**Instituto de Ciências Matemáticas e de Computação**

ISSN - 0103-2569

# MANUAL DE INSTRUÇÃO DA *VERSIONWEB* - UMA FERRAMENTA QUE GERENCIA VERSÕES DE ARQUIVOS ATRAVÉS DA WEB

**MARINALVA DIAS SOARES**
**RENATA PONTIN DE MATTOS FORTES**
**DILVAN DE ABREU MOREIRA**

**N° 130**

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
**FEVEREIRO/2001**

# Manual de Instrução da *VersionWeb* - uma ferramenta que gerencia versões de arquivos através da *Web*

**Marinalva Dias Soares**
**Renata Pontin de Mattos Fortes**
**Dilvan de Abreu Moreira**

Departamento de Ciências da Computação e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
Caixa Postal 668, CEP: 13560-970, São Carlos, SP

# SUMÁRIO

**LISTA DE FIGURAS** ..... **III**

**RESUMO** ..... **IV**

**1. INTRODUÇÃO** ..... **1**

**2. INTERFACES DA *VERSIONWEB*** ..... **5**

2.1. AUTENTICAÇÃO DE USUÁRIOS ..... 5

**3. GERENCIAMENTO DE USUÁRIOS** ..... **8**

3.1. FUNCIONALIDADES DA INTERFACE DE GERENCIAMENTO DE USUÁRIOS ..... 8

**4. GERENCIAMENTO DE ARQUIVOS** ..... **10**

4.1. FUNCIONALIDADES DA INTERFACE DE GERENCIAMENTO DE ARQUIVOS ..... 10

**5. LISTA DE VERSÕES DA PÁGINA (PARA INTERNAUTAS E/OU GRUPOS ESPECÍFICOS DE INTERNAUTAS)** ..... **19**

5.1. FUNCIONALIDADES DA INTERFACE DE RECUPERAÇÃO DE VERSÕES PELOS INTERNAUTAS ..... 21

**6. CONCLUSÕES** ..... **25**

**6. REFERÊNCIAS BIBLIOGRÁFICAS** ..... **27**

# LISTA DE FIGURAS

FIGURA 1.1 - ARQUITETURA BÁSICA DA *VERSIONWEB* ..... 3
FIGURA 2.1 - INTERFACE DE AUTENTICAÇÃO DE USUÁRIOS DA *VERSIONWEB* ..... 6
FIGURA 3.1 - INTERFACE DE GERENCIAMENTO DE USUÁRIOS DA *VERSIONWEB* ..... 8
FIGURA 4.1 - INTERFACE PRINCIPAL DE GERENCIAMENTO DE ARQUIVOS DA *VERSIONWEB* ..... 10
FIGURA 4.2 - LISTA DE ARQUIVOS PARA *DOWNLOAD* QUANDO SE FAZ *CHECKOUT* LOCAL ..... 12
FIGURA 4.3 - ÁREA DE ALTERAÇÃO DO CONTEÚDO DE UM ARQUIVO TEXTO COM *CHECKOUT* REMOTO ..... 13
FIGURA 4.4 - LOG DE HISTÓRIA DE UM ARQUIVO ..... 14
FIGURA 4.5 - LISTA DE VERSÕES DE UM ARQUIVO E OPERAÇÕES PERMITIDAS SOBRE SUAS VERSÕES ..... 15
FIGURA 4.6 - ÁREA DE ALTERAÇÃO DO CONTEÚDO DE UM ARQUIVO TEXTO COM OPÇÃO PARA GERAR *BRANCHES* ..... 16
FIGURA 4.7 - INTERFACE PARA O USUÁRIO ESCOLHER AS VERSÕES PARA VISUALIZAR AS DIFERENÇAS ..... 18
FIGURA 5.1 - EXEMPLO DE UMA PÁGINA QUE CONTÉM UM *LINK* PARA A *VERSIONWEB* ..... 19
FIGURA 5.2 - INTERFACE PRINCIPAL DE RECUPERAÇÃO DE VERSÕES PELOS INTERNAUTAS ..... 20
FIGURA 5.3 - VISUALIZAÇÃO DAS DIFERENÇAS ENTRE DUAS VERSÕES DE UMA PÁGINA NO FORMATO DO CVS ..... 22
FIGURA 5.4 - VISUALIZAÇÃO DAS DIFERENÇAS (NO FORMATO EM HTML) ENTRE DUAS VERSÕES DE UMA PÁGINA ATRAVÉS DE CORES ..... 23
FIGURA 5.5 - VISUALIZAÇÃO DAS DIFERENÇAS (NO CÓDIGO FONTE) ENTRE DUAS VERSÕES DE UMA PÁGINA ATRAVÉS DE CORES ..... 24

# RESUMO

Em um mundo computacional em constante evolução, a *Web* se apresenta como um ambiente caracterizado por um desenvolvimento acelerado de suas informações. Além das informações na *Web* sofrerem muitas mudanças e com extrema freqüência, os autores (ou desenvolvedores) das páginas enfrentam dificuldades nas suas atividades quando envolvem muitas pessoas trabalhando em paralelo no desenvolvimento de uma página ou de um conjunto de páginas. Diante desses problemas, a ferramenta *VersionWeb* foi desenvolvida com os objetivos principais de: proporcionar que os internautas obtivessem as versões das páginas durante a navegação e fornecer um modo fácil de controle de versões de páginas da *Web* aos autores, através da própria *Web*.

# 1. Introdução

O processo de desenvolvimento de um *sotware* muitas vezes é considerado como uma atividade de construção de um produto abstrato, uma vez que *software* é tipicamente um produto "lógico" não paupável, não concreto. Esta visão, de que o produto *software* é abstrato, é possível aos olhos dos usuários, mas sob a perspectiva de seus engenheiros, um *software* é constituído de diversos itens que o representam de forma concreta.

Durante o seu desenvolvimento, um *software* se compõe de uma coleção de itens tais como programas, dados e documentos. Ainda durante o desenvolvimento, esses itens podem ser facilmente modificados: o projeto, o código e mesmo os requisitos são freqüentemente modificados. Além disso, essas mudanças ocorrem a qualquer momento. Tudo isso faz com que um controle sobre as mudanças seja necessário, de maneira a tornar os itens que representam o *software* realmente consistentes com o *software*.

O Gerenciamento de Configuração de Software (GCS) é a disciplina encarregada de controlar sistematicamente as mudanças que acontecem durante o desenvolvimento do *software*. O GCS é uma atividade abrangente que é aplicada em todo o processo de engenharia de software e é responsável por gerenciar a evolução de sistemas de *software* grandes e complexos.

Semelhante às mudanças que sempre ocorrem em ambientes de engenharia de software (principalmente quando se está desenvolvendo um programa), a *World Wide Web* também sofre constante evolução e alteração de informações. Portanto, todos esses conceitos de controle de mudanças aplicados em ambientes de engenharia de software tornam-se importantes também no mundo da *Web*, visto que para a construção de uma página ou de um conjunto de páginas relacionadas é preciso passar por etapas de engenharia de *software*, ou seja, o desenvolvimento de um site é comparado ao desenvolvimento de um *software* e precisa ser projetado. Além disso, um grupo de pessoas pode estar trabalhando no desenvolvimento de uma página onde cada membro do grupo trabalha em sua própria cópia para gerar um hiperdocumento final juntando as cópias uns dos outros. Isso pode se tornar crítico se não houver nenhum controle das alterações, pois pode ocorrer inconsistência, perda e sobreposição de informações.

Os usuários (internautas em geral) percebem toda essa evolução de informação principalmente quando visitam uma página já conhecida e não encontram mais as mesmas informações, os mesmos links, etc, e ficam somente na vontade de rever aquela página novamente.

Neste contexto, a ferramenta *VersionWeb*[1] [Soares 2000; Soares et al. 2000] foi desenvolvida com o objetivo de auxiliar os autores no desenvolvimento de páginas, especificamente para dar um suporte à evolução das mesmas, de forma coordenada e controlada evitando, assim, perdas ou sobreposições de informações. Outro objetivo da *VersionWeb* foi o de permitir que os internautas, durante a navegação, tivessem acesso às informações que alguma vez estiveram disponíveis, mas que, geralmente, como conseqüência das constantes atualizações das páginas não estão mais, naquele momento, sendo apresentadas. Através da recuperação de versões anteriores dessas páginas e localização de alterações, o internauta pode rever informações disponibilizadas em algum momento anterior.

A principal característica da *VersionWeb* é a combinação de um bom sistema de controle de versão com uma interface amplamente acessível através dos *browsers Web*. Para o desenvolvimento dessa ferramenta, foi utilizado o CVS (*Concurrent Versions System*) [Cederqvist 1993; CVS 2000], que é um *software* que gerencia todas as versões geradas de um arquivo, e *scripts* CGIs (*Common Gateway Interface*) [Jamsa et al. 1997] que realizam a chamada ao CVS para que os comandos requisitados pelo usuário sejam executados. O CVS é um sistema de versões concorrentes que permite o acesso simultâneo por várias pessoas sobre o mesmo arquivo e fornece vários mecanismos úteis ao controle de versão como: localização das diferenças entre duas versões de um arquivo, localização do autor, data, hora e comentários sobre a alteração feita no arquivo, gerencia versões de arquivos binários, etc.

A *VersionWeb* proporciona uma interface amigável baseada em formulários HTML que oferece as operações básicas de controle de versão para executar os comandos do CVS, uma vez que este é totalmente orientado a linha de comando. Para incluir os recursos do CVS em uma aplicação baseada na *Web*, uma abordagem arquitetural foi adotada e está ilustrada na **Figura 1.1**.

[1] A *VersionWeb* está atualmente disponível em: http://versionweb.sourceforge.net

De acordo com a abordagem arquitetural da *VersionWeb* esquematizada na **Figura 1.1**, um cenário de uso geral da ferramenta foi desenvolvido.

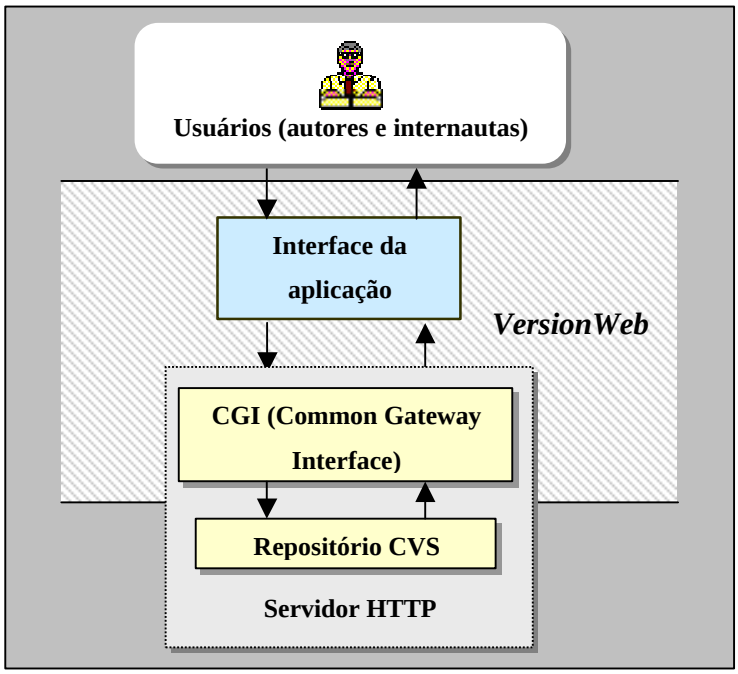


**Figura 1.1 - Arquitetura básica da *VersionWeb***

Este cenário de interação do usuário com a ferramenta (representado na figura acima) juntamente com os seus dois componentes funcionais principais (interface da aplicação e os programas CGIs) realizam o seguinte procedimento para a realização das tarefas de controle de versão através da *Web*:

- os usuários, que estão em suas próprias máquinas (locais ou clientes) em qualquer lugar do mundo, interagem com o CVS através da interface de aplicação da *VersionWeb* (baseada em formulários HTML) e submetem pedidos a um programa CGI que reside fisicamente na mesma máquina que o servidor de *Web*; esses pedidos, geralmente, são comandos CVS a serem executados como *checkout*, *commit*, etc.;
- o servidor de *Web* recebe os pedidos do usuário e os passa ao CGI;
- o CGI recebe e decodifica os pedidos do usuário;
- o CGI que recebeu o pedido faz uma chamada ao CVS (também fisicamente localizado na mesma máquina) com os comandos a serem executados;
- o CVS executa os comandos e retorna o resultado ao CGI que o chamou;

- o CGI envia os resultados ao servidor de *Web*;
- servidor de *Web* devolve os resultados aos usuários que iniciaram a interação com a ferramenta.

A partir dessa arquitetura, o usuário precisa apenas de um *browser Web* em sua máquina local para utilizar a *VersionWeb*, o que favorece ainda mais a sua mobilidade em relação ao acesso à ferramenta. Os arquivos ficam em um repositório central (repositório do CVS) no servidor, mas as modificações nos arquivos podem ser remotas (no próprio servidor) ou locais (na máquina do cliente). Essas são duas características presentes na *VersionWeb* dentre outras, descritas mais detalhadamente nos Capítulos seguintes.

Este manual visa descrever as interfaces da *VersionWeb* e suas respectivas funcionalidades. O próximo Capítulo descreve brevemente as principais interfaces da *VersionWeb* e o processo de autenticação de usuários.

# 2. Interfaces da *VersionWeb*

A ferramenta *VersionWeb* é composta de três interfaces básicas:

1. Interface de gerenciamento de usuários.
2. Interface de gerenciamento de arquivos.
3. Interface de recuperação de versões de uma página por parte dos internautas.

Existem três tipos básicos de usuários da ferramenta *VersionWeb*:

1. **Administradores**: pessoas autorizadas a gerenciarem todos os usuários da ferramenta.
2. **Autores**: pessoas autorizadas a manipularem os arquivos e diretórios que residem em um repositório do CVS.
3. **Internautas e/ou grupos específicos de internautas**: pessoas às quais é permitido apenas visualizar as diferentes versões de uma página e localizar as alterações entre elas.

## 2.1. Autenticação de usuários

O primeiro passo para o uso da ferramenta *VersionWeb* é a autenticação do usuário (obrigatório para autores e administradores), o que dá acesso às funcionalidades principais da ferramenta. Assim, a *VersionWeb* inicia a identificação do usuário através da janela de autenticação de usuários ilustrada na **Figura 2.1**, para que a seguir, a interface relacionada com a devida permissão do usuário seja então disponibilizada.

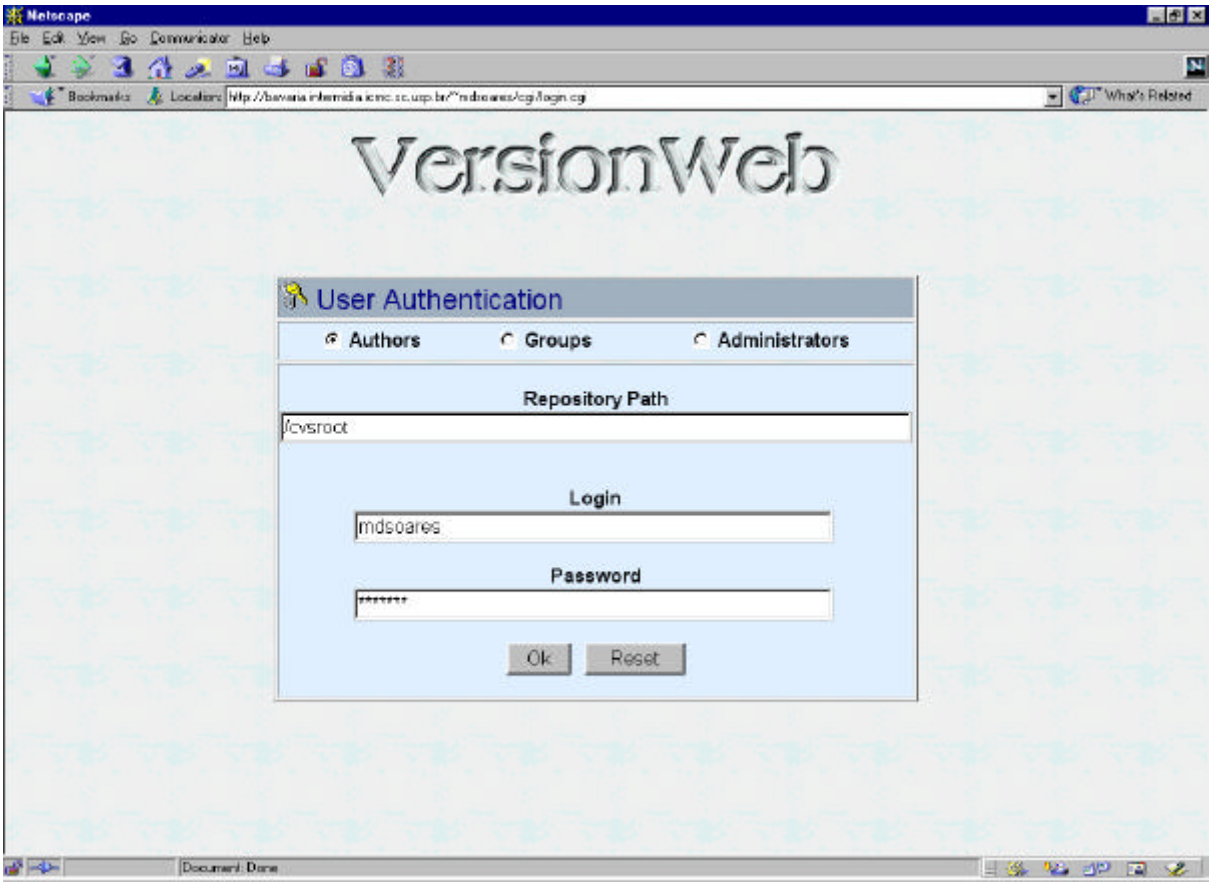


**Figura 2.1 - Interface de autenticação de usuários da *VersionWeb***

A janela de autenticação de usuários ilustrada na **Figura 2.1** possui os seguintes campos para entrada de dados:

- **Tipos de usuário**: Authors (autores), Groups (grupos) e Administrators (administradores).
- **Repository Path**: caminho absoluto onde reside o repositório CVS no servidor.
- **Login**: *username* do usuário.
- **Password**: senha do usuário.

As pessoas autorizadas (normalmente autores) à manipulação dos arquivos e diretórios (armazenados no repositório CVS) devem selecionar o tipo de usuário a que ele pertence, informar o caminho completo onde reside o repositório com os arquivos que ele deseja trabalhar, informar seu *login* e sua *password*.

Grupos específicos de internautas têm acesso à *VersionWeb* através de um *link* da página que ele estiver visitando, desde que essa página esteja sob o gerenciamento do CVS. Esse tipo de usuário não precisa informar o caminho do repositório na janela de autenticação de usuários, mas apenas o tipo de usuário a que ele pertence, seu *login* e sua *password*. Se o administrador decidir que todos os internautas poderão visualizar as versões da página, eles não terão que passar pelo processo de autenticação de usuários, ou seja, ao clicar sobre o *link* que dá acesso à ferramenta eles serão levados diretamente à interface de recuperação de versões para os internautas ilustrada na **Figura 5.2**.

Os administradores, para que tenham acesso à funcionalidade de gerenciamento de usuários da *VersionWeb*, devem também informar o tipo de usuário, o caminho absoluto onde reside o repositório CVS no servidor, seu *login* e sua *password*. Após o processo de autenticação cada usuário tem acesso à interface que lhe for permitido. As funcionalidades de cada uma delas são descritas mais detalhadamente nos Capítulos seguintes.

# 3. Gerenciamento de Usuários

Após o processo de autenticação, cada tipo de usuário tem acesso a um determinado conjunto permitido de serviços. Se o usuário se autenticou como um administrador, ele é responsável por gerenciar todos os usuários da ferramenta, e isso é feito através da interface ilustrada na **Figura 3.1**.

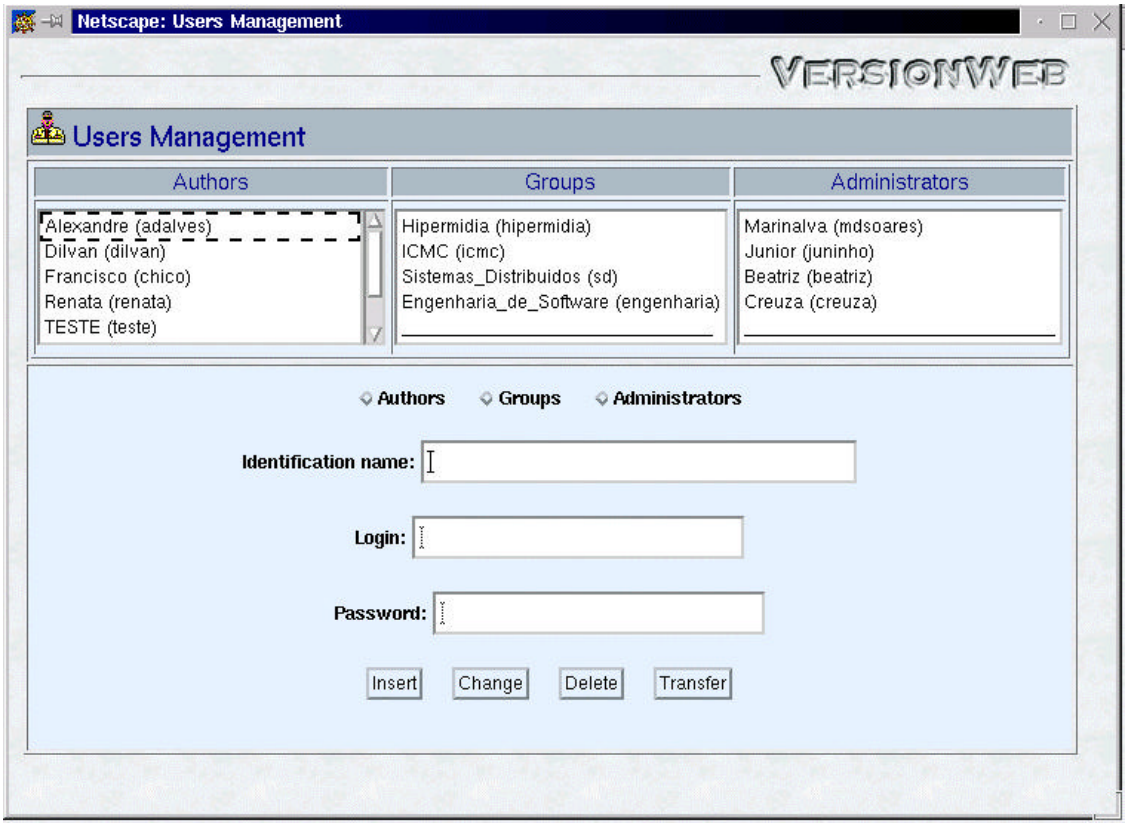


**Figura 3.1 - Interface de gerenciamento de usuários da *VersionWeb***

A Seção seguinte descreve todas as operações que o administrador pode realizar na interface ilustrada na **Figura 3.1**.

## 3.1. Funcionalidades da interface de gerenciamento de usuários

A partir da interface de gerenciamento de usuários (**Figura 3.1**), ao administrador são disponibilizadas operações para controle de informações dos usuários em cada um dos tipos definidos. Essas operações são:

- **Insert**: insere um usuário em uma das listas de usuários da ferramenta (*Authors*, *Groups* ou *Administrators*); para isso, deve-se selecionar a lista que o usuário irá pertencer (opções no botão de rádio da **Figura 3.1**), preencher os campos *name*, *login*, *password* e clicar sobre o botão "Insert".
- **Change**: permite alterar os dados de um usuário (*name*, *login* e *senha*); para isso, deve-se selecionar o usuário em uma das listas, preencher os campos *name*, *login* e *password* com os novos valores e clicar sobre o botão "Change".
- **Delete**: permite remover um usuário de uma das listas de usuários da ferramenta; para isso, deve-se selecionar o usuário em uma das listas e clicar sobre o botão "Delete".
- **Transfer**: permite transferir um usuário de uma lista para outra; para isso, deve-se selecionar o usuário em uma das listas, especificar a nova lista para a qual ele será transferido (marcando no botão de rádio) e clicar sobre o botão "Transfer".

Após inserir um usuário, o administrador deve sair do ambiente da *VersionWeb*, efetuar o seu *login* no CVS através da linha de comando no servidor para que o novo usuário possa manipular os arquivos (se o usuário for um autor ou administrador). Para os usuários do tipo "grupo" não é preciso que o administrador faça seu *login* no CVS, pois eles não têm permissão para manipular os arquivos do repositório. O Capítulo seguinte descreve os serviços de auxílio aos autores, relacionados com o controle de versões das páginas da *Web*, disponibilizados pela ferramenta.

# 4. Gerenciamento de Arquivos

Se o usuário se identificou como um autor, ele terá acesso aos serviços relacionados com o gerenciamento de arquivos (interface ilustrada na **Figura 4.1)**. Nessa interface é exibido todo o conteúdo (arquivos e diretórios) do repositório CVS requisitado na tela de autenticação de usuários pelo autor. Os arquivos gerenciados pelo CVS podem ser tanto texto como binários.

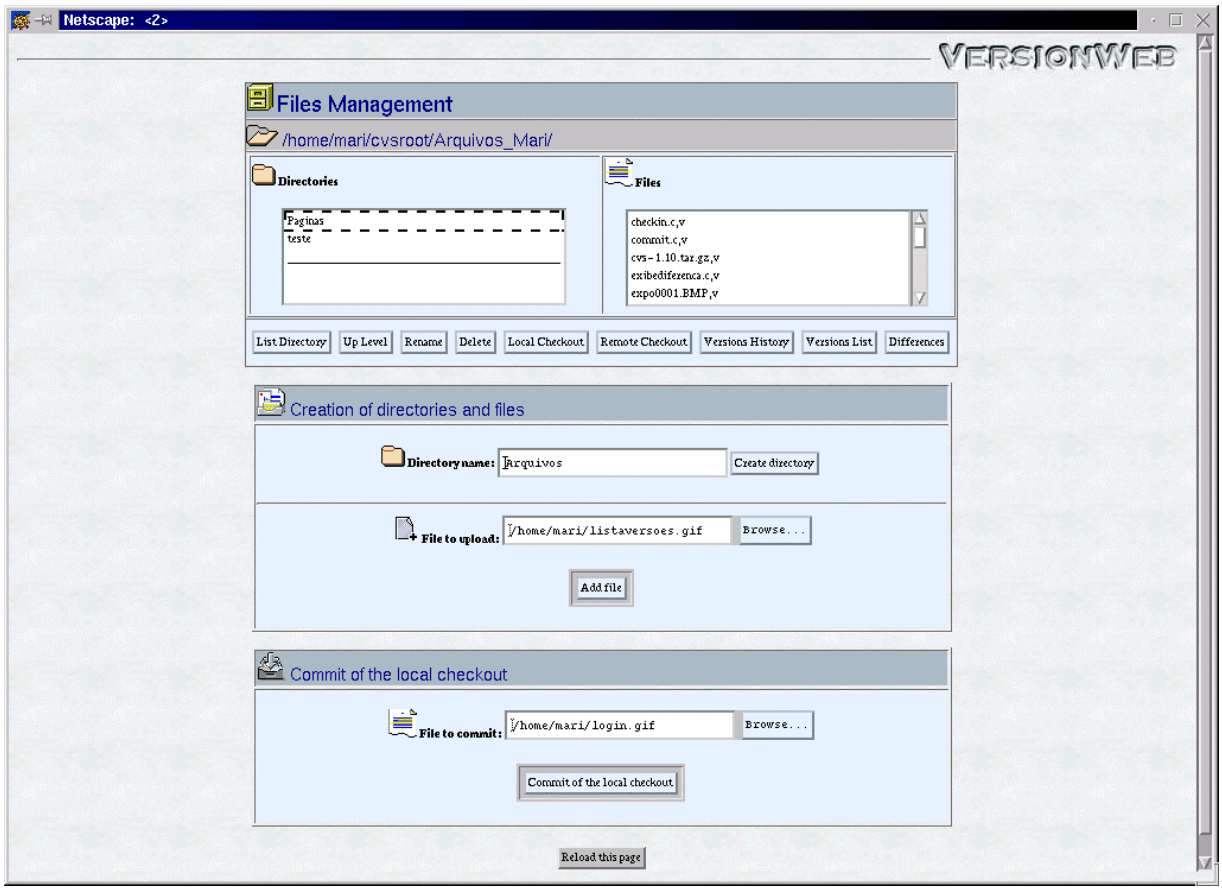


**Figura 4.1 - Interface principal de gerenciamento de arquivos da *VersionWeb***

A Seção seguinte descreve todas as operações disponíveis na interface de gerenciamento de arquivos, bem como as operações de outras interfaces que são obtidas de acordo com a maioria das funções de cada operação da interface ilustrada na **Figura 4.1**.

## 4.1. Funcionalidades da interface de gerenciamento de arquivos

As operações disponíveis na interface ilustrada na **Figura 4.1** são:

- **List Directory**: lista o conteúdo do diretório selecionado.

- **Up Level**: sobe um nível na árvore de diretório.
- **Rename**: renomear um diretório/arquivo selecionado.
- **Delete**: remover um diretório/arquivo selecionado.
- **Remote Checkout**: faz *checkout* remoto (da versão corrente) do arquivo (ascii) selecionado no próprio servidor. Essa opção deve ser usada somente para alterações rápidas e pequenas, para que o *commit* seja feito logo em seguida.
- **Local Checkout**: faz *checkout* (da versão corrente), de um arquivo/diretório para a máquina do usuário. Essa opção deve ser utilizada para alterações mais longas e demoradas, ou seja, quando o *commit* não tiver que ser feito logo em seguida.
- **Versions Log**: lista as versões de um arquivo selecionado com seus respectivos autores, data, hora e mensagem de commit (esta última, se houver).
- **Versions**: abre uma janela com uma lista das versões do arquivo selecionado.
- **Diffs**: permite o autor visualizar as diferenças (localizar alterações) entre as diversas versões do arquivo selecionado.

De acordo com suas funções, a maioria dessas operações prossegue abrindo outras janelas de interação com o usuário, de maneira a efetuar as tarefas necessárias à sua realização. Se o usuário fizer *checkout* local de algum arquivo ou diretório (opção disponível na interface da **Figura 4.1**), ele obterá a tela da **Figura 4.2** para fazer *download* dos arquivos a serem transferidos para a máquina local.

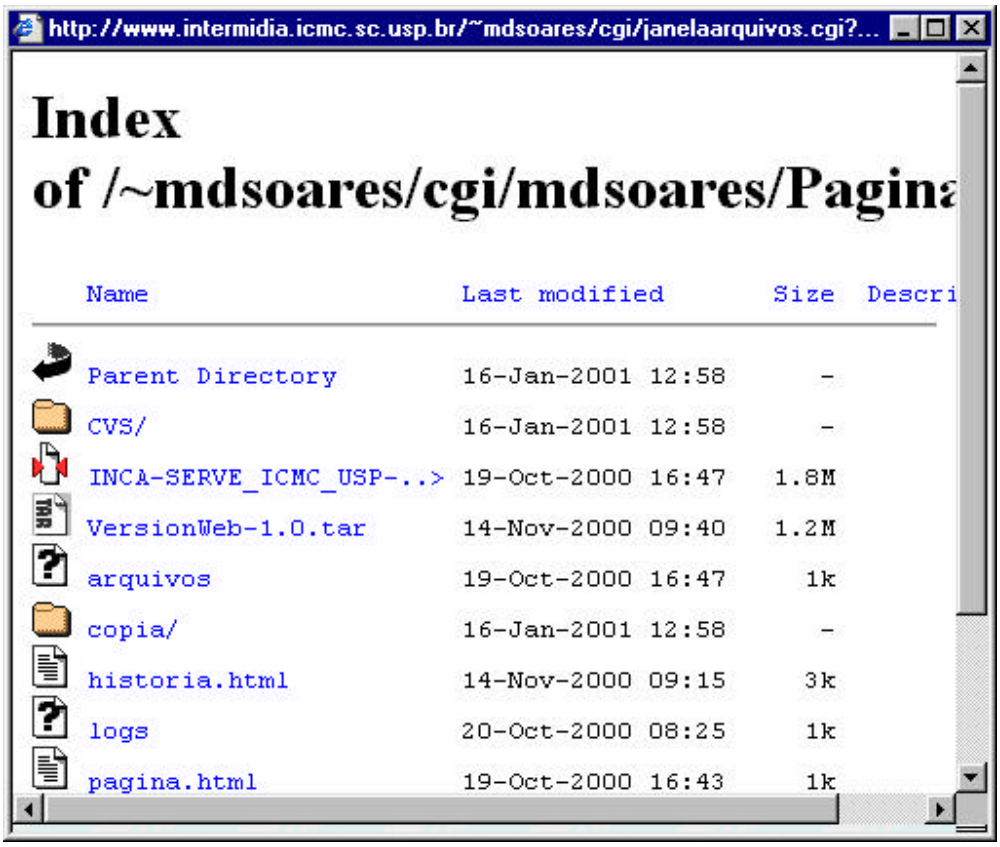


**Figura 4.2 - Lista de arquivos para *download* quando se faz *checkout* local**

A interface da **Figura 4.2** permite que o autor faça o *download* (*checkout* local) de um arquivo ou de um diretório inteiro para alteração em sua própria máquina. A opção da interface principal de gerenciamento de arquivos que permite fazer o *commit* de um arquivo do qual foi feito *checkout local* é "Commit of the local checkout" também ilustrada na tela da **Figura 4.1**.

Se o usuário fez *checkout* local de um diretório inteiro e alterou todos os seus arquivos, o *commit* deverá ser feito para cada arquivo, um a um, e os seus nomes devem ser os mesmos, ou seja, o usuário não pode alterar os nomes dos arquivos em sua máquina e depois fazer o *commit* com nomes diferentes, senão o CVS não irá reconhecer esses arquivos e não irá gerar as versões subseqüentes àquelas das quais se efetuou o *checkout* local. Ao fazer o *commit*, a *VersionWeb* verifica se foi feito *checkout* local daquele arquivo em algum momento, e caso não tenha sido feito o *checkout* local, o *commit* retornará como resultado uma mensagem de erro dizendo que o usuário não efetuou o *checkout* daquele arquivo e que portanto o *commit* não poderá proceder.

Para a modificação de arquivos texto de fácil edição, que não exijam nenhuma ferramenta ou editor específico para essa alteração, a *VersionWeb* possibilita um procedimento ágil, por meio da opção "*Remote Checkout*" também ilustrada na **Figura 4.1**. Quando se faz o *checkout* remoto de um arquivo (arquivo texto, pois a alteração de arquivos binários não pode ser feita diretamente no *browser*), obtém-se a interface da **Figura 4.3** que exibe o seu conteúdo e que permite fazer alterações nele.

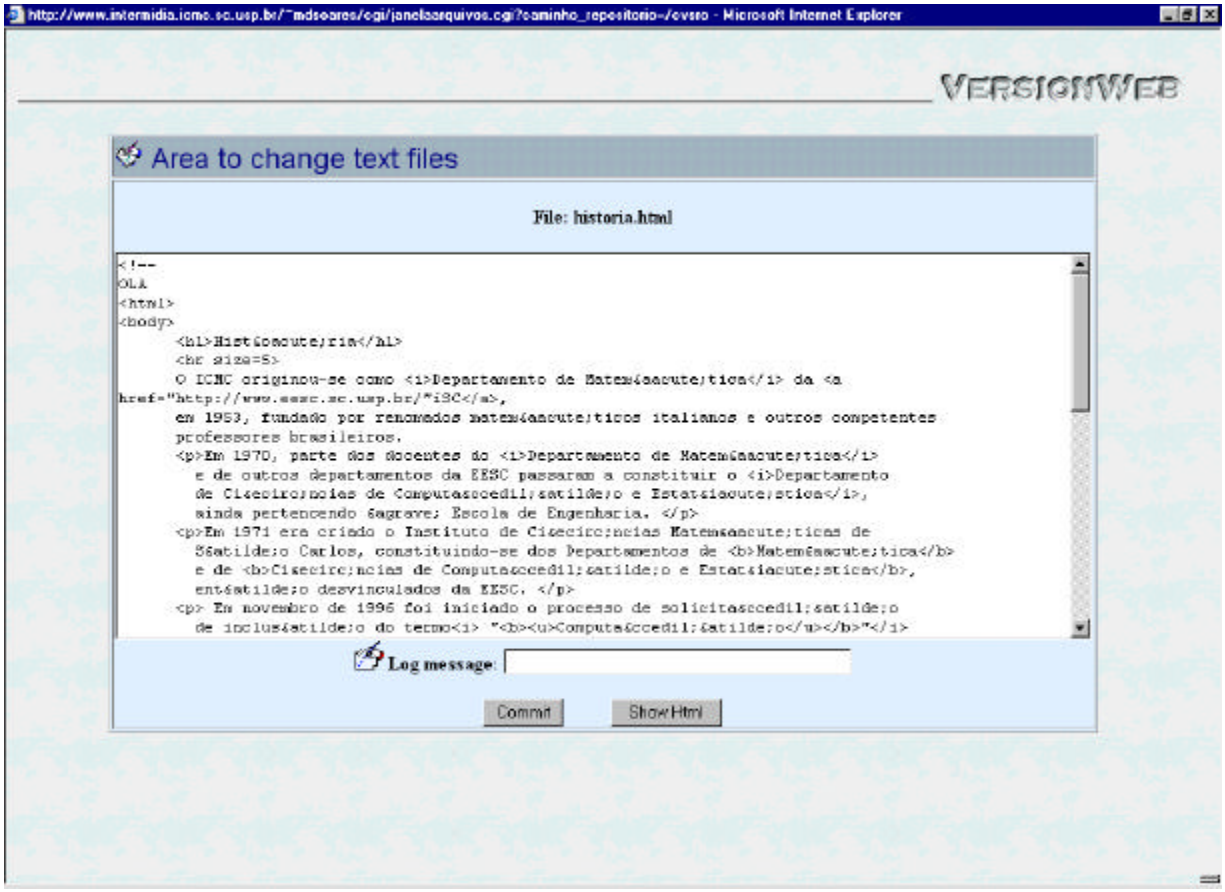


**Figura 4.3 - Área de alteração do conteúdo de um arquivo texto com *checkout* remoto**

A interface ilustrada acima exibe todo o conteúdo do arquivo, do qual se efetuou o *checkout* remoto, para alteração. Além disso, pode-se inserir comentários sobre as alterações que foram feitas no conteúdo do arquivo por meio de um campo de entrada (Log message) na interface. Esses comentários são registrados juntamente com o autor, a data, a hora e a nova versão gerada do arquivo após o *commit*.

O usuário poderá também, através da opção "Versions History" da interface ilustrada na **Figura 4.1,** obter mais informações sobre os arquivos do repositório, tais como: lista das versões existentes, autores, datas e comentários sobre as revisões (esses comentários são inseridos no momento em que se faz o *commit* de um arquivo para gerar uma nova versão). Essas informações são exibidas em um tela como mostra a **Figura 4.4**.

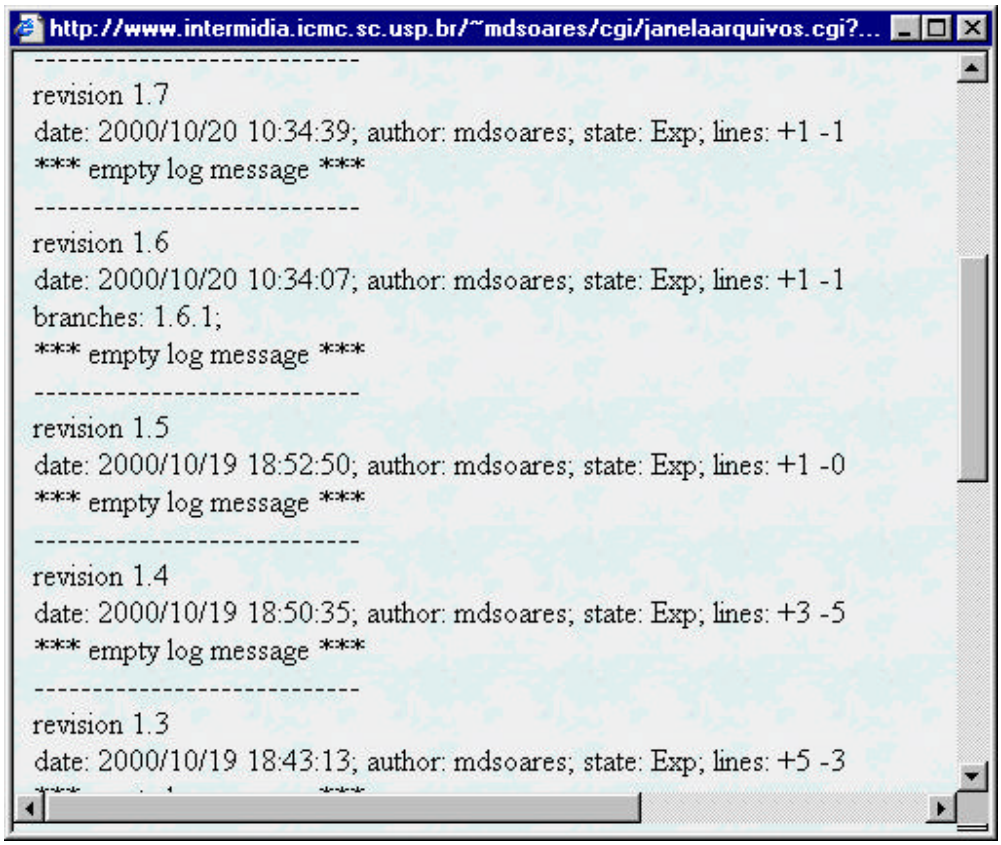


**Figura 4.4 - Log de história de um arquivo**

O autor poderá também, através da opção "Versions List" da interface ilustrada na **Figura 4.1**, obter a lista de todas as versões e *branches* de um arquivo com suas respectivas datas e autores (veja **Figura 4.5**), fazer *checkout* remoto e/ou local sobre qualquer versão do arquivo e fazer o *commit* das alterações efetuadas. Neste caso, para o *commit*, o autor deverá especificar uma *branch* a ser gerada (mais adiante está sendo explicado como gerar *branches*), pois o *checkout* não foi efetuado, normalmente, sobre a versão corrente do arquivo, e sim sobre uma versão anterior à atual.

Além disso, o autor poderá fazer *checkout* remoto e/ou local de uma *branch* (quando se faz *checkout* de uma *branch*, o CVS na verdade sempre faz o *checkout* da última versão daquela *branch*) e, neste caso, o *commit* gera uma versão automática para aquela *branch* e o autor não precisa especificar um número para ela.

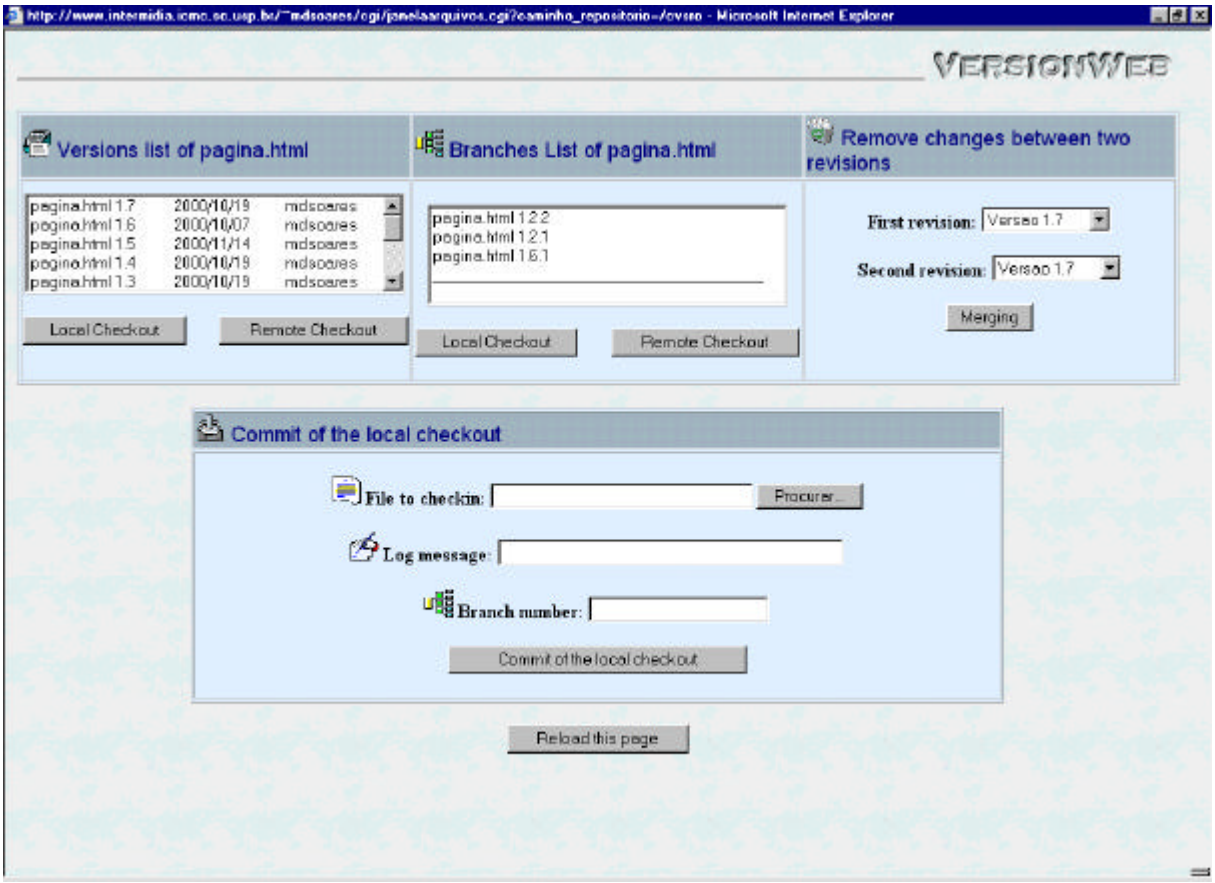


**Figura 4.5 - Lista de versões de um arquivo e operações permitidas sobre suas versões**

A lista de versões ilustrada na interface da **Figura 4.5** inclui todas as versões principais (linha principal de desenvolvimento - por exemplo, 1.1, 1.2, 1.3, etc) e versões das *branches* existentes (por exemplo, 1.1.1.1, 1.1.2.1, 1.1.2.1.2.1, etc), e inclui também uma lista de *branches* e opção para remover alterações entre versões e opção para fazer o *commit* de um *checkout* local.

Se o usuário fizer *checkout* (remoto ou local), por exemplo, da revisão 1.2 e já existir a revisão 1.3, ele deverá especificar um número de versão (normalmente uma *branch*) a ser gerado no momento do *commit*, caso contrário irá surgir um conflito entre versões e o *commit* não poderá

ser efetuado. A *branch* a ser gerada poderá ser 1.2.1, 1.2.2, 1.2.3, etc., e não 1.1.1, 1.1.2, etc (a menos que o *checkout* tenha sido feito da revisão 1.1).

O CVS, no momento do *commit*, gera uma versão automática para a *branch* especificada, ou seja, se o autor especificar o número 1.2.2 para a *branch* a ser gerada, o resultado será 1.2.2.1, onde "1" é a versão da *branch* (neste caso é a primeira versão da *branch*).

Se o *checkout* de uma das versões da lista de versões for remoto (opção "Remote Checkout" da **Figura 4.5**), o autor obterá a tela da **Figura 4.6**. Se for local, as alterações deverão ser feitas localmente na máquina do usuário e o *commit* deverá ser feito através da opção "Commit of the local checkout" da **Figura 4.5**.

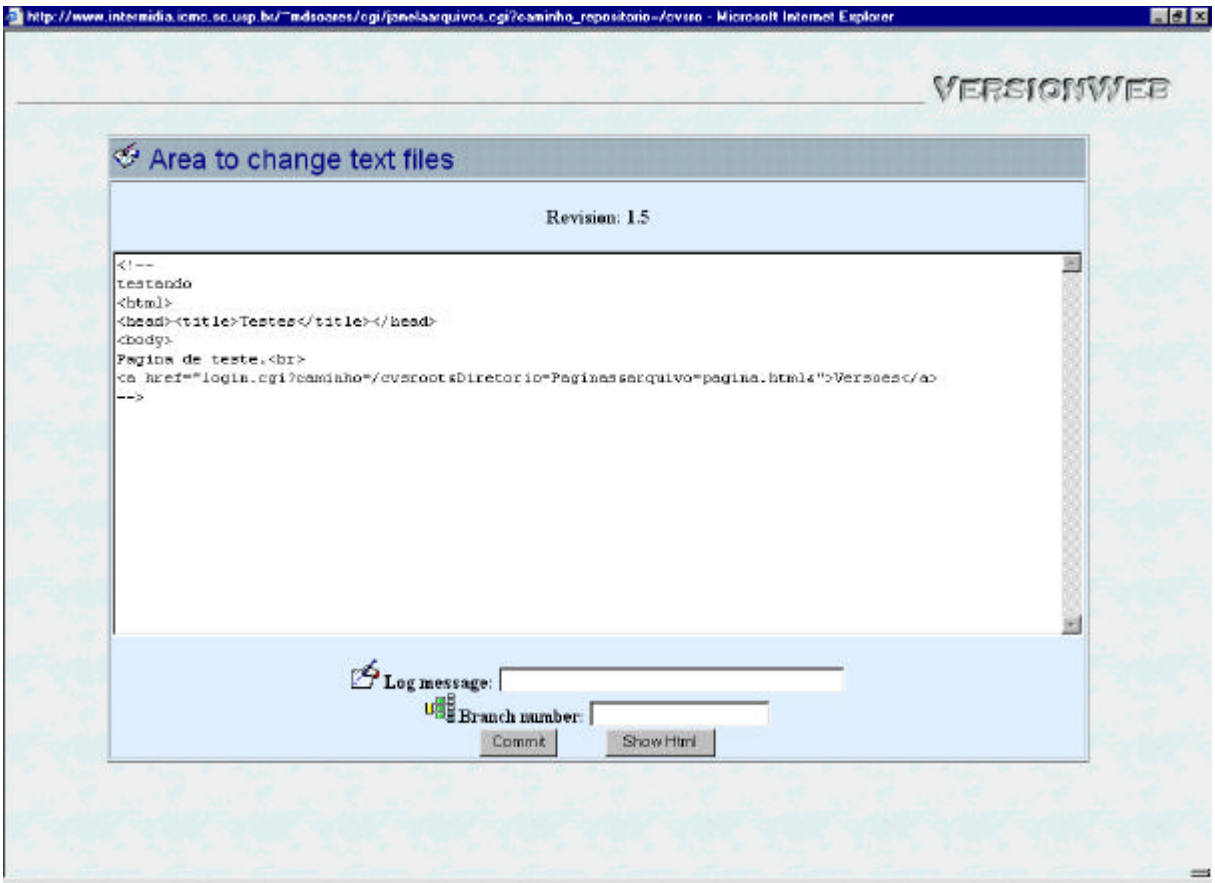


**Figura 4.6 - Área de alteração do conteúdo de um arquivo texto com opção para gerar *branches***

A interface da **Figura 4.6** é semelhante àquela da **Figura 4.3**, diferenciando-se apenas pela adição de um campo para a especificação de um número para a *branch*.

O autor poderá ainda, através da interface ilustrada na **Figura 4.5**, fazer *checkout* remoto e/ou local de uma *branch* (que aparece na lista de *branches*). Neste caso, não é preciso especificar um número de versão a ser gerado, pois o *checkout* é feito sempre da última versão da *branch* e uma versão automática será gerada para ela no momento do *commit*. Por exemplo, suponha que existam 3 versões para a *branch* 1.3.1 (1.3.1.1, 1.3.1.2, 1.3.1.3). Ao fazer o *checkout* da *branch*, o CVS pega automaticamente sua última versão (1.3.1.3) e o *commit* irá gerar a versão 4, ou seja, 1.3.1.4. Se o *checkout* for local, o *commit* deve ser feito através da opção "Commit of the local checkout" ilustrada na mesma interface da **Figura 4.5**. Se for remoto, a tela obtida é a mesma ilustrada na **Figura 4.3**, sem a opção de gerar *branches*, pois nesse caso a versão é gerada de forma automática como explicado anteriormente.

Ainda na interface da **Figura 4.5**, o autor poderá remover alterações feitas entre duas versões diferentes do arquivo. Para isso, ele deve selecionar as duas versões do arquivo (a primeira versão selecionada deve ser maior que a segunda) e clicar sobre o botão "Remove Changes". Por exemplo, se o autor selecionar as versões 1.5 e 1.2, o CVS remove todas as alterações que foram feitas entre a versão 1.2 e 1.5 e gera uma nova versão com a remoção efetuada, ou seja, essa nova versão irá conter, na verdade, o conteúdo da versão 1.2. Isso pode ser útil quando se deseja recuperar uma versão e trabalhar sobre ela na linha principal de desenvolvimento, e não em uma *branch* separada.

O autor poderá também visualizar as diferenças de conteúdo entre duas versões de um arquivo através da opção "Diffs" da **Figura 4.1**. A **Figura 4.7** ilustra a tela que permite escolher as versões e a forma para a visualização das diferenças quando a opção "Diffs" é acionada.

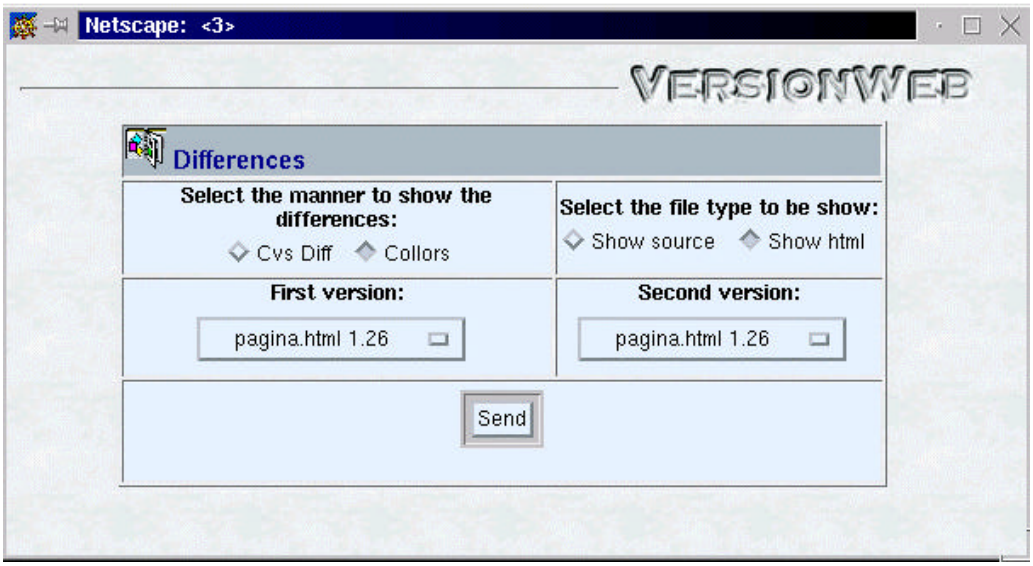


**Figura 4.7 - Interface para o usuário escolher as versões para visualizar as diferenças**

Os resultados obtidos através das operações disponíveis na interface ilustrada na **Figura 4.7** serão mostrados no Capítulo seguinte, o qual descreve a interface acessível pelos internautas e que também possui o recurso de localizar diferenças entre versões.

# 5. Lista de versões da página (para internautas e/ou grupos específicos de internautas)

Um dos objetivos da *VersionWeb* foi o de permitir que os internautas, durante a navegação, tivessem acesso às informações que alguma vez estiveram disponíveis, mas que, geralmente, como conseqüência das constantes atualizações das páginas não estão mais, naquele momento, sendo apresentadas. Por meio da recuperação de versões anteriores dessas páginas e localização de alterações, o internauta pode rever informações disponibilizadas anteriormente.

Para atender esse objetivo, o administrador pode escolher entre permitir que todos os internautas ou um grupo específico destes tenham acesso às versões da página. Se o administrador optar por deixar que apenas grupos específicos de internautas tenham esse acesso, a *VersionWeb* requer que o internauta faça parte de algum grupo já cadastrado. Para isso, a página que ele estiver visitando, se estiver sob controle de versão, conterá um *link* para a ferramenta *VersionWeb*, como a página ilustrada na **Figura 5.1**.

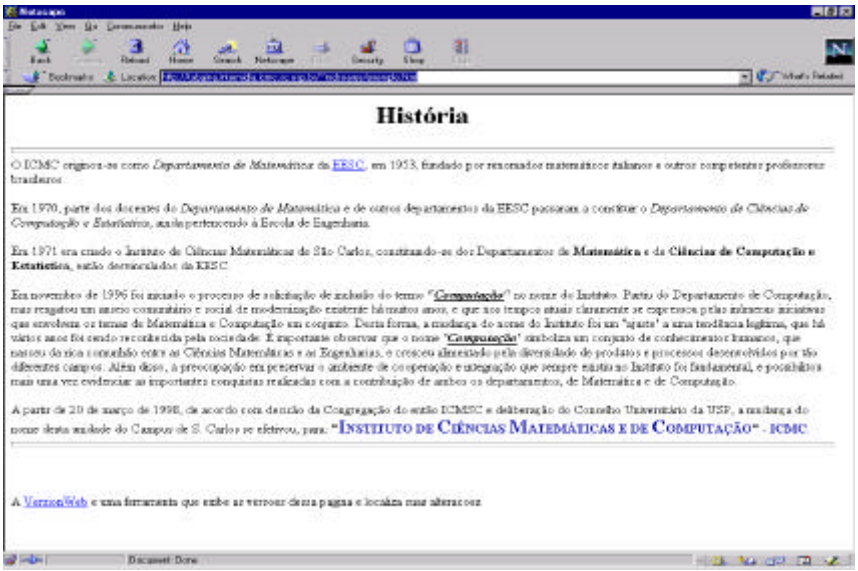


**Figura 5.1 - Exemplo de uma página que contém um *link* para a *VersionWeb***

Esse *link* leva o internauta, primeiramente, à tela de autenticação de usuários ilustrada na **Figura 2.1** e, a seguir, o leva à interface de visualização das versões da página que está apresentada na **Figura 5.2**.

O *link* que dá acesso à *VersionWeb* pode estar definido em qualquer lugar da página. Nesse *link*, o autor deve incluir o caminho onde reside o repositório CVS na máquina servidora, o diretório onde está a página e o nome do arquivo HTML referente à página. Dessa forma, a ferramenta localiza a página no servidor e permite a recuperação de suas versões para o internauta. Após sua correta identificação (através da tela de autenticação de usuários ilustrada na **Figura 2.1)**, o internauta obterá a tela mostrada na **Figura 5.2** a seguir.

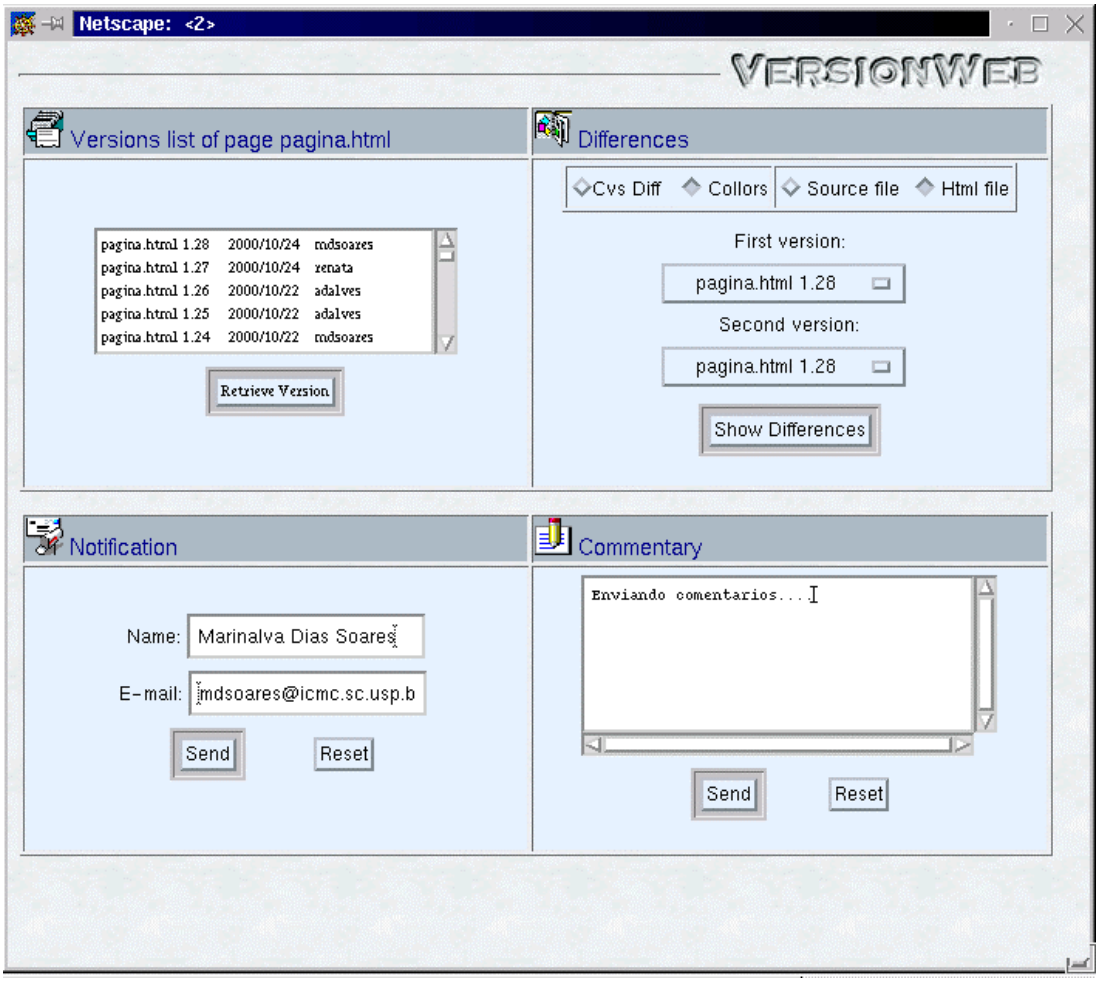


**Figura 5.2 - Interface principal de recuperação de versões pelos internautas**

Se o administrador permitir que todos os internautas tenham acesso às versões da página, eles terão acesso direto à interface ilustrada na **Figura 5.2** quando acionarem o *link* para a ferramenta, ou seja, eles não terão de passar pela tela de autenticação de usuários. A Seção seguinte descreve as funcionalidades da interface ilustrada na **Figura 5.2**.

## 5.1. Funcionalidades da interface de recuperação de versões pelos internautas

A partir da interface ilustrada na **Figura 5.2**, o internauta poderá realizar quatro operações:

- Recuperar uma versão para visualização.
- Visualizar as diferenças de conteúdo entre duas versões através de cores ou do próprio formato exibido pelo CVS.
- Pedir para ser notificado de novas versões disponíveis da página.
- Enviar comentários ou críticas sobre a página e/ou ferramenta para os autores.

Como pode ser observado na **Figura 5.2**, a lista de versões para recuperação é exibida juntamente com a data e o autor da geração de cada versão. Se o usuário desejar recuperar uma versão da página para visualização, ele deverá selecionar a versão desejada e clicar sobre o botão "Retrieve Version". Feito isso, a versão da página é recuperada em uma nova janela do *browser* para visualização como uma página normal, ou seja, como se ele estivesse especificado diretamente na URL uma página para ser mostrada no *browser*.

Geralmente, quando as alterações efetuadas na página são pequenas, fica difícil para o usuário identificar o que realmente mudou naquela página apenas através da visualização de uma versão de cada vez. Dessa forma, a *VersionWeb* possibilita que o internauta visualize as diferenças entre duas versões quaisquer da página, ou seja, o que realmente mudou de uma versão para outra.

Para isso, o internauta deverá informar se quer que essas diferenças sejam exibidas através de cores ou no formato mostrado pelo CVS (o que não é muito aconselhável para quem não está familiarizado com o CVS), e se deseja ver as diferenças no código fonte da página ou no seu formato em HTML. Feito isso, ele deverá então especificar as versões para as quais as diferenças deverão ser exibidas. Após essas escolhas, ele poderá clicar sobre o botão "Show Differences" e as diferenças serão mostradas em uma nova janela do *browser*.

Se o internauta optar por ver as diferenças no formato do CVS, ele obterá uma tela com o conteúdo do arquivo com caracteres de marcação das diferenças como ilustra a **Figura 5.3**. Com

o formato do CVS é preciso uma análise minuciosa do conteúdo para ver e entender essas diferenças.

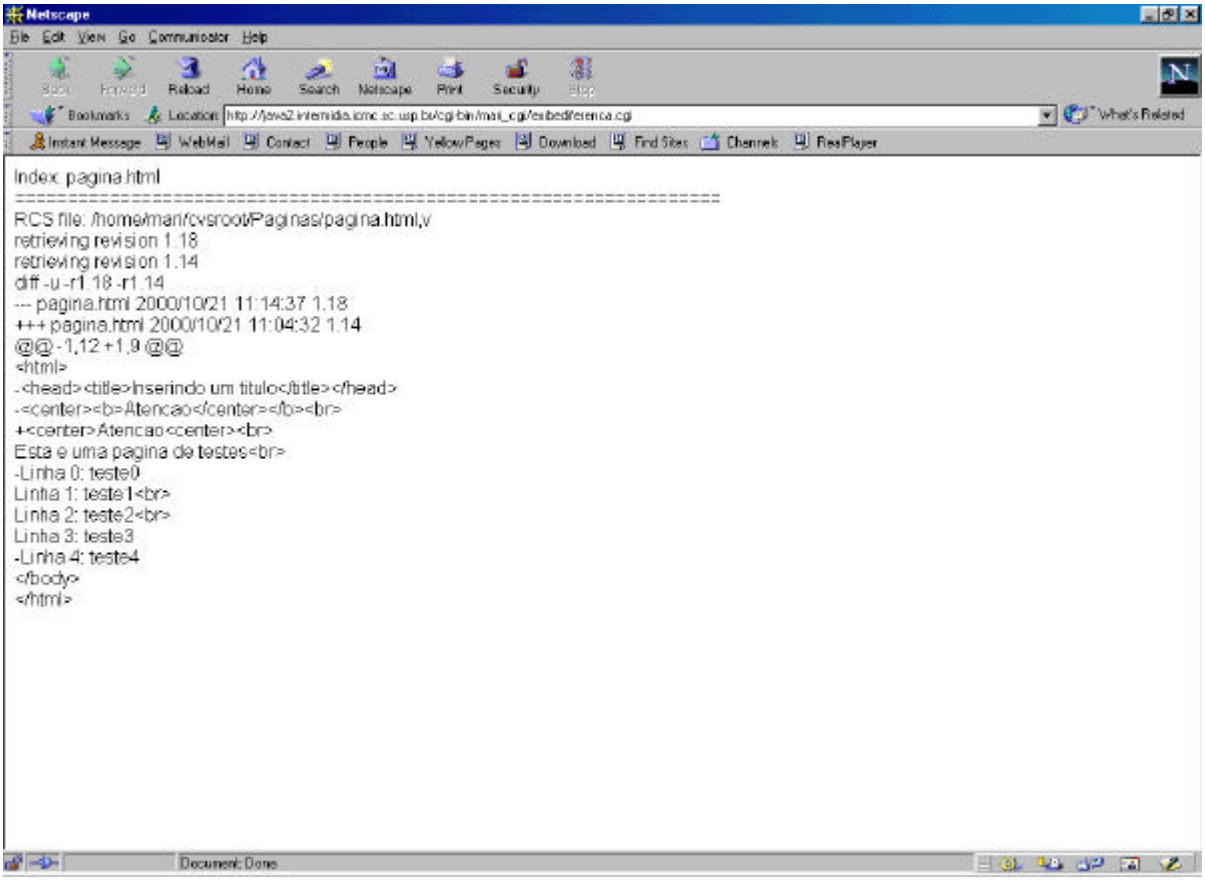


**Figura 5.3 - Visualização das diferenças entre duas versões de uma página no formato do CVS**

Por outro lado, se o internauta optar por visualizar as diferenças através de cores, ele obterá a tela da **Figura 5.4**, que visa facilitar a identificação das alterações. Porém, é bom lembrar que, algumas modificações como título e cabeçalho, por exemplo, não são visíveis no formato em HTML. Então, nesse caso, deve-se optar pelo código fonte. A **Figura 5.5** mostra as diferenças (do código fonte) entre as mesmas versões da página correspondente àquela ilustrada na figura acima através de cores. As **Figuras 5.3**, **5.4** e **5.5** estão ilustrando as diferenças entre as mesmas versões referentes ao mesmo arquivo.

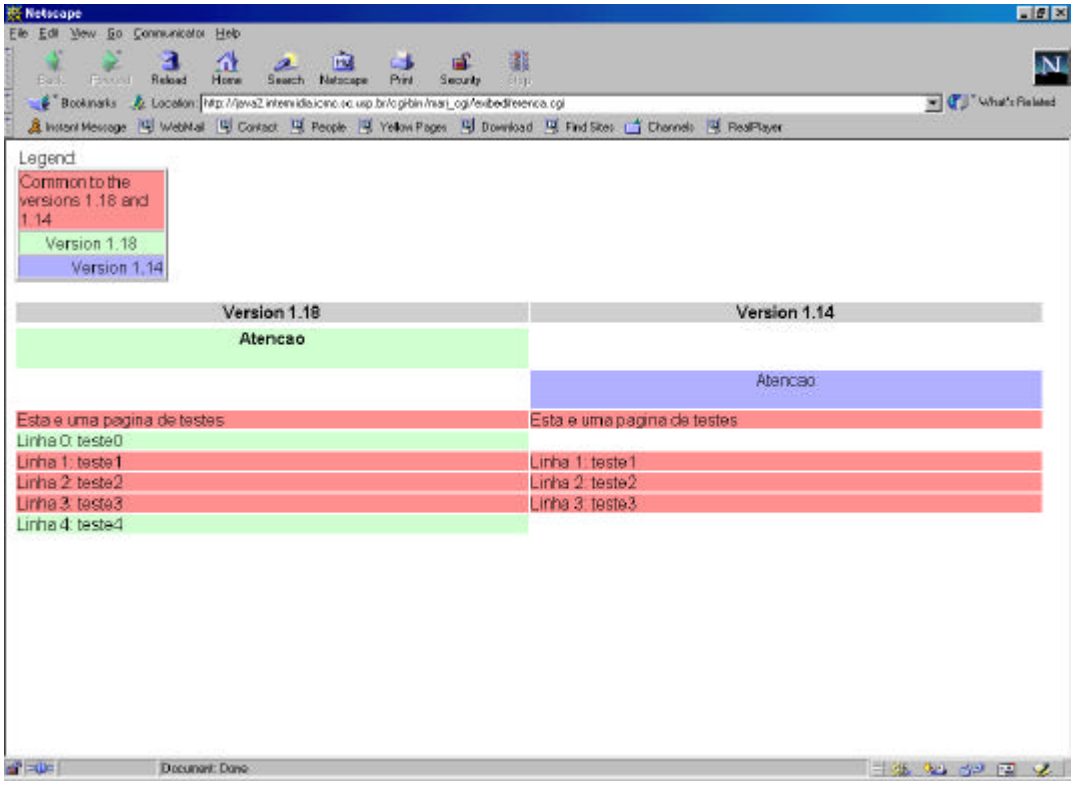


**Figura 5.4 - Visualização das diferenças (no formato em HTML) entre duas versões de uma página através de cores**

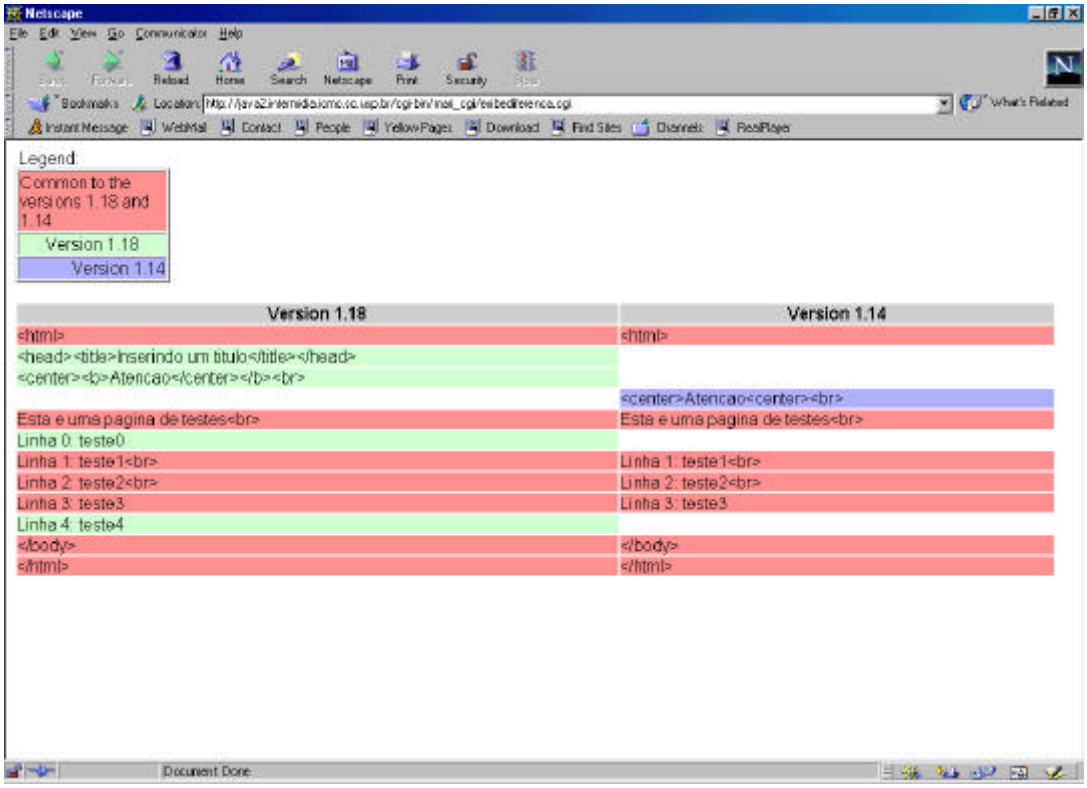


**Figura 5.5 - Visualização das diferenças (no código fonte) entre duas versões de uma página através de cores**

Ainda através da interface ilustrada na **Figura 5.2**, o internauta poderá pedir para ser notificado de novas versões disponíveis da página. Essa notificação é feita automaticamente pelo CVS quando uma nova versão daquela página é gerada. Porém, o autor pode não disponibilizar essa nova versão, o que vai causar uma certa decepção ao internauta, pois ele vai receber uma notificação de que uma nova versão da página foi gerada, mas que pode não estar disponível. Além disso, o internauta pode enviar sugestões ou críticas sobre as alterações efetuadas na página. Todos os autores da *VersionWeb* terão acesso aos comentários enviados pelos internautas.

# 6. CONCLUSÕES

O processo de mudança é uma realidade que ocorre com extrema freqüência na *World Wide Web*. No ambiente WWW, os leitores freqüentemente se queixam quando ao visitar uma página, percebem que esta já não possui o mesmo conteúdo ou até mesmo que ela não existe mais, decorrente da rápida e natural evolução das informações na WWW [Sommerville et al. 1998].

Os desenvolvedores de páginas *Web*, por sua vez, encontram dificuldades quando muitas pessoas estão envolvidas na construção em paralelo de uma mesma página ou de um conjunto de páginas relacionadas. Isso se deve do fato de que os autores trabalham independentemente em suas próprias cópias, tendo como principal problema a integração dessas cópias em um hiperdocumento final [Sommerville et al. 1998]. Além disso, em geral, o volume de documentos envolvidos é significativamente grande e foge a um controle simples da evolução de suas cópias, pois pouco ou nenhum gerenciamento de informação é fornecido.

As várias formas de atuação nesses dois cenários típicos para o controle de versões das páginas *Web* mostram que um suporte ao controle de versão dos arquivos para os autores, os quais trabalham em um desenvolvimento colaborativo, e um suporte à navegação por versões anteriores das páginas, por parte dos internautas, são alvos de investigação de muito interesse. Neste contexto foi desenvolvida a ferramenta *VersionWeb*.

De forma geral, a disponibilização de uma ferramenta para o gerenciamento de versões de arquivos na *Web* com a arquitetura adotada na *VersionWeb* visa estimular a colaboração entre os autores que podem estar localizados em diferentes lugares, pois eles não precisam se preocupar tanto em comunicar uns aos outros que vão trabalhar em um determinado arquivo e não necessariamente precisam ter os arquivos locais em suas máquinas.

A abordagem adotada pela *VersionWeb*, de possuir o repositório centralizado no servidor, facilita ainda mais esse processo, pois as informações se tornam mais prontamente disponíveis aos autores à medida que cada um vai modificando os arquivos e fazendo o *commit*. O fato do CVS não bloquear os arquivos para edição por mais de um autor ao mesmo tempo gera uma produtividade maior entre os integrantes da equipe, pois a espera de liberação de um arquivo para escrita não é necessária e isso com certeza retardaria a obtenção do produto final.

Os recursos de localização das alterações feitas pelo autor, juntamente com data e comentários da versão gerada, dão um certo conforto e agilidade ao processo de desenvolvimento no sentido de que os autores não precisam estar se comunicando e analisando cada linha de código para ver o que foi alterado e quem alterou, pois isso é feito automaticamente. Além disso, os desenvolvedores dos *sites* não precisam colocá-los "indisponíveis" (fora do ar) enquanto estiverem fazendo modificações ou adaptações. Isso certamente causa menos aborrecimento aos internautas que estão visitando essas páginas, pois têm sempre uma versão disponível. Além disso, os desenvolvedores (empresários, por exemplo) não correm o risco de perder clientes pelo fato do *site* estar sofrendo alterações e estar fora do ar.

# 6. Referências Bibliográficas

[Cederqvist 1993] Cederqvist, P. Version Management with CVS. Disponível *on-line* em: ftp://java.icmc.sc.usp.br/library/books/cvs.pdf. Visitado em janeiro de 2001.

[CVS 2000] Disponível *on-line* em: http://www.cvshome.org. Visitado em janeiro de 2001.

[Jamsa et al. 1997] Jamsa, K.; Lalani, S.; Weakley, S. – **Web Programming**, Jamsa Press, 1997.

[Soares 2000] Soares, M. D. *Dissertação de Mestrado do ICMC - USP*, 2000.

[Soares et al. 2000] Soares, M. D.; Fortes, R. P. M.; Moreira, D. A. *VersionWeb: A Tool for Helping Web Pages Version Control*. Proceedings. In: International Conference on Internet Multimedia Systems and Applications (IMSA 2000), Las Vegas, EUA, pp. 275-280.

[Sommerville et al. 1998] Sommerville, I., Rodden, T., Rayson, P., Kirby, A., Dix, A. Supporting information evolution on the WWW. **World Wide Web**, Vol. 1, Nº 1, 45-54, 1998.