 - Disposição das páginas no protótipo

As ligações da *figura 6.5*, ilustram para onde o utilizador pode ser direccionado, no caso de se encontrar num determinado interfaces.

3ª Uso do protótipo

Nesta fase, o utilizador é encorajado a trabalhar com o protótipo para, determinar até que ponto o protótipo vai de encontro às suas necessidades, e fazer sugestões para a sua melhoria. Se o protótipo satisfizer as necessidades do utilizador, torna-se no protótipo operacional que fornece as especificações finais para a aplicação.

4ª Revisão e melhoria do protótipo

De acordo com as sugestões feitas pelo utilizador, é feito o aperfeiçoamento do protótipo. Após isso, retorna-se à 3ª fase. Este ciclo é repetido até que o utilizador se sinta satisfeito.

6.4 Implementação do modelo

Após análise exaustiva do protótipo, pelos utilizadores, é hora de passar à implementação do modelo. Este (modelo), foi implementado num servidor que possui como sistema operativo o *Windows 2000* e como servidor *Web* o *Internet Information Server 5.0(IIS 5.0)*.

A escolha do Windows2000 como sistema operativo do servidor deve-se ao facto de este possuir maior segurança no acesso à informação que as demais plataformas Windows.

O servidor está incorporado na rede interna da Exi, que se encontra protegida, contra invasões vindas da Internet, pelo mecanismo de segurança *Firewall*.

6.4.1 Alguns interfaces do modelo

Ao aceder o site, a partir de qualquer browser, o utilizador terá como primeiro interface a *figura 6.5.* Este interface, contém informação geral sobre o que o utilizador encontrará no site, e o que fazer para obter permissão de acesso, caso não a tenha.

a) ***Interface Inicial***

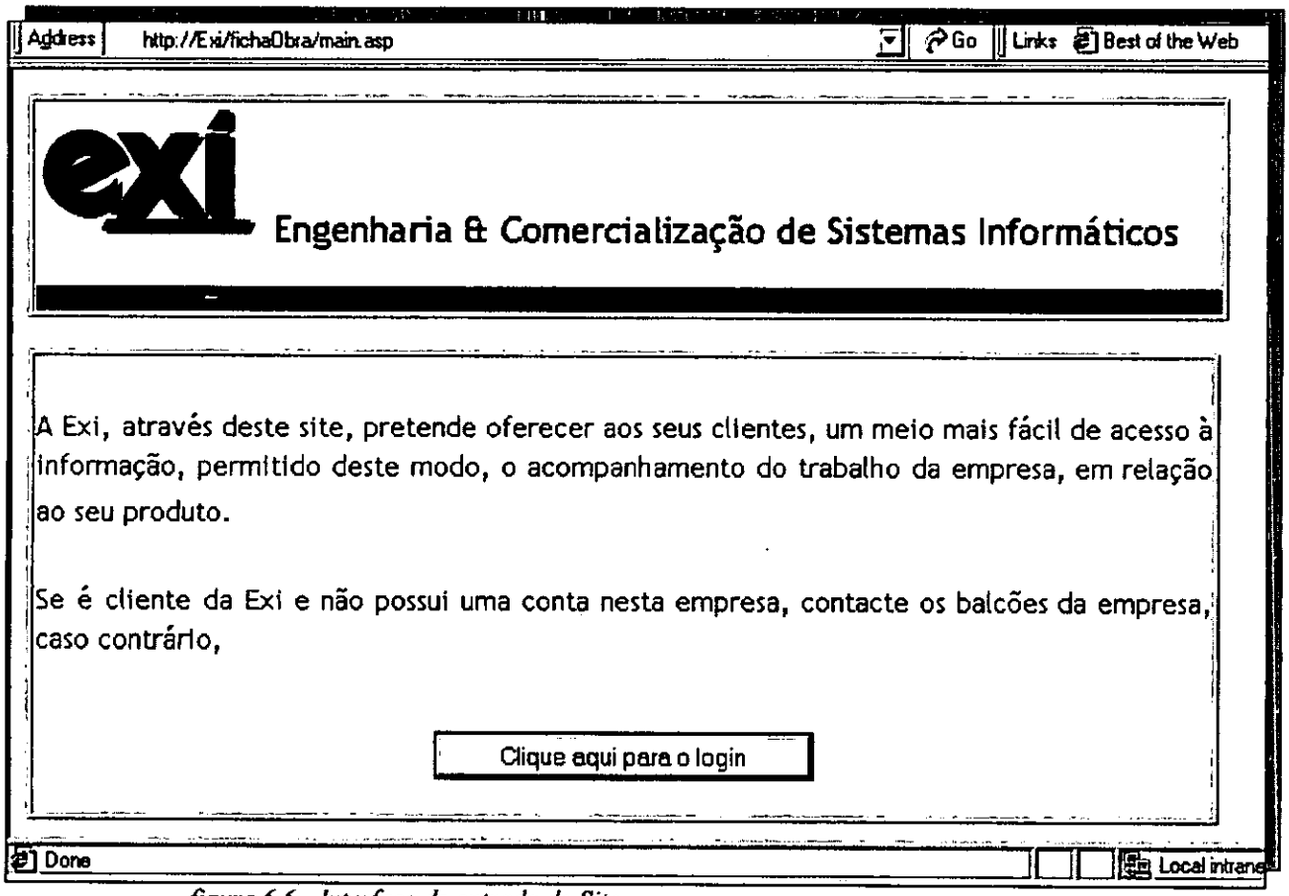


figura 6.6 - Interface de entrada do Site.

Ao fazer clique sobre o botão que se encontra no interface da figura 6.6, o utilizador é encaminhado para o interface de autenticação. *Figura 6.7.*

b) ***Autenticação do utilizador***

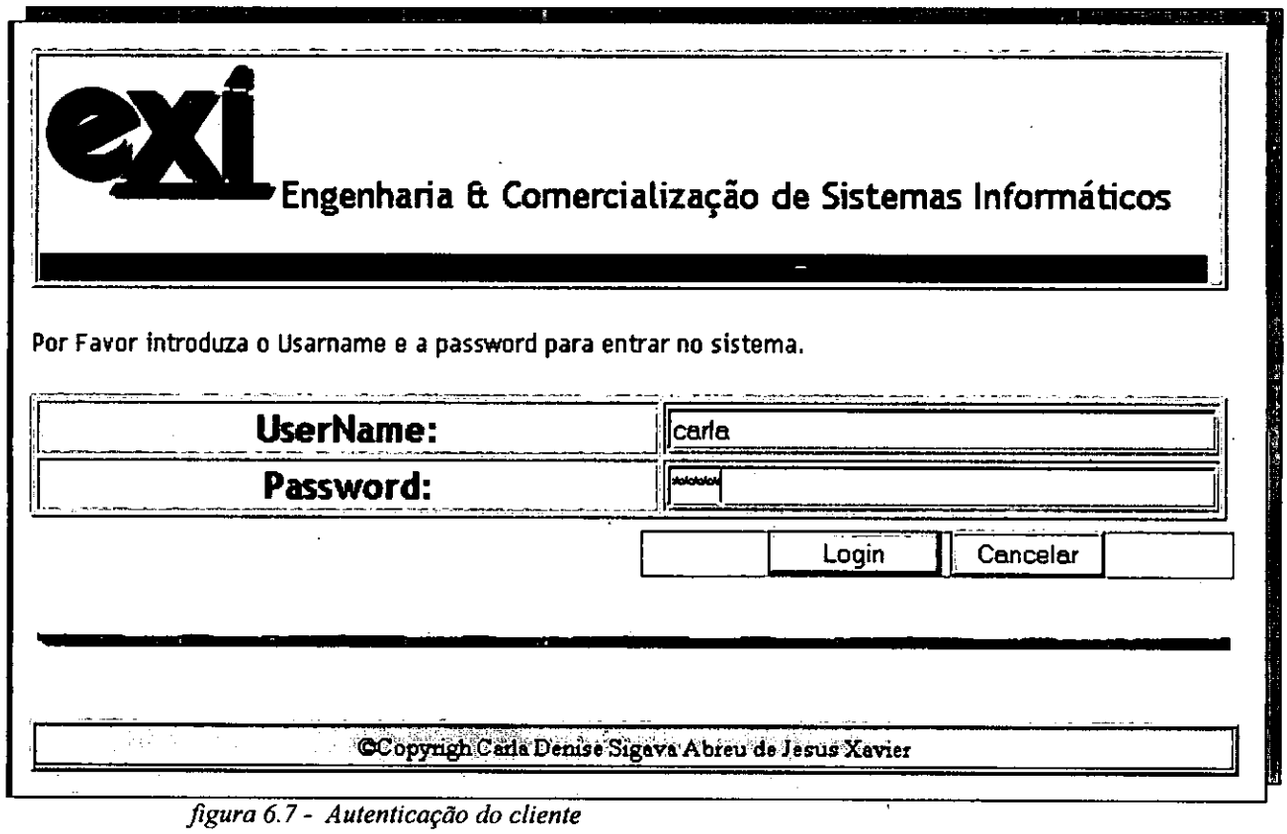


figura 6.7 - Autenticação do cliente

Neste interface, é solicitado ao utilizador que introduza o seu *username* (nome do utilizador) e a respectiva senha. Se tiver introduzido correctamente, ele é encaminhado para o interface que contém informação sobre as fichas de obra por ele abertas, veja *figura 6.8*, caso contrário, é pedido ao utilizador que introduza novamente o nome do utilizador e a respectiva senha.

c) ***Historial de Fichas de obra***

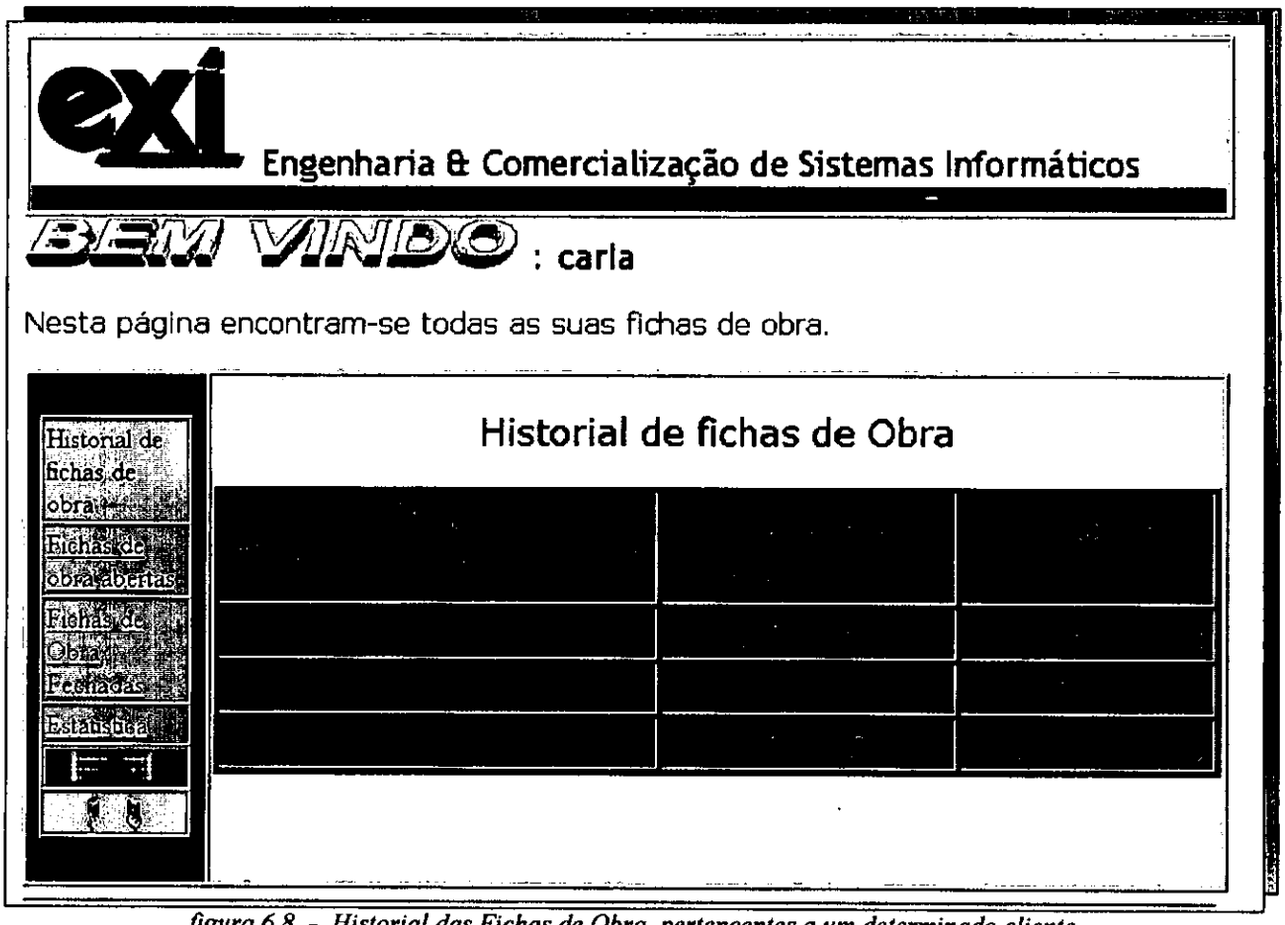


figura 6.8 - Historial das Fichas de Obra, pertencentes a um determinado cliente

Este interface contém informação sobre todas fichas de obra pertencentes a um determinado cliente. A partir deste interface pode-se ter acesso à informação referente às fichas de obra abertas e as que já foram concluídas. Pode-se também ter acesso à informação detalhada de uma determinada ficha de obra, ao fazer clique sobre o seu número.

As funções dos *icons* do interface possuem a seguinte função:

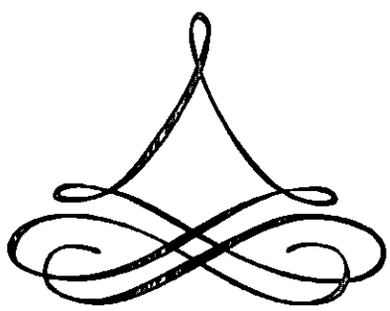
 - é uma porta que permite ao utilizador, retirar-se da aplicação.

 - Este *icon* permite ao utilizador enviar *e-mail* para a empresa.

Para mais detalhes sobre o código usado para a construção do modelo, consulte o *anexo1.*

Capítulo VII

Conclusões e Recomendações



7. Conclusões e Recomendações

7.1 Conclusões

Do trabalho realizado, pode-se concluir o seguinte:

- A evolução da Internet, partindo das páginas estáticas às dinâmicas, tem feito com que este meio seja usado para reformular as relações entre as empresas e os seus clientes. Com efeito, existem tecnologias de desenvolvimento de páginas dinâmicas e interactivas que permitem que o cliente aceda à aplicações residentes em uma determinada empresa e consulte informação exclusivamente do seu interesse, independentemente da sua localização geográfica. Estas tecnologias estão divididas em *Server Side* e *Client Side*. As tecnologias *Server Side* possuem *scripts* que são processados no servidor, e as *Client Side* são processados no cliente;

- A escolha destas tecnologias, para a sua posterior implementação, deve ser feita de acordo com as infra-estruturas existentes na empresa, de modo que a sua implementação não acarrete custos inesperados, tais como de formação, manutenção e segurança. Por esta razão, das várias tecnologias existentes, foi escolhida a tecnologia *Active Server Page*;

- *Active Server Page (ASP)* é uma tecnologia, *Server Side* executada em servidores *Windows*, e permite o processamento de *scripts* no servidor e consequente geração de páginas em *HTML*.

- ASP possui objectos e componentes que auxiliam no manuseamento da informação, que passa do cliente para o servidor e vice-versa, permitindo o acesso aos mais variados tipos de bases de dados através de: *Open Data Base Connectivity* (ODBC) e *Object Linking and Embedding Database* (OLEDB);

- A segurança de uma rede ligada à Internet pode ser garantida através do *protocolo SSL*, que faz uso de algoritmos de criptografia e de certificados digitais, e também do mecanismo de segurança *firewall*, que a protege de invasões através Internet. Por outro lado, a tecnologia *ASP*, pelo facto de ser uma tecnologia *Server Side*, possui uma certa segurança da informação, pois o código não é visível para o utilizador;

- A técnica de prototipificação contribuiu para a redução de tempo e custos de produção, durante o desenvolvimento do modelo;

- O modelo foi implementado e testado internamente. Deste teste verificou-se que o modelo responde favoravelmente aos requisitos, permitindo a um determinado utilizador consultar informação de seu interesse;

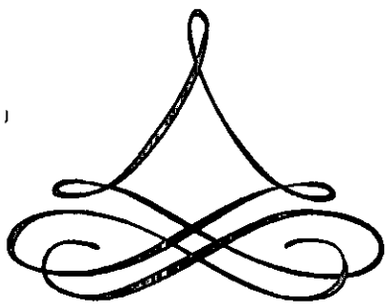
- A implementação deste modelo, na empresa, permitiu solucionar a dificuldade de comunicação existente entre a empresa e os clientes, possibilitando a obtenção da informação em tempo útil, para todos os intervenientes no processo decisório.

7.2 Recomendações

1. Efectuar questionários aos técnicos e clientes para avaliar os benefícios do modelo.
2. O desenho do interface do modelo não teve o envolvimento de especialistas na área de desenho de páginas. Recomenda-se que se efectue uma consulta aos mesmos, de modo a melhorar os interfaces. Esta melhoria teria como objectivo, cativar o cliente no uso da aplicação.

Capítulo VII

Bibliografia



9. Bibliografia

9.1 Bibliografia Referenciada

[Barreto, 2000] Barreto, A.(2000) *ASP Active Server Page*, brasil, http://www.tol.pro.com.br

[Branski, 1998] Branski, R. M.(1998), *Estratégias de negócios na internet*, Branski@obelix.unicamp.br

[Busel et al, 2000] Buser, D., J. Kauffman, J. T. Liibre, B. Francis, D. Sussman, C. Ullman, J. Duckett (2000) *Beginning Active Server Pages 3.0*, 2° edição, 1198 páginas, USA, Wrox Press;

[Laudon e Laudon, 1998] Laudon, K. e J. Laudon (1998). *Management Information Systems*, 5° edição, 693 páginas, New Jersey, Prentice Hall.

[Lúcio, 1999] Lúcio, L., S. Vilar, L. Campos (1999), *Programação em Visual Basic 6*, 3ª edição, 445 páginas, Portugal, FCA

[Pereira, 2001] Pereira, D. W. R.(2001) *Programação Internet – ASP*, http://www.asparena.eti.br

[Reiff,2001] Reiff F. (2001), *Migration from character – based interfaces to web – baed interface*, Lancaster University, United Kingdom

[Santos, 2000] Santos, L. A. L (2000), *Curso de Active Server Pages*, Universidade Federal do Piaui

[Silva,1999] Silva, E. (1999), *Estratégia de tecnologia de informação Transformação de organização*, http://www.gdysafety.com.br/erp.htm .

[Albino, 2000] Albino, A. (2000), Programação Web, http://www.clikando.com.br/coluna/progweb.htm

Puttini [1] Sousa, R.T., R.S. Puttini, *Tutorial*, http://www.rede.unb.br/security/ssleay/ssleay.htm

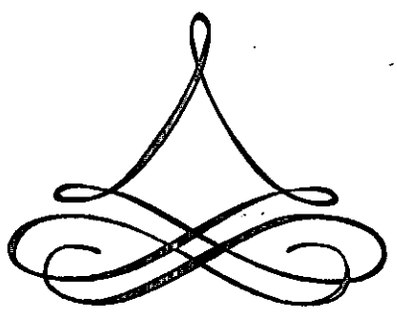
Puttini [2]	Sousa, R.T., R.S. Puttini, *Cenario Actual*, http://www.rede.unb.br/security/ssl3/ssl3.htm
Puttini [3]	Sousa, R.T., R.S. Puttini, *Autenticidade*, http://www.rede.unb.br/security/Introdução/Introdução.htm
[URL1]	*Redes e protocolos de comunicação*, http://www.fe.up.pt/
[Werner,1998]	Werner, J.A.V, (1998) *Curso REDES DE COMPUTADORES - Internet e arquitetura TCP/IP* - PUC Rio/CCE

9.2 Bibliografia utilizada

- http://www.tol.pro.br (2000), *Cartilha de segurança na Internet* consultada à 25.02.2002
- Sobreira, F.C (1998), *Apostila de Internet*, http://www.sidenet.com.br
- Flevry, *A. Criptografia: Criptus. Grafos- uma questão de segurança Nacional*, http://www.jseg.net/criptografia.html
- Gates, B.(1999), *Negócios @ velocidade do pensamento*, Editora Temas e Debates, Lisboa, Portugal.
- Foquisato, M.C. (1998), *Segurança de Sistemas e internet firewalls*, Unicamp
- Almeida, A. P. (1997), *Trabalho de redes - Internet*, Universidade Veiga de Almeida
- Soares, L.F.G, L.Guido e S. Colcher (1995) *Redes de Computadores, Das LANS MANS e WANS às Redes ATM*, 2º edição, editora Campus, 705pp
- Date, C. J.(1989), *Introdução ao sistema de Bancos de dados*, Editora Campos, Rio de Janeiro, Brasil.
- Homer A, A. Enfield, C. Jakob, B. Hartwell, D. Gill, B. Francis, R. Harrison (1997), *Active Server Pages Professional*, Wrox Press, 596 páginas.
- http://www.esperanto.org.nz/jsp/jspfaq.html, consultada à 15 de Abril de 2002

- http://www.php3.net, consultada à 02 de Março de 2002
- http://www.allaire.com, consultada à 25 de Abril de 2002
- http://www.allaire.com/products/coldfusion, consultada à 25 de Abril de 2002

Anexos



Índice de Anexos

Anexo A

A.1. Protocolo TCP/IP

TCP/IP é um acrónimo para o termo *Transmission Control Protocol/Internet Protocol*, ou seja é um conjunto de protocolos, onde dois dos mais importantes (o IP e o TCP) deram os nomes à arquitectura.

O TCP/IP é um protocolo estruturado por camadas, estando a camada TCP(*Transmission Control Protocol*) acima da camada IP(*Internet Protocol*). A camada TCP gere o envio das mensagens ou ficheiros, necessitando por vezes de dividi-los em vários pacotes de tamanho apropriado. No receptor, a aplicação que implementa a camada TCP será a responsável pela sua reconstrução.

A camada IP tem a responsabilidade de fazer chegar o pacote ao endereço IP de destino, o pacote é-lhe entregue pela camada TCP juntamente com o endereço do computador a que se destina. Durante a transmissão, o pacote passará eventualmente por vários sistemas, onde eventualmente será verificado o endereço do destinatário de forma a rotear o melhor possível o trajecto[URL1].

A arquitectura TCP/IP, realiza a divisão de funções do sistema de comunicação em estruturas de camadas. Em TCP/IP as camadas são:

- ✓ Aplicação
- ✓ Transporte
- ✓ Inter-Rede
- ✓ Rede

A *figura A.1* ilustra a divisão em camadas da arquitectura TCP/IP:

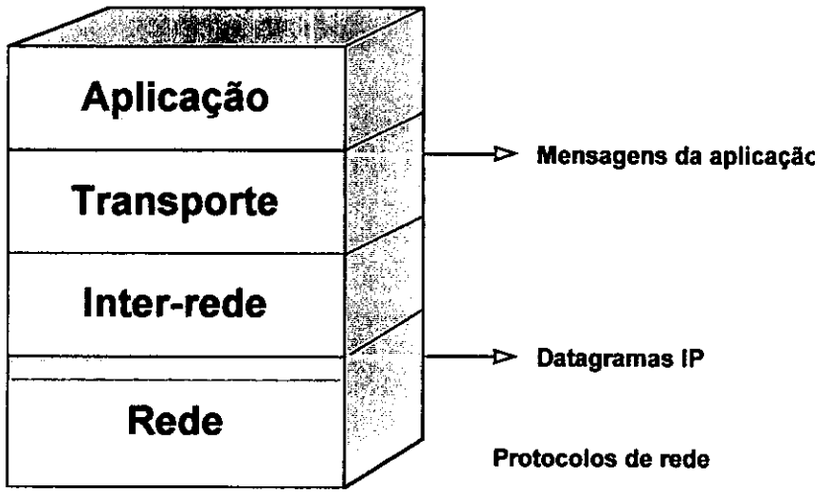


Figura A.1 Arquitectura TCP/IP

A.1.1 Camada de rede

A camada de rede é responsável pelo envio de datagramas construídos pela camada Inter-Rede. Esta camada realiza também o mapeamento entre um endereço de identificação de nível Inter-rede para um endereço físico ou lógico do nível de Rede[Werner,1998].

Os protocolos deste nível possuem um esquema de identificação das máquinas interligadas por este protocolo. Por exemplo, cada máquina situada em uma rede, possui um identificador único chamado endereço *MAC* ou endereço físico que permite distinguir uma máquina de outra, possibilitando o envio de mensagens específicas para cada uma delas.

As redes ponto-a-ponto, formadas pela interligação entre duas máquinas não possuem, geralmente, um endereçamento de nível de rede (modelo TCP/IP), uma vez que não há necessidade de identificar várias estações.

A.1.2 Camada Inter-Rede

Esta camada realiza a comunicação entre máquinas vizinhas através do protocolo IP. Para identificar cada máquina e a própria rede onde estas estão situadas, é definido um identificador, chamado endereço IP, que é independente de outras formas de endereçamento que possam existir nos níveis inferiores. No caso de existir endereçamento nos níveis inferiores é realizado um mapeamento para possibilitar a conversão de um endereço IP em um endereço deste nível.

Os protocolos existentes nesta camada são:

Protocolo de transporte de dados: IP - *Internet Protocol*

Protocolo de controle e erro: ICMP - *Internet Control Message Protocol*

Protocolo de controle de grupo de endereços: IGMP - *Internet Group Management Protocol*

Protocolos de controle de informações de roteamento

O protocolo IP realiza a função mais importante desta camada que é a própria comunicação inter-redes. Para isto ele realiza a função de **roteamento** que consiste no transporte de mensagens entre redes e na decisão de qual rota uma mensagem deve seguir através da estrutura de rede para chegar ao destino.

O protocolo IP utiliza a própria estrutura de rede dos níveis inferiores para entregar uma mensagem destinada a uma máquina que está situada na mesma rede que a máquina de origem. Por outro lado, para enviar mensagem para máquinas situadas em redes distintas, ele utiliza a função de roteamento IP. Isto ocorre através do envio da mensagem para uma máquina que executa a função de roteador. Esta, por sua vez, repassa a mensagem para o destino ou a repassa para outros roteadores até chegar no destino, veja a *figura A2.*

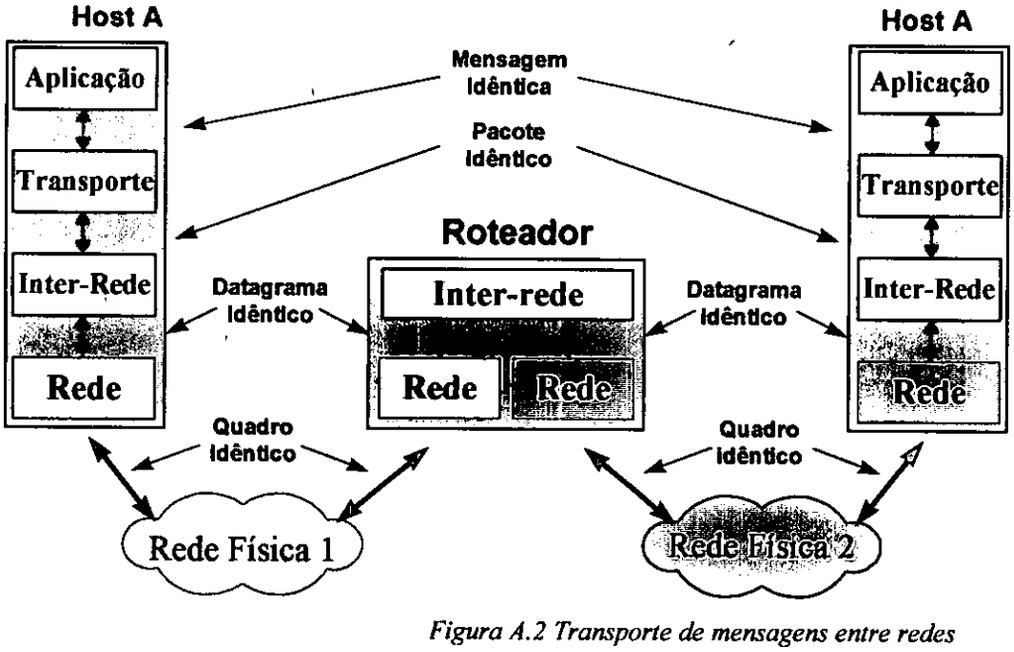


Figura A.2 Transporte de mensagens entre redes

A.1.3 Camada de Transporte

Esta camada reúne os protocolos que realizam as funções de transporte de dados fim-a-fim, ou seja, considerando apenas a origem e o destino da comunicação, sem se preocupar com os elementos

intermediários. A camada de transporte possui dois protocolos que são o *UDP* (*User Datagram Protocol*) e *TCP* (*Transmission Control Protocol*).

O protocolo *UDP* realiza apenas a multiplexação para que várias aplicações possam aceder o sistema de comunicação de forma coerente.

O protocolo *TCP* realiza, além da multiplexação, uma série de funções para tornar a comunicação entre origem e destino mais confiável. São responsabilidades do protocolo *TCP*: o controle de fluxo, o controle de erro, a sequência e a multiplexação de mensagens.

A camada de transporte oferece para o nível de aplicação um conjunto de funções e procedimentos para acesso ao sistema de comunicação de modo a permitir a criação e a utilização de aplicações de forma independente da implementação.

A.1.4 Camada de Aplicação

A camada de aplicação reúne os protocolos que fornecem serviços de comunicação ao sistema ou ao utilizador. Pode-se separar os protocolos de aplicação em protocolos de serviços básicos ou protocolos de serviços para o utilizador:

Protocolos de serviços básicos, que fornecem serviços para atender as próprias necessidades do sistema de comunicação *TCP/IP: DNS, BOOTP, DHCP*

Protocolos de serviços para o usuário: *FTP, HTTP, Telnet, ICQ,* e outros

Anexo B

B.1. Comunicação entre o Servidor Web e o Browser

Quando o utilizador faz o pedido de uma página, o browser empacota a instrução usando o protocolo *Transmission Control Protocol* (TCP). Antes dos pacotes serem enviados para a rede, eles precisam de ser endereçados, por isso o protocolo *Hyper Text Transfer Protocol* (HTTP) coloca um campo de endereço no pacote [Busel et al, 2000].

A mensagem que passa de um *browser* para o Servidor *Web* é denominada de *HTTP request*. Quando o Servidor *Web* recebe esta solicitação, ele procura a página solicitada, caso a encontre, empacota-a[26], endereça o pacote ao *browser*[27] e envia-o de volta. Se não a encontrar, ele emite uma página de erro, empacota-a e envia-a ao browser. A mensagem enviada do Servidor *Web* para o *browser* é denominada *HTTP response*. O processo de comunicação pode ser ilustrado com maior detalhe na *figura B.1*.

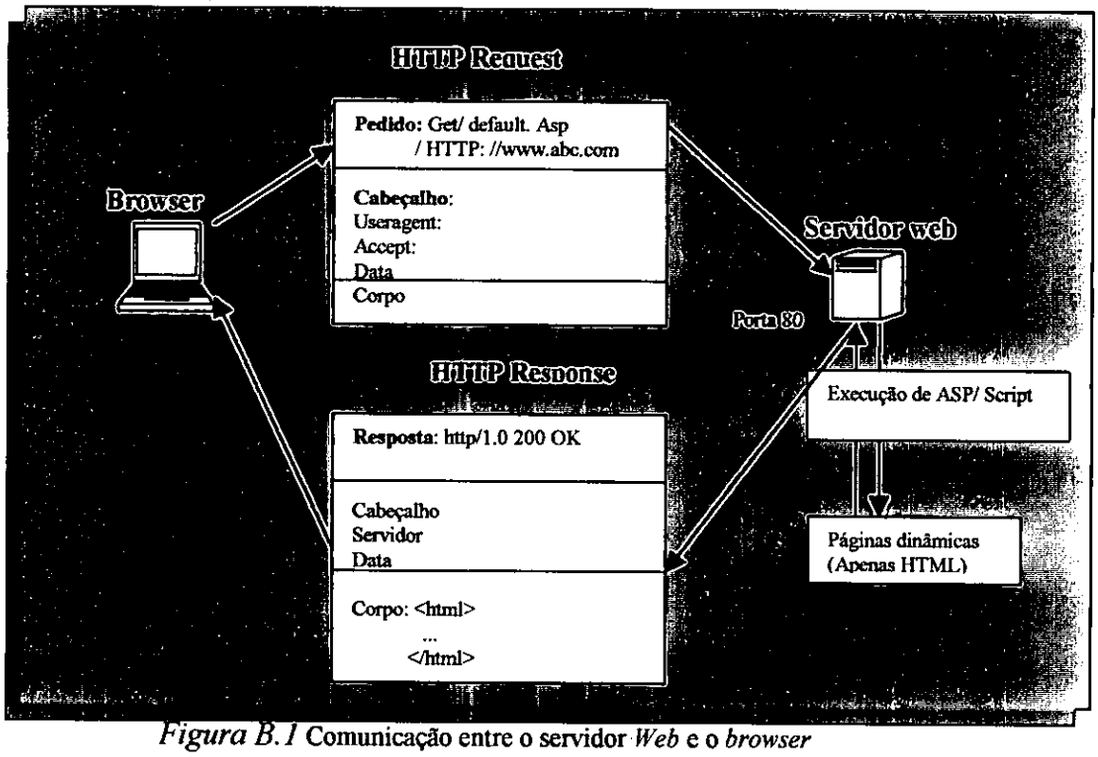


Figura B.1 Comunicação entre o servidor *Web* e o *browser*

B.1.1 HTTP Request

É enviado pelo browser ao servidor e contém o seguinte:

- *A linha do Pedido(Request line)*: que contém três peças de informação

  - Um comando HTTP, conhecido como método;
  - O URL do ficheiro que o cliente está a requisitar;
  - O número da versão do HTTP.

Por exemplo:

GET /testpage.htm HTTP/1.1

- *O cabeçalho do HTTP(Header): o* próximo *bit* da informação enviada é o cabeçalho(*header*). Que contém detalhes de informação sobre que tipo de documento o cliente aceitará de volta, o tipo de *browser* que requisita a página, a data, e informações gerais sobre configuração.

- *O corpo do HTTP(body)*: se o método *post* tiver sido usado na linha do pedido, então o corpo do HTTP conterá qualquer dado que estiver a ser enviado para o servidor, caso contrário o corpo do HTTP estará vazio.

B.1.2 HTTP Response

É enviado, pelo servidor, de volta ao browser do cliente, e contém o seguinte:

- *A linha de resposta(Response line):* Contém apenas dois *bits* de informação

  - O número da versão do HTTP;
  - Um código do *HTTP Request*, que reporta o sucesso ou a falha do pedido.

Por exemplo:

[26] Usando o *Transmission Control Protocol* (TCP)
[27] Usando o *Hyper Text Transfer Protocol* (ou HTTP)

HTTP/1.0 200 OK

- *Cabeçalho do HTTP:* é semelhante ao cabeçalho do *HTTP Request*

- *Corpo do HTTP:* Se o pedido tiver tido sucesso, então o corpo do HTTP conterá o código HTML (Juntamente com todos os *cripts* que serão executados pelo *browser*), pronto a ser interpretado pelo *browser*.

Anexo C

C.1. Páginas Estáticas e Dinâmicas

C.1.1 Páginas Estáticas

Páginas Estáticas são páginas que contém HTML introduzido directamente num editor de texto e gravado em um ficheiro com extensão *.htm ou *.html. Numa página estática, *veja a figura C.1.1*, a aparência da página é sempre a mesma e a comunicação é feita da seguinte forma:

1. O autor escreve uma página composta por HTML puro, e grava num ficheiro com extensão *.htm;
2. Mais tarde o utilizador requisita a página ao Servidor;
3. O servidor localiza a página *.htm;
4. O servidor envia a sequência HTML de volta ao *browser*;

O *browser* processa o HTML e edita a página.

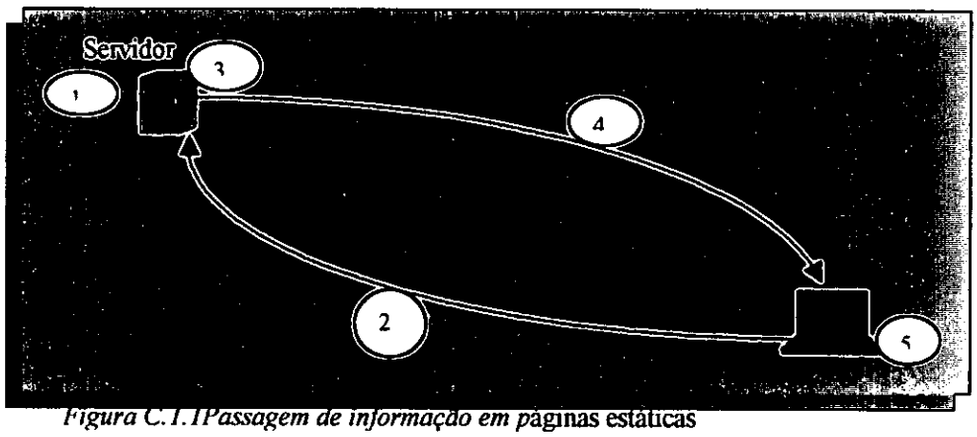


Figura C.1.1Passagem de informação em páginas estáticas

C.1.2 Páginas Dinâmicas

Páginas dinâmicas são geradas dinamicamente durante o pedido e a comunicação é feita da seguinte forma, veja a *figura C.1.2*:

1. Um autor de páginas escreve um conjunto de instruções para a criação do HTML, e grava estas instruções dentro de um ficheiro;

2. Mais tarde um utilizador faz o pedido da página a partir do seu *browser*, o pedido é passado *do browser* para o servidor;

3. O servidor localiza o ficheiro de instruções;

4. O servidor envia o recém criado HTML de volta ao *browser*;

5. *O browser* processa o HTML e edita a página.

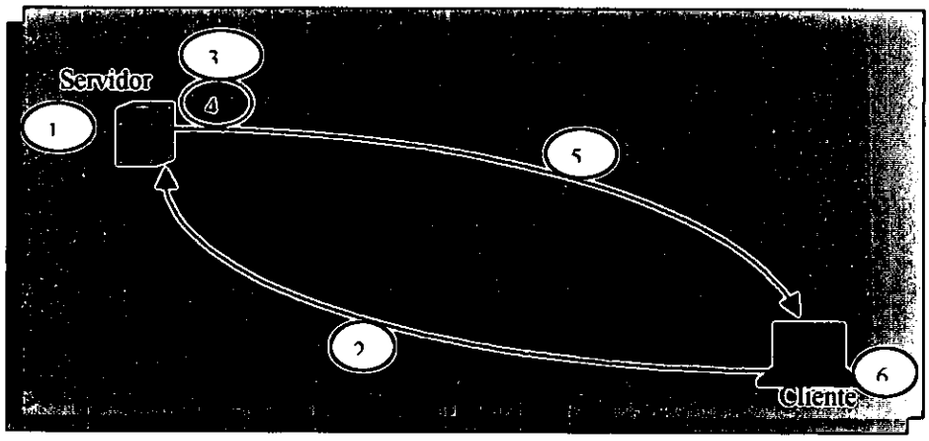


Fig C.1.2 Passagem de informação em páginas dinâmicas

O processo de servir uma página dinâmica é um bocado diferente do processo de servir uma página estática. Mas a diferença é crucial, pois o HTML que define a página não é gerado até que a página seja requisitada.

As páginas dinâmicas permitem capturar informação, que até a altura da escrita do código não nos é conhecida, tal como:

- Identidade do utilizador e preferências pessoais;

- O tipo de *browser*;

- Informações providenciadas pelo pedido do utilizador;

- Informações contidas em bases de dados, etc.

Anexo D

D.1. Objectos ASP

O Diagrama seguinte mostra, conceptualmente, a relação dos objectos com o servidor e o cliente, as requisições feitas pelo cliente e as respostas dadas pelo servidor.

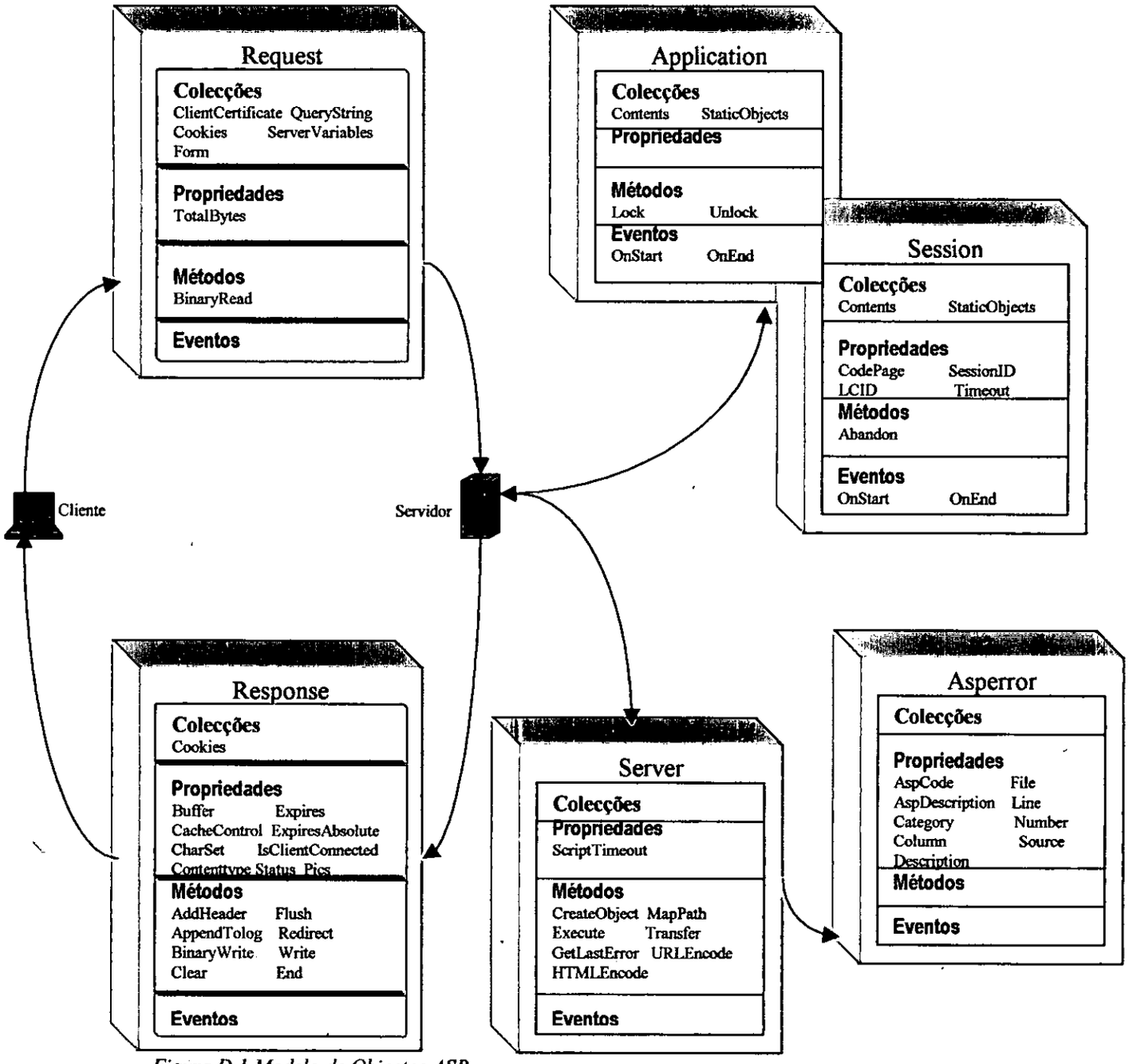


Figura D.1 Modelo de Objectos ASP

D.1.1 Objecto Application

O objecto *Application* é criado quando o código *ASP* é carregado em resposta ao primeiro pedido de uma página ASP. Ele providencia um repositório para armazenamento de variáveis e referências à objectos que estarão disponíveis em todas as páginas abertas pelos visitantes..

Colecção	**Descrição**
Contents	Uma colecção de todas as variáveis e seus valores que são armazenadas no objecto Application, e não são definidas usando o elemento <OBJECT>
StaticObjects	Uma colecção de todas as variáveis que são armazenadas num objecto Application usando o elemento< OBJECT >

Método	**Descrição**
Content.Remove (nome da variável*)*	Remove uma variável da colecção Application.Contents
Contents.RemoveAll()	Remove todas as variáveis da colecção Application.Contents
Lock()	Usada para garantir que o conteúdo não seja corrompido, não permitindo, deste modo, que dois utilizadores leiam e actualizem os valores simultaneamente.
Unlock()	Liberta as páginas ASP, anteriormente trancadas pelo método lock.

Evento	**Descrição**
OnStart	Ocorre no primeiro instante em que um utilizador requisita uma das páginas da aplicação, antes de a página requisitada ser executada. Usada para inicializar variáveis, criar objectos, ou correr outros códigos.
OnEnd	Ocorre quando a aplicação ASP termina, isto é, quando o Servidor web desliga.

Um evento é uma subrotina automaticamente chamada quando o sistema sofre alguma acção específica.(Santos, 2000) Estas subrotinas são escritas directamente nas páginas ASP, num ficheiro à parte denominado *GLOBAL.ASA*. Cada aplicação tem somente um ficheiro *Global.ASA* e situa-se na raiz do directório virtual.

Quando um directório virtual é acedido pela primeira vez, o servidor web procura, nesse directório, pelo ficheiro. Ao encontrar o ficheiro, ele procura a subrotina *application_OnStart* para executar os

seus comandos. Quando o servidor web desliga-se, ele chama a subrotina application_OnEnd. (Busel et al,2000).

D.1.2 Objecto ASPError

O objecto *ASPError* está disponível através do método *GetLastError* do Objecto *Server*. Ele providencia um conjunto de informação detalhada sobre o último erro ocorrido no ASP.

Propriedades	**Descrição**
ASPCode	*Integer*. Número de erro gerado pelo IIS.
ASPDescription	*Integer*.Uma descrição detalhada do erro, caso se encotre relacionado ao ASP.
Category	*String*. Indica a fonte do erro.Por exemplo, o próprio ASP, a linguagem script ou um objecto.
Column	*Integer*. A posição do caracter dentro do ficheiro que gera o erro.
Description	*String*. Uma breve descrição do erro.
File	*String*. O nome do ficheiro que estava a ser processado quando o erro ocorreu.
Line	*Integer*. O número da linha no ficheiro que gerou o erro.
Number	*Integer*. Código de erro COM padráo
Source	*String*. O código actual da linha que causou o erro.

D.1.3 Objecto Request

O objecto *Request* torna disponível ao *script* toda informação que o cliente providencia quando requisita uma página. Ele inclui, as variáveis do servidor que identifica o *browser* e o utilizador, *cookies* que estão armazenados no *browser*, e qualquer valor anexado ao URL como *QueryString*, ou em controlos HTML. Ele também providencia acesso à qualquer certificado que esteja a ser usado através do ***Secure Sockets Layer (SSL)*** ou outro protocolo de comunicação encriptado, e à propriedades que ajudam na gestão da conexão.

Colecção	**Descrição**
ClientCertificate	Uma colecção de valores de todos os campos ou entradas no certificado do cliente que o utilizador apresenta ao servidor quando acede a uma página ou recurso.
Cookies	Uma colecção de valores de todos os *cookies* enviados do sistema de utilizadores com os pedidos. Apenas *cookies* válidos para o domínio que contém o recurso são enviados para o servidor.
Form	Uma colecção de valores de todos os elementos do controle HTML na secção <FORM>, submetidos como pedido, onde o valor do

	atributo METHOD é POST.
QueryString	Uma colecção de todos os pares Nome/Valor anexado ao URL no pedido do utilizador, ou de todos os elementos do controlo HTML na secção <FORM> submetidos como pedido, onde o valor do METHOD é GET ou o atributo é omitido.
ServerVariables	Uma colecção de todos os valores do cabeçalho HTTP enviados a partir do cliente com o seu pedido, mais os valores de várias variáveis do ambiente servidor.

Propriedade	**Descrição**
TotalByte	*Integer*. Valores apenas de leitura que suportam um número total de *bytes* no pedido enviados pelo cliente

Método	**Descrição**
BinaryRead(Count)	Devolve *bytes (Count)* de dados do pedido do cliente quando os dados são enviados ao servidor como parte de um pedido *POST*. Este método não pode ser usado se o código ASP possuir uma referência para a colecção *Request.form*. Por outro lado a colecção *Request.form* não poderá ser acedida se o método *BinaryRead* estiver a ser usado.

D.1.4 Objecto Response

É usado para aceder às respostas criadas, e que tem que ser enviadas de volta ao cliente. Torna disponível as variáveis HTTP que identificam o servidor, suas capacidades, informações sobre o conteúdo que está a ser enviado ao browser, e qualquer *cookie* novo que será armazenado no *browser*. Providencia também uma série de métodos que são usados para criar a página retornada.

Colecção	**Descrição**
Cookies	Uma colecção contendo valores de todos os *cookies* que serão enviados de volta ao cliente na resposta corrente.

Propriedade	**Descrição**
Buffer = True!false	*Boolean. Read/Write*. Especifica se a saída criada por uma página ASP será mantida no *buffer* do IIS até que todos os *scripts* da página corrente sejam processados, ou até que os métodos *Flush* ou *End* sejam chamados. Tem que ser a primeira linha do

	ficheiro **.asp* depois da declaração <%@Language = ...%>.
CacheControl *" Setting "*	*String. Read/ write.* É colocada em "*Public*" para permitir que servidores proxy façam o armazenamento(cache) da página e em "*Private*", para prevenir que tal aconteça.
Charset = *" value "*	*String. Read/write.* Anexa o nome de um conjunto de caracteres ao cabeçalho do *Content-type* HTTP, criado pelo servidor para cada resposta.
CopntentType *" MIME-type "*	O *content type* informa ao browser sobre o tipo de conteúdo à esperar.
Expires minutes	*Number. Read/write.* Especifica o tempo em minutos pelo qual a página ficará válida.
ExpiresAbsolute *#date[time]#*	Date/Time. *Read/write.* Especifica a data e a hora na qual a página irá expirar.
IsClientConnected	*Boolean. Read – Only.* Retorna uma indicação sobre a conecção, ou não, do cliente e carrega a página do servidor. Pode ser usado para terminar um processamento, através do método *Response.End*, se um cliente mover para outra página antes que a página corrente termine a execução.
PICS *(" PICS-label-string ")*	*String. Write - only.* Cria um cabeçalho *PICS* e adiciona-o ao cabeçalho HTTP na resposta. Cabeçalho PICS define o conteúdo da página em termos de violência, sexo, palavrões, etc.
Status = *"código da mensagem"*	*String. Read/write.* Especifica o valor do estado e a mensagem que será enviada ao cliente, no cabeçalho HTTP da resposta, para indicar um erro ou sucesso no processamento da página.

Método	**Descrição**
AddHeader *("Nome", "Conteúdo")*	Cria um cabeçalho HTTP usando os valores de nome e conteúdo, e adiciona-os à resposta. Não irá substituir um cabeçalho existente e com o mesmo nome. Assim que o cabeçalho estiver criado, ele não poderá ser removido. Deve ser usado antes que qualquer conteúdo de uma página seja enviado ao cliente.
AppendToLog*("String")*	Adiciona uma linha no fim da entrada do "*log*" do servidor para um determinado pedido, quando o ficheiro *W3C Extended* está em uso.
BinaryWrite("*SafeArray*")	Escreve o conteúdo de uma variável "*Safe Array*" para a linha de saída do HTTP corrente, sem a conversão de nenhum caracter. Útil para a escrita de informações que não sejam *string*, tais como

	dados binários requisitados por uma aplicação do cliente, ou dados para editar um ficheiro de imagens.
Clear()	Apaga o conteúdo de qualquer página temporariamente armazenada no *buffer* da resposta do IIS quando o *Response.Buffer* for verdadeiro. Não apaga os cabeçalhos HTTP das respostas. Pode ser usada para cancelar uma página parcialmente completa.
End()	Pára o processamento de um *script* da página e retorna o conteúdo recentemente criado, cancela qualquer outro processamento na página.
Flush()	Envia todo o conteúdo das páginas, recentemente armazenadas no *buffer* do IIS, para o cliente quando o *response.Buffer* for verdadeiro. Pode ser usado para enviar partes de uma longa página ao cliente.
Redirect(*"url"*)	Instrui ao browser para carregar a página na linha URL, enviando um cabeçalho HTTP "*302 Object Moved*" na resposta.
Write("*String*")	Escreve uma linha específica na resposta HTTP Corrente e no *buffer* IIS de tal modo que ela se torne parte da página retornada.

D.1.5 Objecto Server

O objecto *Server* providencia uma série de métodos e propriedades que são úteis no *script* ASP. O mais obvio é o método *Server.Createobject*, que instancia outros objectos *COM* dentro do contexto da página corrente ou sessão. Existem também métodos para traduzir *strings* em um formato que possa ser usado em URLs e em HTML.

Property	**Descrição**
ScriptTimeout	*Integer*. Tem por defeito 90 segundos. Coloca ou retorna o número de segundos que o *script* permanece em execução na página antes do servidor abortar a execução da página e reportar um erro. Pára e remove, automaticamente, da memória páginas que contém erro e que podem trancar a execução num *loop*, ou páginas que param enquanto esperam que recurso se torne disponível.

Método	**Descrição**
CreateObject("identificador")	Cria uma instancia do objecto que é identificado pelo "identificador", e retorna uma referência, que pode ser usado no código. Pode ser usado no ficheiro *global.asa* para criar objectos do nível de sessão ou aplicação.
Execute("url")	Pára a execução da página corrente e transfere o

	controle para a página especificada no "*url*". O ambiente do utilizador corrente é carregado para a nova página. Após a execução desta página, o controle passa novamente para a página original e a execução é retomada .
GetLastError()	Retorna uma referência ao objecto *ASPError* que guarda detalhes do último erro ocorrido durante o processamento ASP, isto é, dentro do *asp.dll*. A informação exposta pelo objecto *ASPError* inclui o nome do ficheiro, o número da linha, código do erro, etc.
HTMLEncode("String")	Retorna um *string*, que é a cópia do valor de entrada "*String*", mas com os caracteres '<', '>', '&' e aspas duplas, convertidos em caracteres *html*.
Mappath("url")	Retorna o caminho completo e o nome do ficheiro do ficheiro ou recurso especificado em "*url*"
Transfer("url")	Pára a execução da página corrente e transfere o controle para a página especificada no "*url*". O ambiente do utilizador corrente é carregado para a nova página. Ao contrário do método execute, a execução não é retomada na página original. Ela termina quando a nova página completa a execução
URLEncode("string")	Retorna um *string* que é a cópia do valor de entrada "*string*" mas com todos os caracteres que não são válidos num *url*, tal como, '?', '&', e espaço, convertidos em caracteres válidos por *url*.

D.1.6 Objecto Session

O objecto *Session* é criado para cada visitante quando eles requisitam a primeira página ASP, e permanece disponível até que expira o período *default* de *timeout*. Ele providencia um repositório para o armazenamento de variáveis e referências de objectos que estiverem disponíveis, apenas para as páginas que o visitante abre durante o tempo de vida da sessão.

Colecção	**Descrição**
Contents	Colecção de todas as variáveis e seus valores que são armazenados neste objecto, e não são definidos usando o elemento <objecto>.
StaticObjects	Colecção de todas as variáveis e seus valores que são armazenados neste objecto, usando o elemento <objecto>.

Propriedade	**Descrição**
CodePage	Integer. *Read/write*. Define o código da página que será usado para editar o conteúdo da página no browser. O código da página é um valor numérico de um conjunto de caracteres.
LCID	Integer. *Read/ write*. Define o identificador do local (LCID), da página, que se encontra a ser enviada ao browser. O LCID é uma abreviatura internacional padrão, que identifica de uma maneira única o local. Por

	exemplo, 2057 define o local onde o seguinte símbolo de moeda '£' está. Este LCID pode ser usado em declarações, tais como, *FormatCurrency*, onde existem argumentos LCID opcionais
SessionID	Long. *Read-only*. Retorna o identificador de uma determinada sessão. Esse identificador é gerado pelo servidor quando a sessão é criada.
Timeout	Integer. *Read/write*. Define o período de *timeout*, em minutos, do objecto *session*. Se o utilizador não reagir ou não pedir uma página dentro do período de *timeout*, a sessão termina. Por defeito, o *timeout* é de 20 minutos.

Método	**Descrição**
Contents.Remove("nome da variável")	Remove uma determinada variável da colecção *session.contents*.
Contets.RemoveAll()	Remove todas as variáveis da colecção *session.contents*.
Abandon()	Termina a sessão do utilizador corrente, e destrói o corrente objecto *session*, assim que a execução da página estiver completa. É possível aceder às variáveis de sessão corrente, mesmo depois da chamada do método *Abandon*.

Não é possível remover variáveis da colecção *Session.StaticObjects* durante a execução

Evento	**Descrição**
OnStart	Ocorre quando uma sessão do utilizador ASP inicia, antes de a primeira página pedida pelo utilizador ser executada. Usada para inicializar variáveis, criar objectos, ou correr outros códigos.
OnEnd	Ocorre quando uma sessão do utilizador ASP termina. Isto acontece quando o período predeterminado de *timeout* termina. Todas as variáveis existentes na sessão são destruídas.

Anexo E

E.1 Gestão de directórios no servidor

Quando um utilizador requisita, via HTTP, uma página, o servidor procura pela localização da página. De facto, existe uma relação entre a informação dada no URL, e a informação física (no sistema de ficheiros do servidor) do ficheiro que contém a fonte da página.

Esta relação funciona através da criação de uma segunda estrutura de directórios, no servidor, que reflecte a estrutura do *Web site*. A primeira estrutura de directórios é a que se vê quando se abre o *Windows Explorer* no servidor, estes directórios são chamados de directórios físicos. Por exemplo c:\MyDocuments.

A segunda estrutura de directórios é a que reflecte a estrutura do *Web site*. Esta consiste de uma hierarquia de directórios virtuais. Estes directórios são criados pelo servidor, e são usados para estabelecer a relação com o directório físico.

Um directório virtual é um apelido ou alias de um directório físico existente no servidor. A ideia é que ao requisitar uma página contida no directório físico, o utilizador não precise de usar o nome do directório físico, mas sim o seu apelido.

Anexo F

F.1 Glossário

Active Data Object – Uma tecnologia de acesso aos dados que contém capacidade de acesso à qualquer tipo de dados, tais como, dados em bases de dados, mensagens, etc.

Active Server Pages - Uma linguagem *scripts*, baseada num servidor da Microsoft, que combina HTML e código script num único ficheiro.

ActiveX - Um conjunto de tecnologias que permitem que componentes de *software* interajam entre eles num ambiente de rede, independentemente da linguagem no qual eles foram criados.

Browser - Uma aplicação de cliente conectado à *world wide web* que requisita recursos de um servidor *web*, normalmente com a finalidade de os exibir. Um exemplo de um *browser* é o *Microsoft Internet Explorer,* também chamado um *web browser*.

Certificate Autority – autoridade competente que emite certificados, para encriptação e verificação de uso.

Common Gateway Interface – Permite, aos *scripts* e programas executáveis, o acesso aos pedidos dos utilizadores e às respostas do servido, de modo a criar páginas dinâmicas.

Component Object Model – Uma arquitectura aberta para desenvolvimento de aplicações cliente/servidor baseadas em tecnologia orientada à objectos. Clientes têm acesso a um objecto através de interfaces implementadas no objecto. *COM* é linguagem neutra, de modo que, qualquer linguagem que produza componentes de *ActiveX* também possa produzir aplicações *COM*.

Database Management System – Um programa, que armazena, gere e recebe dados.

Directório Virtual – é um directório criado pelo servidor *Web* e relacionado à um path (caminho) de um directório real no disco. Na requisição de um página, faz-se referência ao directório virtual, e não real, ficando desconhecido, para o utilizador, a real localização do ficheiro.

File Transfer Protocol – um protocolo de Internet, usado para a transferência de ficheiros na Internet.

Hypertext Markup Language – uma forma de inserção de *tags* em página, para enriquecer a formatação, o conteúdo e outras informações

Hypertext Tranfer Protocol – um protocolo que corre em IP e que foi desenhado pela *World Wide Web*. Providencia o empacotamento de informações que contém cabeçalhos de instruções e outros dados sobre conteúdo.

Internet Information Server – um *software* de servidor *Web*, incluso no *Windows NT*.

Internet Protocol – O nível baixo do protocolo TCP/IP. Junta os pacotes do TCP, adiciona a informação do endereço e os envia para a rede.

Java server Pages – Linguagem de programação do lado do lado do servidor que combina HTML e o código *Java* para gerar páginas dinâmicas.

Java Server-based executable – um programa executável, escrito em *java* que corre num servidor *web* em resposta à um pedido.

Object Linking and Embedding Database – novo interface de acesso aos dados, desenhado pela Microsoft para substituir ODBC e providenciar uma larga cobertura a diferentes tipos de dados.

Open Database Connectivity – permite o acesso a qualquer tipo de dado.

Personal Home Page – Plataforma do lado do servidor, com HTML embebido na linguagem *script*.

Script – Pedaços de código de uma linguagem de *script*, integrados directamente no código HTML da página, e que podem ser executados pela própria página ou como resposta a algum evento desencadeado pelo servidor.

Secure Socket Layer – uma tecnologia originalmente desenvolvida pela *Netscape*, para providenciar uma verificação, no cliente e no servidor, e assegurar a comunicação entre o servidor e o browser. Usa uma tecnologia de combinação de chave pública e privada..

Server Side Include – uma instrução dentro da página ou *script*, que faz com que o servidor execute um programa, insira um ficheiro ou outra informação no segmento HTML enviado ao cliente.

Servidor web-é um software, conectado à *World Wide Web*, que fornece, ao browser, recursos pedidos pelos utilizadores. O recurso pedido normalmente é identificado por um URL.

Structured Query Language – uma linguagem padrão de acesso aos dados em bases de dados relacionais.

Tags – etiquetas usadas na construção de páginas *html*, para a formatação de instruções embutidas no texto. Estão cercadas por parênteses de ângulo (< >), de modo à distingui-los do texto circunvizinho. São também usadas aos pares ao redor do texto que se pretende formatar — um *tag* de abertura, o texto, e um *tag* final marcado pelo caracter (/). Por exemplo, a linha seguinte mostra como formatar um texto em *bold* (<B> texto</B>) e outro em *itálico* (<I> texto </I>)

Transport Control Protocol – é um protocolo orientado á conexão que fornece um serviço fiável de transferência de dados. O nível alto do protocolo *TCP/IP*. Cria os pacotes de dados e passa-os ao IP, para que possam ser enviados à rede.

Transport Control Protocol/Internet Protocol – protocolo base da Internet, também usado em redes internas e em *intranets*. Passa os dados em pacotes roteáveis, entre servidores.

Universal Resource Locator – uma combinação de protocolo, nome do *host*, porta (opcional), caminho e nome de recurso. Identifica de forma única um recurso na Internet.

Users Browser - é definido como cliente.

World Wide Web- Um sistema de informação cliente/servidor que usa múltiplos protocolos Internet. O cliente é um *browser* e o servidor é um servidor *web*.

Anexo G

G.1 Acrónimo

ADO- *Activa Data Object*

ARPANET - *Advanced Research Projects Network*

ASP – *Active Server Pages*

CA – *Certificate Autority*

CGI – *Common Gateway Interface*

DBMS – *Database Management System*

DSW – *Departamento de SoftWare*

DT – *Departamento Técnico*

FTP – *File Transfer*

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Tranfer Protocol*

IIS – *Internet Information Server*

IP – *Internet Protocol*

JSP – *Java server Pages*

MAC- Message Authentication Code

***ODBC** – Open Database Connectivity*

***OLE-DB** – Object Linking and Embedding Database*

***PHP** – Personal Home Page*

***Servlet** – Java Server-based executable*

***SQL** – Structured Query Language*

***SSI** – Server Side Include*

***SSL** – Secure Socket Layer*

***TCP** – Transport Control Protocol*

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

***URL** – Universal Resource Locator*

WWW – World Wide Web

Anexo H

H1 Esquema de tabela do sistema actual

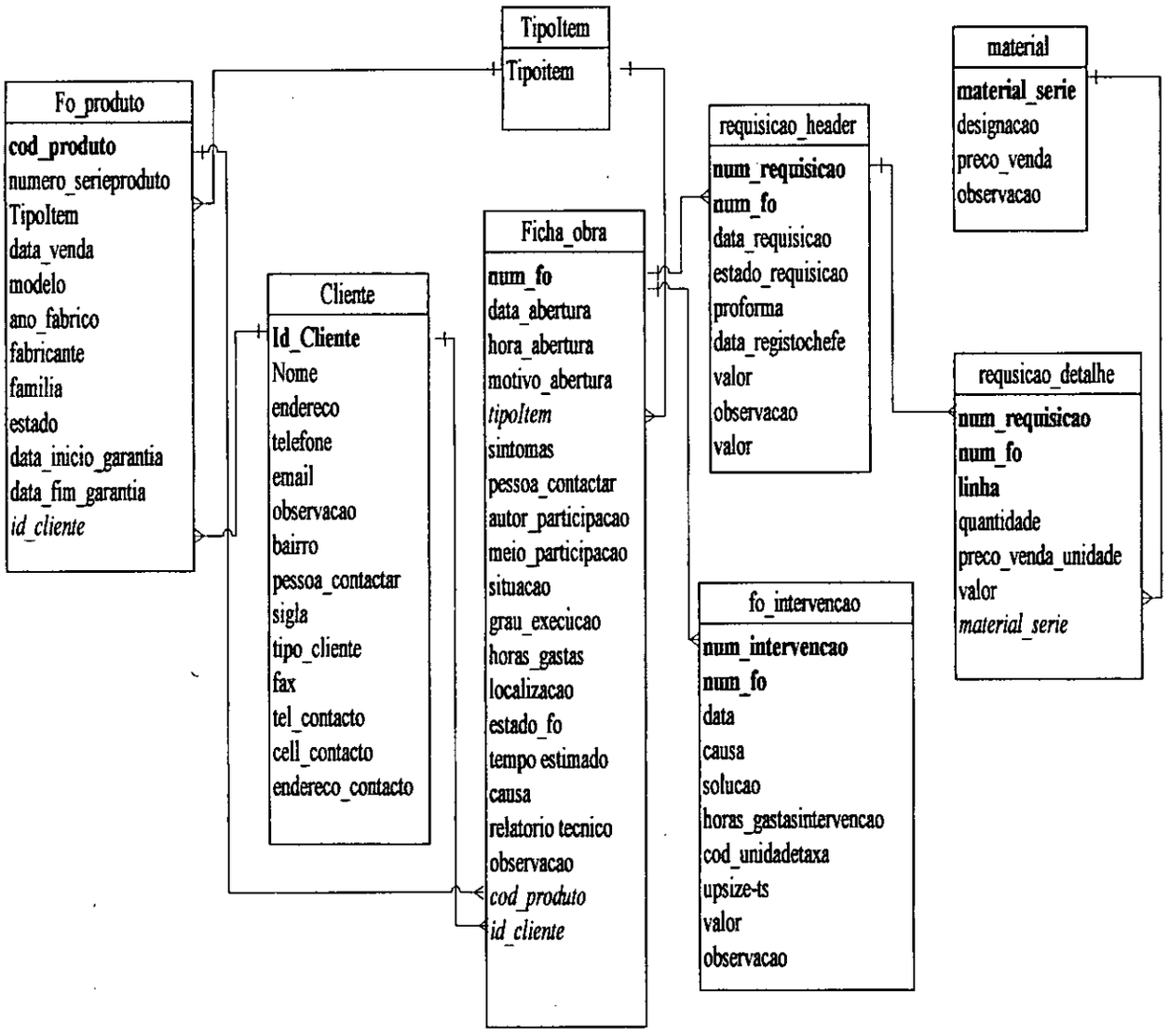


Figura H.1 Esquema de tabelas do sistema actual

H2 esquema de tabela do modelo proposto

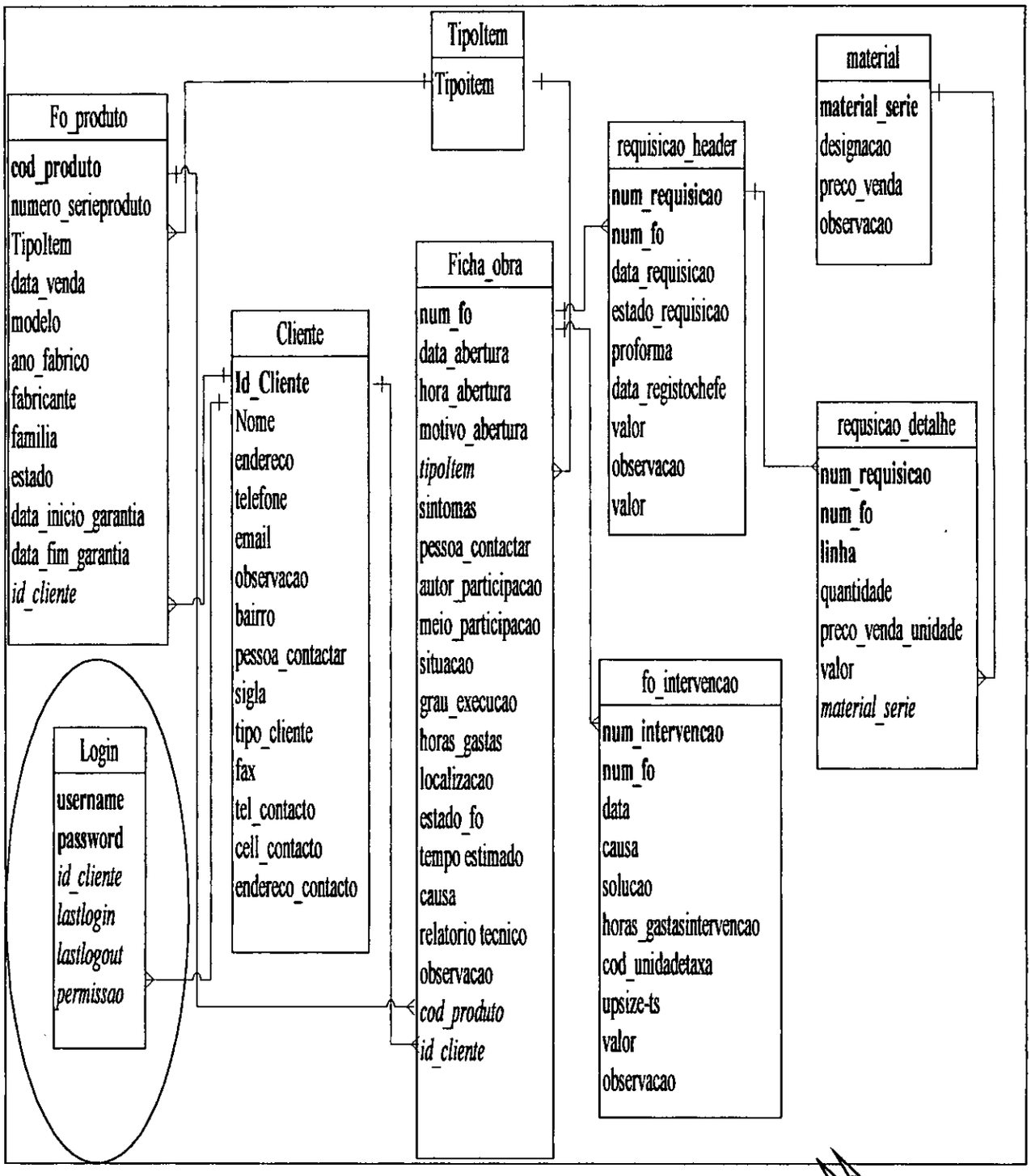


Figura H.2 Esquema de tabelas do modelo proposto

Anexo I

Neste anexo é ilustrado parte do código usado na construção do modelo.

I.1. Código usado para a construção do modelo

Interface de entrada do site

```
<html>

<head>

<title>Ficha de Obra - Main</title>

</head>

<body>

        <IMG alt ="" src="exilogo.gif" >

                <font face="Trebuchet MS" size=3>

                  Engenharia &amp;amp; Comercialização de Sistemas Informáticos

                </font>

                <p><IMG alt ="" height=22 src  ="barra_feichedeluz.gif">

                </p>

    <font size="1">

  <font face="Trebuchet MS">

      A Exi, através deste site, pretende oferecer aos seus clientes, um meio mais  fácil de acesso à
      informação. Permitido deste modo, o acompanhamento do  trabalho da empresa, em relação
      ao seu produto.

      Se é cliente da Exi e não possui uma conta nesta empresa, contacte os  balcões&amp;nbsp;da
      empresa, caso contrário,
```

```
    </font>
    <form action="login.asp" method="post" name="FORM1">
        <input name="submit1" type="submit" value="Clique aqui para  o login">
    </form>
<p><IMG alt ="" height=14 src="_Themes/bluemood/blmhorsd.gif">
</p>
</font>
<font size="1" face="Times New Roman">
</font>
</body>
</html>
```

Autenticação do cliente

```
<%@ Language=VBScript %>
<html>
<head>
<title>Ficha de Obra - log in</title>
</head>
<body>
<IMG alt ="" src="exilogo.gif" >
<font size="3">
<font face="Trebuchet MS">Engenharia &amp;amp; Comercialização de Sistemas Informáticos </font>
```

```
</font>
    <img alt src="barra_feichedeluz.gif" >
    <p style="TEXT-ALIGN: justify">
    <font size="1">
    <font face="Trebuchet MS">
<%
  if request("Secondtry")="True" then
     if request("wrongPW")="True" then
       response.write "Password Invalida. Por favor tente de novo: " & _
                 " Se o erro ocorrer novamente por favor contacte o tecnico. <br>"
     else
       response.write "Nao foi encontrado o UsarName.Por favor Tente Novamente:" &_
                 " Se o erro ocorrer novamente por favor contacte o tecnico.<br>"
     end if
   end if
     Response.write "Por Favor introduza o Usarname e a password para entrar no sistema."
%>
<form action="Checklogin.asp<% if request("SecondTry")="True" then response.write
"?secondTry=true" end if%>" method="post">
    <b>UserName: </b>
       <b>
```

```
        <input name="Username" <%if request("Secondtry")="True" then%>
        Value="<%=session("UserName")%>" <%End if%>size="40">
        </b>
      <b>Password: </b>
      <div align="left">
      <input type="password" name="Password" size="40">
      </div>
      <div align="center" >
       <input name="submit" type="submit" value="Login">
       <input type="reset" value="Cancelar" name="reset" >
      </div>
    </form>
<p><img alt height="14" src="_Themes/bluemood/blmhorsd.gif" ></p>
<p>
<font face="Times New Roman" size="1">
</font>
</p>
</font>
</font>
</body>
</html>
```

Validação do Cliente

```
<!--#include file="ConnFicha.asp"-->

<%

dim strUserName, strPassword

   strUserName = request("UserName")

   strPassword = request("Password")

Dim rsusers

 set rsUsers = Server.CreateObject("ADODB.Recordset")

 strSQL = " Select * from Login where Username= '" & strUserName & "'"

   rsUsers.Open strSQL, objconn,adOpenDynamic,adLockOptimistic, adCmdText

 if rsUsers.EOF then

    Session("UserName")=Request("UserName")

    if Request("SecondTry") = False Then

      Response.Redirect "Login.asp?SecondTry=True"

    end if

  else

    while not rsUsers.EOF

       if UCase(rsUsers("Password"))=UCase(strPassword) then

          rsusers("Activo")=True

          rsusers("lastlogin")=now

          rsusers.Update

          dim strname, strValue
```

```
        for each strfield in rsUsers.fields
          strName=strfield.name
          strvalue=strfield.value
          Session(strName) = strValue
        next
        Session("blnValidUser")=True
          Response.Redirect "fo.asp"
      else
        rsusers.MoveNext
      end if
    wend
    session("UserName")=Request ("UserName")
    if request("SecondTry") = "True" then
      Response.Write "Nao esta Registrado. Contacte o tecnico para o registo"
      Response.Redirect "Main.asp"
    else
      response.redirect "login.asp?SecondTry=True&WrongPW=True"
      Response.Redirect "Main.asp"
    end if
End if%>
```

Conexão a base de dados

```
<%

dim objconn

 set objconn= server.createObject("ADODB.Connection")

 datasource="ficha"

objconn.Open Datasource

if session("blnValidUser")=true and session("Id_Cliente")="" then

 dim rsIdClienteCheck

 set rsIdClienteCheck= server.CreateObject("ADODB.Recordset")


 dim strsql

 strsql = "select Id_Cliente from LogIn" & _

        " Where UserName = '" & session("UserName") & "';"

  rsIdClienteCheck.open strSQL, objconn

 if rsIdClienteCheck.eof then

    session("blnValidUser")=false

 else

    session("Id_cliente") = rsIdClienteCheck("ID_Cliente")

 end if

rsIdClienteCheck.close

 set rsIdClienteCheck = Nothing

end if
```

```
%>
```

Acesso à informação da base de dados

```
<!-- #Include file="ConnFicha.asp"-->

<%

if session("ID_Cliente") ="" then

  response.redirect "Login.asp"

end if

%>

<html>

<head>

<title>EXI-Fichas de Obra</title>

</head>

<body>

<IMG alt ="" src="exilogo.gif" >

<font face="Trebuchet MS" size=3>

Engenharia &amp;amp; Comercialização de Sistemas  Informáticos

</font>

 <IMG alt ="" height=22 src  ="barra_feichedeluz.gif" t>

  <IMG height=30 src="WellSpinTwo.gif">

<font color="black" size="2">: <%= session("UserName")%>

</font>
```

```
<p>
<font size="1">
Nesta página encontram-se todas as fichas
de obra. Para visualizar informação detalhada de uma determinada ficha
clique no seu número.
</font>
</p>
<h3 align="left">
      <div align="left">
        <font face="Times New Roman" size="1">
            Historial de fichas de  obra
       </font>
        <p>
         <font face="Times New Roman" size="1">
           <A href="Foabertas.asp">Fichas de obra abertas</a>
         </font>
      </p>
        <p>
        <font face="Times New Roman" size="1">
        <A href="FoFechadas.asp">Fichas de Obra Fechadas</a>
        </font>
```

```
        </p>
        <p align="center">
<font face="Times New Roman" size="1">
<A href="logout.asp">
<IMG alt ="" border=0 height=20 src="_Themes/inmotion/inmprebd.gif" >Sair</a>
</font>
</p>
<p align="center">
<font size="1">
<font face="Times New Roman">
<A href="mailto:Exi@co.mz">
<IMG border=0 height=51 src="Email18.gif" >
</a>E-mail
</font>
</font>
</p>
</div>
 <p align="center">Historial de fichas de Obra
<%
dim rsFo
 set rsFO=Server.createObject("ADODB.RecordSet")
```

```
rsFO.filter = "ID_Cliente= " & Session("Id_Cliente")

rsFO.Open "Ficha_Obra", Objconn, adopenForward, adlockOptimistic; adCmdTable

if not rsFo.eof then

  Response.Write _

   "<TABLE bgColor=darkcyan BORDER= ""1"" CELLSPACING=""3"" >" & _

   "<TR><FONT face=""Times New Roman"" size=""1"">" & _

   "       <TH> Número <BR><Font Size= "" -2""> clique sobre o nr. Para mais detalhes</font></th>" & _

   "  <TH> Data de abertura      </th>" & _

   "  <TH> Tipo de Item          </th>" & _

   "</font></Tr>"

  Do while not rsFo.EOF

    Response.write _

     " <TR ALIGN=Center>" & _

     "  <TD><A HREF=""DetalheFo.asp?num_fo=" & rsFo("num_fo")& """>" & _

           rsFo("num_fo") & "</A></TD>" & _

     "  <TD>" & rsFo("data_abertura") & "</TD>" & _

     "  <TD>" & rsFo("tipoItem")     & "</TD>" & _

     " </TR>"

    rsFO.Movenext

  loop

  Response.write "</TABLE>"
```

```
Else

  Response.write "<CENTER><h2> Não existe historial de nenhuma ficha de
obra</H2></center>"

end if

rsFO.close

%>

</p>

 <hr>

<font size="1">

<table bgColor="navajowhite" border="1">

 <tr>

  <td>

    <p align="center" class="MsoNormal">

<font size="1">

<span lang="PT">©Copyrigh Carla

    Xavier</span>

</font>

</p></td></tr></table></font></h3>

</body>

</html>
```