UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HÉLTON RIBEIRO NUNES

Sistema de Informação para Apoio à Administração Contábil

Dissertação submetida à Universidade Federal de Santa Catarina como requisito final para a obtenção do titulo de Mestre em Ciência da Computação

Prof° João Bosco da Mota Alves, Dr.
Orientador

Florianópolis, Fevereiro de 2003.

Sistema de Informação para Apoio à Administração Contábil

HÉLTON RIBEIRO NUNES

Esta dissertação foi julgada adequada para a obtenção do título de MESTRE EM CIÊNCIA DA COMPUTAÇÃO, na área de concentração COMPUTAÇÃO APLICADA, e aprovada em sua forma final pelo programa de Pós-Graduação em Ciência da Computação.

_____

Prof° Fernando A. Ostuni Gauthier, Dr.

BANCA EXAMINADORA

_____

Prof° João Bosco da Mota Alves, Dr.

Orientador

_____

Prof° Luiz Alfredo Soares Garcindo, Dr.

Examinador

_____

Prof° Luiz Fernando Jacintho Maia, Dr., Dr.

Examinador

DEDICATÓRIA

A meus pais

Hilton e Romilda

e em memória de minha avó

Francisca

AGRADECIMENTOS

Em primeiro lugar a Deus, por me iluminar no caminho percorrido.

A meus pais, que nunca mediram esforços para me dar uma educação de qualidade.

A meus irmãos, Romilton e Hilton, pela força sempre que necessária.

A minha namorada Mônica, que tem me acompanhado nos momentos bons e ruins.

Aos colegas Richard, Alessandro e Alex pelas caronas e auxílios.

O pessoal do RexLab que de uma forma ou de outra sempre ajudou.

Ao aluno Jorge Motta dos Santos que colaborou no desenvolvimento.

As funcionárias da secretária, Verinha e Val, que sempre nos atendeu de forma gentil.

Ao Prof. Garcindo que sempre que precisei prontamente me ajudou.

Ao Prof. Bosco pela ajuda e paciência.

Também as demais pessoas que de alguma forma ajudaram no de alguma forma no projeto.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho tem como propósito apresentar um modelo para um sistema de informação *WEB*, que permite o gerenciamento e a troca de informações entre profissionais da área contábil e seus clientes. Em uma contabilidade, todos os meses, enfrenta-se o problema de envio das diversas guias (Darf, Folha de pagamento, etc.) aos seus clientes para pagamento em banco. Isso trás um enorme transtorno, além de muitas vezes o escritório ter que contratar um funcionário para fazer esse serviço. A proposta é que o próprio cliente, utilizando o *site* da contabilidade ou a tecnologia *WAP*, tenha acesso a informações necessárias, além de poder imprimir guias de pagamento, sem a necessidade de esperar que o funcionário faça a entrega em mãos ou ter que se dirigir até o escritório.
Para demonstrar esse serviço, será utilizada a tecnologia *JSP* para o módulo *WEB* e a tecnologia *WAP* para o acesso sem fio.

Palavras-Chave: Contabilidade, *JSP*, *Internet*, *WAP*.

ABSTRACT

This project intents to make a study showing the evolution and also propose a new service for Internet, the contabilitian documents exchange.
In one contability, every month, we face the problem of sending many guides (Darf, payroll, etc) to your clients for bank payment. This brings a big upset, besides, several times the office has to hire an employ to do this job.
The propose is that the own client, using the contability web site, print the guides, with no need to wait the employ deliver them or wasting time going to the office.
To show this service, it will be used JSP technology to the WEB module. This language uses Java technology to create dynamic web sites, similar to the known PHP and ASP. Another module will use WAP technology; where through wireless dispositives can also be accessed.

Key Words: Contability, JSP, Internet WAP

1 INTRODUÇÃO

1.1 Apresentação

Desde que se tem noticias dos primeiros contabilistas eles se tornaram imprescindíveis para o bom funcionamento das empresas em geral.

Nesta área a utilização da informática tem se desenvolvido bastante. Como comparação basta lembrar que algum tempo atrás para ser feita a declaração de imposto de renda havia a necessidade de que se fizesse o preenchimento de diversos formulários para entregá-la, algum tempo depois veio a possibilidade da entrega em disquetes, hoje podemos faze-la no computador e enviar via *Internet*. Além desta, outras declarações já podem ser entregues via *Internet*: DARF (Documento de Arrecadação de Receitas Federais), GIA (Guia de Informação e Apuração do ICMS), entre outros. Uma outra forma de analisarmos a informática dentro da contabilidade é que a grande maioria dos escritórios é informatizada, realizando quase todos os serviços via computador.

Mas ainda persiste a necessidade do contato entre contador e cliente para a troca de informações e documentos. Porque não tentar diminuir a necessidade de contato entre as partes? A idéia aqui não é fazer com que as partes deixem de ter encontros para tratar de determinados assuntos, mas sim evitar, quando possível, a locomoção escritório-empresa, ou vice-versa.

Esse problema foi detectado através da convivência com alguns profissionais da área contábil onde havia a preocupação a respeito da quantidade de

tempo dispensada para a troca de documentos com os clientes para pagamento de tarifas e impostos. Isso tornava, muitas vezes, necessário a contratação, pelo escritório, de uma pessoa que ficasse responsável pela entrega desses documentos.

Figura 01 - Processo atual de entrega de documentos

Outro problema comum era o atraso no pagamento por dois principais motivos:

1. Muitos documentos levam em consideração o faturamento do mês anterior, mas o período, entre o fechamento do mês e a data de pagamento, em certas ocasiões é curto;
2. em determinadas ocasiões o responsável não consegue completar o serviço por motivos alheios a sua vontade, acarretando atrasos no pagamento.

Sendo assim verificamos a real importância deste trabalho realizado para a melhoria dos serviços prestados bem como a satisfação do cliente.

1.2 Justificativa

Após a verificação dos problemas surgiu a necessidade de se desenvolver soluções para eliminação ou diminuição dos mesmos, sendo que as soluções deveriam utilizar a maior parte possível dos recursos já disponíveis. Esses recursos a serem utilizados podem ser a grande informatização já existente nas contabilidades.

No modelo que se propõe, utiliza a *Internet* como forma também de distribuição de documentos, onde o cliente poderá, a qualquer momento, saber dos documentos que possui para pagamento.

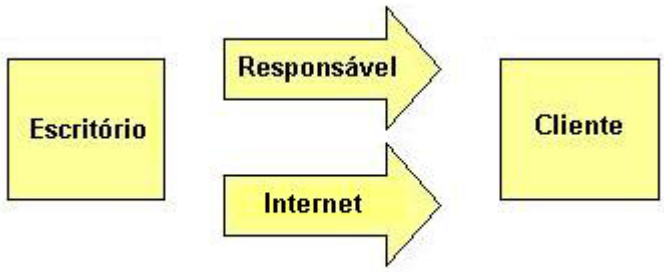


Figura 02 - Processo como o modelo proposto

A partir do sistema e base de dados já existente no escritório, serão desenvolvidos módulos adicionais para *WEB* e *WAP*.

No módulo *WEB* o cliente poderá fazer acesso à página do escritório contábil e, utilizando-se de um login e uma senha, verificar as suas informações. Esses dados acessados serão, a princípio, referentes a documentos a pagar ou já pagos, também poderá, se for necessário, fazer a impressão para pagamento em agência bancária.

O outro módulo, chamado módulo *WAP*, o cliente poderá fazer acesso as informações utilizando-se de um dispositivo sem fio, como telefone celular, Pager, etc..

Figura 03 - Formas de acesso no modelo

1.3 Objetivos

1.3.1 Geral

Este trabalho tem como objetivo principal apresentar um modelo de sistema de informação *WEB* para contabilidades, onde o cliente e o próprio contador poderão acessar informações a distância.

1.3.2 Específicos

- Apresentar estudo de ambientes de desenvolvimento para *WEB*;
- Possibilitar que o cliente possa fazer um auto-atendimento;
- Controle de documentos pagos e a pagar;
- Tornar o processo de distribuição de documentos mais eficaz;

1.4 Metodologia

Para o desenvolvimento desta dissertação houve um estudo bibliográfico intenso, principalmente no que se refere a tecnologias.

Havia também a necessidade de compreender de forma mais detalhada o funcionamento dos serviços contábeis envolvidos no processo além de verificar as deficiências a serem tratadas. Com os dados em mãos partiu-se para a tarefa de definição do modelo do sistema através de diversas reuniões realizadas com o profissional da área.

Após iniciou-se o estudo dos ambientes e ferramentas para o desenvolvimento do protótipo, onde várias tecnologias foram analisadas as suas características e viabilidade de uso.

Com o modelo e as tecnologias a serem utilizadas já definidas foi desenvolvido um protótipo para a demonstração do sistema. Apesar de ser considerado uma versão inicial, algumas coisas ainda necessitaram de alteração, como o acesso *WAP*, por se tratar de uma tecnologia nova na área de contabilidade.

Durante o desenvolvimento do protótipo surgiu à idéia da utilização de agentes inteligentes para melhorar a execução de determinadas tarefas, essa parte do trabalho não entrou no desenvolvimento inicial do protótipo, mas faz parte do modelo proposto.

Para terminar, o protótipo passou pela aprovação de um contador, que analisou e sugeriu algumas alterações que são apresentadas também no modelo proposto. Entre os itens analisados podemos citar:

- Segurança;
- usabilidade;
- facilidade de uso;
- informações disponíveis.

1.5 Estrutura da Dissertação

Além deste capítulo introdutório, esta dissertação possui a seguinte estrutura:

- Capítulo 2 – Apresenta um histórico da Internet, onde se procuramos trazer informações um pouco diferentes do que costuma se encontrar em bibliografias;
- Capítulo 3 – É apresentado de outras linguagens de desenvolvimento para *WEB* (*ASP*, *PHP* e *Cold Fusion*), busca-se aqui mostrar características de cada uma delas e não como utiliza-las. Também foi detalhado o funcionamento da tecnologia utilizada para desenvolver do módulo principal do projeto, o *JSP*;
- Capítulo 4 – Apresentamos a tecnologia *WAP*;
- Capítulo 5 – Este capítulo trata do assunto Agentes Inteligentes;
- Capítulo 6 – O último capítulo desta dissertação trata diretamente do modelo proposto.

2 A INTERNET

> [...] numa altura em que toda a gente fala da Internet, e a mídia não para de nos bombardear com histórias de feitos fantásticos nesta rede das redes, fica claro que a Internet chegou, e veio para ficar. Mas a Internet não foi "planejada". Não houve nenhum grupo de pessoas que se sentou a uma mesa e disse "vamos criar a Internet" (CLIX (2003)).

A *Internet* tem revolucionado o mundo de tal maneira que nenhuma invenção foi capaz de fazer antes. Ela é um mecanismo de disseminação de informações e uma forma de colaboração e interação entre pessoas e seus computadores independentemente de suas localizações geográficas.

> A *Internet* é uma grande rede que integra computadores de todos os tipos e tamanhos. O computador conectado à rede pelo telefone sai do seu ambiente isolado e limitado para acessar às informações de seu disco rígido e ingressa numa enorme teia que coloca o mundo ao seu alcance. (BIANCHI, 2000, p.266)

A *Internet* é um dos investimentos que mais deram certo na história. Desde as primeiras pesquisas, vários áreas tem sido colaboradores na evolução desta tecnologia. Hoje, termos como nome@nomedeempresa.com.br e http://www.nomedeempresa.com são usados diariamente por milhões de pessoas.

A história da *Internet* é complexa e envolve muitos aspectos tecnológicos, organizacionais e comunitários. Sua influência já atinge toda a sociedade, na medida em que usamos cada vez mais ferramentas on-line para fazer comércio eletrônico, adquirir informações e operar em comunidade.

2.1 A origem da Internet

As primeiras informações que se tem a respeito de que comunicação entre computadores poderia ser realizada através de uma rede foram memorandos publicados por J.C.R. Licklider em agosto de 1962. No documento imaginavam-se computadores interligados por todo o mundo, onde as conexões eram feitas de forma transparente, acessando informações e programas, este conceito era muito parecido com a *Internet* de hoje.

Os primeiro trabalho sobre troca de pacotes foram publicados em julho de 1961 e o primeiro livro em 1964 por Leonard Kleinrock, do *MIT*. Apartir daí ele convenceu outros estudiosos sobre a possibilidade de comunicação usando pacotes, o que representou um grande passo para tornar possíveis às redes de computadores. O outro grande passo foi fazer os computadores se comunicarem.

Em 1965 foi conectado um computador em Massachussets com um da Califórnia através de uma linha discada, o que foi a primeira comunicação entre computadores do mundo. Assim consegui-se comprovar que era possível fazer com que computadores trabalhassem juntos, executando programas e compartilhando dados quando necessário, em compensação o circuito do sistema telefônico era totalmente inadequado em virtude da baixa velocidade.

Segundo TOLHURST (1994, p. 60), no final de 1966, Roberts começou a trabalhar no *DARPA*[1] com a finalidade de desenvolver o conceito das redes computadorizadas e elaborou um plano para a *ARPANET*, publicado em 1967. Em uma conferencia ele apresentou este trabalho onde também houve uma apresentação sobre redes de pacotes, desenvolvida pelos ingleses Donald Davies e Roger Scantlebury, da *NPL-Nuclear Physics Laboratory*. Scantlebury conversou com Roberts sobre o trabalho da *NPL* e do trabalho de Paul Baran e outros em *RAND*. O grupo do projeto *RAND* havia escrito um trabalho para voz segura sobre o papel das redes de trocas de pacotes, quando serviam militarmente em 1964. O que se percebeu então é que os trabalhos desenvolvidos no *MIT* (1961-67), *RAND* (1962-65) e *NPL* (1964-67) estavam sendo feitos em paralelo sem que nenhum dos pesquisadores soubesse dos outros trabalhos.

> A *ARPANET* foi uma rede criada em 1969 pela *Advanced Research Projects Agency* do Departamento de Defesa dos EUA, com a intenção de partilhar o processamento de dados, armazenamento e troca de dados entre os pesquisadores e fornecedores ligados ao Departamento, servia

[1] *Defense Advanced Research Projects Agency*. Nome pelo qual a *ARPA* passou a ser designada a partir da década de 70.

> também como mecanismo de troca de correspondência entre os pesquisadores (TOLHURST, 1994, p. 66).

O primeiro ponto da *ARPANET* a ser escolhido foi a *UCLA* em 1969, devido à teoria de trocas de pacotes de Kleinrock. O segundo foi um projeto de Doug Engelbart, no *SRI - Stanford Research Institute*, foi o segundo, que passou também a manter as tabelas de *Host Name*[2] que fariam o mapeamento dos endereços e diretórios. A primeira mensagem entre servidores foi enviada um mês depois com vários outros pontos da rede sendo conectados, principalmente universidades.

O primeiro protocolo a ser desenvolvido foi o servidor da *ARPANET*, chamado *Network Control Protocol* (*NCP*), em 1971, onde as aplicações poderiam ser desenvolvidas.

Em outubro de 1972, foi organizada uma demonstração sobre a *ARPANET* na Conferência Internacional de Comunicação entre Computadores que foi a primeira demonstração pública da nova tecnologia de rede. Foi também em 1972 que o correio eletrônico foi introduzido quando também surgiu o software básico de e-mail com as funções de enviar e ler.

2.2 Os conceitos iniciais da Internet

A *ARPANET* original cresceu e se tornou a *Internet* que temos hoje. Ela foi criada a partir de que existiriam inúmeras redes independentes umas das outras e com desenhos diferentes, começando com a *ARPANET*, mas logo incluindo redes de satélites, de rádio, etc. A *Internet* como conhecemos hoje incorpora uma idéia-chave: rede de arquitetura aberta. A opção pela tecnologia da rede não é ditada por nenhuma arquitetura em particular, mas sim escolhida livremente pelo provedor, que deve ser capaz de se comunicar com outras redes.

Em uma rede com arquitetura aberta, as redes individuais podem ser desenvolvidas separadas e cada uma pode ter suas próprias características para ser oferecida a usuários e outros provedores. Essa idéia de redes com arquitetura aberta foi criado por Kahn em 1972. Este trabalho foi parte de um programa de pacotes, mas tornou-se um programa em separado depois.

O *NCP* era limitado em relação ao endereçamento das redes e máquinas, além de que se qualquer pacote fosse perdido, o protocolo e qualquer aplicação que

[2] Nome de um computador na Internet, equivalente a um endereço IP.

ele suportasse iria simplesmente parar a transferência dos dados. Também não tinha controle de erro ponta a ponta, isso em virtude de se pensar que a *ARPANET* seria a única rede e ela seria confiável. Então se decidiu pelo desenvolvimento de uma nova versão do protocolo que deveria satisfazer as necessidades de um ambiente de redes com arquitetura aberta. Este protocolo iria ser chamado *Transmission Control Protocol/Internet Protocol* (*TCP/IP*).

Segundo CLIX (2003), três regras foram críticas:

- As redes deveriam ser independentes e mudanças nelas não deveriam ser requisitadas para conectá-las à *Internet*;
- Se um pacote não chegasse em seu destino final, ele seria retransmitido da fonte;
- Caixas pretas seriam usadas para conectar as redes. Mais tarde elas seriam chamadas *gateways* e roteadores.

Kahn começou a trabalhar às comunicações dos princípios do sistema operacional e documentou alguns dos seus pensamentos num memorando chamado "Princípios de Comunicações para Sistemas Operacionais". Então percebeu que era necessário estudar a implementação de cada sistema operacional para ter a chance de embutir neles novos protocolos. Assim, na primavera de 1973, depois de começar o projeto, Kahn chamou Vint Cerf para trabalhar com ele na criação do protocolo. Cerf tinha se envolvido no desenvolvimento do *NCP* original e já tinha o conhecimento com os sistemas operacionais existentes. A conhecimento sobre comunicação de Kahn e a experiência em *NCP* de Cerf possibilitou a construção do que se tornou *TCP/IP.* A primeira versão escrita foi distribuída numa reunião especial do *International Network Working Group* (*INWG*).

O trabalho original descreveu um protocolo chamado *TCP*, que possui controle sobre o transporte e serviços de encaminhamento na *Internet*. A versão inicial funcionou bem para transferência de arquivos e aplicações de *logins* remotos, mas em algumas aplicações avançadas, como pacotes de voz, mostraram que, em alguns casos, a perda de pacotes deveria ser corrigida pela aplicação e não pelo protocolo *TCP*.

> [...] os dois protocolos básicos da Internet são o TCP e o IP, usados para viabilizar a transmissão e troca de dados de redes diferentes, permitindo assim que os computadores se comuniquem. Como o TCP/IP foi desenvolvido a partir de fundos públicos, ele não pertence a uma empresa

> específica e pode ser utilizado por qualquer computador para o compartilhamento de informações com outro computador (EDDINGS, 1994, p. 92).

Isso levou a uma reorganização do protocolo original em outros dois: o *IP* que possuía o endereçamento e o roteamento dos pacotes individuais e o *TCP* em separado, que tinha como função cuidar do controle do fluxo e a recuperação de pacotes perdidos.

Uma das maiores motivações para o desenvolvimento da *ARPANET* e da *Internet* foi o compartilhamento de recursos. A conexão das duas redes foi muito mais econômica do que a implantação de novos computadores. Enquanto a transferência de arquivos e o *Telnet* foram aplicações muito importantes, o correio eletrônico ou *e-mail* teve o impacto mais significativo. Onde foi criado um serviço no qual as pessoas poderiam comunicar-se, primeiro na construção da própria *Internet* e mais tarde na sua utilização por grande parte da sociedade.

> [..] o Telnet é uma ferramenta utilizada para estabelecer comunicação com outras máquinas em outros lugares. Quando é estabelecida a conexão via Telnet, você está no computador remoto, ou seja, é como se você estivesse usando o computador no lugar onde ele está instalado. Permite uma conexão pura e simples entre duas máquinas, sem interface gráfica (EDDINGS, 1994, p58).

Diversas aplicações foram também propostas nos dias iniciais da *Internet*, incluindo comunicação de voz, modelos de compartilhamento de arquivos e discos e os primeiros programas que mostraram o conceito de agentes. Um detalhe importante da *Internet* é que ela não foi criada para apenas uma aplicação, mas sim como uma infra-estrutura genérica na qual novas aplicações podem ser criadas, como aconteceu com a *World Wide Web*. Foi e é a natureza do serviço provido pelos protocolos *TCP* e *IP* que tornam isso possível.

2.3 A Internet no Brasil

No Brasil, as universidades estão ligadas com redes mundiais desde 1989. Naquele ano, havia conexões com a *Bitnet*, uma rede semelhante à *Internet*, em várias instituições, como as universidades federais do Rio Grande do Sul e do Rio de Janeiro. Os serviços disponíveis eram apenas o correio eletrônico e transferência de arquivos. Em 1990, a Fapesp (Fundação de Amparo à Pesquisa de

São Paulo) conectou-se com a *Internet*. Nesse mesmo ano, foi criada a RNP, uma iniciativa do Ministério da Ciência e Tecnologia.

Financiada pelo CNPq, a RNP interligou inicialmente capitais de onze estados. Essa arquitetura de linhas de comunicações e equipamentos compõe o que se chama de espinha dorsal (*backbone*) da RNP. A partir de abril de 95, o Ministério das Comunicações e o Ministério da Ciência e Tecnologia decidiram lançar um esforço de implantação de uma rede integrada entre instituições acadêmicas e comerciais. Desde então vários fornecedores de acesso e serviços privados começaram a operar no Brasil.

3 DESENVOLVIMENTO PARA WEB

Muita coisa evoluiu na *Internet* desde a sua criação, e um dos 'braços' dela que mais cresceu foi a *WWW*. Uma das evoluções dizem respeito à criação das *Home Pages* (páginas web). Desde o inicio elas são criadas utilizando-se da linguagem *HTML* (*Hiper Text Markup Language* – Linguagem para marcação de hipertexto).

3.1 Desenvolvendo Aplicações Web

Em um sistema cliente/servidor, o cliente mantém uma conexão com servidor, consultando-o periodicamente para verificar se a conexão está aberta. Se o servidor não estiver em funcionamento, o cliente não conseguirá a conexão e então tomará as medidas apropriadas como, por exemplo, enviar uma mensagem de erro para o usuário.

Porém, as aplicações baseadas na *Web* são muito diferentes de outros tipos de programação. Diferente de aplicações cliente/servidor, essas aplicações compreendem páginas como resposta ao cliente. Estas são estáticas, ou seja, depois do cliente navegador solicitar a página, a mesma é processada no servidor e então, o documento é retornado.

3.2 HTML

Segundo RUAS (2002, p.16) *HTML* é uma linguagem interpretada, ou seja, não necessita de um compilador que crie outro aplicativo em linguagem de máquina, mas sim de um interpretador que leia o arquivo fonte e gere o resultado final, no caso a página. Quem faz o trabalho de interpretação da linguagem *HTML* é o *Web Browser* também chamado de navegador ou cliente.

HTML, na verdade é uma linguagem de roteiro que utiliza marcações especificas e distintas para dizer ao navegador como exibir o documento. A esses marcadores damos o nome de *Tags*, as quais servem para indicar de que forma os elementos da página serão apresentados.

A linguagem é simples e fácil de aprender, motivos pelos quais se tornou tão popular. Mas a necessidade de recursos nas páginas aumentou muito e a linguagem não apresentava formas de supri-las. Assim surgiram várias ferramentas (animação, vídeos, sons, etc) de 'incrementar' a apresentação. Mas ainda, para algumas situações, ainda não eram suficientes. Apesar de tornarem as páginas rápidas ainda possuíam limitações:

- Conteúdo estático;
- Funcionalidades limitadas;
- Pouca interação com o usuário.

Com essas dificuldades era impossível criar, por exemplo, sistemas de Comercio Eletrônico e *Internet Banking*. Surgiu então a necessidade de incrementar as páginas sem a necessidade de serem instalados, no cliente, novos softwares (*plugins*[3]). Para algumas situações surgiram linguagens de *script*[4] que eram também interpretadas pelo navegador, mas ainda não era o suficiente. Então foram sendo desenvolvidas linguagens que possuam recursos maiores que o *HTML* e que são interpretados no servidor. Como exemplo podemos citar:

- ASP;
- PHP;
- Cold Fusion;
- JSP, que é o ponto central de nosso trabalho.

[3] Plugins - Aplicativos que podem ser facilmente instalados e usados como parte do navegador, em geral para execução de som, imagem de vídeo e animações.
[4] Script - Um atalho de programação que fornece ao usuário não técnico uma maneira de criar um conteúdo mais rico em seu respectivo computador e fornece aos programadores uma maneira rápida de criarem aplicativos simples.

3.3 Outras Linguagens

Essas linguagens citadas funcionam de forma semelhante e em conjunto com o *HTML*: na forma de *scripts*, ou seja, são colocadas no mesmo arquivo, mas entre *Tags* especiais. Elas possuem características que trazem muitas possibilidades no desenvolvimento das páginas, principalmente acesso à banco de dados e disponibilização de conteúdo dinâmico, ou seja, variando de acordo com parâmetros fornecidos pelo usuário.

Uma chamada a uma página que utilizam essas linguagens faz-se da seguinte maneira:

1. O usuário, através de seu navegador, faz uma chamada, como qualquer outra página, ao servidor web;
2. O servidor vai detectar, através da extensão do arquivo, que esta página possui *scripts* de alguma outra linguagem e repassa a chamada para um *Engine*;
3. O *Engine* faz a chamada a outros componentes (banco de dados, por exemplo) e retorna, para o servidor, uma página onde todo o conteúdo foi traduzido para *HTML*;
4. Por último o servidor devolve ao cliente um *HTML* puro que será interpretado normalmente.

Com isso temos duas situações importantes:

- O cliente não precisa de *plugin* ou qualquer outro recurso instalado;
- O código fonte fica escondido, diferente do *HTML* que fica visível ao cliente.

Apresentaremos a seguir informações sobre as linguagens citadas.

3.4 ASP

Active Server Pages, segundo ASPBRASIL[5], são páginas web que possuem conteúdo dinâmico. Tais páginas consistem em arquivos de extensão .asp que contêm combinações de *Server-Side scripts* e Tags *HTML*.

[5] ASPBRASIL. Disponível em:. <www.aspbrasil.com.br> . Acesso em: 07/01/2003.

Todo o código de programação existente em páginas *Asp* é executado no servidor, e este retorna ao cliente somente respostas em *HTML* padrão – o que faz com que aplicações *Asp* possam ser acessadas por qualquer navegador existente no mercado. Uma aplicação feita em *Asp* pode ainda conter linhas de *Client-Side script*, que serão executados na estação cliente. Essas páginas devem estar hospedadas num servidor *Microsoft Information Server*.

3.4.1 Client Side scripts

> [..] são códigos de programa que são processados pela estação cliente. Geralmente em aplicações voltadas à Internet, o código que é executado no cliente cuida apenas de pequenas consistências de telas e validações de entrada de dados (WEISSINGER, 2000, p. 25).

Em se tratando de páginas *web*, os *client-side scripts* serão processados por um navegador. O maior problema de se utilizar este tipo de artifício em uma aplicação é a incompatibilidade de interpretação da linguagem entre os navegadores. O *Microsoft Internet Explorer*, por exemplo, é capaz de interpretar o *Visual Basic Script*, porém o *Netscape* não o faz sem o auxílio de um *plugin* (que foi desenvolvido por terceiros). Há ainda o problema de versões muito antigas de navegadores, que não conseguem interpretar nenhum *script*.

Em grande parte das situações, não é possível exigir que o usuário final disponha de determinado produto para acessar a aplicação. Portanto é importante pesar todos estes fatores ao planejar alguma aplicação com *client-side scripts*.

A linguagem script mais indicada para se construir *client-side scripts* é o *JavaScript*, devido a sua compatibilidade com os navegadores existentes no mercado.

Na verdade a tecnologia esta em um arquivo de aproximadamente 300kb chamado asp.dll. Sempre que é solicitada uma página .asp, a dll faz a interpretação e transforma o código resultante em *HTML*, que é retornado ao cliente.

3.4.2 Server Side scripts

Segundo WEISSINGER (2000, P.25), são códigos de programa que são processados no servidor. Devido a este fato, não é necessário preocupar-se com a linguagem que o código foi criado: o servidor é quem se encarrega de interpretá-lo e

de devolver uma resposta para o cliente. Em páginas *Asp*, são esses códigos os maiores responsáveis pelos resultados apresentados.

3.4.3 Pré-Requisitos de funcionamento

As páginas *Asp* necessitam ser hospedadas no servidor *Web* da *Microsoft: o Internet Information Server* (*IIS*) na versão 3 ou superior. Este servidor deve ser instalado numa máquina *NT Server* 4. Para o *IIS* 3, ainda é preciso instalar um pacote adicional do *Asp* para que as aplicações funcionem. A partir da versão 4 este pacote já vem incorporado ao *IIS*. Para efeitos de desenvolvimento e testes pode-se utilizar o *Personal Web Server* (*PWS*), servidor *Web* que acompanha o windows 98.

3.4.4 ADO (ActiveX Data Objects)

É uma coleção de objetos utilizados para recuperação, alteração, inclusão e exclusão de registros em bancos de dados *ODBC* e *OLE DB*. O *ADO* é instalado com *Microsoft Internet Information Server* (*IIS*), versão 3.0.

O *ADO* é escrito em páginas *ASP* e executado no servidor Web, que retorna as informações dos bancos de dados em *HTML*. A página retornada pode ser visualizada por qualquer navegador em qualquer plataforma, pois o código é todo executado pelo servidor.

Estes objetos se constituem em uma camada entre a página ASP e o banco de dados. Para acessar o banco de dados, são escritos códigos que configurarão propriedades e métodos dos objetos do *ADO*.

A comunicação do *ADO* com o banco de dados é feita através de *OLE-DB* ou banco de dados que possuem drives *ODBC*.

3.5 Cold Fusion

É uma ferramenta de desenvolvimento de aplicações *web*, desenvolvida atualmente pela *Macromedia*, que permite a criação de páginas dinâmicas através de integração sofisticada entre elementos como: banco de dados, ambiente *web* e aplicações de *e-mail*, além de permitir aplicações *Java*.

O *Cold Fusion* utiliza-se de uma linguagem denominada *CFML* (*Cold Fusion Markup Language*) também baseada em *Tags* especiais como *ASP* e *HTML*.

3.5.1 Arquitetura Cold Fusion

A arquitetura também é semelhante ao *ASP*. Quando uma página dentro da aplicação *Cold Fusion* é requerida por um navegador, é processado o CFML e gerada dinamicamente uma página de retorno para o cliente com as informações requeridas.

3.5.2 Cold Fusion Markup Language (CFML)

CFML dispõe de um ambiente de script baseado em *Tags*, o que torna o desenvolvimento de aplicações muito simples. Por também ser uma linguagem baseada em *Tags*, a *CFML* se integra facilmente ao *HTML*.

Através de suas *Tags* pode-se manipular variáveis, utilizar funções de data, hora, matemáticas, pesquisa e string, fazer declarações condicionais, loop, além de utilizar declarações *SQL* avançadas.

3.5.3 Trabalhando Banco de Dados

O *Cold Fusion* utiliza o *ODBC* para se comunicar com uma larga escala de banco de dados. Antes de utilizar um banco de dados (ou *Data Source*) em uma aplicação é necessário registrá-lo no *Cold Fusion Administrator*.

Quando registramos um *Data Source* no *Cold Fusion* daremos um nome a esta conexão e definimos o banco de dados a ser utilizado no desenvolvimento do *script*.

Vamos assumir que já configuramos um *Data Source* com o nome de Agenda e que ele é um banco de dados agenda.mdb onde temos uma tabela chamada Pessoal, que tem como campos Nome e Telefone. Com estas configurações podemos criar o script para acesso a sua base de dados:

```
<CFQUERY DATASOURCE=”Agenda” NAME=”AgendaPessoal”>
SELECT * FROM Pessoal
</CFQUERY>
```

No script acima foi feito uma consulta na tabela Pessoal de nosso *Data Source* ao final da consulta o resultado é armazenado em uma variável chamada "AgendaPessoal". Agora podemos, por exemplo, listar o resultado na página:

```
<CFOUTPUT QUERY= "AgendaPessoal" >
#nome# - #telefone#
</CFOUTPUT>
```

O resultado desta pesquisa, assumindo ter apenas três registros no banco de dados da forma a seguir que será utilizado no *HTML*:

Antonio Carlos – 48-626-9980

Carlos Augusto – 48-626-3328

Maria da Silva – 48-626-9915

3.6 PHP

> [..]é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. PHP é semelhante às linguagens descritas anteriormente onde o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial (CONVERSE, 2001, p. 22).

Basicamente, qualquer coisa que pode ser feita com *PHP*, como coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookies*[6].

O *PHP* também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como *dBase*, *Interbase*, *mySQL*, *Oracle*, *Sybase*, *PostgreSQL* e vários outros. Construir uma página baseada em um banco de dados torna-se uma tarefa extremamente simples com *PHP*.

Além disso, *PHP* tem suporte a outros serviços através de protocolos como *IMAP*, *SNMP*, *NNTP*, *POP3* e, logicamente, *HTTP*. Ainda é possível abrir *sockets* e interagir com outros protocolos.

---

[6] Cookie - Mensagem enviada ao browser por um servidor Web. Normalmente, esta mensagem é gravada no micro cliente como um arquivo de texto. Sempre que o cliente volta a acessar aquele site, a mensagem é reenviada ao servidor. O objetivo do cookie é identificar o usuário e, por exemplo, exibir páginas personalizadas. Para isso, o usuário precisa ter fornecido informações pessoais numa visita anterior ao site.

3.6.1 Surgimento do PHP

A linguagem *PHP* foi criada em 1994 por Rasmus Lerdorf. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua página apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como *Personal Home Page Tools* (ferramentas para página pessoal). Era composta por um sistema bastante simples que interpretava alguns *Scripts* e alguns utilitários que rodavam no servidor: um livro de visitas, um contador e algumas outras coisas.

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de *PHP/FI*, o "*FI*" veio de um outro pacote escrito por Rasmus que interpretava dados de formulários *HTML* (*Form Interpreter*). Ele combinou os scripts do pacote *Personal Home Page Tools* com o *FI* e adicionou suporte a banco de dados, nascendo assim o *PHP/FI*, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.

Estima-se que em 1996 *PHP/FI* estava sendo usado por cerca de 15.000 sites pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000. Nessa época houve uma mudança no desenvolvimento do *PHP*. Ele deixou de ser um projeto de Rasmus com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por Zeev Suraski e Andi Gutmans, e esse novo interpretador foi a base para a versão 3.

O lançamento do *PHP4*, ocorrido em 2000, trouxe muitas novidades aos programadores de *PHP*. Uma das principais foi o suporte a sessões. Além das mudanças referentes a sintaxe e novos recursos de programação, o *PHP4* trouxe como novidade um otimizador que permite a execução muito mais rápida de *scripts PHP*.

3.7 JSP – JAVA SERVER PAGES

3.7.1 Introdução

Uma página *JSP*, segundo BOMFIM (2002, p. 229) é "um documento baseado em texto que descreve como processar uma solicitação (*Request*) para criar uma Resposta (*Response*)". E complementa: "A descrição mistura internamente

dados modelares com ações dinâmicas e é alavancada pela plataforma *Java* 1.2". A junção de dados estáticos com códigos de uma linguagem de programação resulta em um resultado para um usuário motivado por uma solicitação dele.

Neste capítulo pretende-se dar ênfase, única e exclusivamente, a tecnologia *JSP*, mostrando comandos, exemplos e características dela.

3.7.2 Necessidades de software

Para trabalharmos com *JSP*, precisamos de alguns softwares instalados no servidor.

O primeiro a utilizar será o *JDK*[7]. Ele será responsável pela base necessária à implantação dos recursos relativos aos aplicativos desenvolvidos em *Java* e para o servidor *JSP*.

Este software é distribuído gratuitamente pela *Sun Microsystems* (http://java.sun.com), empresa desenvolvedora do *Java.*

O próximo software necessário será o servidor *Internet* com suporte a *JSP*.

Para conseguir ver os resultados das páginas que iremos construir é preciso instalar um servidor *JSP*. Por se tratar de uma extensão da programação *Java*, o *JSP* pode ser executado em qualquer plataforma. Existem diversos servidores *JSP* espalhados na *WEB*, alguns deles, dispõe de versões para plataforma *Windows* e *Linux*. Dois dos mais famosos são o *Tomcat*, do projeto *Jakarta* (http://jakarta.apache.org/tomcat) e o *Resin* da Empresa *Caucho* (www.caucho.com).

3.7.3 Como Funciona

Quando o cliente faz a solicitação de um arquivo *JSP*, é enviado um *Object Request* para a *JSP Engine*. A *JSP Engine* envia a solicitação de qualquer componente especificado no arquivo. O componente controla a requisição possibilitando a recuperação de arquivos em banco de dados ou outro dado armazenado, em seguida, passa o objeto *Response* de volta para a *JSP Engine*. A *JSP Engine* e o *WEB Server* enviam a página *JSP* revisada de volta para o cliente, onde o usuário pode visualizar os resultados através do navegador.

[7] Java Developers Kit – Kit para desenvolvimento de aplicações *JAVA*.

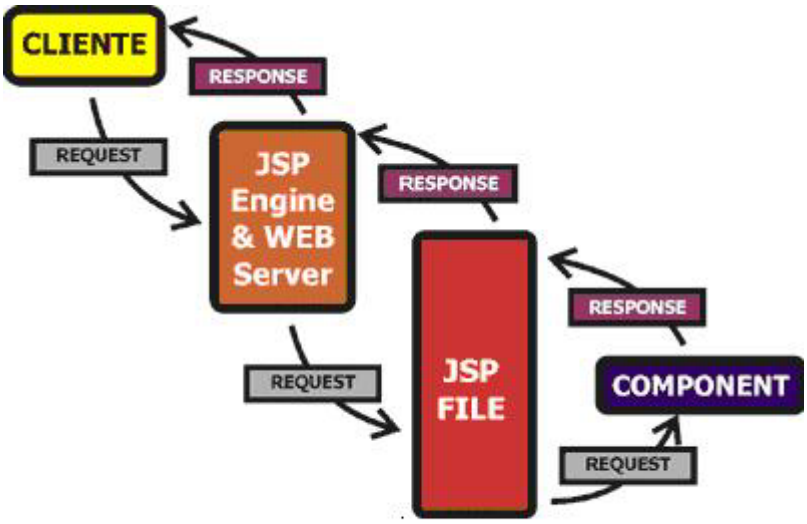


Figura 04 - Funcionamento da requisição de páginas JSP

3.7.4 Sintaxe

Na sintaxe *JSP* é apresentada dois tipos de dados para a formação de uma página:

- Estáticos - São informações inseridas que não possuem relação alguma com a sintaxe *JSP*, como *HTML* e *XML*, portanto, não podem ser interpretadas.
- Elementos sintáticos – São informações que fazem parte da sintaxe *JSP*. Há três tipos de elementos sintáticos:

    1. Diretivas;
    2. Ações;
    3. Elementos de Script.

3.7.5 Diretivas

No momento da tradução das páginas *JSP*, as diretivas são essenciais, pois por meio delas são obtidas informações globais. A sintaxe básica é:

<%@ diretiva %>

Falaremos mais sobre as diretivas ainda neste capítulo.

3.7.6 Ações

As informações fornecidas pelas ações são utilizadas no momento da execução, ou seja, durante a fase de processamento de uma solicitação. Por estar ligada à fase de solicitação, a interpretação de uma ação depende diretamente dos detalhes da solicitação recebida.

As ações podem ser separadas em dois grupos: Padrões e Personalizadas,

No primeiro grupo estão aquelas definidas pela especificação *JSP* 1.2 no segundo estão as *Tags* personalizadas. Ambas empregam a sintaxe XML.

A ação jsp:include é um exemplo de ação padrão, ela faz a inclusão de um documento, estático ou dinâmico, na página atual. Sintaxe:

<jsp:include page="busca.jsp" flush="false" />

3.7.7 Elementos de Script

Os elementos de *script* têm como função unir as informações estáticas e as dinâmicas. Há três tipos de elementos de *script*:

- Declarações - São utilizadas para declararmos métodos e variáveis a serem utilizados pela página. Elas são executadas apenas uma vez, quando da primeira solicitação. A sintaxe básica é:

  <%! declaração %>

- Scriptlets - São utilizados com códigos de programação e serão executados a cada nova solicitação da página. A sintaxe é:

  <% código; %>

- Expressões - São utilizadas para que sejam gerados os dados que serão exibidos ao usuário na resposta. Assim como os scriptlets, as expressões são executadas a cada nova solicitação. A básica é:

  <%= expressão %>

3.7.8 Exemplo

Exemplo 1: um.jsp

```
<%@ page import="java.util.*" %>
<html>
<body>
<%! int i, t; %>
```

```
<%
        for(i=0;i<5;i++)
        {
                %><h1> <%
                out.print("Valor "+i); %>
                </h1><%
        }
%>
<%= i+i %>
</body>
</html>
```

No exemplo acima temos a utilização de vários dos recursos *JSP* citados anteriormente, onde o seu resultado esta presente na figura 05.

Como já comentado, temos, junto aos comandos *JSP*, várias *tags HTML*. Podemos diferencia-las pela utilização das *tags JSP* que iniciam e terminam códigos da linguagem.

Na primeira linha utilizamos a diretiva *page* com o atributo *import*, que tem como finalidade importar um pacote para esta página. Recursos semelhantes são facilmente encontrados em diversas linguagens de programação. Conforme mostrado neste exemplo, vemos a possibilidade da utilização de caracteres coringas (*) para a busca de vários pacotes em uma única vez.

Outro exemplo da diretiva *page* é na utilização de páginas de erro:

<%@ page errorPage ="erro.html" %>

Quando, na execução da página *JSP*, algum erro acontece, uma página de erro é mostrada ao usuário que nem sempre é muito amigável. Com a diretiva acima, no momento que houver o erro, a página erro.html será chamada, onde nela podemos personaliza-la e dar as devidas informações ao usuário.

Na linha 4 encontramos uma declaração de variáveis. Para a declaração de variáveis utilizam-se os mesmos tipos da linguagem *JAVA*: int, String, double, char, boolean, entre outros.

Logo abaixo temos uma estrutura de repetição (for), onde dentro dela há comandos *HTML*. Em situações como essa devemos tomar cuidados redobrados, pois a *tag JSP* deve ser fechada antes do comando *HTML* e reaberta após.

E ao fim utilizamos o último elemento de script, que são as expressões. Nela geramos dados com base em informações e exibimos ao usuário.

Como já estudamos anteriormente, a página exemplificada será executada no servidor e retornará apenas código *HTML*, que será semelhante ao do quadro abaixo:

```
<html>
<body>
<h1> Valor 0
</h1><h1> Valor 1
</h1><h1> Valor 2
</h1><h1> Valor 3
</h1><h1> Valor 4
</h1>10
</body>
</html>
```

A página visualizada ficará como na figura a seguir:

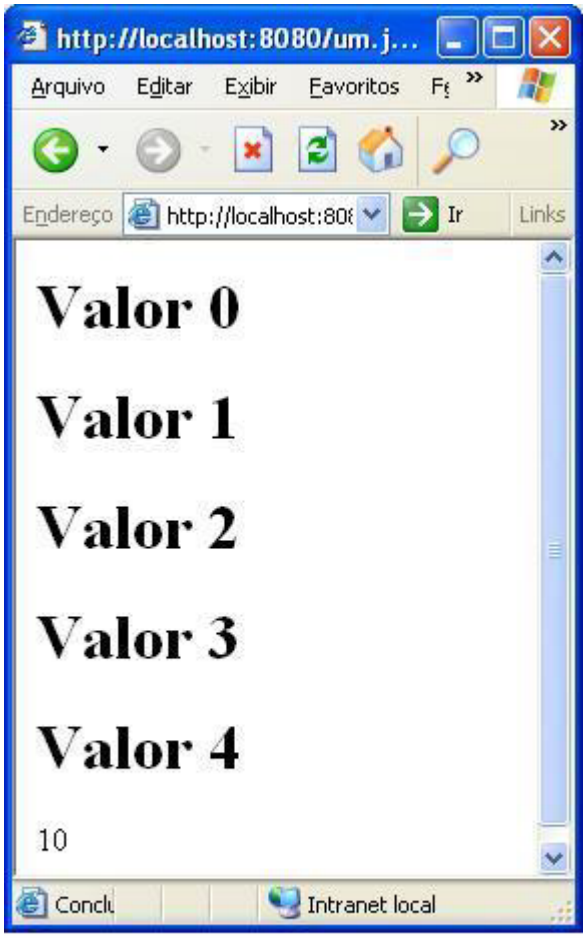


Figura 05 - Visualização da página JSP no cliente

3.7.9 Diretivas

Há, dentro da especificação JSP, duas diretivas principais: page e include.

3.7.9.1 Diretiva Page

Possui a seguinte sintaxe:

<%@ page a1=v1 a2=v2... %>

Onde a1 e a2 são atributos e v1 e v2 são os valores.

Os atributos desta diretiva podem ser declarados em uma única linha ou em linhas separadas, mas nunca podem ser repetidos na mesma página.

- Atributo language - Tem como finalidade especificar a linguagem a ser utilizada, apesar de a especificação *JSP* apenas citar a possibilidade da utilização da linguagem *JAVA*.
- Atributo extends - Define se a superclasse deve ser gerada.
- Atributo import - Conforme já citado anteriormente tem como finalidade importar um pacote para a página *JSP*.
- Atributo session - Define se a página faz parte ou não de uma sessão.
- Atributo buffer - Através deste atributo definimos o tamanho do buffer para as informações enviadas ao objeto *Response*. Caso não seja definida, será utilizado 8kb como tamanho.
- Atributo autoFlush - Indica se, quando estiver cheio, o buffer deve ser esvaziado.
- Atributo isThreadSafe - Informa se a página *JSP* pode ou não lidar com solicitações simultâneas.
- Atributo info - Neste atributo pode-se armazenar qualquer informação em forma de cadeia, esses dados podem ser recuperados a qualquer momento.
- Atributo isErrorPage - Utilizada para definir se uma página pode atuar como uma página de erro para uma outra página *JSP*.
- Atributo errorPage - Conforme já mencionado serve para definirmos uma página em caso de ocorrer algum erro na atual.
- Atributo contetType - Define, para o conteúdo enviado como resposta, o tipo *MIME*.
- Atributo PageEncoding - Utilizado para definir como será codificação da página *JSP*.

3.7.9.2 Diretiva include

Tem como finalidade incluir arquivos na página atual.

3.7.10 Utilização de Objetos

Como o *JSP* utiliza a linguagem *JAVA* ela também se aproveita de recursos existentes na linguagem, como a utilização de objetos.

Cada objeto que é criado, de forma implícita ou explicita, faz parte de um escopo, que nada mais é do que um atributo onde temos a referencia ao objeto e quando ela deixa de existir.

Os possíveis escopos são:

page – o objeto é acessível apenas na página onde foi criado;

request – o objeto é acessível pelas páginas que fazem a solicitação;

session – podem ser acessados por páginas da mesma sessão.

application – podem ser acessados por páginas da mesma aplicação.

3.7.10.1 Objetos Implícitos

Os objetos implícitos são: request, response, pageContext, session, application, out, config, page e exception.

- Objeto request - representa a solicitação que requisitou a página. Esse objeto é um dos mais complexos e mais utilizados. Ele é responsável por armazenar informações em atributos, recuperar essas informações e parâmetros, entre outras funções.
- Objeto response - este objeto é semelhante ao anterior. O request representa a solicitação já o response é a resposta.
- Objeto pageContext - este objeto tem como função facilitar o gerenciamento de sessões, atributos, páginas de erro, inclusões e encaminhamento de requisições.
- Objeto session - é um objeto muito importante, pois é através dele que é feito todo o controle das sessões nas páginas JSP. Ele armazena e recupera as informações dos atributos para transmitir para as outras páginas.

- Objeto application - representa a aplicação a qual as páginas pertencem. Normalmente utilizam o nome do diretório corrente como uma aplicação.
- Objeto out - este objeto é responsável pela saída de dados em uma página JSP. É nele que esta implementado, por exemplo, os métodos print e println que fazem a impressão de strings.
- Objeto config - neste objeto podemos armazenar configurações de inicialização.
- Objeto page - o objeto page representa a própria página JSP.
- Objeto exception - é uma instância da classe Throwable que é a superclasse das exceções em JAVA. Este objeto apenas estará disponível se o atributo isErrorPage estiver configurado como true na diretiva page.

3.8 Conclusão

Após fazermos a análise de quatro linguagens, mais uma vez a questão da melhor linguagem entra em cena, porque foi escolhido *JSP*? Alguns motivos nos levaram a utiliza-la:

- Por ser baseada em *JAVA* e esta estar em constante evolução;
- É totalmente *Free*;
- Possui uma grande empresa por trás no desenvolvimento (*Sun*);
- Os demais softwares por ela utilizados podem ser também adquiridos gratuitamente;

Com estes motivos não estamos afirmando que as demais linguagens citadas anteriormente não seriam capazes de fazer o mesmo que o *JSP*, mas apenas uma poderia ser escolhida.

4 WAP

O *WAP* (*Wireless Application Protocol*, ou Protocolo para Aplicações Sem Fio) é o resultado dos esforços do *WAP Forum*, visando promover uma tecnologia padronizada a ser utilizada pela industria no desenvolvimento de aplicações e serviços para ambientes móveis.

Ele especifica um ambiente de aplicação e uma pilha de protocolos para dispositivos sem fio, como telefones celulares, *pagers* e *PDAs* (*Personal Digital Assistants*, ou Assistentes Pessoais Digitais).

Grande parte das tecnologias desenvolvida para a Internet é destinada à maquinas com alto poder de processamento, grande quantidade de memória, com uma boa largura de banda e redes confiáveis. Por isso, a utilização em dispositivos sem fio obrigou a criação de um padrão que leve em consideração as características dos dispositivos móveis:

- Menor poder de processamento;
- quantidade de memória reduzida;
- consumo restrito de energia;
- pequenos visores;
- diferentes dispositivos de entrada de dados;
- pouca largura de banda;
- pequena estabilidade entre e durante a conexão;

4.1 Dispositivos WAP

O protocolo pode ser utilizado com um grande grupo de aparelhos móveis como telefones, *pagers*, rádios, *smartphones*, etc.

Para desenvolvimento de aplicações pode utilizar-se das linguagens *WML* e *WMLScript*, desde que o aparelho suportem essa nova tecnologia, os mais modernos já possuem este suporte. Os celulares, por exemplo, vêm equipados com um pequeno navegador para visualização de páginas *WEB* que, em virtude das características dos aparelhos, deverão ser construídas de forma bastante simples.

A linguagem *WML* (Wireless Markup Language) é similar a *HTML* e baseada no padrão *XML*, lido e interpretado por um pequeno navegador instalado num dispositivo (um celular) *WAP*. Para ela também temos novos tipos de objetos, em especial *WBMP* (*Wireless Bitmap*) que são imagens mostradas nos dispositivos *WAP*, feitos em preto e branco. Os navegadores que conhecemos interpretam páginas *HTML* da *Web*. Um navegador *WAP* é um software similar, muito mais simples, desenvolvido para funcionar em celulares e outros aparelhos sem fio, interpretando páginas *WML*. Os *sites WAP*, criados na linguagem *WML*, são criados baseados em texto e com poucas imagens, sempre monocromáticas que lembram muito os primeiros sites da *Web*.

O *WMLScript* é uma linguagem de *script* que funciona semelhantemente ao *JavaScript,* que é executada no cliente, mas muito limitado.

O *WBMP* é o padrão de imagens utilizado nas páginas *WML*. Em sua versão 1.0, o *WBMP* não possui suporte para imagens coloridas.

4.2 Conexão

A conexão de um dispositivo móvel com a *Internet* funciona de forma semelhante ao computador, ou seja, o aparelho faz a conexão com a rede utilizando-se de um modem interno e recebe um número *IP* da operadora. Lembrando que apenas os modelos compatíveis com a tecnologia *WAP* podem acessar a rede.

O aparelho visualiza as páginas *WAP* através de um pequeno navegador, que é uma versão reduzida dos navegadores utilizados na *Web*, preparado especialmente para esta finalidade. Com este pequeno navegador podemos acessar

pelo conteúdo de um site *WAP* (texto, alguma imagem e com a diagramação em menus) e utilizar os seus serviços (formulários).

Os navegadores podem ter características diferentes uns dos outros, de forma que páginas podem ser visualizadas com pequenas diferenças, semelhante ao que acontece com os navegadores *WEB*. Com eles podemos utilizar todos os serviços que já estamos acostumados na *WEB* e fazem parte do dia-a-dia, como lojas virtuais, portais, agendas, e-mail, etc, desde que tenham sido construídos para a tecnologia *WAP*. Já existem hoje diversos sites *WAP* na *Internet* de empresas que disponibilizam seus serviços produtos e informações diversas. Isso esta gerando novas perspectivas no mercado de trabalho para diversas áreas como *webmasters*, programadores, consultores, jornalistas e profissionais de tecnologia de informação.

4.3 WML

A linguagem de programação utilizada para a criação de conteúdos para a tecnologia *WAP* é o *WML*, *Wireless Markup Language,* onde, segundo AREHART (2001, p. 37) “é uma linguagem de marcação semelhante ao *HTML* baseada no *XML*”.

O *WML* é uma linguagem *case sensitive*, ou seja, os seus comandos são interpretados de forma diferente quando utilizamos letras maiúsculas e minúsculas, por exemplo, Card e card são interpretados de forma diferente pelo *WML*.

Outra característica do *WML* é a sua forma de organização que é feita através de *Cards* e de *Decks*. Esse conceito utilizado é semelhante a um baralho de cartas, onde as cartas do baralho são os *Card* e os baralhos são os *Decks*. Quando o navegador faz o pedido de uma página, na verdade o que será carregado será um baralho, onde o dispositivo utilizado irá apresentar uma carta por vez.

Exemplo:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
        <card id="um" =”Exemplo Um”>
                <p> PRIMEIRA MENSAGEM </p>
        </card>
</wml>
```

Todo baralho começa com o mesmo cabeçalho *XML* que é formado pelas três primeiras linhas do exemplo acima. Nessas linhas informamos que o será usado comandos *XML* e também qual a versão esta sendo utilizada.

Todo o código *WML* de um baralho fica entre as *Tags* <wml> </wml>. Esse é o corpo do documento e as cartas são definidas usando-se as *Tags* <card> </card>.

Resultado:

Figura 06 - Exemplo de WML

4.4 Número de Usuários WAP

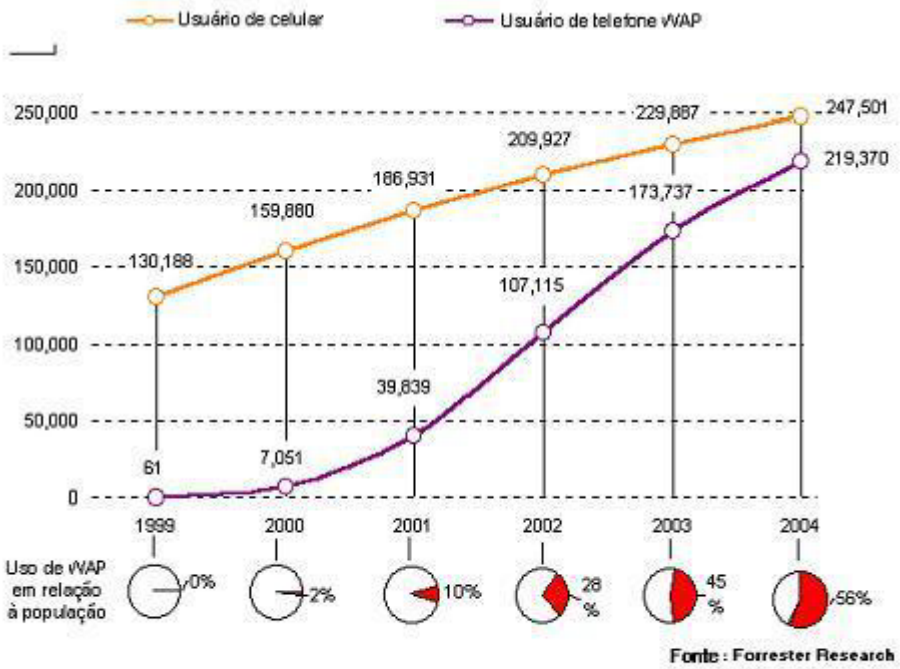


Figura 07 - Estimativas do número de usuários de tecnologia WAP na Europa

Pela estimativa mostrada no gráfico podemos ver que o padrão *WAP* tende a se estabelecer apesar dos problemas que ainda não foram totalmente resolvidos. Além dos problemas relacionados à segurança existem certas lutas dentro do *WAP Forum* pelo fato de algumas companhias integradas ao fórum seguem ainda fornecendo soluções que não tem base *WAP*. Por exemplo, Phone.com oferece ainda um navegador que não tem suporte *WML*. Outro problema é em relação a patentes, pois as especificações foram criadas por diversas companhias, como ocorre com a *Geoworks Corps de Alameda*, Califórnia. Esta companhia afirma ter uma patente do ano 1994 sobre tecnologia sem fio que consideram como parte *standard* do *WAP*.

5 AGENTES INTELIGENTES

> [..] é uma entidade que possui um sistema interno de tomada de decisões, age sobre o mundo e também sobre os outros além de funcionar sem a necessidade de qualquer outra coisa o guiar, ou seja, possui mecanismos próprios de percepção do exterior (SILVA, 2000, p. 02).

Em muitas situações é usado o termo agente, desde interfaces de usuários a complexos sistemas distribuídos. Um agente deve possuir algumas das características associadas com a inteligência humana: conhecimento, adaptabilidade, independência, criatividade, etc. É suposto que um usuário deva delegar uma tarefa ao agente e não comandar o agente para executar a tarefa. As dificuldades de definição do termo se deve principalmente ao uso indiscriminado do termo pelas indústrias de software, as quais utilizam-no de diferentes formas e com significados variados como um apelo para a aceitação de seus produtos no mercado. Uma maneira de substituir uma definição formal é enumerarmos uma lista das características gerais que devem ser associadas a um agente:

- Autonomia - Agentes devem operar sem a intervenção humana e devem ter controle sobre suas ações;
- Cooperação - Agentes interagem com outros agentes e possivelmente seres humanos;
- Reatividade - Um agente conhece seu ambiente e responde prontamente a alterações que ocorram neste;
- Continuidade temporal - Agentes são processos que estão continuamente rodando podendo estar ativos ou passivos;

- Mobilidade - A habilidade de um agente em se mover sobre uma rede eletrônica;
- Benevolência - Os agentes não possuem conflitos de objetivos e sempre tentarão fazer o que for solicitado a eles;
- Racionalidade - Um agente sempre vai agir de forma a alcançar seus objetivos;
- Aprendizado/Adaptabilidade - Um agente deve estar apto a aprender e a se ajustar aos hábitos, métodos de trabalho e preferências do usuário;
- Colaboração - Um agente não deve impensadamente executar instruções, mas deve considerar que os seres humanos cometem erros, seja através de instruções mal formuladas ou omissão de informações. Um agente pode também se recusar a executar determinadas tarefas se estas causarem danos para outros usuários.

5.1 Aplicações

Grande maioria das pesquisas desenvolvidas tem como foco aplicações básicas, pois os resultados são em curto prazo. Tais aplicações envolvem principalmente:

- Agentes que manipulam e-mails;
- Agentes que filtram e pesquisam através da rede novos artigos, procurando por informações de interesse do usuário;
- Agentes que planejam reuniões baseados em lista de participantes ou na agenda eletrônica de cada participante em particular.

Para as funcionalidades acima, há um grande número de produtos sendo comercialmente vendidos e protótipos em fase final.

Podemos identificar áreas de aplicações mais complexas, com trabalhos de pesquisa ainda em desenvolvimento, onde o uso de agentes pode trazer grandes benefícios:

- Gerenciamento de redes e sistemas - Simplificar o gerenciamento de sistemas complexos;
- Acesso Móvel - Agentes podem residir na rede e realizar as requisições do usuário enviando informações compactadas como uma forma de burlar as limitações tais como largura de banda;

- Gerenciamento e acesso à informação - Auxílio ao usuário para pesquisar e filtrar informações;
- Trabalho colaborativo - Suporte a equipes de design para compartilhar documentos ou recursos através da rede;
- Gerenciamento administrativo - Agentes podem ser usados para automatizar processos;
- Comércio eletrônico - Agentes podem ir às compras para os usuários a partir de especificações retornando sugestões de compras;
- Interface com o usuário - Agentes inteligentes permitem aos sistemas monitorar as ações do usuário ajudando-os em seus problemas.

A colaboração entre agentes e a mobilidade de agentes consistem em áreas que ganham espaço nas pesquisas e consistem em uma tendência para o futuro.

5.2 Sistemas Multiagentes

Agentes podem funcionar de maneira associada a outros Agentes para resolver um problema comum, essa associação é chamada Sistema Multiagente (*Multi-Agent System – MAS*). As razões para o crescente interesse na pesquisa em *MAS* incluem habilidades para:

- Resolver grandes problemas que muitas vezes são muito grandes para um único agente devido a limitações de recursos;
- Prover soluções que advém de fontes de informações distribuídas;
- Aumentar velocidade, confiabilidade (capacidade de se recuperar de uma falha), prover extensibilidade (capacidade de alterar o número de processadores dedicados a um problema);
- Oferecer clareza e simplicidade conceitual do projeto.

Para um *MAS* resolver problemas comuns os agentes devem se comunicar entre eles, coordenar as atividades e negociar quando entrarem em conflito. Conflitos podem resultar em problemas de recursos até questões mais complexas de computação. Uma boa organização é imprescindível para determinar a estrutura entre os Agentes e também para evitar problemas com alocação de tarefas e recursos.

5.3 Agentes Móveis

Uma importante área de pesquisa atualmente é a tecnologia de Agentes Móveis. Um Agente é móvel se for capaz de se locomover de máquina para máquina por conta própria.

Um Agente Móvel deve conter os seguintes modelos:

- Modelo de Agentes - Define a estrutura interna do agente inteligente. Esta estrutura especifica as características de autonomia, aprendizado, cooperação;
- Modelo de Ciclo de Vida - Define os diferentes estados de execução de um agente móvel e os eventos que causam movimento de um estado para outro;
- Modelo Computacional - Define como o agente móvel se comporta no estado "*running*".
- Modelo de Segurança – Define a proteção contra Agentes Destrutivos;
- Modelo Navegacional – Diz respeito aos aspectos de mobilidade do agente móvel;
- Modelo de Comunicação - Implementa um ou mais protocolos para a comunicação entre agentes móveis e outras entidades.

Muitos estudiosos da área afirmam que toda e qualquer tarefa que é executada pelos Agentes Móveis podem ser realizadas de outras formas, como a computação cliente-servidor, entretanto acredita-se que os Agentes Móveis possuem vantagens sobre as abordagens convencionais. Algumas vantagens são:

- Baixo consumo dos recursos de rede;
- Redução no tráfego na rede;
- É uma boa alternativa para interações com entidades de tempo real, onde atrasos não podem ser tolerados;
- Sistemas que utilizam Agentes Móveis são independentes do computador e da rede.

5.4 Conclusão

Com este breve comentário foi possível analisar que a utilização dos Agentes, desde que de forma correta, pode ser muito interessante principalmente

para ganho de processamento e organização das tarefas dentro do sistema, mas infelizmente em tarefas simples que poderíamos utilizá-los não o fazemos.

6 O MODELO PROPOSTO

O objetivo desde modelo de sistema contábil é agilizar a troca de documentos entre escritório e o seu cliente e também uma maior organização de ambas as partes quanto a contas (tributos, taxas, etc.).

6.1 Ambiente de Software

O protótipo foi desenvolvido utilizando um conjunto de tecnologias de *software*:

- *JSP*;
- *HTML*;
- Banco de dados *MYSQL*;
- Servidor *WEB Tomcat.*
- *WML;*

6.2 Representação do Processo

O processo de acesso ao sistema é representado como na ilustração abaixo:

Figura 08 - Representação do acesso no Modelo

Para o reconhecimento pelo sistema e saber que tipo de usuário esta acessando o sistema, se cliente ou administrador, há na tabela de usuários um atributo responsável pela definição do privilégio do usuário.

6.2.1 Acesso Cliente

No modelo o cliente tem permissão para realizar apenas dois tipos de ações:

- Consultas – O cliente pode consultar suas informações cadastrais e também contas previamente cadastradas pelo administrador, que pode ser feito via *WEB* ou *WAP*;
- Relatórios – O cliente poderá realizar a impressão das consultas e também os formulários para pagamento das contas, que pode ser feito somente através do módulo *WEB*;

6.2.2 Acesso Administrador

O administrador terá, obviamente, maior direito de acesso que o cliente:

- Consultas – Poderá consultar, via *WEB* ou *WAP*, informações cadastrais dos demais e também detalhes dos documentos de todos os clientes;
- Relatórios – A impressão poderá ser feita de todos os dados cadastrados e também dos documentos, somente via *WEB*;
- Cadastros – O Administrador tem total liberdade de cadastros, tanto de novos clientes quanto de documentos;

6.3 Especificações do sistema

Para o desenvolvimento do modelo proposto utiliza-se a união de algumas tecnologias conforme ilustração abaixo:

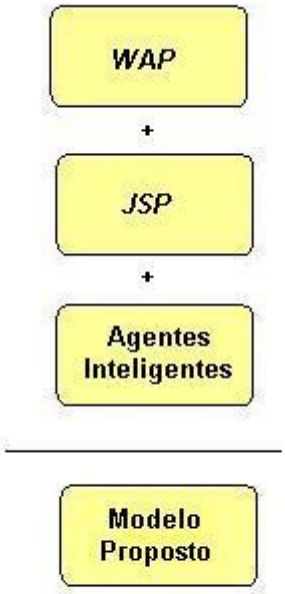


Figura 09 - Tecnologias utilizadas no modelo

- *WAP* – Será utilizado para acesso ao sistema por dispositivo móvel sem fio;
- *JSP* – Será utilizado para construção do portal do escritório contábil;
- Agentes Inteligentes – Esta tecnologia será utilizada para automatizar algumas ações para facilitar o trabalho do contabilista.

6.3.1 Tecnologias utilizadas

Abaixo uma breve descrição das tecnologias utilizadas:

6.3.1.1 WAP

Através desde modulo ambas as partes envolvidas no processo, poderão acessar as informações do sistema utilizando-se de, por exemplo, um telefone celular compatível, lembrando que os aparelhos mais recentes já possuem este recurso.

Esta é uma tecnologia que ainda esta sendo pouco explorada pelas empresas devido a vários fatores, entre eles podemos citar:

- Custo de utilização do telefone – Para a utilização do recurso há o custo da ligação telefônica, o que ainda para nossos padrões ainda é um alto preço,

visto que a grande maioria das pessoas que tem telefone celular utiliza a modalidade pré-pago para fins de economia;

- Baixa velocidade de transmissão – A velocidade de transferência de dados ainda é muito baixa, fazendo que a consulta deva ser feita sempre de poucos dados para que a conexão seja rápida;
- Restrição quanto aos aparelhos – Os aparelhos existentes possuem um visor pequeno que resulta em um certo desconforto na utilização. Já existem aparelhos com visores maiores e até coloridos, mas possuem alto preço para aquisição. Outro problema é quanto ao teclado que se mostra também muito incomodo na sua utilização.

Mesmo com essas restrições a utilização do *WAP* é de grande importância, pois em momentos de urgência pode-se utilizar para fazer pequenas consultas como:

- Se há alguma conta a ser paga;
- se o cliente fez a confirmação do pagamento;
- cerificação de dados dos clientes;
- acesso a índices econômicos, entre outros.

- **Estrutura**

Em virtude dos fatores de problemas citados acima este módulo deverá ter uma estrutura bem reduzida em relação ao portal.

Ao entrar no *site WAP* o usuário deverá entrar com seu *Login* e sua Senha. Como tanto o cliente como o administrador do sistema poderão entrar no sistema, deverá haver uma diferenciação de comportamento do sistema dependendo do usuário. A idéia é que se crie na tabela de usuários no banco de dados um campo "*Privilegio*", onde saberemos se o usuário irá se conectar como Administrador ou como Cliente do escritório. Caso o usuário tenha privilégio de Administrador ele poderá acessar informações do tipo:

- Clientes em débito;
- débitos por clientes;
- informações cadastrais dos clientes;
- contas pagas pelos clientes;
- índices econômicos.

E caso o usuário se conecte ao sistema como Cliente poderá acessar as suas informações abaixo:

- Débitos;
- informações cadastrais;
- contas pagas;
- índices econômicos.

Como já descrito, as informações deverão ser apresentadas de forma reduzida para que a conexão seja o mais breve possível.

6.3.1.2 JSP

Este módulo refere-se ao portal da contabilidade. A idéia é que se mostrem informações não apenas para cliente e contabilista, mas sim para a comunidade em geral que necessite de informações contábeis, além da apresentação do próprio escritório.

- **Estrutura**

O método de conexão segue a mesma forma do módulo *WAP*, aonde através de um *Login* e uma senha o usuário irá se conectar ao sistema e irá acessar informações dependendo do seu privilégio.

Entre as informações do portal que não necessitam que o usuário faça o *Login* podemos citar:

- Dados e histórico do escritório;
- principais clientes;
- serviços prestados;
- perguntas e respostas para assuntos contábeis;
- downloads de softwares contábeis;
- agenda com datas para pagamentos de tributos. Essas datas não são as mesmas dos documentos a pagar, mas sim datas que são fixas.
- contato com o escritório;
- links;
- Informações para abertura de empresas;
- índices econômicos.

Como já comentado somente usuários terão acesso a determinadas áreas do portal, onde haverá diferenças entre conteúdo para o administrador e para o cliente.

Se a conexão for feita pelo administrador, ele poderá, entre outras coisas:

- Cadastrar e alterar informações dos clientes. Entre essas informações esta a criação de um *Login* e uma senha;
- cadastrar os documentos;
- documentos que ainda estão em aberto;
- documentos já pagos;
- visualizar e imprimir documentos. Estes documentos podem ser: Darf (Documento de arrecadação de receitas federais), GRFC (Guia de recolhimento rescisório do FGTS e da Contribuição Social), DAR (Documento de arrecadação estadual), GPS (Guia da Previdência Social), entre outros.

Em caso da conexão ao sistema ser feita através de um usuário com privilégio de cliente, este terá um acesso restrito a poucas opções, sendo elas restritas a informações a ele próprio:

- verificar *status* dos documentos;
- visualizar e imprimir documentos;
- informações cadastrais.

6.3.1.3 Agentes Inteligentes

O uso dos agentes no modelo serve para que algumas tarefas que deveriam ser feitas pelo administrador possam ser realizadas de forma automatizada e transparente. Essas tarefas seriam principalmente as seguintes:

- Informar aos clientes, através de e-mail ou mensagens *SMS*, a respeito da necessidade de efetuar algum tipo de pagamento;
- Informar ao administrador, através de e-mail ou mensagens *SMS*, que algum cliente deixou de entrar no sistema para verificação de documentos;
- Informar ao administrador, através de e-mail ou mensagens *SMS*, se algum documento deixou de ser pago;

A implementação dos Agentes ainda não há uma forma definida, mas podemos considerar duas formas a se propor:

- Diretamente na base de dados – Seria uma forma mais simples de implementação, onde seriam utilizados recursos do *SGBD* do sistema, onde utilizaríamos Agentes do tipo mais simples, os chamados “reativos”. Esta implementação poderia ser feita através de *triggers*, onde ela seria executada pela combinação de situações, por exemplo: Todo documento possui uma data de vencimento, então se a data de vencimento for igual a data do sistema e ele não foi pago ainda então o cliente e o administrador deve ser informado que este é o último dia para pagamento.
- No sistema interno – Quase todo escritório contábil hoje em dia já é informatizado. Com a possibilidade da implementação do sistema atual juntamente do sistema *WEB* pode-se utilizar os Agentes internamente ao sistema interno. A dificuldade nesta forma seria a implementação, visto que grande parte dos sistemas existente no mercado não possui este recurso, o que acarretaria no desenvolvimento de um novo.

6.3.2 Diagrama ER – Entidade Relacionamento

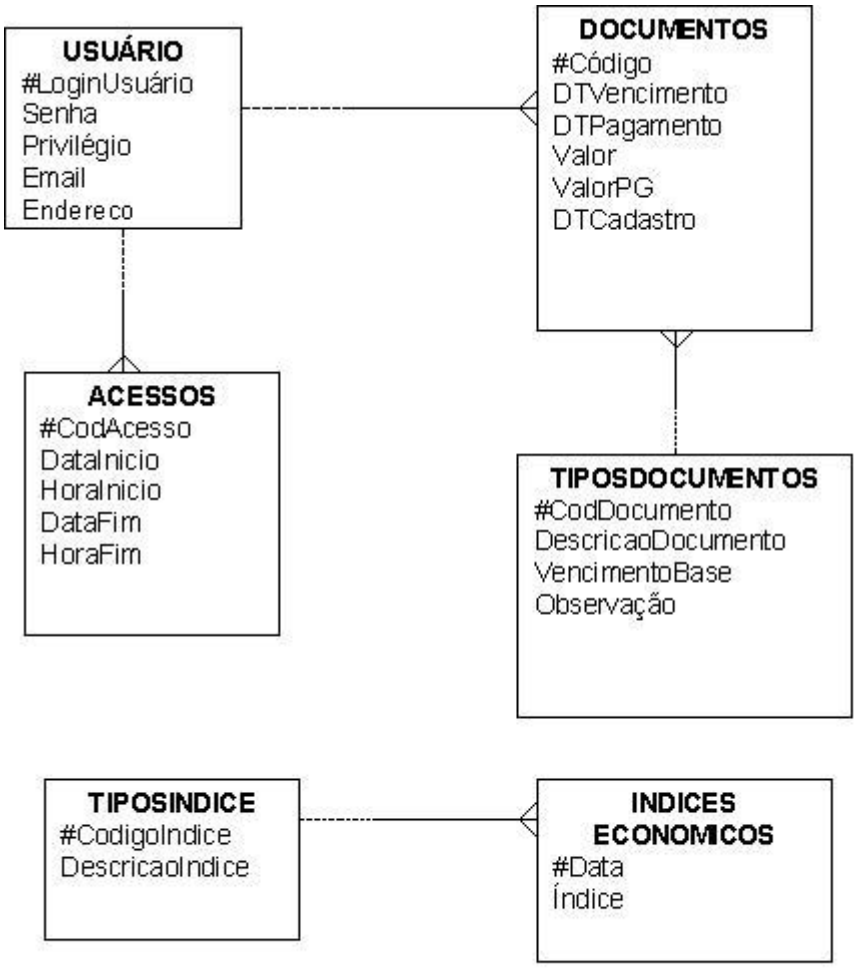


Figura 10 - Diagrama ER - Entidade relacionamento

6.3.3 Dicionário de Dados

6.3.3.1 Descrição das tabelas

Nome	Descrição
Usuário	Cadastro dos usuários que podem acessar o sistema
Acessos	Mantém informações sobre o acesso ao sistema pelos usuários.
TiposDocumentos	Tipos de Documentos para cobrança.
Documentos	Documentos para cobrança.
TiposIndice	Tipos de índices econômicos.
IndicesEconomicos	Tabela com os Índices econômicos.

6.3.3.2 Descrição dos Atributos

- **Tabela - Usuário**

Nome	Descrição	Tipo	Tamanho	PK
LoginUsuário	Login do usuário	Varchar	10	X
Senha	Senha do usuário	Varchar	10	
Privilégio	Tipo do privilégio do usuário	Integer		
Email	Email do usuário	Varchar	50	
Endereço	Endereço do usuário	Varchar	50	

- **Tabela – Acessos**

Nome	Descrição	Tipo	Tamanho	PK
CodAcesso	Código do acesso efetuado	Integer		X
DataInicio	Data inicial do acesso	Data		
HoraInicio	Hora inicial do acesso	Hora		
DataFim	Data final do acesso	Data		
HoraFim	Hora final do acesso	Hora		
LoginUsuário	Login do usuário que realizou o acesso	Varchar	10	

- **Tabela – TiposDocumentos**

Nome	Descrição	Tipo	Tamanho	PK
CodDocumento	Código do tipo do documento	Integer		X
DescricaoDocumento	Descrição do documento	Varchar	10	
VencimentoBase	Dia base de vencimento no mês	Integer		
Observação	Observação sobre o documento	Varchar	70	

- **Tabela - Documentos**

Nome	Descrição	Tipo	Tamanho	PK
Código	Código do documento emitido	Integer		X
DTVencimento	Data de vencimento	Data		
DTPagamento	Data em que foi pagamento	Data		
Valor	Valor do documento	Numérico	10,2	
ValorPG	Valor pago	Numérico	10,2	
DTCadastro	Data em que o documento foi cadastrado	Data		

- **Tabela – TiposIndice**

Nome	Descrição	Tipo	Tamanho	PK
CodigoIndice	Código do indice	Integer		X
DescricaoIndice	Descrição do índice	Varchar	40	

- **Tabela – IndicesEconomicos**

Nome	Descrição	Tipo	Tamanho	PK
Data	Data do índice	Data		X
CodigoIndice	Código do índice	Integer		X
Índice	Índice econômico	Numeric	5,3	

6.3.4 Diagrama Hierárquico das Funções

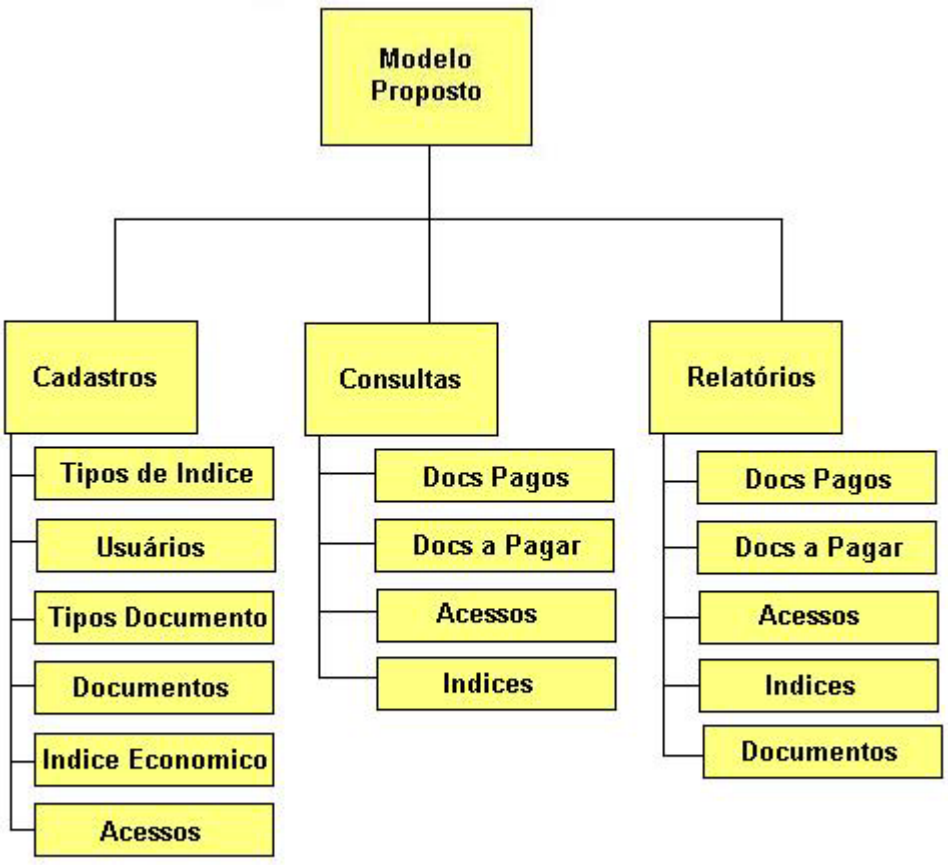


Figura 11 - Diagrama Hierárquico das Funções

6.3.5 Matriz de Uso: Entidade x Funções

	Usuário	Acessos	TiposDocumentos	Documentos	TiposIndice	Índices Econômicos
Cadastro Tipos de Índice					CIEA	

Cadastro Usuários	CIEA					
Cadastro Tipos Documento			CIEA			
Cadastro Documentos			C	CIEA		
Cadastro Índice econômico					C	CIEA
Cadastro Acessos	C	CIA				
Consulta Docs Pagos	C		C	C		
Consulta Docs a Pagar	C		C	C		
Consulta Acessos	C	C				
Consulta Índices					C	C
Relatório Docs Pagos	C		C	C		
Relatório Docs a Pagar	C		C	C		
Relatório Acessos	C	C				
Relatório Índices					C	C
Relatório Documentos	C		C	C		

Legenda:

C - Consultar

I - Incluir

E - Excluir

A – Alterar

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 Conclusões

Este trabalho tratou da apresentação de um modelo de sistema contábil com a intenção de facilitar a troca de documentos e acesso a dados de contabilistas e seus clientes.

Durante o desenvolvimento do projeto foram encontradas muitas dificuldades:

- Utilização de novas tecnologias – Como algumas tecnologias utilizadas são recentes, a aquisição de bibliografias tornou-se um pouco difícil o que em determinadas ocasiões atrasou o cronograma inicial;
- Compreender o funcionamento da cobrança de tributos – Houve a necessidade de entendimento quanto os métodos, datas e tipos de cobrança realizada pela contabilidade.

Esse trabalho também apresenta a possibilidade de criação de um grande software contábil que reúna todos os sistemas de uso interno do escritório com o sistema proposto. Com esse pensamento, haveria a possibilidade do desenvolvimento de um sistema, de um único desenvolvedor, que atendesse todas as necessidades e que tenha os novos recursos apresentados, inclusive utilizando o conceito de Agentes inteligentes diretamente no sistema.

Entretanto para realizar este novo software seriam necessários novos estudos na área contábil. Estes estudos deveram ser realizados em todas as áreas contábeis: Tributos, fiscal, leis trabalhistas, entre outros.

7.2 Trabalhos Futuros

Como idéias para trabalhos de complementação fica a possibilidade, como já citado, de desenvolvimento de um software contábil que se interligue com o sistema proposto.

Outra possibilidade seria a adaptação do modelo para outras áreas desenvolvidas em nossa região onde haja a necessidade de troca de documentos de cobrança, como:

- Indústria – Uma empresa cerâmica da região poderia mais facilmente controlar os pagamentos com seus clientes;
- Comércio – Um cliente assíduo de uma loja de confecções poderia ter acesso a suas contas a pagar;
- Prestação de serviços – O paciente de uma clinica médica poderia acessar suas informações;

Enfim a apresentação deste modelo de sistema abre uma gama de aplicações a serem desenvolvidas que trazem a possibilidade de implantação dele com poucas alterações.

8 REFERÊNCIAS

ANSELMO, Fernando. **PHP e MySQL para Windows**. Florianópolis: Visual Books, 2000.

AREHART, Charles et al. **Professional WAP**. São Paulo: Makron Books, 2001.

ARTIFICIAL INTELLIGENCE GROUP. Disponivel em:. <http://www.cs.tcd.ie/research_groups/aig/old_pages>. Acesso em: 15/02/2003.

ASPBRASIL. Disponível em:. <www.aspbrasil.com.br> . Acesso em: 07/01/2003.

BIANCHI, Luiz. **Curso Prático de Informática Básica**. Blumenau: Acadêmica, 2000.

BOMFIM. Francisco Tarcizo. **JSP, A Tecnologia Java Na Internet**. São Paulo: Érica, 2002

CLIX. Disponível em:. < http://bvi.clix.pt/> . Acesso em: 05/01/2003.

CONVERSE, Tim. **PHP 4: A bíblia**. Rio de Janeiro: Campus, 2001.

COULDFUSION BRASIL. Disponível em:.< http://www.coldfusionbrasil.com.br>. Acessado em: 13/01/2003.

EDDINGS, Joshua. **Como funciona a Internet**. 2. ed. São Paulo: Quark, c1994.

FACUNTE, Emerson. **WAP: guia de tecnologia**. Rio de Janeiro: Brasport, 2000.

FORT, Ben. **Desenvolvendo WAP com WML e WMLScript**. Rio de Janeiro: Campus, 2001.

FRANKLINT, Kleitor. **ASP: Active Server Pages : técnicas e estratégias**. São Paulo: Érica, 2001.

IMASTER. Disponível em:.< http://www.imasters.com.br/>. Acessado em: 13/01/2003.

MACORATTI, José Carlos. **ASP, ADO e banco de dados na internet**. 2. ed. Florianópolis: Visual Books, 2000.

MACROMEDIA BRASIL. Disponível em:. < http://www.macromedia.com/br/>. Acesso em: 08/01/2003.

MUKHI, Vijay. **Java Servlets JSP**. São Paulo: Makron, 2002.

MYSQL. Disponível em:. <http://www.mysql.com/>. Acesso em: 01/11/2002.

PHP BRASIL. Disponível em:. < http://www.phpbrasil.com/>. Acesso em: 10/01/2003.

RUAS, Nilson. **Criando sites web com HTML**. Florianópolis: Visual Books, 2002.

SCHIAVONI, Flávio Luiz. **Agentes Autônomos Inteligentes**. Disponivel em:. <http://www.em.pucrs.br/manuals/agentes>. Acessado em:15/02/2003.

SCHMITZ, Richard S. **WAP aplicado ao monitoramento remoto de informações - Um hospital Virtual**. Mestrado – UFSC, Florianópolis.

SILVA, Henrique Oliveira da. **Agentes EDI como ferramenta de apoio a Sistemas de Informação baseado em Web**. Doc. Porto Alegre, 2000.

SOARES, Walace. **Programando em PHP: Conceitos e Aplicações**. São Paulo: Érica, 2000.

TOLHURST, William A. **A Internet: um guia rápido de recursos e serviços**. Rio de Janeiro: Campus, 1994.

WEISSINGER, A. Keyton. **ASP: o guia essencial**. Rio de Janeiro: Campus, 2000.

Anexos

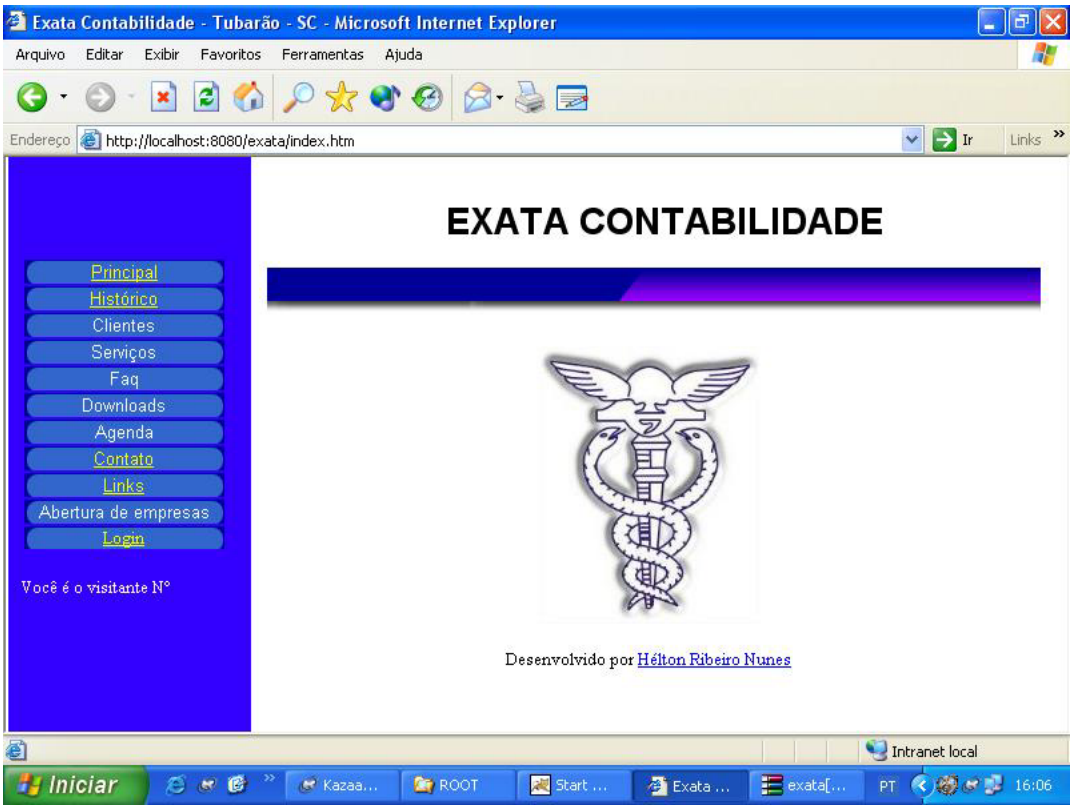
Protótipo

Como forma de demonstrar parte de nosso trabalho foi desenvolvido um protótipo em conjunto com o escritório contábil onde algumas funções já foram implementadas como será demonstrada a seguir.

Este protótipo foi desenvolvido com as seguintes tecnologias:

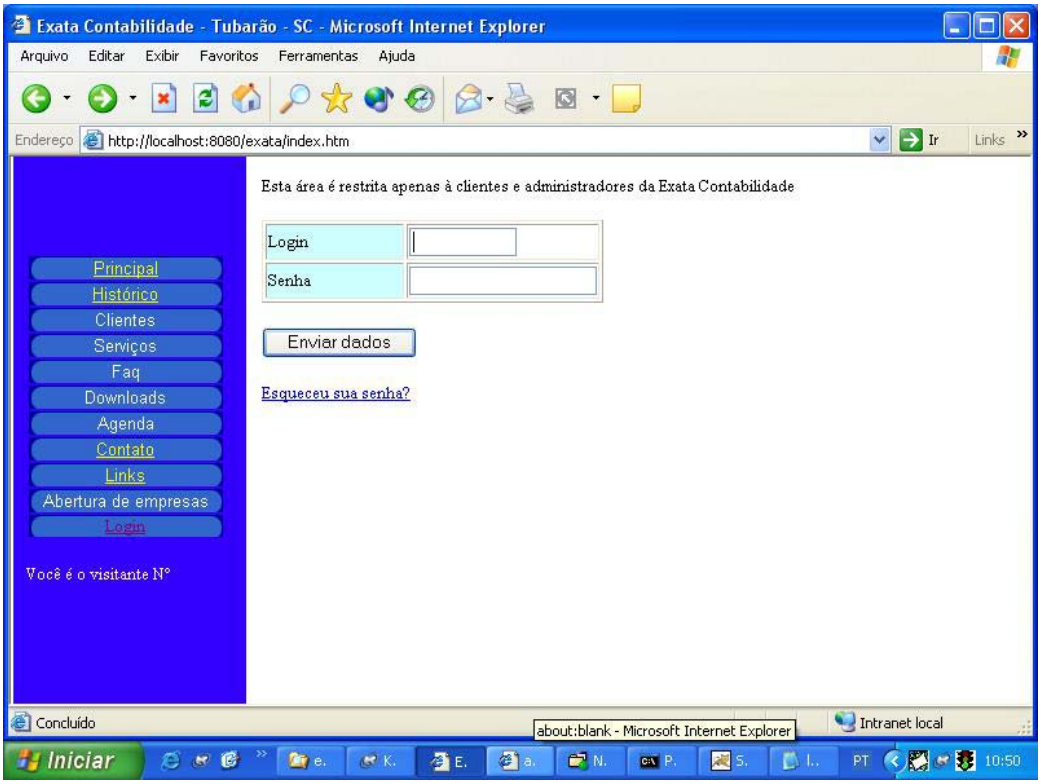
- HTML – Utilizado na construção da página em suas partes estáticas;
- JSP – Utilizado na construção da página em suas partes dinâmicas;
- MySQL – Banco de dados utilizado para armazenamento das informações.

Conexão



Na tela inicial do portal acima, além de apresentarmos a empresa (Exata Contabilidade), também o usuário poderá utilizar as opções do menu. A opção *Login* apenas usuários cadastrados podem fazer uso, apenas o responsável pelo escritório e seus clientes possuem este cadastro.

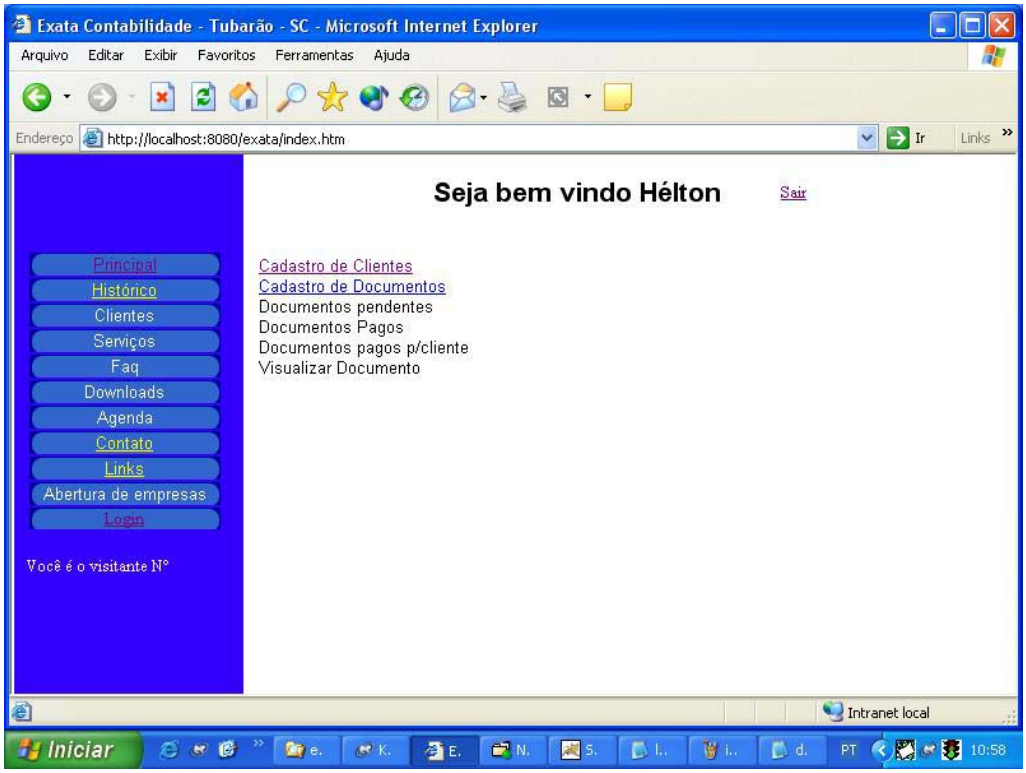
Com o cadastro feito, o usuário poderá fazer a sua conexão através da tela de *Login* abaixo:



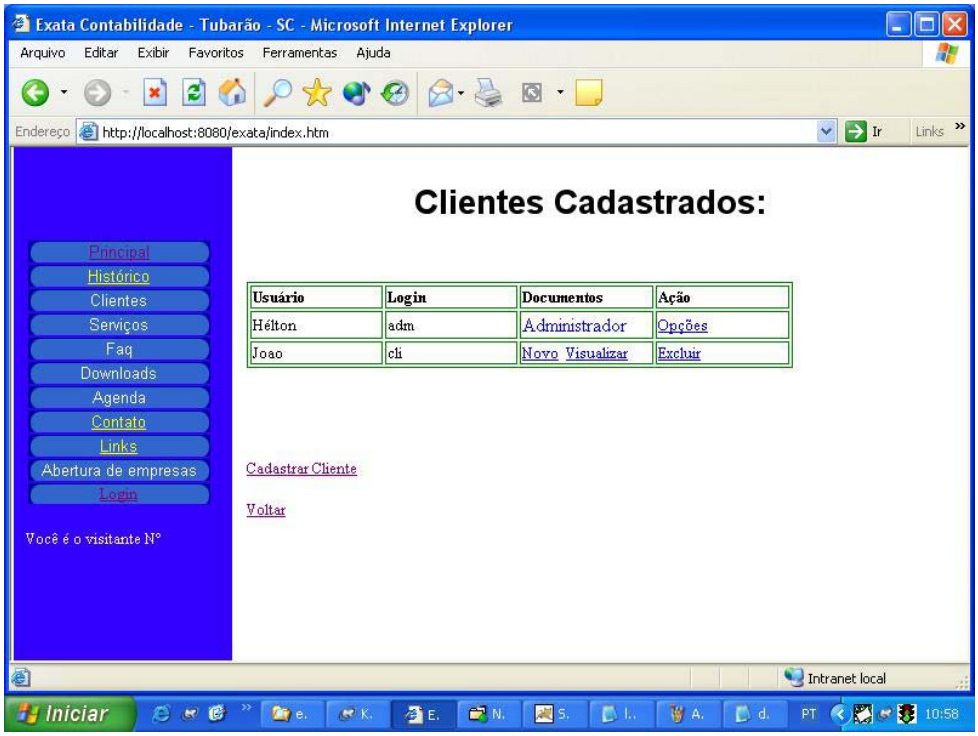
Para diferenciar se o acesso esta sendo feito pelo Administrador ou pelo Cliente, foi criado na tabela de usuários um campo que informa o privilégio de uso do usuário. Quando da conexão, é feita a consulta no banco de dados sobre existência do usuário e sua senha. Com esta validação realizada será verificado o seu privilégio de uso que levará para partes especificas do sistema.

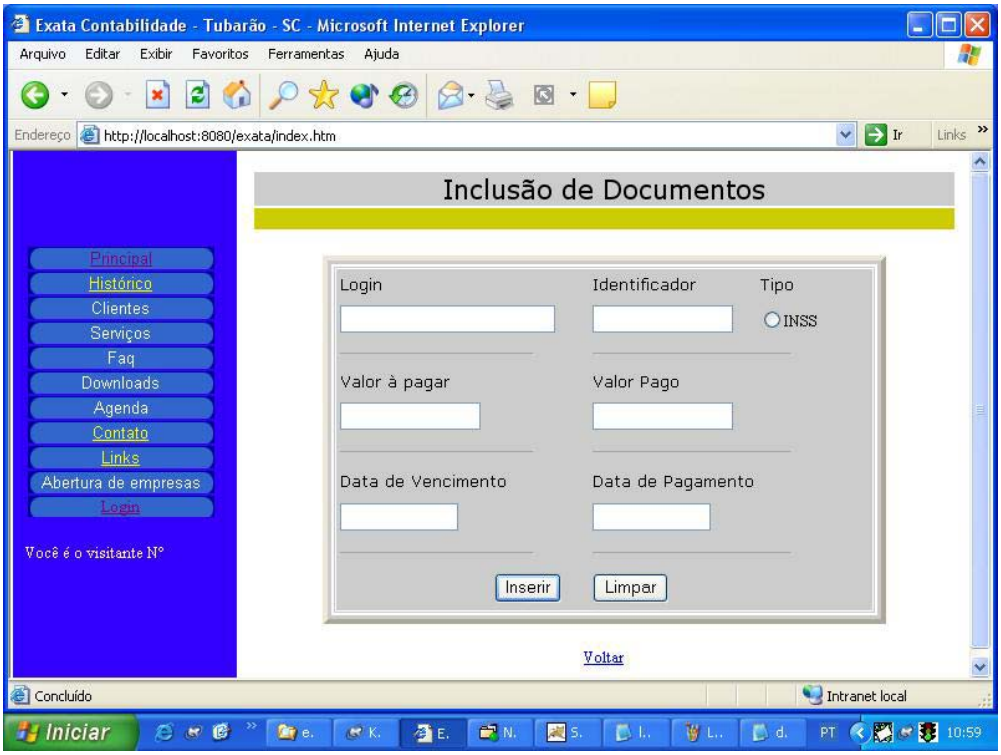
Acesso Administrador

No acesso do administrador, será apresentada a página a seguir.



Nesta página o usuário terá acesso a informações e ações exclusivas do contabilista como cadastrar novos usuários e documentos a serem acessados pelos clientes (abaixo) e emitir relatórios.





Acesso Cliente

Ao fazer o acesso com privilegio de cliente o usuário terá um acesso bem restrito, ficando apenas com a possibilidade de acesso a informações próprias e impressão de guias para pagamento, abaixo o exemplo de Guia de Previdência Social (GPS):