UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

APLICAÇÃO DA TECNOLOGIA WEBSNAP PARA O DESENVOLVIMENTO DE PÁGINAS HTML

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

JEANDERSON GRIPA

BLUMENAU, DEZEMBRO/2002

2002/2-37

APLICAÇÃO DA TECNOLOGIA WEBSNAP PARA O DESENVOLVIMENTO DE PÁGINAS HTML

JEANDERSON GRIPA

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Maurício Capobianco Lopes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Maurício Capobianco Lopes

_____

Prof. Francisco Adell Péricas

_____

Prof. Wilson Pedro Carli

AGRADECIMENTOS

Agradeço ao professor Maurício Capobianco Lopes pela orientação, críticas e principalmente apoio dado no decorrer do estudo.

Aos meus pais, que sempre lutaram para proporcionar aos filhos tudo o que eles jamais tiveram para si, sem os quais eu não teria chegado a este momento.

Aos meus irmãos e namorada pelo apoio, incentivo, compreensão e carinho que recebi durante a elaboração deste trabalho, principalmente nos momentos difíceis que tive.

A Deus, por estar sempre presente em todos os momentos e me dando força nos momentos de dificuldade.

Agradeço em especial, ao grande amigo e companheiro de faculdade, Jaime Stähelin Junior, pelo seu apoio e lealdade. Um verdadeiro escudeiro que sempre esteve ao meu lado por todos esses anos.

A todos os meus colegas de faculdade, que contribuíram para o meu crescimento e tornaram este período de faculdade inesquecível.

Finalmente, agradeço a todos que de alguma forma contribuíram para elaboração deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Este trabalho tem como propósito demonstrar a utilização da nova tecnologia WebSnap encontrada no Delphi 6, destacando seus recursos e performance como tecnologia de desenvolvimento de páginas para a internet. Para demonstrar a efetividade do mesmo, foi desenvolvido um protótipo de reserva de laboratório de ensino *on-line*, utilizando esta tecnologia.

ABSTRACT

This work aims to show WebSnap technology that can be found Delphi 6. The principal purpose is to analyse the performance of WebSnap as technology for Internet sites design. Intending to show how effective it is, a reservation prototype was created for on-line teaching laboratories, using this technology.

1 INTRODUÇÃO

A internet foi criada na década de 1960, como uma rede restrita, destinada aos segmentos militar, governamental e acadêmico. No início da segunda metade da década de 1990, com a regulamentação para exploração comercial da internet e com a popularização dos programas gráficos para navegação na mesma, pequenas redes, antes operando isoladamente, interligaram-se numa única rede (Fleury, 2002).

Segundo Cusumano (1998), a primeira forma de comunicação entre os usuários da internet foi o correio eletrônico, com suas mensagens transmitidas para qualquer parte do mundo ao custo de uma ligação local. Posteriormente surgiram os primeiros *sites* para consultas, onde as informações eram exibidas sem que o usuário pudesse interagir com o mesmo. Por último, o usuário tornou-se capaz de transmitir informações aos *sites*, concluindo todas as etapas necessárias para que fosse possível estabelecerem-se transações em tempo real entre duas partes através da rede.

Antes do surgimento da tecnologia de documentos hiper-textos para navegação em documentos da rede, a internet não possuía uma *interface* amigável com os usuários, principalmente para com os usuários iniciantes e amadores. Somente após seu surgimento em 1993, com o protocolo *Hyper Text Transfer Protocol* (HTTP) e navegadores capazes de interpretá-lo surgiu a *World Wide Web* (WWW), tornando o ambiente da internet mais amigável, facilitando a vida dos usuários comuns e o uso da rede para fins comerciais. Atualmente grande parte dos usuários conhece a internet apenas como uma rede mundial de computadores onde se navega pelas páginas *Hyper Text Markup Language* (HTML) em seus Navegadores *Netscape* ou *Explorer*, o que não reflete a verdade, pois a WWW é somente um subconjunto da internet.

Junto com o HTML, surgiram diversas linguagens de programação para um melhoramento na interação de aplicações de internet com usuários. Uma das tecnologias existentes atualmente para o desenvolvimento de páginas HTML é o WebSnap, ferramenta essa que permite evoluir o conceito de formulários simples, fornecendo uma melhor *interface* e incrementando o *WebBroker*, antiga solução para desenvolvimento de páginas HTML da *Borland*.

Numa publicação recente (Pauli, 2002) é exposto que o WebSnap é uma coleção de componentes voltados ao desenvolvimento de aplicações *Web* facilitando muito a vida de empresas que desenvolvem aplicações de comércio eletrônico, principalmente as que possuem programadores e *Web designers* em uma mesma equipe. Usando o WebSnap, o programador Delphi desenvolve a *interface* da aplicação, à parte do acesso a dados e das regras de negócio uma vez que estas ficam do lado do servidor.

Por ser ainda uma tecnologia pouco conhecida e pouco utilizada, por estar inserida em um dos ambientes de programação mais utilizados no mercado atualmente, o Delphi, decidiu-se neste trabalho, desenvolver uma aplicação para testar a tecnologia Websnap. Para isso, foi desenvolvido um protótipo de reserva de laboratório de ensino, a fim de possibilitar o uso e teste da tecnologia.

1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho é desenvolver uma aplicação utilizando os recursos de internet do Delphi 6, mais especificamente o WebSnap.

Como objetivos específicos podem ser citados:

a) o desenvolvimento de um protótipo de reserva de laboratório de ensino *on-line*;
b) a análise das facilidades e performance do WebSnap como tecnologia de desenvolvimento de páginas interativas para a internet.

1.2 ESTRUTURA DO TRABALHO

O trabalho foi estruturado da seguinte maneira:

O primeiro capítulo apresenta a contextualização e justificativa para o desenvolvimento da proposta do trabalho.

O segundo capítulo aborda conceitos sobre internet, suas formas de utilização e as tecnologias utilizadas para o desenvolvimento de aplicações *Web*.

O terceiro capítulo, que é o ponto chave do trabalho, apresenta a tecnologia WebSnap encontrada no Delphi 6.

O quarto capítulo descreve a especificação do protótipo, bem como detalhes de sua implementação, resultados e discussão.

O quinto e último capítulo apresenta as considerações finais, abrangendo as conclusões do desenvolvimento deste trabalho e as sugestões para próximos trabalhos.

2 INTERNET

De acordo com Fleury (2002), a internet foi criada na década de 1960, como uma rede restrita destinada aos segmentos militar, governamental e acadêmico, em uma organização de pesquisas do governo norte-americano, a *Advanced Research Projects* (Arpanet), que pesquisava a formação de redes de computadores. A idéia era criar uma rede para conectar centros de pesquisa que estivessem distantes uns dos outros e que não pudesse ser destruída por bombardeios, na impossibilidade de uma comunicação seria uma forma alternativa de ligação que seria usada entre os computadores do exército e das universidades.

Ao longo dos anos 70 e meados dos anos 80 muitas universidades se conectaram a essa rede, o que moveu a motivação militarista do uso da rede para uma motivação mais cultural e acadêmica. Nos meados dos anos 80 a *National Science Foundation* (NSF) constituiu uma rede de fibra ótica de alta velocidade conectando centros de supercomputação localizados em pontos chave nos EUA. Essa rede da NSF, chamada de *backbone*, teve um papel fundamental no desenvolvimento da internet por reduzir substancialmente o custo da comunicação de dados para as redes de computadores.

No início da segunda metade da década de 1990, com a regulamentação para exploração comercial da internet e com a popularização dos programas gráficos para navegação na mesma, pequenas redes, antes operando isoladamente, interligaram-se numa única rede. Mais ainda, qualquer pessoa que possuísse um computador pessoal e uma linha telefônica tornou-se capaz de conectar-se através de redes de acesso, à internet.

Segundo Cusumano (1998), a difusão na utilização da internet, que pode ser considerada numa primeira análise como uma evolução na comunicação eletrônica, possibilitou uma nova revolução tecnológica, alterando a maneira como pessoas e organizações vivem e interagem.

O uso da internet continua crescendo em velocidade elevada, e sua difusão supera em muito à de outros artefatos. Em apenas três anos a internet superou a marca de 90 milhões de usuários. O rádio levou mais de 30 anos para atingir 60 milhões de usuários e a televisão levou 15 anos para atingir o mesmo volume.

A internet representa um dos mais bem sucedidos exemplos dos benefícios da manutenção do investimento e do compromisso com a pesquisa e o desenvolvimento de uma infra-estrutura para a informação. Começando com as primeiras pesquisas em trocas de pacotes, o governo, a indústria e o meio acadêmico têm sido parceiros na evolução e uso desta nova tecnologia. Hoje, termos como nome@nomedeempresa.com 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 e sua influência atinge não somente os campos técnicos das comunicações via computadores, mas toda a sociedade, na medida em que se usa cada vez mais ferramentas *on-line* para fazer comércio eletrônico, adquirir informação e operar em comunidade.

2.1 FORMAS DE UTILIZAÇÃO

A internet pode ser utilizada de diversas maneiras, entre as quais destacam-se:

a) correio eletrônico (*e-mail*): foi a primeira aplicação surgida na Internet, com o objetivo de facilitar a comunicação e a troca de idéias e observações entre o grupo de acadêmicos que estavam construindo e experimentando a Internet. Entre as principais características do correio eletrônico destacam-se:
   - agilidade: em segundos ou minutos é enviado para qualquer parte do mundo;
   - gratuito: não paga por *e-mail* enviado ou recebido;
   - escrito: facilita o acompanhamento de solicitações;
   - permite envio de mensagens para muitas pessoas ao mesmo tempo;
   - permite respostas a *e-mails* recebidos ou encaminhamentos a terceiros;
   - permite o envio de arquivos de dados anexados.

b) listas de discussão (*mailing lists*): são listas de endereços de correio eletrônico de pessoas interessadas em determinados assuntos. Uma lista de discussão é formada quando existe um número relativamente grande de pessoas que pretendem discutir algum assunto *on-line* através de *e-mails*. Quando esse número torna-se difícil ou impraticável o endereçamento do *e-mail* para cada um dos destinatários, o recurso mais prático e barato é criar ou usar uma lista de distribuição;

c) transferência de arquivos (FTP): tem como objetivo promover o compartilhamento de arquivos sejam eles programas ou dados, encorajar indiretamente o uso de servidores remotos que funcionam como backup, proteger os internautas de variações em sistemas de armazenamento de arquivos entre servidores e transferência de dados com confiabilidade, rapidez e eficiência;
d) conversas *on-line* (*Chat*): é o nome popular que foi dado para o *Internet Relay Chat* (IRC). O IRC ou *chat* é o encontro virtual onde pessoas podem se encontrar e conversar em tempo real através de mensagens escritas, tanto participando de discussões grupais como em conversas particulares;
e) *sites* (*Web*): consiste na rede mundial de computadores denominada *World Wide Web* (WWW) composta por páginas *Hypertext Markup Language* (HTML) onde se navega, pesquisa-se e compram-se produtos utilizando navegadores como *Netscape* ou *Explorer*.

2.2 HTML

Segundo Cantu (2002), o *Hypertext Markup Language* (HTML), linguagem de marcação de hipertexto é o formato mais difundido para conteúdo na *Web*. HTML é o formato que os navegadores *Web* normalmente lêem; trata-se de um padrão definido pelo *World Wide Web Consortium* (W3C), que é um dos organismos que controlam a internet.

Os arquivos HTML são basicamente arquivos de textos ASCII. Além do texto simples, um arquivo HTML contém muitas *tags*, as quais podem determinar o estilo da fonte, o tipo do parágrafo ou um *link* para outro arquivo HTML ou para uma imagem, entre outras coisas.

A maioria das *tags* ocorrem em pares de abertura e fechamento, normalmente a *tag* de fechamento é igual a *tag* de abertura, mas é precedida por uma barra (/) para indicar onde o estilo ou conteúdo começa e onde termina .

Um documento HTML começa com a *tag* <html> e é dividido em duas partes, marcadas como <head> (cabeçalho da página) e <body> (corpo da página). Cada uma dessas três *tags* exige o terminador correspondente. Na parte do cabeçalho do arquivo HTML, geralmente escreve-se o título (freqüentemente apresentado na barra de título do navegador) e alguns outros elementos genéricos.

No corpo da página, escreve-se o conteúdo do arquivo, geralmente começando com seu título visível. Pode-se usar cabeçalhos com diferentes níveis, marcados com a *tag* <hX>, onde se troca o X por um número de 1 a 6. Isso é seguido por parágrafos simples (<p>), parágrafos previamente formatados (<pre>, um estilo geralmente usado para listagens de programas), vários tipos de listas e muitos outros elementos. O texto freqüentemente terá *links* para outras páginas ou outras partes da página corrente, usando-se a *tag* <a> (“âncora”).

No QUADRO 1 é apresentado um exemplo de um arquivo HTML e na FIGURA 1, tem-se o resultado deste arquivo.

QUADRO 1 – EXEMPLO DE ARQUIVO HTML

```
<html>
<head>
<title> Aplicação WebSnap para Páginas HTML</title>
</head>
<body>
<center> Exemplo de Arquivo HTML
<br> <br>
===> HTML <===
</center>
</body>
</html>
```

FIGURA 1 – EXEMPLO DE PÁGINA HTML

Outro elemento relevante da HTML são as tabelas. As *tags* <table> e </table> indicam o início e o fim da tabela, e seu atributo opcional *border* exibe bordas com determinada largura. As *tags* <tr> e </tr> introduzem e fecham cada linha, e as *tags* <th> e </th> e <td>_</td> indicam uma célula de cabeçalho de tabela e uma célula de dados de tabela, respectivamente. O número de colunas depende dos itens presentes em cada linha. Diferentes linhas podem ter diferentes números de itens.

Recentemente, a HTML foi aprimorada pelo W3C para ser mais consistente, flexível e intercambiável com sistemas avançados, como *Extensible Markup Language* (XML); a nova versão é chamada *Extensible* HTML (XHTML).

2.3 TECNOLOGIAS PARA A INTERFACE *WEB*

Segundo Frydrych (apud Fleury, 2002), a aplicação *Web* utiliza-se de uma página HTML, interpretada pelo *browser*, para interagir com o usuário. Outras tecnologias podem ser misturadas ao HTML para a construção de uma melhor *interface*, com um visual mais adequado, além de proporcionar recursos que o HTML isoladamente não é capaz.

Estas tecnologias são:

a) DHTML: *Dynamic* HTML é um termo utilizado para agrupar as tecnologias de *script*, cascatas de estilo e *applets*, as quais podem ser utilizadas em conjunto com o HTML tornando as páginas *Web* mais interativas e animadas. O uso da tecnologia DHTML é possível graças à concepção do *Document Object Model* (DOM), que aplica os conceitos da orientação a objetos a todos os elementos de uma página HTML;

b) APPLET JAVA: a linguagem *Java* da *Sun Microsystems*, utilizada na forma de *applets*, é capaz de estender as funcionalidades do *browser*, adicionando recursos antes impossíveis de serem construídos com o HTML puro. Os *applets* são miniprogramas executados sob o *browser*, através da *Java Virtual Machine*;

c) ACTIVE X: numa forma similar aos *applets Java*, o *Active* X da *Microsoft* também oferece formas de ampliar as funcionalidades do *browser*, podendo interagir com sistemas instalados no computador cliente. É capaz de, por exemplo, permitir a visualização no *browser* de documentos do editor de texto Word;

d) JAVASCRIPT: também é capaz de aumentar a capacidade de processamento do *browser*. O *JavaScript* é uma linguagem de *script* que pode ser embutida na página HTML, oferecendo algumas formas de controle da página, como a validação de campos;
e) VBSCRIPT: possui a mesma filosofia do *JavaScript*, mas utiliza a sintaxe da linguagem *Visual Basic* da *Microsoft*, ao invés da sintaxe da linguagem *Java*;
f) CSS (*Cascading Style Sheet*): permite que os estilos dos elementos da página (espaçamento, cores, fontes, margens, etc.) sejam especificados separadamente da estrutura do documento, facilitando dessa forma, uma futura modificação no estilo da página;
g) XML (*Extensible Markup Language*): é uma linguagem de marcação, tal como o HTML. O XML lida com rótulos (*tags*) sendo possível definir conjuntos de *tags* próprias. A definição do padrão de *tags* possibilita a criação de documentos num formato XML que podem ser facilmente interpretados pelo *Browser*. Diferentemente do HTML, no XML não há *tags* para aparência dos dados. O XML é também muito utilizado para padronizar a troca de informações entre sistemas.

2.4 TECNOLOGIAS PARA O PROCESSAMENTO NO SERVIDOR

Uma aplicação na *Web* é normalmente desenvolvida em uma arquitetura de três camadas: a camada do cliente, responsável pelo gerenciamento da *interface*, a camada intermediária, responsável pelos modelos de negócios e a camada de dados, responsável pelo acesso aos dados da aplicação.

Segundo Zoltán (apud Fleury, 2002), na camada intermediária, ocorre realmente o trabalho de programação do aplicativo *Web*, sendo esta camada a responsável por processar a informação enviada pelo cliente (*browser*), processar a regra de negócio (que pode estar em outra camada), interagir com o banco de dados, preparar a resposta (quase sempre na forma de uma página HTML) e enviá-la ao cliente. Os componentes dessa camada estão no servidor *Web* e são capazes de utilizar os recursos desses servidores e dos demais recursos conectados para realizar o processamento. É importante perceber que a forma com que todas essas

tecnologias trabalham é similar: recebem uma solicitação do cliente, processam essa solicitação e respondem na forma de uma página HTML.

A seguir, são destacadas algumas destas tecnologias:

a) CGI (*Common Gateway Interface)*: é um padrão para interfaceamento de aplicações externas com servidores, como um servidor *Web*, por exemplo. O CGI é a aplicação mais básica para acessar os recursos do sistema no servidor e foi também a primeira tecnologia para o desenvolvimento de aplicações *Web*. Pode ser escrito em diversas linguagens, sendo as principais o *Perl* e o *C/C++;*

b) SSI (*Server Side Includes)*: utiliza rótulos especiais (*tags*), inseridos no documento HTML que são interpretados pelo servidor *Web*, possibilitando assim que as *tags* sejam substituídas por conteúdo dinâmico, de acordo com o processamento realizado no servidor. As *tags* do SSI são específicas para cada servidor *Web*;

c) ASP (*Active Server Pages)*: é uma tecnologia da *Microsoft* que utiliza os conceitos de SSI e CGI para a construção de conteúdo dinâmico, somente funcionando no *Internet Information Server* (IIS), o software servidor *Web* da *Microsoft*, ou seja, é exclusiva para a plataforma Windows. O código ASP é inserido no HTML e interpretado pelo servidor a cada requisição recebida. O ASP é talvez a mais popular linguagem de *script* servidora atualmente em uso;

d) PHP (*Hypertext Preprocessor*): segue a mesma filosofia do ASP, porém pode ser executada por diferentes servidores, principalmente na plataforma *Unix* (*Solaris*, *Linux*, Etc.). Diferentemente do ASP, o PHP utiliza sintaxe baseada em *C*, *Java* e *Perl*. É uma tecnologia não-proprietária;

e) ISAPI/NSAPI: a tecnologia *Information Server Application Programming Interface* (ISAPI) é baseada no acesso a *Application Programming Interface* (API) do servidor *Web*, através do qual a aplicação servidora ISAPI ou NSAPI utiliza diretamente a API do servidor *Web* para executar a função desejada. A NSAPI é voltada para o *Netscape Server* e a ISAPI é a tecnologia para o servidor IIS da *Microsoft*. Algumas linguagens possibilitam o desenvolvimento de tais aplicativos, como é o caso do Delphi e C++;

f) SERVLETS: é um tipo de aplicativo *Java* que, executado no servidor *Web*, permite um funcionamento similar ao CGI. Os *Servlets Java* são multiplataforma e oferecem bom desempenho;

g) JSP: (*Java Server Pages)*: é uma tecnologia baseada em *Java* que utiliza o mesmo princípio do ASP, com código *Java* embutido na página HTML, o qual é interpretado a cada requisição pelo servidor *Web*. Tem se mostrado uma tecnologia bastante promissora;
h) COLDFUSION: linguagem de *script server* que também utiliza uma filosofia similar ao ASP e JSP. Possui sintaxe própria e é uma tecnologia proprietária;

Uma outra tecnologia disponível atualmente é o WebSnap, da *Borland*. Por ser objeto de estudo neste trabalho, ela será detalhada no próximo capítulo.

3 WEBSNAP

De acordo com Matos (2002), até o Delphi 5, o conjunto de componentes que encapsulava toda a tecnologia para a criação de aplicativos *Web* se chamava *WebBroker*. A melhor opção era o conjunto de componentes *Internet Express*, que fornecia uma *interface* de programação, unindo XML, *JavaScript* e HTML. A partir do Delphi 6 surgiu o WebSnap.

O WebSnap não substitui o que o Delphi já oferecia para desenvolvimento internet, mas surgiu para completar e estender as tecnologias *WebBroker* e *Internet Express*, evoluindo o conceito de formulários e fornecendo uma melhor *interface*, contendo uma coleção de componentes voltados ao desenvolvimento *Rappid Application Development* (RAD) de aplicações *Web*.

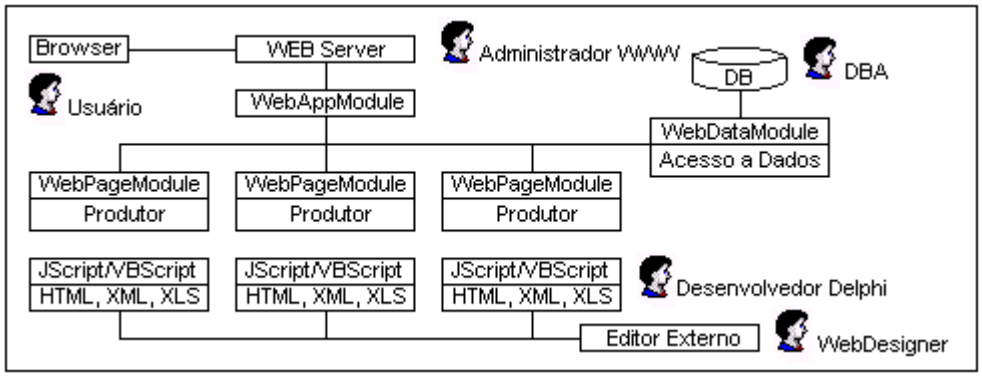
O Websnap é um novo paradigma para o desenvolvimento de aplicativos *Web*, possuindo diversas vantagens em relação ao *WebBroker*, como o fato de existir muitos componentes prontos para uso, para manipular tarefas comuns, como *login* de usuários, gerenciamento de sessões e inúmeros novos recursos, como a criação de múltiplos módulos, suporte a módulos do *Apache* (uma alternativa ao *Microsoft Internet Information Server*), *server-side scripts* (*scripts* no lado do servidor), facilidades no uso de banco de dados e muitas outras novidades.

3.1 ARQUITETURA WEBSNAP

A primeira diferença notável entre WebSnap e *WebBroker* é que, em vez de ter um único módulo de dados com várias ações eventualmente conectadas aos componentes produtores, o WebSnap tem vários módulos de dados, cada um correspondente a uma ação e tendo um componente produtor com um arquivo HTML anexado. Pode-se adicionar várias ações em uma página/módulo, mas a idéia é estruturar os aplicativos em torno de ações. Assim como nas ações, o nome da página é indicado no caminho da requisição (Cantu, 2002).

A FIGURA 2 ilustra a arquitetura WebSnap, com a ligação entre os módulos e onde cada profissional atua no processo (Pauli, 2002).

FIGURA 2 – MODELO WEBSNAP E OS PAPÉIS NA EQUIPE DE DESENVOLVIMENTO



Fonte: (Pauli, 2002).

Nesta arquitetura destacam-se:

a) *WebAppModule*: representa a criação de um novo módulo de aplicação WebSnap;

b) *WebPageModule*: representam páginas web criadas na aplicação WebSnap;

c) *WebDataModule*: equivalente ao módulo de dados tradicional do Delphi;

Pauli (2002) destaca que o WebSnap facilita muito a vida de empresas que desenvolvem aplicações de comércio eletrônico, principalmente às que possuem programadores e *webdesigners* em uma mesma equipe. Usando o WebSnap o programador Delphi desenvolve a aplicação principal à parte de acesso a dados e das regras de negócio que ficam do lado do servidor. O *webdesigner*, por sua vez, customiza a *interface* e adiciona alguma funcionalidade extra, usando um editor HTML/*JavaScript* apropriado.

Normalmente, os *scripts* WebSnap são escritos em *JavaScript*, uma linguagem baseada em objetos muito comum para programação de internet, pois ela é a única linguagem de *scripts* geralmente disponível nos navegadores (no lado do cliente). Na verdade, o WebSnap usa o mecanismo *ActiveScripting* da *Microsoft*, que suporta *JScript* (uma variação de *JavaScript*) e *VBScript*.

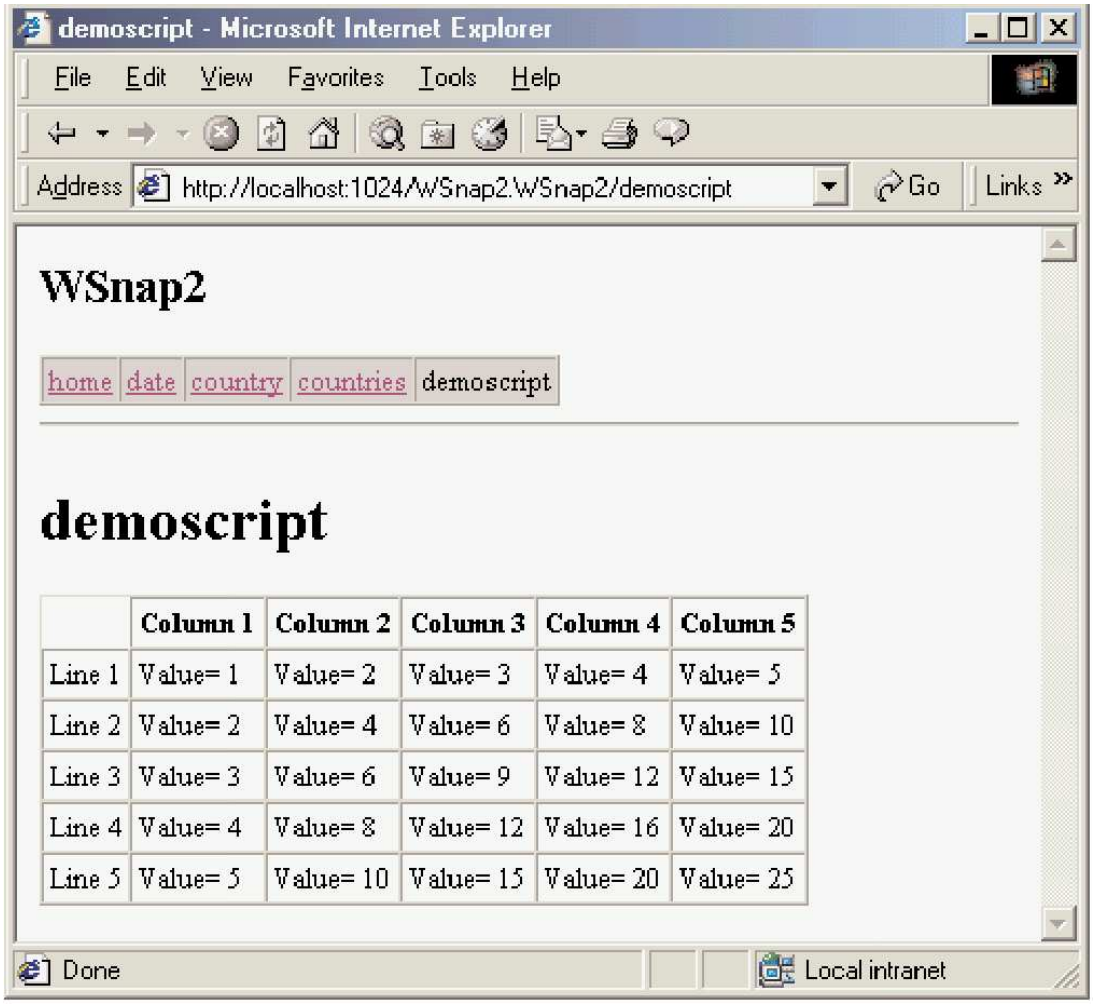
Como exemplo do que pode-se fazer com *scripts* é apresentado no QUADRO 2 e na FIGURA 3 um exemplo de uma página *demoscript*. O *script* dessa página pode gerar uma tabela completa de valores multiplicados.

QUADRO 2 – CÓDIGO COM *SCRIPTS*

```
<table border=1 cellspacing=0>
<tr>
  <th>&hbsp;</th>
  <% for (j=1;j<=5;j++){%>
  <th>Column <%=j %></th>
  <% } %>
</tr>
<%for (i=1;i<=5;i++){%>
<tr>
  <td>Line <%=i %></td>
  <% for (j=1;j<=5;j++){%>
  <td>Value=<%=i*j%></td>
  <% } %>
</tr>
<% } %>
</table>
```

Fonte: Cantu, 2002.

FIGURA 3 – EXEMPLO DE *SCRIPTS*



Fonte: Cantu, 2002.

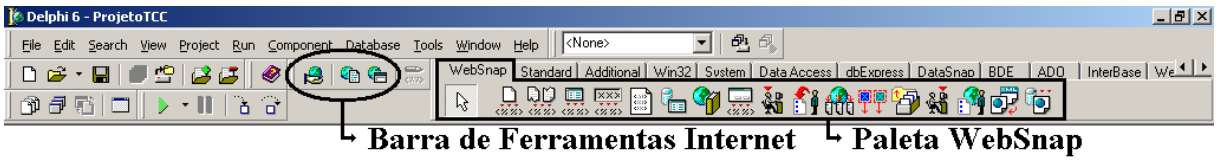
No *script* apresentado no QUADRO 2, o símbolo <%= substitui o comando *Response.Write* que é utilizado para escrever uma linha na página.

Outro recurso importante do *script* no lado do servidor é a inclusão de páginas dentro de outras páginas. Por exemplo, para modificar o menu, pode-se incluir o código HTML e o *script* relacionado em um único arquivo, em vez de mudá-lo e mantê-lo em várias páginas (Cantu, 2002).

3.2 CRIANDO APLICATIVOS COM WEBSNAP

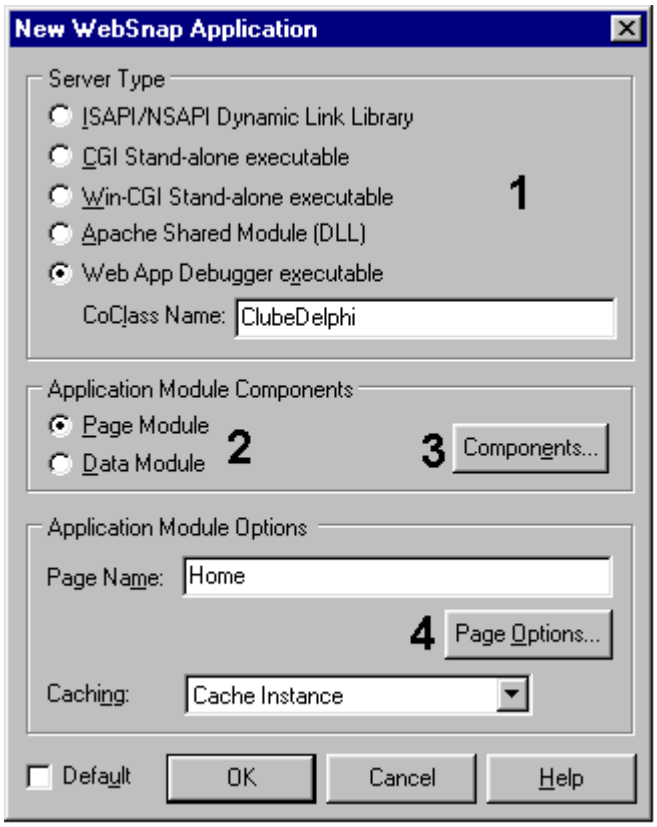
O ponto de partida do desenvolvimento de um aplicativo WebSnap é uma caixa de diálogo que pode ser chamada na página WebSnap da caixa de diálogo *New Items* (*File* | *New* | *Other*) ou usando a barra de ferramentas Internet do IDE, que contém atalhos para os itens principais do WebSnap conforme ilustra a FIGURA 4 (Pauli, 2002).

FIGURA 4 – PALETA WEBSNAP E A BARRA DE FERRAMENTAS INTERNET



Na barra de ferramentas, quando escolhido *New WebSnap Application*, uma caixa de diálogo resultante, ilustrada na FIGURA 5, é apresentada, permitindo configurar o tipo de servidor, o tipo do módulo de aplicação, serviços do módulo principal e configurar qual o componente produtor será utilizado no módulo principal.

FIGURA 5 – CRIAÇÃO DE UMA APLICAÇÃO WEBSNAP



As opções de configuração da aplicação são:

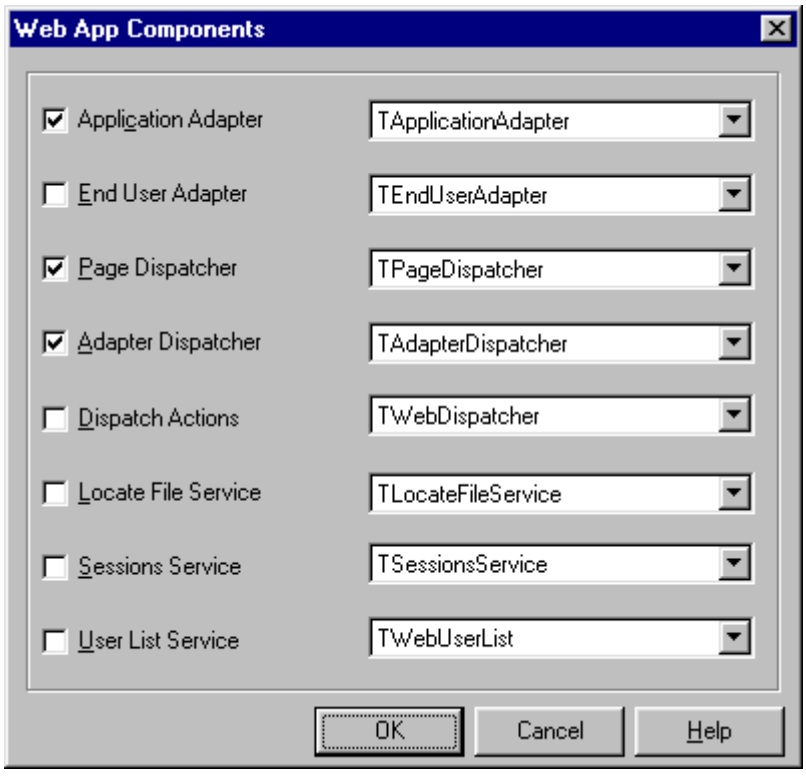
a) tipo de servidor (item 1): especifica o tipo de servidor *Web* a ser utilizado na aplicação:
   - ISAPI/NSAPI: cria um DLL, que pode ser compartilhada entre vários usuários;
   - CGI: cria um servidor que é instanciado uma vez para cada usuário;
   - WINCGI: cria um executável que opera com servidores Web de 16bit;
   - Apached Shared Module: novidade trazida pelo Delphi 6. Cria uma DLL compatível com o servidor Web Apache (uma alternativa ao *Microsoft Internet Information Server*);
   - Web App Debugger Executable: versão utilizada somente para teste. Este tipo de aplicativo permite a depuração e execução de aplicações WebSnap sem a necessidade de um servidor *web*. Quando a aplicação estiver pronta, deve-se convertê-la para um dos outros quatro tipos;

b) tipo de módulo da aplicação (item 2): define como será o módulo principal da aplicação:

- *Page Module*: representa uma página *Web*. O módulo principal será uma *WebAppPageModule* gerando conteúdo HTML, feito através de um componente produtor associado (como um *DataSetPageProducer* ou *AdapterPageProducer*). Em *Page Options* (item 4), configura-se qual componente produtor será utilizado no módulo principal;
- *Data Module*: é o equivalente ao módulo de dados tradicional do Delphi. O módulo principal será um *WebAppDataModule*, que não produz conteúdo. O *Web Data Module* é usado quando vários *Web Page Modules* necessitam compartilhar alguns componentes, como *datasets* e *databases;*

c) serviços do módulo principal (item 3): o botão *Components* mostra a caixa de diálogo ilustrada na FIGURA 6, na qual se escolhe quais serviços devem ser adicionados ao módulo principal de uma aplicação que contém vários componentes, com cada um responsável por implementar uma determinada funcionalidade:

FIGURA 6 – CONFIGURAÇÃO DOS COMPONENTES BÁSICOS DO APLICATIVO



A funcionalidade de cada componente listado na FIGURA 6 será detalhada a seguir. Na FIGURA 7 é apresentada a tela com os componentes setados na FIGURA 6.

FIGURA 7 – COMPONENTES SETADOS

3.3 PALETA WEBSNAP

Na paleta Websnap encontram-se os seguintes componentes:

a) *Adapter*: permite que campos e ações fiquem acessíveis através de *scripts*. Quando for necessário construir um formulário para entrada de valores e depois recuperar estes valores, pode-se utilizar o componente *Adapter*. Um exemplo é a utilização do mesmo para fornecer uma entrada para campos NOME e SENHA, usando a propriedade *Data* para colocar os campos e *Actions* para as ações;

b) *PagedAdapter*: tem suporte interno para exibir seu conteúdo em múltiplas páginas;

c) *DataSetAdapter*: é um tipo especializado de *Adapter* que obtém e seta valores de um *Dataset*. O *DataSetAdapter* tem um mecanismo para recuperar informações sobre o registro original que está sendo alterado. Isso é importante em um ambiente como a *Web*, onde cada requisição é processada como um novo pedido;

d) *LoginFormAdapter*: utilizado para construir um formulário de *login* para o aplicativo, automaticamente ligado à lista de usuários;

e) *StringValuesList*: este componente possui uma propriedade chamada *Strings*, onde são fornecidos os possíveis valores para um campo. Para que a lista de valores seja exibida no formulário, basta alterar a propriedade *ValuesList* num campo qualquer do *DataSetAdapter*;

f) *DataSetValuesList*: este recurso é utilizado para puxar valores de outra tabela. Esse componente possui uma propriedade *Dataset* que indica de onde serão retirados os valores para exibição e os códigos correspondentes. Para fazer a vinculação no *DataSetAdapter*, altera-se a propriedade *ValuesList* de qualquer campo no *DataSetAdapter*;

g) *WebAppComponents*: centraliza o acesso aos componentes e serviços da aplicação. Suas propriedades nada mais são que ponteiros para os demais componentes do módulo principal;

h) *ApplicationAdapter*: torna campos e ações da aplicação acessíveis via *server-side scripts*, por meio do objeto Application;

i) *EndUserAdapter*: é utilizado para acessar informações de usuário, sessão e para construir um formulário de login para o aplicativo;

j) *EndUserSessionAdapter*: usado pelo serviço de *login*, em conjunto com o componente *TSessionsService*. A propriedade *LoginPage* indica a página responsável por gerar o conteúdo do *login*, geralmente usando um *LoginFormAdapte*r e um *AdapterPageProducer*. Este componente armazena informações sobre o usuário que está logado na aplicação servidora;

k) *PagedDispatcher*: responsável por traduzir as mensagens da requisição http e associar o respectivo *Web Page Module*. A propriedade *DefaultPage* indica a página padrão da aplicação;

l) *AdapterDispatcher*: encarregado de manipular o envio de informações geradas por uma ação de um formulário HTML;

m) *LocateFileService*: pode-se utilizar este componente para controlar como o servidor deve localizar os *templates*. Templates são arquivos HTML criados junto ao *Web Page Module*, contendo HTML e *server-side scripts*. Pode-se, por exemplo, especificar outra localização (diretório) para os arquivos de *template*, diferente da usada pelo servidor;

n) *SessionsService*: utilizado quando se precisa manter informações persistentes sobre um determinado usuário, como o conteúdo de um “carrinho de compras” por exemplo. A propriedade *DefaultTimeOut* indica quanto tempo deve-se aguardar pela liberação de uma sessão depois de um período de inatividade especificado em minutos. O serviço de sessões permite que a aplicação servidora mantenha

informações sobre tudo o que o usuário está fazendo: páginas acessadas, operações feitas, acessos e *logins*. Como exemplo, pode-se usar este serviço para construir facilmente uma loja virtual, onde o servidor armazena informações sobre todos os produtos que o usuário comprou durante toda a vida útil da sessão (neste caso um processo de compra), no final fazendo uma listagem de produtos com base nos valores da sessão do usuário.

o) *WebUserList*: mantém uma lista com os usuários que podem acessar as páginas que estejam com o *flag wpLoginRequierd* habilitado. Utiliza-se a propriedade *UserItems* para gerenciar usuários;

p) *XslPageProducer*: cria uma página *Web* transformando os dados descritos com o *Extensible Markup Language* (XML) em *Extensible Stylesheet Language* (XSL);

q) *AdapterPageProducer*: permite passar informações de código Delphi compilado para o *script* interpretado, fornecendo uma *interface* de *script* para um aplicativo Delphi.

3.3.1 ADAPTADORES

Segundo Cantu (2002), um adaptador permite passar informações do código Delphi compilado para o *script* interpretado, fornecendo uma *interface* de *script* para o aplicativo Delphi. Os adaptadores contêm campos que representam dados e contêm ações que representam comandos. Os *scripts* no lado do servidor podem acessar esses valores e executar esses comandos, passando parâmetros específicos para eles.

Tecnicamente, os adaptadores implementam uma *interface* chamada *IDispatch* que pode ser acessada pelo *script* através de uma linguagem de mecanismo *Active Scripting*, como o *JavaScript*. O componente produtor de página é responsável por chamar o mecanismo *Active Scripting* e tem uma propriedade indicando a linguagem do *script*.

Para recuperar ou validar os valores dos campos quando a ação for chamada, utiliza-se o evento *OnExecute* da ação, como apresentado no QUADRO 3.

QUADRO 3 – RECUPERAR OU VALIDAR VALORES

```
Procedure TpageProducerPageDemo.ENVIARexecute(Sender: TObject;
Params: Tstrings);
Begin
IF NOME_CAMPO.Action.Value= ‘ ’ Then
  Adapter.Errors.AddError(Exception.Create(‘Nome não pode ser nulo’));
End;
```

Fonte: Pauli, 2002.

Desenvolver a *interface* com o usuário para o formulário e o *script* relacionado levaria algum tempo, usando HTML simples. Mas o componente *AdapterPageProducer* tem um projetista de HTML integrado, que a *Borland* chama de *Web Surface Designer*. Usando essa ferramenta, pode-se adicionar visualmente um formulário à página HTML e adicionar o componente *AdapterFieldGroup* a ela. Para ter editores para os dois campos automaticamente apresentados, é necessário conectar esse grupo de campos ao adaptador. Então, para ter botões para todas as ações do adaptador, deve-se adicionar um componente *AdapterCommandGroup* e conectá-lo ao componente *AdapterFieldGroup* (Cantu, 2002).

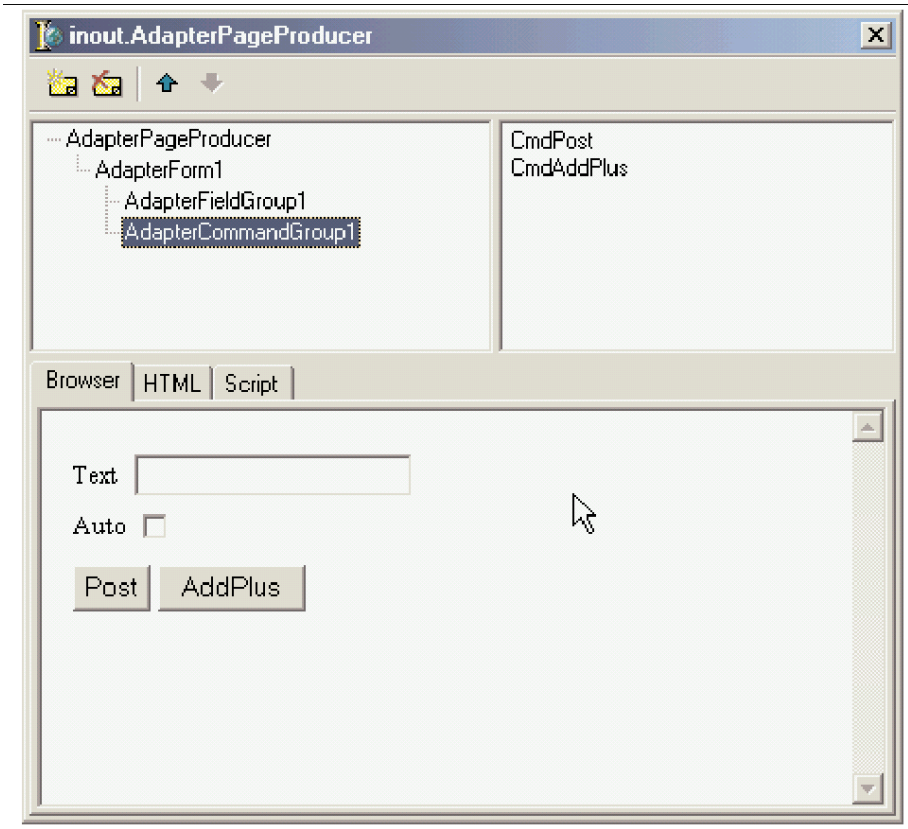
Os campos e os botões são apresentados automaticamente, se as propriedades *AddDefaultFields* e *AddDefaultCommands* do grupo de campos e do grupo de comandos forem configuradas. O efeito das operações visuais executados para construir o formulário ilustrado na FIGURA 8, está resumido no trecho do QUADRO 4.

QUADRO 4 – *SCRIPT* PROJETISTA DO *ADAPTERPAGEPRODUCER*

```
object AdapterPageProducer: TAdapterPageProducer
  object AdapterForm1: TAdapterForm
    object AdapterFieldGroup1: TAdapterFieldGroup
      Adapter = Adapter1
      object FldText: TAdapterDisplayField
        FieldName = 'Text'
      end
     object FldAuto: TAdapterDisplayField
       FieldName = 'Auto'
     end
   end
    object AdapterCommandGroup1: TAdapterCommandGroup
      DisplayComponent = AdapterFieldGroup1
      object CmdPost: TAdapterActionButton
        ActionName = 'Post'
      end
      object CmdAddPlus: TAdapterActionButton
        ActionName = 'AddPlus'
      end
    end
  end
end
```

Fonte: Cantu, 2002.

FIGURA 8 – *WEB SURFACE DESIGNER*

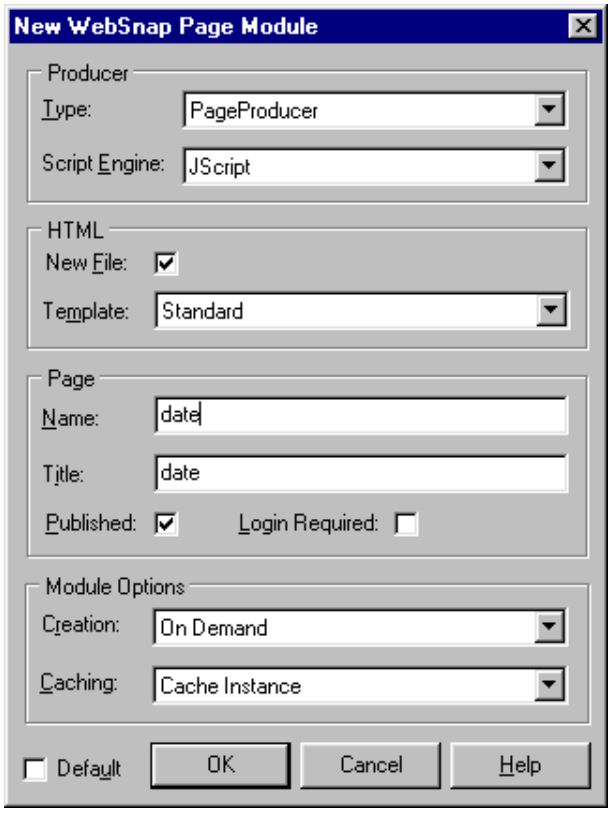


Fonte: Cantu, 2002.

3.4 CRIANDO UMA *WEB PAGE MODULE*

Quando é criado um novo *WebSnap Page Module* (página WebSnap) a caixa de diálogo na FIGURA 9 é apresentada. Pode-se criar um novo *Web Page Module* para cada página *Web* da aplicação. No momento da criação é escolhido o tipo de componente a ser usado para produzir o conteúdo da página. As opções de produtores são: *AdapterPageProducer*, *DataSetPageProducer*, *InetXPageProducer*, *PageProducer* ou *XSLPageProducer*.

FIGURA 9 – CRIAÇÃO DE UM *WEB PAGE MODULE* COM UM *ADAPTERPAGEPRODUCER*



Dependendo do tipo de produtor escolhido, a opção *New File* poderá ser marcada. Caso a opção estiver marcada, um novo arquivo HTML será criado com o mesmo nome da *unit*, ficando automaticamente associado ao *WebPageModule*. Este arquivo conterá alguns *tags* HTML ou *server-side scripts* e pode ser facilmente alterado em um editor externo, ou no próprio Delphi.

A opção *Module Options* permite especificar como deverá ser feita a criação e a destruição do objeto. A opção *Creation* pode ser *On Demand* (o módulo é criado quando

alguém solicitar seu conteúdo) ou *Always* (o módulo é criado quando o aplicativo é iniciado). A opção *Caching* permite configurar a instância do objeto para permanecer em memória após o uso (*Cache Instance*), ou para que seja destruído após devolver o conteúdo do produtor associado (*Destroy Instance*).

3.5 WEBSNAP E BANCO DE DADOS

De acordo com Cantu (2002), uma das áreas em que o Delphi sempre se destaca é a programação em bancos de dados. Por isso, não é surpresa existir um grande suporte a manipulação de conjuntos de dados dentro da estrutura WebSnap. Especificamente pode-se usar o componente *DataSetAdapter* para conectar-se a um conjunto de dados e exibir seus valores em um formulário ou em uma tabela, usando o editor visual do componente *AdapterPageProducer*.

Como exemplo, um novo aplicativo WebSnap foi criado com um componente *AdapterPageProducer* como página principal, para apresentar uma tabela em uma grade, e outro componente *AdapterPageProducer* em uma página secundária, para mostrar um formulário com um único registro. Ao aplicativo, também foi adicionado um módulo de dados WebSnap, como um *container* dos componentes de conjunto de dados. O módulo de dados tem um componente *ClientDataSet* conectado a um conjunto de dados *dbExpress* através de um provedor e baseado em uma conexão *InterBase*, como segue no QUADRO 5.

QUADRO 5 – EXEMPLO DE *SCRIPT* DO COMPONENTE *CLIENTDATASET*

```
object ClientDataSet1: TClientDataSet
  Active = True
  ProviderName = 'DataSetProvider1'
end
object SQLConnection1: TSQLConnection
  Connected = True
  ConnectionName = 'IBLocal'
  LoginPrompt = False
end
object SQLDataSet1: TSQLDataSet
  SQLConnection = SQLConnection1
  CommandText =
    'select CUST_NO, CUSTOMER, ADDRESS_LINE1, CITY, STATE_PROVINCE, ' +
    ' COUNTRY from CUSTOMER'
end
object DataSetProvider1: TDataSetProvider
  DataSet = SQLDataSet1
end
```

Fonte: Cantu, 2002.

Segundo Cantu (2002), pode-se adicionar um componente *DataSetAdapter* à primeira página e conectá-lo ao componente *ClienteDataSet* do módulo da *Web*. O adaptador torna automaticamente disponíveis todos os campos do conjunto de dados e várias ações predefinidas para operar sobre ele (como *Delete, Edit, Apply*).

Assim como o componente *PagedAdapter*, o componente *DataSetAdapter* tem uma propriedade *PageSize* em que se pode indicar o número de elementos a serem exibidos em cada página. O componente também tem comandos que podem ser usados para navegar entre as páginas. Essa estratégia é particularmente conveniente quando se quer apresentar um conjunto de dados grande em uma grade.

O produtor de página correspondente tem um formulário contendo dois grupos de comandos e uma grade. O primeiro grupo de comandos (exibido abaixo da grade) tem comandos predefinidos pra manipular páginas: *CmdPrevPage, CmdNextPage* e *CmdGotoPage*. Esse último comando gera uma lista de números para as páginas, de modo que um usuário pode pular diretamente para cada uma delas. O componente *AdapterGrid* tem as colunas padrão, mais uma extra, contendo dois comandos, *Edit* e *Delete*. O grupo de comandos inferior tem um botão, usado para criar um novo registro.

3.5.1 MESTRE-DETALHE NO WEBSNAP

O componente DataSetAdapter tem suporte específico a relacionamentos mestre-detalhe entre conjuntos de dados. Depois de ter criado o relacionamento entre os conjuntos de dados, define-se um adaptador para cada conjunto de dados e, em seguida, conecta-se a propriedade *MasterAdapter* do adaptador do conjunto de dados de detalhes. Configurar o relacionamento mestre-detalhe entre os adaptadores os faz trabalhar de uma maneira transparente. Por exemplo, quando se muda o modo de trabalhar do mestre ou insere-se novos registros, os detalhes entram automaticamente no modo *Edit* ou são atualizados (Cantu, 2002).

3.6 SESSÕES, USUÁRIOS E PERMISSÕES

Segundo Cantu (2002), outra área muito interessante da arquitetura WebSnap é seu suporte a sessões e usuários. As sessões são suportadas usando-se uma estratégia clássica: *cookies* temporários. Esses *cookies* são enviados para o navegador, de modo que as requisições seguintes do mesmo usuário podem ser reconhecidas pelo sistema. Adicionando

dados a uma sessão, em vez de um adaptador de aplicativo, pode-se ter dados que dependem de sessão ou do usuário específico (embora um usuário possa executar várias sessões, abrindo várias janelas de navegador no mesmo computador). Para suportar sessões, o aplicativo mantém os dados na memória, recurso este que não está disponível no caso de programas CGI, por exemplo.

Além de ter páginas que exigem um login para acessar, pode-se fornecer a usuários específicos o direito de ver mais páginas do que outros. Todo usuário, tem um conjunto de direitos separados por pontos-e-vírgulas ou vírgulas. O usuário deve ter todos os direitos definidos para a página solicitada, geralmente listados nas propriedades *ViewAccess* e *ModifyAccess* dos adaptadores, que indicam, respectivamente, se o usuário pode ver os elementos determinados enquanto navegam ou se podem até edita-los. Essas configurações são muito granulares e podem ser aplicadas a adaptadores inteiros ou a alguns campos de adaptadores específicos. Por exemplo, pode-se ocultar algumas das colunas de uma tabela para determinados usuários, ocultando os campos correspondentes.

O componente global *PageDispacher* também tem os eventos *OnCanViewPage* e *OnPageAccessDenied* que podem ser usados para controlar o acesso às várias páginas do programa dentro do código do programa, permitindo o controle ainda maior.

4 DESENVOLVIMENTO DO TRABALHO

Neste capítulo são apresentados os requisitos do protótipo, as fases de seu desenvolvimento, considerando a seqüência de suas etapas e as características do protótipo, proporcionando assim o entendimento das funcionalidades do mesmo.

Além disso, serão também relacionados e discutidos os resultados obtidos a partir das avaliações da análise e implementação do protótipo desenvolvido.

4.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Com a realização deste trabalho pretendeu-se fazer uma verificação da efetividade dos novos recursos de desenvolvimento do Delphi 6 para aplicações de internet mais especificamente o WebSnap. Esta proposta baseou-se no fato da pouca ênfase dada a esta tecnologia atualmente no desenvolvimento de aplicações *web*.

Para isto foi utilizada essa nova coleção de componentes na construção de um protótipo de sistema de reserva de laboratório de ensino *on-line.*

Os requisitos identificados para o trabalho são:

a) ao acessar o protótipo, o mesmo deverá solicitar o login do usuário para que sejam dadas as devidas permissões;
b) devem existir dois perfis de usuários: administrador e usuário padrão;
c) o protótipo deve ser capaz de permitir que o usuário administrador mantenha os cadastros de usuário, disciplina, software, laboratório e horário;
d) o protótipo deve ser capaz de permitir que o usuário faça reservas de laboratório *on-line*;
e) o protótipo deve permitir que um usuário faça consultas de reservas realizadas;
f) o protótipo deve permitir que um usuário desmarque reservas realizadas.

4.2 ESPECIFICAÇÃO

A especificação do protótipo foi realizada utilizando a linguagem de modelagem *Unified Modeling Language* (UML). A UML é a padronização da linguagem de

desenvolvimento orientado a objetos para visualização, especificação, construção e documentação de sistemas (Furlan 1998).

Para a modelagem foram utilizados os diagramas de casos de uso, diagramas de classes e diagramas de seqüência. Os mesmos foram feitos utilizando a ferramenta *Rational Rose* da empresa *Rational Software Corp*.

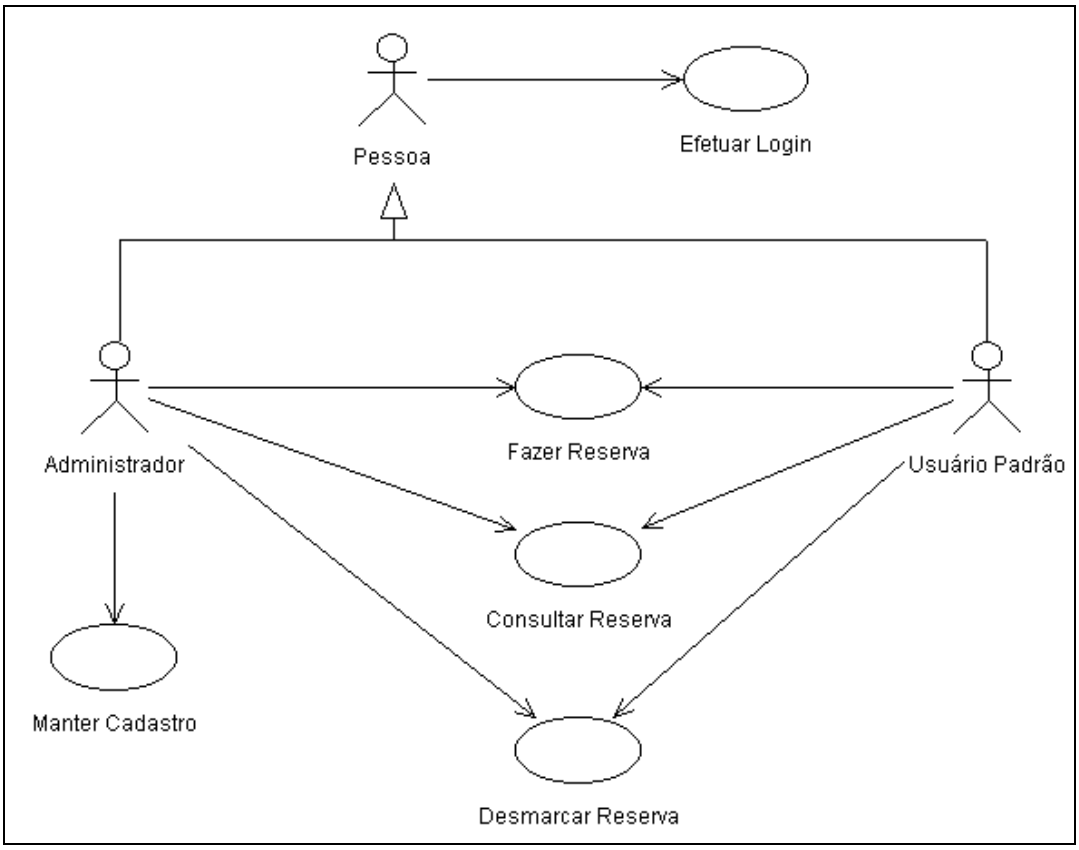
4.2.1 CASOS DE USO

Neste protótipo foram identificados cinco casos de uso. Eles são descritos a seguir:

a) efetuar Login: responsável pelo login do usuário e pela liberação de permissões de acesso e manutenção das páginas;
b) manter Cadastro: responsável pela entrada dos dados referentes aos cadastros de usuários, disciplinas, softwares, laboratórios e horários. Esses cadastros e eventuais manutenções são de total responsabilidade do usuário Administrador. Ao cadastrar um usuário, o mesmo pode ser administrador ou usuário padrão e ter disciplinas ligadas a ele. Ao cadastrar os laboratórios, pode-se incluir quais os softwares com licenças de uso disponíveis nesses laboratórios;
c) fazer Reserva: neste caso de uso, ao acessar a página de reservas o usuário informa a disciplina, software e horário. Através do software, o protótipo verificará quais os laboratórios que possuem a disponibilidade de licenças de uso, informando quais estão disponíveis no momento, para que possam ser registradas as reservas;
d) consultar Reserva: ao acessar a página de consultar reservas, são demonstradas todas as reservas, sendo que o usuário padrão só poderá modificar suas próprias reservas;
e) desmarcar Reserva: ao acessar as consultas de reserva, o usuário padrão tem direito de desmarcar suas próprias reservas. Caso o mesmo for administrador poderá desmarcar as reservas de qualquer usuário.

A FIGURA 10 demonstra os cinco casos de uso descritos anteriormente:

FIGURA 10 – DIAGRAMA DE CASO DE USO

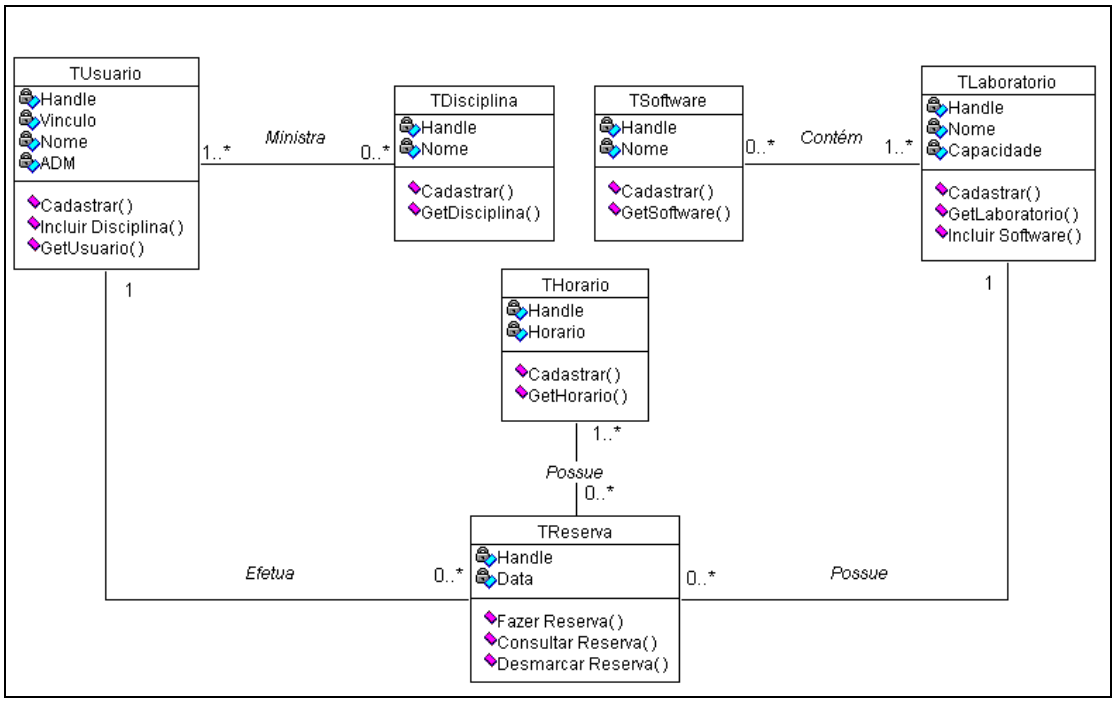


4.2.2 DIAGRAMA DE CLASSES

Segundo Furlan (1998), o diagrama de classe é a essência da UML resultado de uma combinação de diagramas propostos pelo OMT, Booch e vários outros métodos. Trata-se de uma estrutura lógica estática em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo, como classes, tipos e seus respectivos conteúdos e relações.

A FIGURA 11 demonstra o diagrama de classes do protótipo.

FIGURA 11 – DIAGRAMA DE CLASSES



Existem seis classes identificadas para o funcionamento do protótipo:

a) TUsuario: corresponde aos usuários cadastrados no protótipo e controla os acessos dos tipos de usuários (administrador, usuário padrão);

b) TDisciplina: corresponde às disciplinas dos usuários padrão cadastradas no protótipo e sempre estará ligada à classe TUsuario. É importante destacar que a ligação entre usuário e disciplina não é obrigatória, uma vez que o protótipo pode ter usuários que não ministram disciplinas. Um exemplo pode ser o próprio administrador do sistema;

c) TSoftware: corresponde aos softwares cadastrados no protótipo e podendo estar ligada a classe TLaboratório;

d) TLaboratorio: corresponde aos laboratórios cadastrados no protótipo e sempre estará ligada a classe TSoftware onde se encontram os softwares com direitos de uso para a classe TLaboratório. O cadastramento dos softwares é de responsabilidade do administrador. Assim, caso necessite de um software não cadastrado no protótipo o usuário padrão deverá entrar em contato com o administrador que verificará a possibilidade de instalar o software em algum laboratório. Isto é particularmente importante para que se mantenha o controle sobre as licenças de uso dos softwares instalados nos laboratórios e para que o

administrador tenha tempo de providenciar as instalações dos mesmos nos equipamentos;

e) TReserva: a classe TReserva é responsável por manter as reservas efetuadas pelos usuários nos laboratórios;

f) THorario: corresponde aos horários cadastrados no protótipo e poderá estar ligada a uma classe TReserva;

4.2.3 DIAGRAMA DE SEQÜÊNCIA

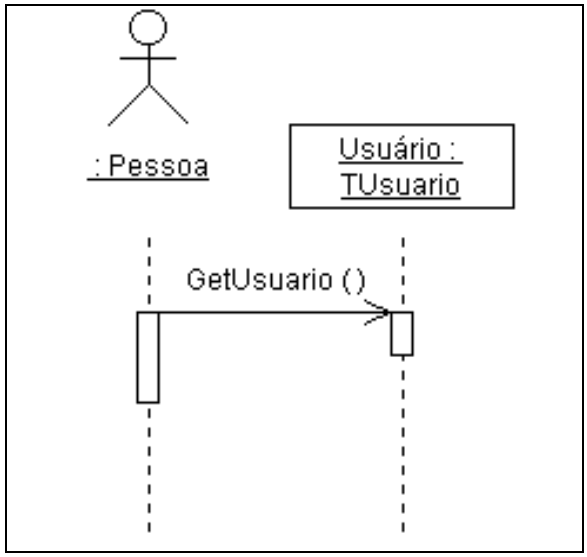
Os diagramas de seqüência representam a seqüência em que as ações ocorrem dentro do protótipo. Eles demonstram como é feita a troca de mensagens entre as classes. Para cada caso de uso apresentado anteriormente, foi feito um diagrama de seqüência, os quais serão demonstrados a seguir:

4.2.3.1 EFETUAR LOGIN

Para entrar no protótipo a pessoa deve se identificar através de seu Login e Senha. É executada a operação *GetUsuario* para assim efetuar o login conforme apresentado na FIGURA 12 do diagrama de seqüência Efetuar Login.

Através do login o protótipo deverá identificar se a pessoa é um administrador ou um usuário padrão, para assim dar seus devidos direitos de acesso ao sistema.

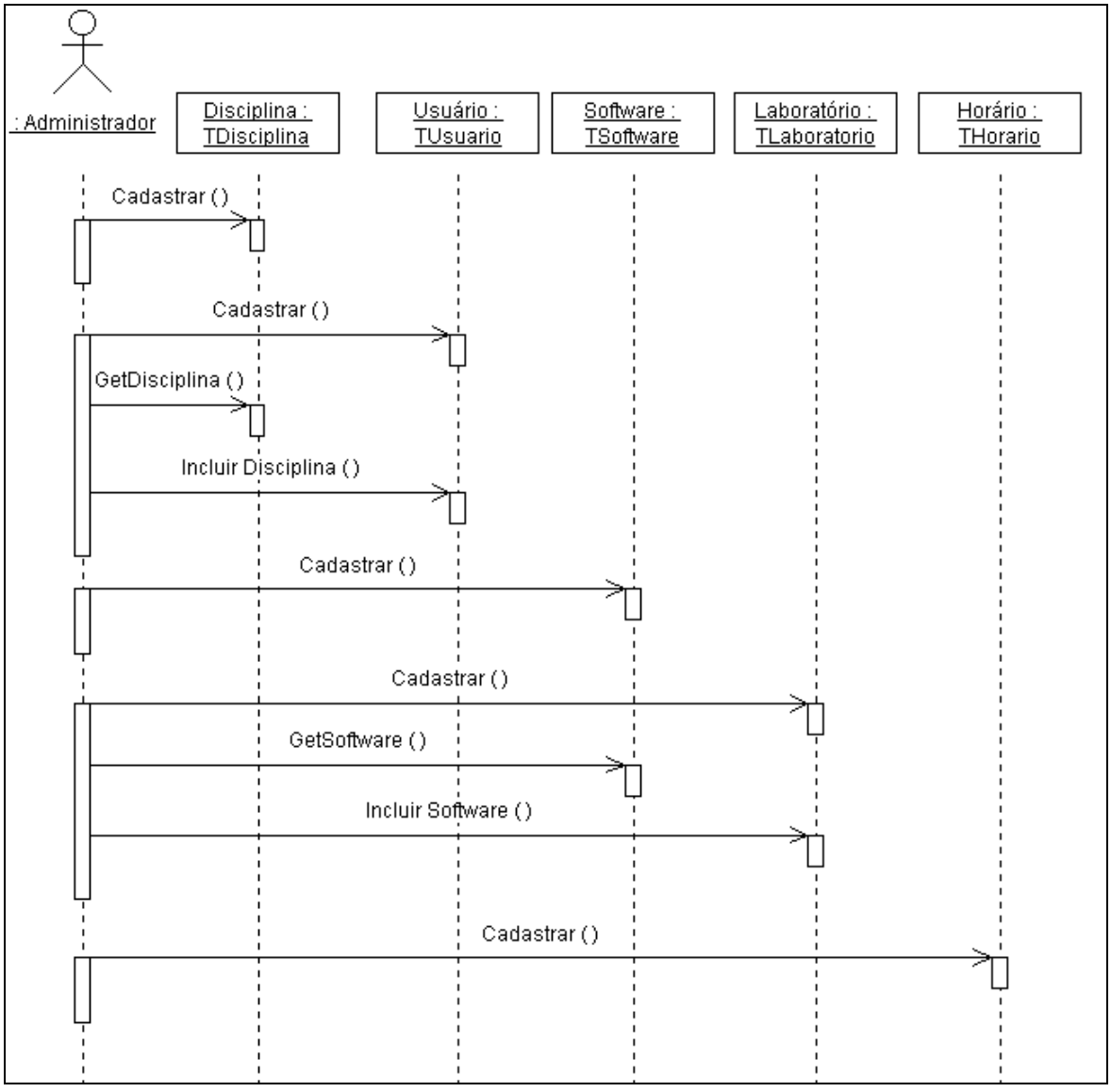
FIGURA 12 – DIAGRAMA DE SEQÜÊNCIA EFETUAR LOGIN



4.2.3.2 MANTER CADASTRO

O ator do tipo administrador efetua os cadastros, que são executados através da rotina "Cadastrar" existentes nas classes *TUsuario*, *TDisciplina*, *TSoftware*, *TLaboratorio* e *THorario* conforme apresentado na FIGURA 13 do diagrama de seqüência Manter Cadastro.

FIGURA 13 – DIAGRAMA DE SEQÜÊNCIA MANTER CADASTRO



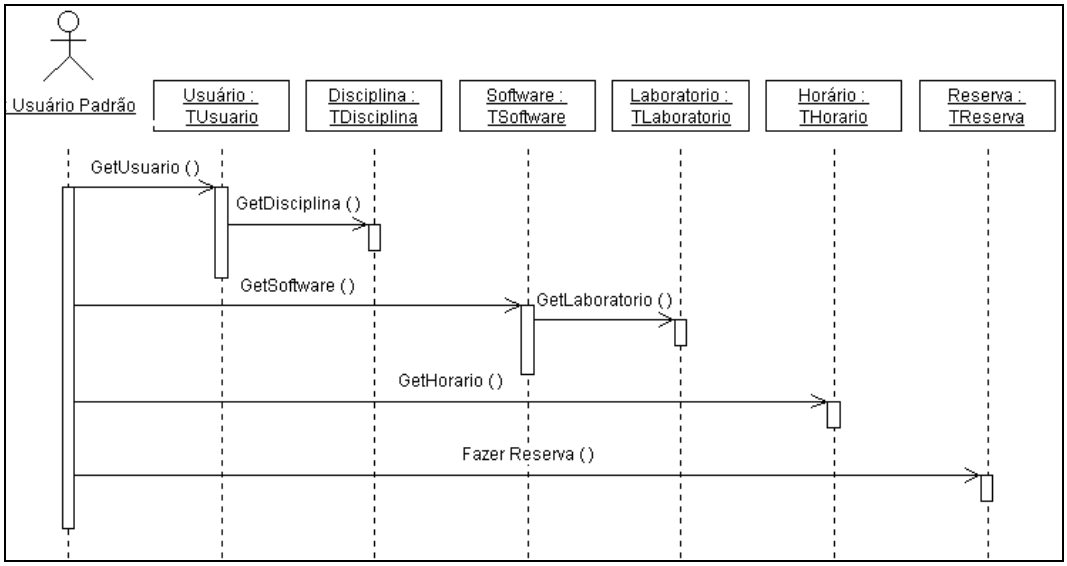
4.2.3.3 FAZER RESERVA

O ator pode fazer uma reserva, que é executada através da rotina “Fazer Reserva” da classe *TReserva*. Essa classe executa várias operações que são:

a) *GetUsuario*: responsável por pegar o usuário na tabela de dados;
b) *GetDisciplina*: caso o usuário for do tipo usuário padrão informar as disciplinas do mesmo;
c) *GetSoftware*: responsável por pegar na tabela de dados o software necessário para reserva do laboratório;
d) *GetLaboratorio*: informar quais os laboratórios disponíveis com direito de uso para o software escolhido anteriormente;
e) *GetHorario*: responsável por pegar na tabela de horários o horário da reserva;
f) *Fazer Reserva*: responsável por cadastrar a reserva.

A FIGURA 14 apresenta o diagrama de seqüência Fazer Reserva.

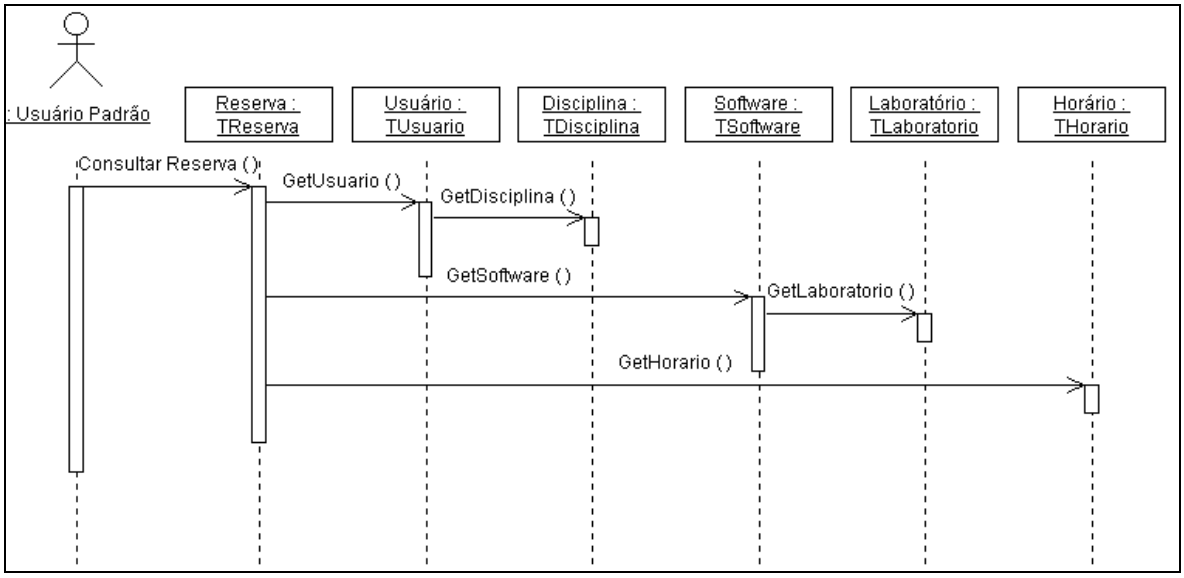
FIGURA 14 – DIAGRAMA DE SEQÜÊNCIA FAZER RESERVA



4.2.3.4 CONSULTAR RESERVA

O ator pode fazer consultas de reservas, que são executadas através da rotina "Consultar Reserva" da classe *TReserva*. Essa classe executa várias operações que são: *GetUsuario*, *GetDisciplina*, *GetSoftware*, *GetLaboratório*, *GetHorario*. A FIGURA 15 apresenta o diagrama de seqüência Consultar Reserva.

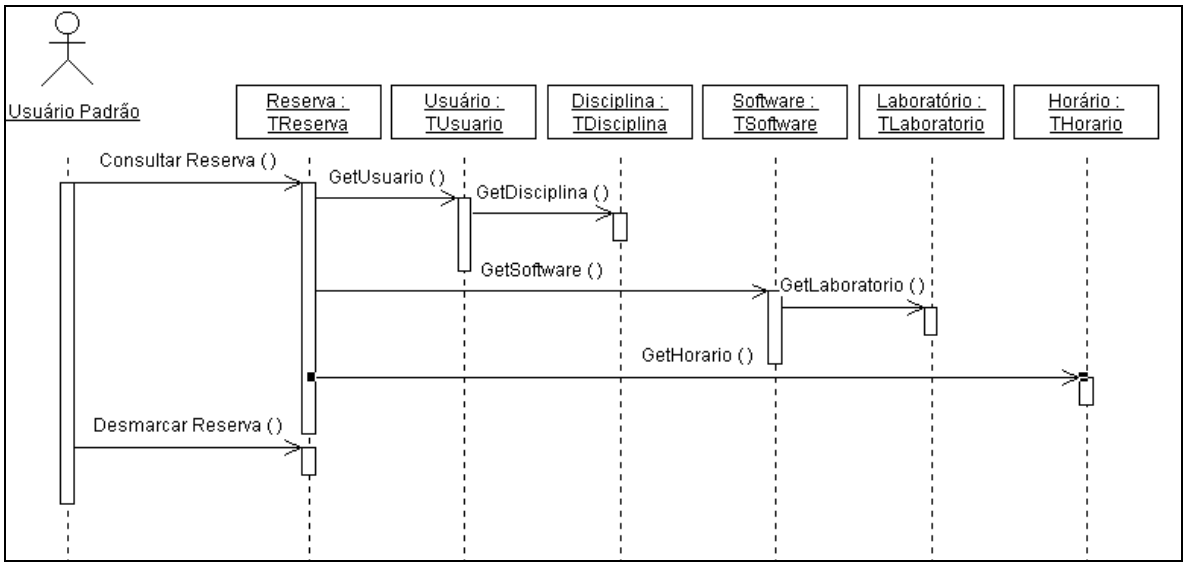
FIGURA 15 – DIAGRAMA DE SEQÜÊNCIA CONSULTAR RESERVA



4.2.3.5 DESMARCAR RESERVA

O ator pode desmarcar reservas, que são executada através da rotina “Desmarcar Reserva” da classe *TReserva.* Essa classe executa várias operações que são: *GetUsuario, GetDisciplina, GetSoftware, GetLaboratório, GetHorario.*

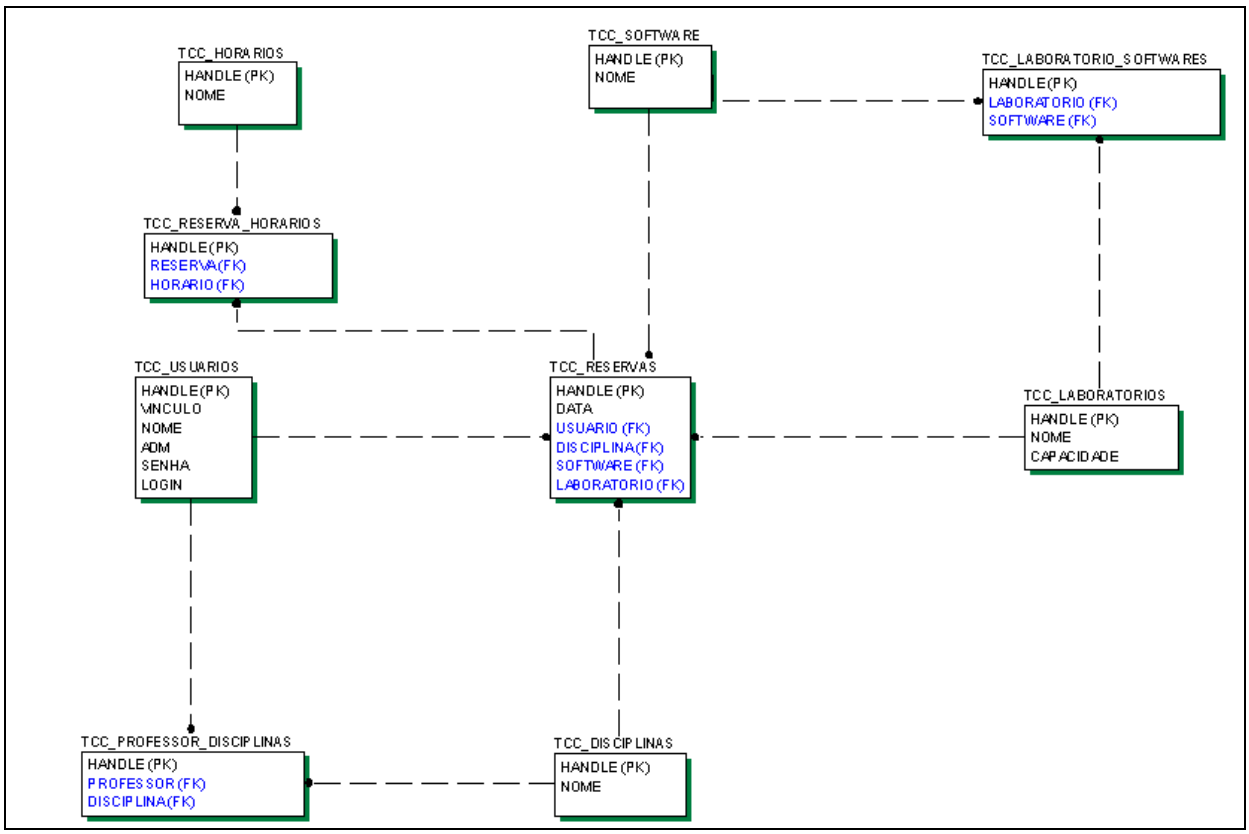
FIGURA 16 - DIAGRAMA DE SEQÜÊNCIA DESMARCAR RESERVA



4.2.4 MODELO DE DADOS

Neste modelo fica a estrutura dos dados utilizados no protótipo. Apesar de ser modelado baseado em objetos, os dados são armazenados em um banco de dados relacional, o MSSQL Server 2000. Sendo assim, na FIGURA 17 é apresentado o modelo de dados do protótipo.

FIGURA 17 - MODELO DE DADOS FÍSICO



4.3 IMPLEMENTAÇÃO

Para ter-se um melhor entendimento da implementação do protótipo, a seguir mostra-se as técnicas, ferramentas utilizadas e quais são as operacionalidades desta implementação.

4.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Durante o desenvolvimento desse trabalho foram utilizadas algumas tecnologias e ferramentas citadas a seguir, visando uma melhor compreensão da pesquisa como um todo.

Para a criação das tabelas de estrutura de dados foi utilizado o *ER/Studio* da empresa Embarcadero.

O banco de dados utilizado no protótipo foi o MSSQL Server 2000 da *Microsoft*.

Para implementação do protótipo foi utilizada a ferramenta *Borland* Delphi 6.0.

O desenvolvimento e testes foram realizados no *Internet Explorer* do sistema operacional *Windows XP Professional*.

Nos testes de portabilidade e implantação do protótipo utilizou-se o *Internet Information Service* (IIS) da *Microsoft* como servidor *Web*.

4.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

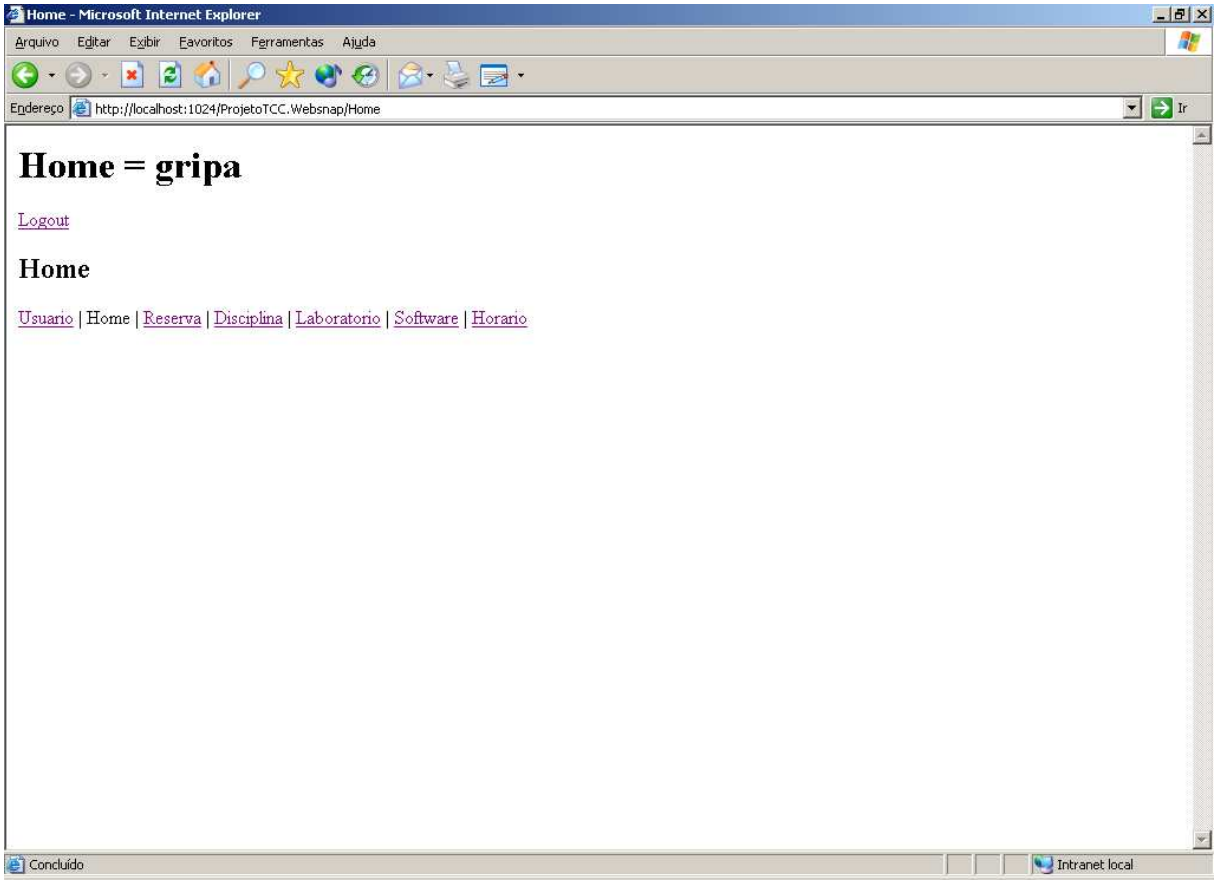
Neste item, será abordado o protótipo que é o objetivo geral deste trabalho, demonstrando o seu funcionamento e apresentando sua interface simples e objetiva.

Nesse protótipo foi explorada à parte de desenvolvimento do lado do programador, sendo assim, para um melhor entendimento será demonstrada detalhadamente a execução do mesmo. Começando com os diferentes comportamentos em conseqüência do login do usuário, podendo esse ser administrador ou usuário padrão.

4.3.2.1 ADMINISTRADOR

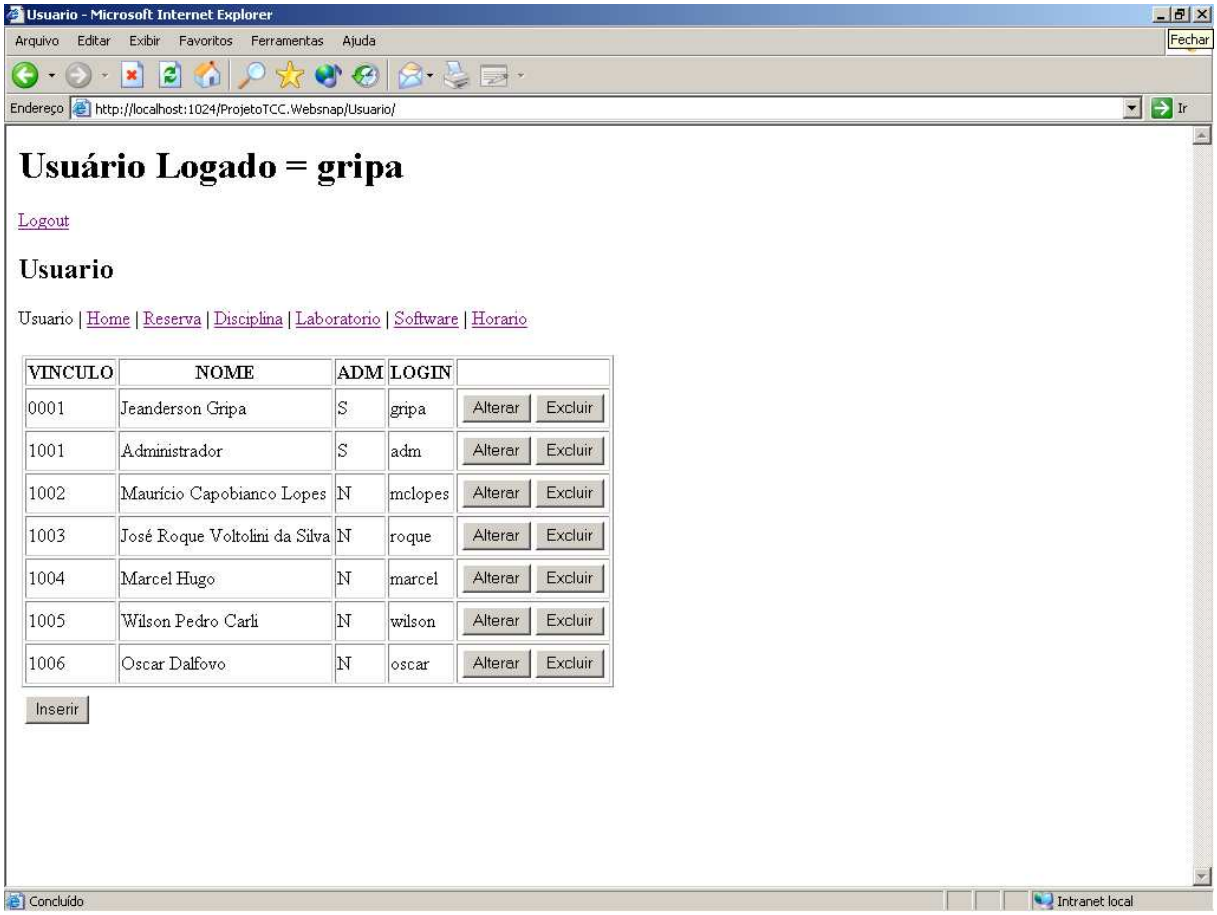
O administrador é responsável pela manutenção dos cadastros (Usuário, Disciplina, Laboratório, Software e Horário) e Reservas. Compreendendo a inserção, alteração e exclusão dos registros nas referidas tabelas. Sendo assim, após logado, o usuário administrador terá acesso a todos os cadastros disponibilizados no protótipo, conforme pode ser observado na FIGURA 18.

FIGURA 18 – TELA HOME DO ADMINISTRADOR



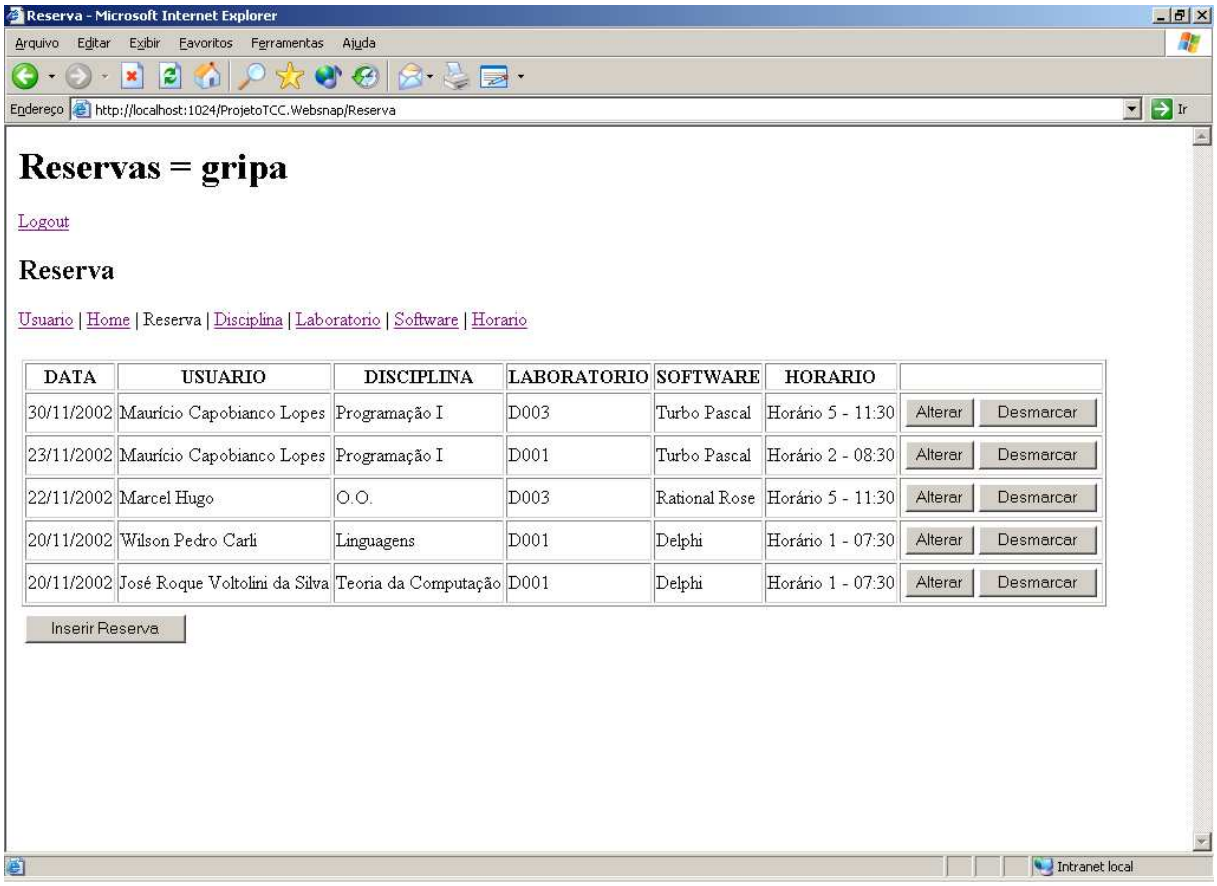
Além dos cadastros naturais é função do administrador manter a tabela de usuários e é apenas ele que pode inserir, excluir ou definir um outro usuário como administrador. Na FIGURA 19 pode-se observar a visão que o usuário administrador tem da página de usuários.

FIGURA 19 – TELA DE USUÁRIO DO ADMINISTRADOR



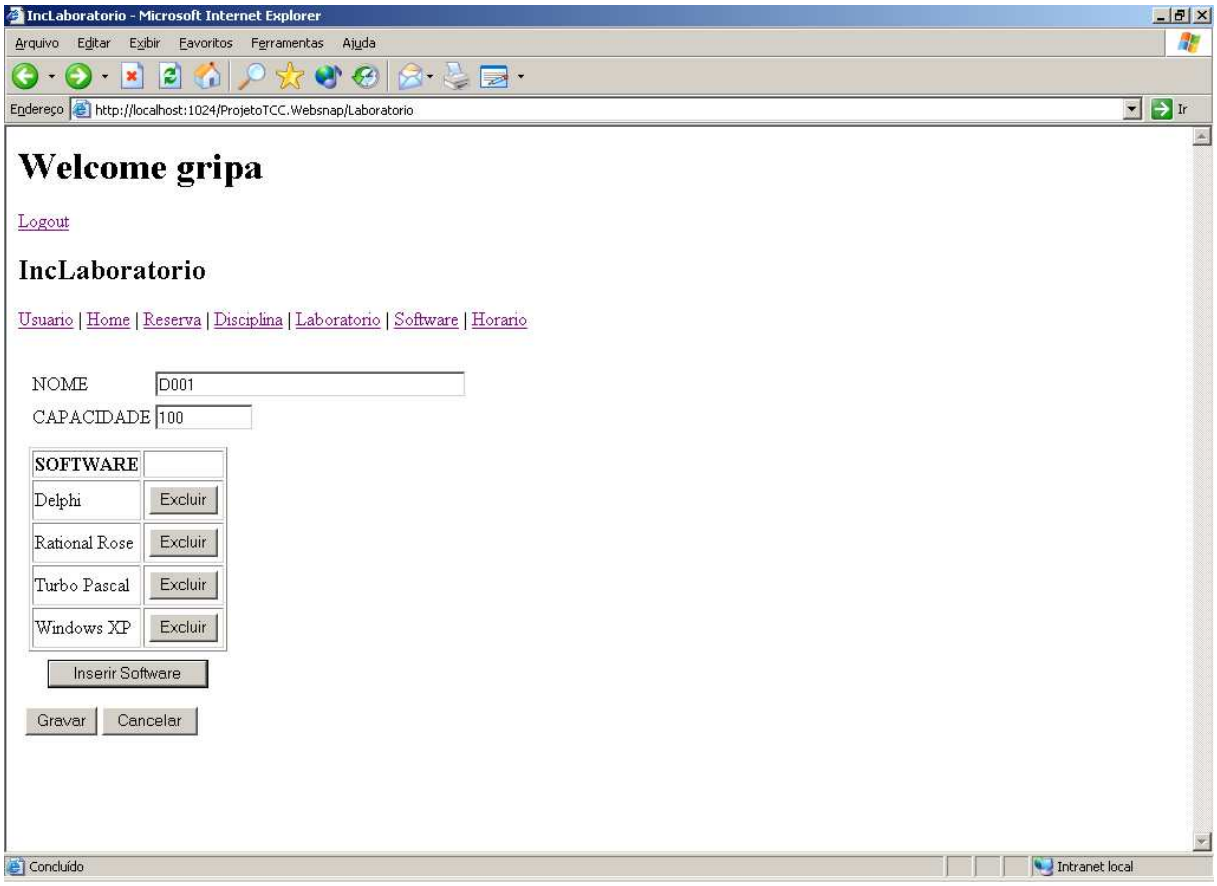
Na página de Reservas observa-se mais um diferencial do administrador. Nessa página o mesmo pode inserir, alterar ou desmarcar qualquer reserva, indiferente do usuário que cadastrou a mesma. A visão da página de Reservas obtida pelo administrador é demonstrada na FIGURA 20.

FIGURA 20 – TELA DE RESERVA DO ADMINISTRADOR



Ao administrador também cabe a função de manter as tabelas “filhas” como é o caso de Professor_Disciplinas e Laboratorio_Softwares. Nesses casos específicos a sua manutenção ocorre através da página de sua tabela “pai”, como se observa na FIGURA 21, onde a tela de Laboratório disponibiliza a inserção ou exclusão dos softwares com direitos de uso para o respectivo laboratório.

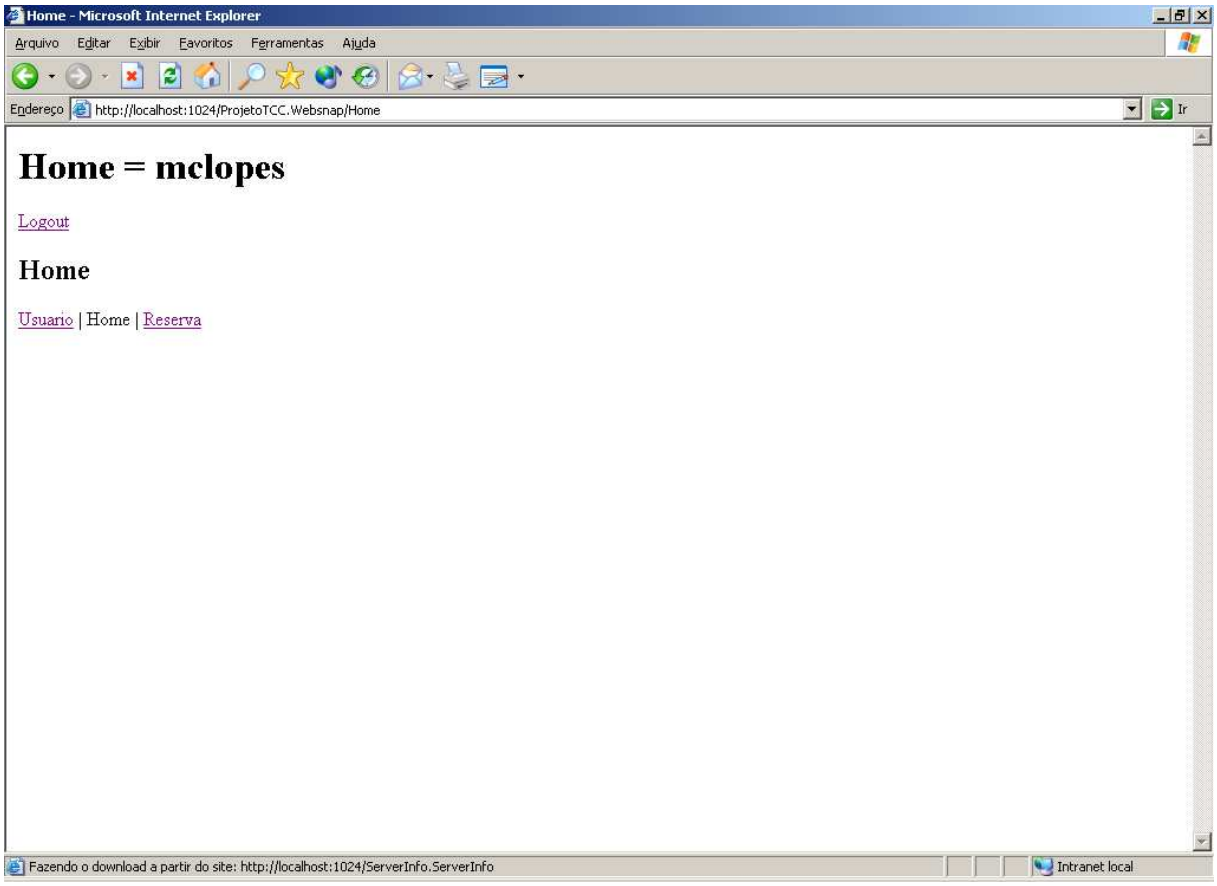
FIGURA 21 – TELA DE LABORATÓRIO DO ADMINISTRADOR



4.3.2.2 USUÁRIO PADRÃO

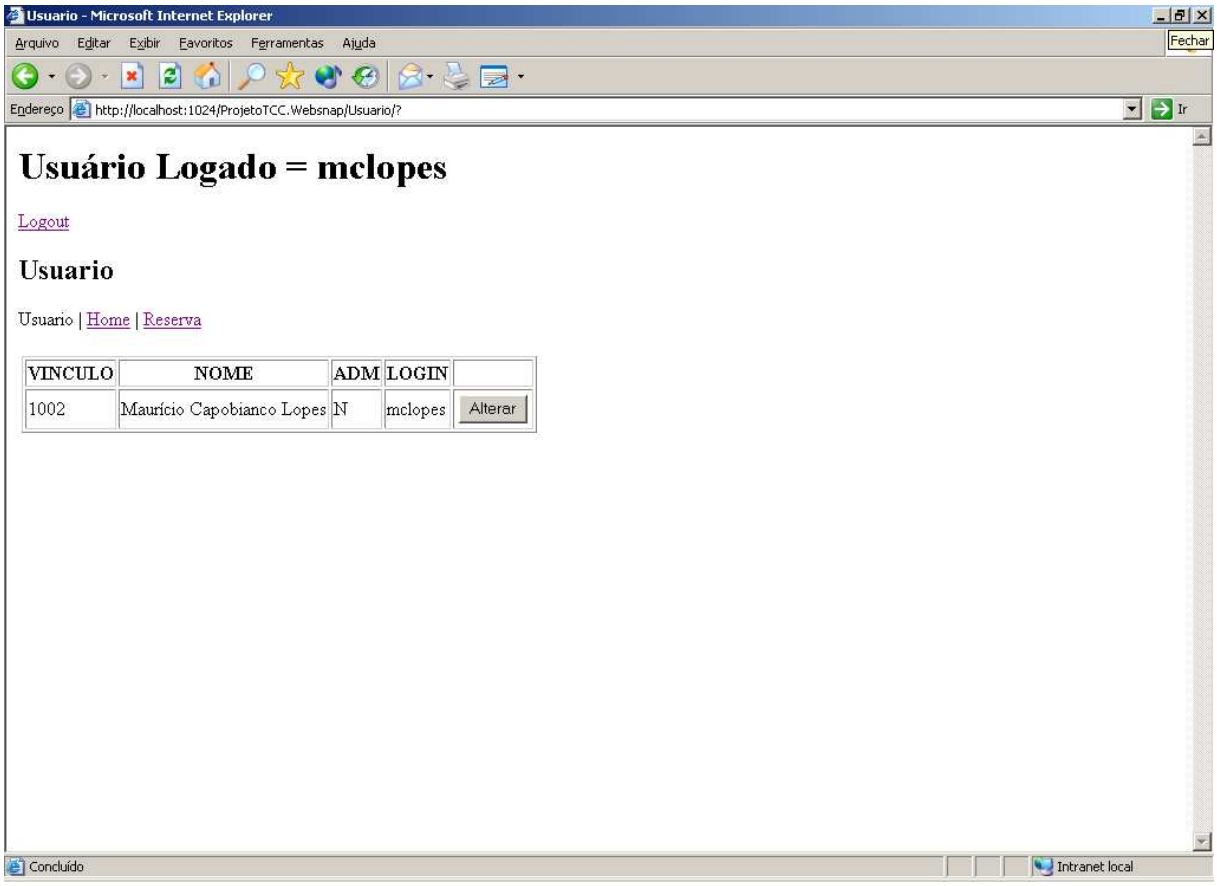
Esse usuário representa a figura da pessoa que faz suas reservas de laboratório. Ao logar-se na página principal ele terá acesso apenas às páginas de Usuário e Reserva, como pode ser observado na FIGURA 22.

FIGURA 22 – TELA HOME DO USUÁRIO PADRÃO



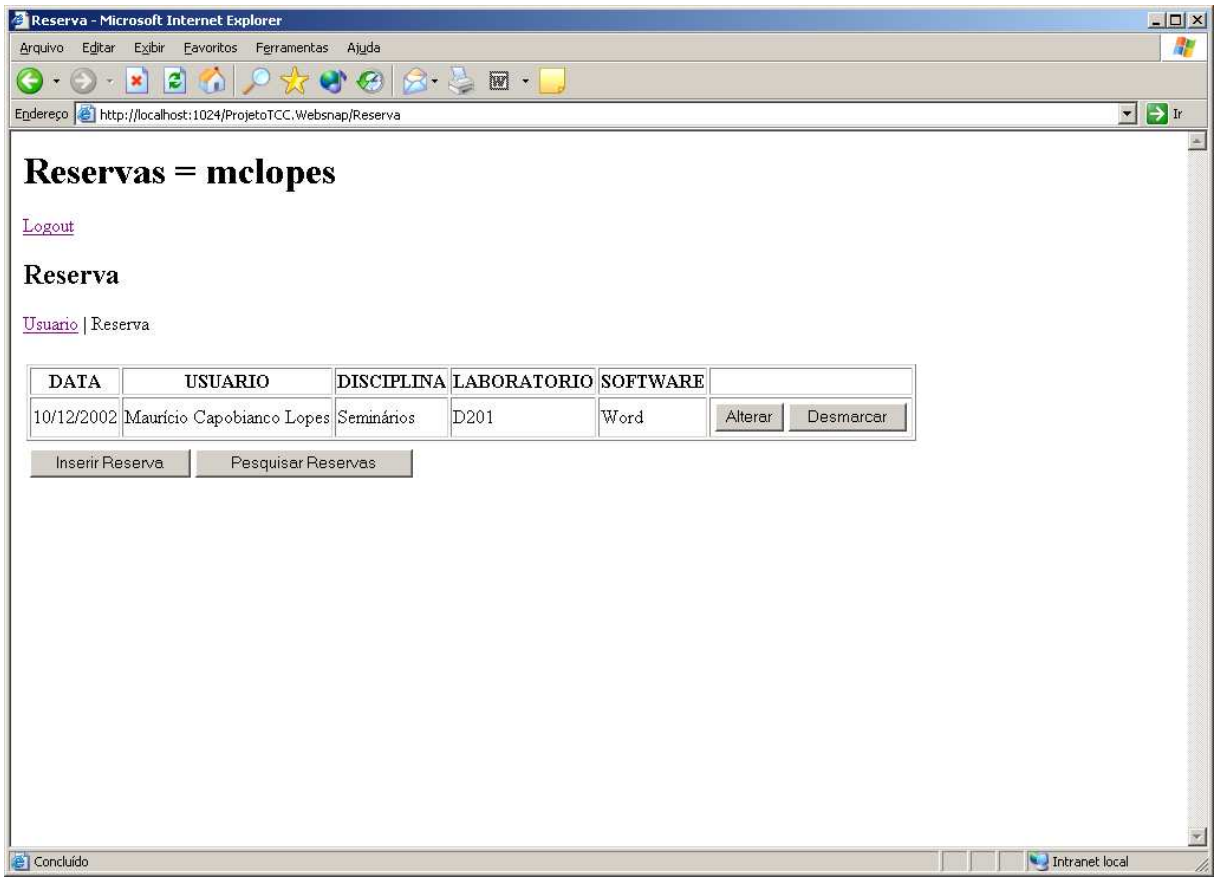
Acessando-se a página de Usuário pode-se observar mais uma limitação que esse tipo de login impõe. Apenas será mostrado o usuário que está logado e nessa tabela não será permitida a inclusão ou exclusão de registros, sendo ainda bloqueada a alteração do campo ADM que corresponde ao perfil do usuário (Administrador ou Usuário padrão). Na FIGURA 23 é mostrada a tela de Usuário do usuário padrão.

FIGURA 23 – TELA DE USUÁRIO DO USUÁRIO PADRÃO



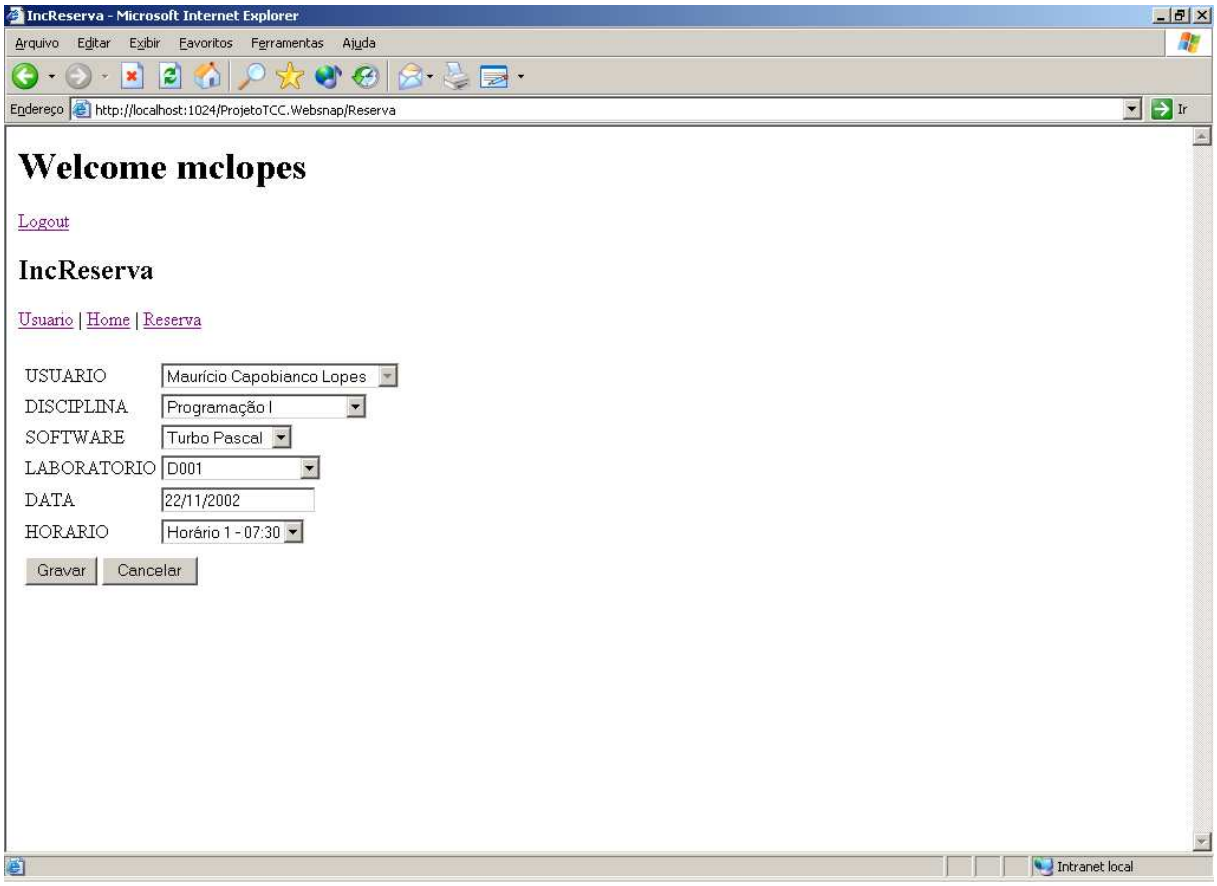
Quando logado com um usuário padrão é permitido consultar todas as reservas existentes, no entanto apenas pode-se alterar ou excluir reservas onde o usuário seja o mesmo que o usuário que está logado atualmente. Tal limitação pode ser vista na FIGURA 24 que mostra a tela de reservas do usuário padrão.

FIGURA 24 – TELA DE RESERVA DO USUÁRIO PADRÃO



Ao cadastrar uma nova reserva quando o usuário logado for do tipo usuário padrão, o mesmo já vem como responsável da reserva, bastando apenas selecionar a disciplina, software, laboratório, data e horário. Pode-se ver na FIGURA 24 tela de incluir reservas do usuário padrão.

FIGURA 25 – TELA DE INCLUIR RESERVA DO USUÁRIO PADRÃO



4.4 RESULTADOS E DISCUSSÃO

Neste item serão comentadas algumas características e dificuldades da tecnologia de desenvolvimento em questão neste trabalho.

Uma das primeiras dificuldades encontradas na fase do levantamento bibliográfico foi a escassez de material, pois a tecnologia é nova e há poucos documentos sobre o assunto. As principais bibliografias sobre desenvolvimento *Web* tratam de tecnologias como ASP, PHP, Dot.Net que têm sido mais utilizadas comercialmente.

Outra dificuldade foi o fato da tecnologia ser originária e proprietária da *Borland*, sendo que a maior parte do material encontrado era da própria empresa, devendo-se portanto tomar muito cuidado com a "propaganda" embutida em tais textos.

Quanto à implementação, foram encontradas pequenas dificuldades também oriundas da novidade, mas tudo contribuiu para o aprendizado e para as avaliações conclusivas.

No desenvolvimento do protótipo utilizou-se o aplicativo *Web App Debugger Executable* que é uma das grandes características referentes ao tipo de Servidor de aplicações WebSnap. O *Web App Debugger Executable* permite que se desenvolva uma aplicação *Web* sem precisar ter um servidor WWW instalado na máquina. Com ele, é gerado um executável que é um servidor COM que pode ser utilizado no desenvolvimento da aplicação, o que facilita muito na depuração do protótipo.

Este tipo de aplicação deve ser usado somente para fins de desenvolvimento, teste e depuração, mas nunca para distribuição. No momento da distribuição, deve-se migrar de uma aplicação do tipo *Web App Debugger Executable* para uma de produção (ISAPI/NSAPI, CGI, WIN-CGI ou módulos Apache).

Para testar a portabilidade do protótipo desenvolvido com *Web App Debugger Executable*, o mesmo foi migrado facilmente para um servidor WWW que suporte ISAPI.. Abaixo são descritos os passos para a migração entre os tipos de servidores suportados pelo WebSnap:

a) abrir o projeto original (desenvolvido para *Web App Debugger Executable*) no IDE do Delphi;
b) mostrar o *Project Manager* usando *View* | *Project Manager*;
c) abrir a "árvore" do projeto para que todas as *units* fiquem visíveis;
d) no *Project Manager*, clicar com o botão direito e criar um novo projeto WebSnap, selecionando o tipo de servidor apropriado. Nesse teste de portabilidade criou-se um projeto para ISAPI;
e) abrir a "árvore" do novo projeto para que sua *unit* seja mostrada, onde se deve remover a mesma;

f) feito esses passos, selecionar cada *unit* do projeto original (exceto o arquivo .DPR) e arrastar para o novo projeto. Neste caso, aparecerá um diálogo solicitando a confirmação para adicionar o arquivo ao novo projeto, devendo-se clicar em YES;
g) Executar o novo projeto.

A homologação da implantação do protótipo foi realizada num sistema operacional *Windows XP Professional* com IIS, onde se pôde constatar a facilidade de implantação. Os requisitos/passos para a implantação estão descritos a seguir:

a) necessidade de um servidor de banco de dados para suportar as estruturas existentes no protótipo. Foi utilizado o MSSQL Server 2000;
b) instalação do BDE;
c) configurar um *Aliase* denominado TCCx apontando para a base de dados definida;
d) necessidade de prévia instalação de um servidor WWW. Utilizou-se IIS que está presente na instalação do *Windows XP Professional*;
e) compartilhar para *Web* a pasta onde se encontra o protótipo;
f) para testar a funcionalidade da implantação, deve-se carregar o *Internet Explorer* e acessar o endereço compartilhado no item anterior.

Quanto à performance do protótipo pode-se dizer que a mesma foi boa, porém deve-se lembrar que não se teve a possibilidade de colocar o mesmo com vários usuários logados simultaneamente.

5 CONCLUSÕES

Com a situação atual no desenvolvimento de sistemas onde Internet é a palavra do momento, começou-se a observar que existem duas áreas bem distintas para o desenvolvimento de *sites*, que podem ser separadas em profissionais diferentes (programador e *webdesigner*). O WebSnap veio justamente ajudar nesse sentido, pois com ele fica bem dividida a parte de programação e a interação com o usuário. Além disto, o mesmo é extremamente ágil na construção de aplicações *web* com ligação a base de dados, como demonstrado no protótipo.

O principal objetivo do trabalho, que era desenvolver uma aplicação utilizando os recursos de internet do Delphi 6, mais especificamente o WebSnap, foi atingido plenamente com o desenvolvimento de um protótipo de reserva de laboratório de ensino *on-line*. O uso do WebSnap como tecnologia de desenvolvimento de páginas para a internet mostrou-se muito interessante para o aprimoramento das técnicas de desenvolvimento de aplicativos *Web*, uma vez que a tecnologia é realmente fácil de usar e o resultado obtido no protótipo satisfez as expectativas.

Em virtude das exigências do mercado, que estão cada vez mais se expandindo e necessitando de informações em qualquer hora e em qualquer lugar, os benefícios que o protótipo fornece são muito interessantes, pois nele destacam-se as facilidades da administração remota dos cadastros, a possibilidade de serem efetuadas reservas de qualquer lugar e um maior controle referente aos direitos de uso de cada software instado nos laboratórios.

O estudo da tecnologia WebSnap mostrou-se muito interessante para o aprimoramento das técnicas de desenvolvimento de aplicativos *Web*. Entretanto, aspectos como performance e segurança não puderam ser totalmente avaliados em função do sistema não ter sido utilizado em produção efetivamente. Apesar disto, com as evidências encontradas no trabalho conclui-se que se tem um conjunto que promete ser muito promissor nas aplicações como um todo. Embora ainda esteja mais na teoria do que na prática, é um paradigma que vem conquistando desenvolvedores e usuários, devido às vantagens que a tecnologia pode oferecer, seja em nível comercial ou em nível de aprendizagem.

5.1 EXTENSÕES

Devido ao WebSnap ser dividido em duas partes e nesse protótipo ser explorada principalmente à parte de desenvolvimento do lado do programador, uma sugestão seria desenvolver um aplicativo explorando a parte de *scripts* da tecnologia.

Sugere-se também que seja desenvolvido um aplicativo utilizando a tecnologia WebSnap junto com outras tecnologias de desenvolvimento *Web* encontradas no Delphi.

Quanto ao protótipo alguns aspectos podem ser melhorados, onde sugere-se um melhor desenvolvimento na parte de interface e um tratamento mais rigoroso na segurança do mesmo.

REFERÊNCIAS BIBLIOGRÁFICAS

CANTU, Marco. **Dominando o Delphi 6:** a bíblia; Tradução João Eduardo Nóbrega Tortello. São Paulo: Makron Books, 2002.

CUSUMANO, M.; YOFFIE, D. **Competing on internet time**: lessons from netscape and its battle with Microsoft. New York: Free Press, 1998.

FLEURY, André Leme. **O surgimento da internet**, Mestrando em Engenharia de Produção-UFSC, . Disponível em: <**http://igti.eps.ufsc.br/ce/surgimento.htm>.** Acesso em: 03 abr. 2002.

FURLAN, José Davi. **Modelagem de objetos através da UML**. São Paulo: Makon Books, 1998.

MATOS, Gladstone. WebSnap essencial, formulários, fields e producers. **Clube Delphi**, Rio de Janeiro, v. 26, p. 16-25, 2002.

PAULI, Guinther. WebSnap essencial, fundamentos, arquitetura e componentes. **Clube Delphi**, Rio de Janeiro, v. 24, p. 24-36, 2002.

ANEXO I

Neste anexo encontra-se o dicionário de dados do protótipo.

TCC_USUARIOS

Table Name	TCC_USUARIOS
Primary Keys	HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	int	N	Código seqüencial utilizado como chave primária
VINCULO	varchar(20)	N	Descrição do vínculo do usuário
NOME	varchar(40)	N	Descrição do nome do usuário
ADM	char(1)	N	Escolha do tipo de usuário (administrador / usuário padrão)
SENHA	varchar(10)	N	Descrição da senha do usuário
LOGIN	varchar(20)	N	Descrição do login do usuário

Foreign Keys

Parent Table	Child Table
TCC_USUARIOS	TCC_RESERVAS
TCC_USUARIOS	TCC_PROFESSOR_DISCIPLINAS

DDL Code

```
CREATE TABLE TCC_USUARIOS(
    HANDLE     int            NOT NULL,
    VINCULO    varchar(20)    NOT NULL,
    NOME       varchar(40)    NOT NULL,
    ADM        char(1)        NOT NULL,
    SENHA      varchar(10)    NOT NULL,
    LOGIN      varchar(20)    NOT NULL,
    CONSTRAINT PK1 PRIMARY KEY NONCLUSTERED (HANDLE)
)
```

TCC_DISCIPLINAS

Table Name TCC_DISCIPLINAS

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	Int	N	Código seqüencial utilizado como chave primária
NOME	varchar(40)	N	Descrição do nome da disciplina

Foreign Keys

Parent Table	Child Table
TCC_DISCIPLINAS	TCC_RESERVAS
TCC_DISCIPLINAS	TCC_PROFESSOR_DISCIPLINAS

DDL Code

```
CREATE TABLE TCC_DISCIPLINAS(
    HANDLE    int            NOT NULL,
    NOME      varchar(40)    NOT NULL,
    CONSTRAINT PK5 PRIMARY KEY NONCLUSTERED (HANDLE)
)
```

TCC_PROFESSOR_DISCIPLINAS

Table Name	TCC_PROFESSOR_DISCIPLINAS
Primary Keys	HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	int	N	Código seqüencial utilizado como chave primária
PROFESSOR	int	N	Código da tabela de usuários
DISCIPLINA	int	N	Código da tabela de disciplinas

Foreign Keys

Parent Table	Child Table
TCC_USUARIOS	TCC_PROFESSOR_DISCIPLINAS
TCC_DISCIPLINAS	TCC_PROFESSOR_DISCIPLINAS

DDL Code

```
CREATE TABLE TCC_PROFESSOR_DISCIPLINAS(
    HANDLE        int    NOT NULL,
    PROFESSOR     int    NOT NULL,
    DISCIPLINA    int    NOT NULL,
    CONSTRAINT PK19 PRIMARY KEY NONCLUSTERED (HANDLE),
    CONSTRAINT RefTCC_USUARIOS29 FOREIGN KEY (PROFESSOR)
    REFERENCES TCC_USUARIOS(HANDLE),
    CONSTRAINT RefTCC_DISCIPLINAS30 FOREIGN KEY (DISCIPLINA)
    REFERENCES TCC_DISCIPLINAS(HANDLE)
)
```

TCC_LABORATORIOS

Table Name TCC_LABORATORIOS

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	Int	N	Código seqüencial utilizado como chave primária
NOME	varchar(40)	N	Descrição do nome do laboratório
CAPACIDADE	Int	Y	Descrição da capacidade do laboratório

Foreign Keys

Parent Table	Child Table
TCC_LABORATORIOS	TCC_RESERVAS
TCC_LABORATORIOS	TCC_LABORATORIO_SOFTWARES

DDL Code

```
CREATE TABLE TCC_LABORATORIOS(
    HANDLE        int            NOT NULL,
    NOME          varchar(40)    NOT NULL,
    CAPACIDADE    int            NULL,
    CONSTRAINT PK8 PRIMARY KEY NONCLUSTERED (HANDLE)
)
```

TCC_SOFTWARE

Table Name TCC_SOFTWARE

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	Int	N	Código seqüencial utilizado como chave primária
NOME	varchar(40)	N	Descrição do nome do software

Foreign Keys

Parent Table	Child Table
TCC_SOFTWARE	TCC_RESERVAS
TCC_SOFTWARE	TCC_LABORATORIO_SOFTWARES

DDL Code

```
CREATE TABLE TCC_SOFTWARE(
    HANDLE    int            NOT NULL,
    NOME      varchar(40)    NOT NULL,
    CONSTRAINT PK6 PRIMARY KEY NONCLUSTERED (HANDLE)
)
```

TCC_LABORATORIO_SOFTWARES

Table Name TCC_LABORATORIO_SOFTWARES

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	Int	N	Código seqüencial utilizado como chave primária
LABORATORIO	Int	N	Código da tabela de laboratórios
SOFTWARE	Int	N	Código da tabela de sofwares

Foreign Keys

Parent Table	Child Table
TCC_LABORATORIOS	TCC_LABORATORIO_SOFTWARES
TCC_SOFTWARE	TCC_LABORATORIO_SOFTWARES

DDL Code

```
CREATE TABLE TCC_LABORATORIO_SOFTWARES(
    HANDLE         int    NOT NULL,
    LABORATORIO    int    NOT NULL,
    SOFTWARE       int    NOT NULL,
    CONSTRAINT PK21 PRIMARY KEY NONCLUSTERED (HANDLE),
    CONSTRAINT RefTCC_LABORATORIOS31 FOREIGN KEY (LABORATORIO)
    REFERENCES TCC_LABORATORIOS(HANDLE),
    CONSTRAINT RefTCC_SOFTWARE32 FOREIGN KEY (SOFTWARE)
    REFERENCES TCC_SOFTWARE(HANDLE)
)
```

TCC_HORARIOS

Table Name TCC_HORARIOS

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	Int	N	Código seqüencial utilizado como chave primária
NOME	varchar(20)	Y	Descrição da grade de horário

Foreign Keys

Parent Table	Child Table
TCC_HORARIOS	TCC_RESERVA_HORARIOS

DDL Code

```
CREATE TABLE TCC_HORARIOS(
    HANDLE    int            NOT NULL,
    NOME      varchar(20)    NULL,
    CONSTRAINT PK3 PRIMARY KEY NONCLUSTERED (HANDLE)
)
```

TCC_RESERVA_HORARIOS

Table Name TCC_RESERVA_HORARIOS

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	Int	N	Código seqüencial utilizado como chave primária
RESERVA	Int	N	Código da tabela de reservas
HORARIO	int	N	Código da tabela de horários

Foreign Keys

Parent Table	Child Table
TCC_HORARIOS	TCC_RESERVA_HORARIOS
TCC_RESERVAS	TCC_RESERVA_HORARIOS

DDL Code

```
CREATE TABLE TCC_RESERVA_HORARIOS(
    HANDLE     int    NOT NULL,
    RESERVA    int    NOT NULL,
    HORARIO    int    NOT NULL,
    CONSTRAINT PKTCC_RESERVA_HORARIOS PRIMARY KEY NONCLUSTERED
    (HANDLE),
    CONSTRAINT RefTCC_HORARIOS35 FOREIGN KEY (HORARIO)
    REFERENCES TCC_HORARIOS(HANDLE),
    CONSTRAINT RefTCC_RESERVAS41 FOREIGN KEY (RESERVA)
    REFERENCES TCC_RESERVAS(HANDLE)
)
```

TCC_RESERVAS

Table Name TCC_RESERVAS

Primary Keys HANDLE

Columns

Column Name	Datatype	Null	Definition
HANDLE	int	N	Código seqüencial utilizado como chave primária
DATA	datetime	N	Descrição da data da reserva
USUARIO	int	N	Código da tabela de usuários
DISCIPLINA	int	N	Código da tabela de disciplinas
SOFTWARE	int	N	Código da tabela de softwares
LABORATORIO	int	N	Código da tabela de laboratórios

Foreign Keys

Parent Table	Child Table
TCC_USUARIOS	TCC_RESERVAS
TCC_DISCIPLINAS	TCC_RESERVAS
TCC_SOFTWARE	TCC_RESERVAS
TCC_LABORATORIOS	TCC_RESERVAS
TCC_RESERVAS	TCC_RESERVA_HORARIOS

DDL Code

```
CREATE TABLE TCC_RESERVAS(
    HANDLE          int          NOT NULL,
    DATA            datetime     NOT NULL,
    USUARIO         int          NOT NULL,
    DISCIPLINA      int          NOT NULL,
    SOFTWARE        int          NOT NULL,
    LABORATORIO     int          NOT NULL,
    CONSTRAINT PK9 PRIMARY KEY NONCLUSTERED (HANDLE),
        CONSTRAINT RefTCC_USUARIOS25 FOREIGN KEY (USUARIO)
    REFERENCES TCC_USUARIOS(HANDLE),
    CONSTRAINT RefTCC_DISCIPLINAS26 FOREIGN KEY (DISCIPLINA)
    REFERENCES TCC_DISCIPLINAS(HANDLE),
    CONSTRAINT RefTCC_SOFTWARE27 FOREIGN KEY (SOFTWARE)
    REFERENCES TCC_SOFTWARE(HANDLE),
    CONSTRAINT RefTCC_LABORATORIOS28 FOREIGN KEY (LABORATORIO)
    REFERENCES TCC_LABORATORIOS(HANDLE)
)
```