UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UMA APLICAÇÃO PARA CONSULTAS ACADÊMICAS UTILIZANDO SERVLETS

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

EMERSON DE PINHO ADAM

BLUMENAU, NOVEMBRO/2001

2001/2-19

PROTÓTIPO DE UMA APLICAÇÃO PARA CONSULTAS ACADÊMICAS UTILIZANDO SERVLETS

EMERSON DE PINHO ADAM

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. Marcel Hugo

_____

Prof. Everaldo Artur Grahl

Aos simples e humildes.

AGRADECIMENTOS

Ao Pai de toda sabedoria, o Deus da bíblia que usou de longanimidade, e ajudou-me nos momentos de angústia, desespero e ansiedade. Senhor obrigado! – A Ele toda a honra, a glória e o louvor.

Aos meus colegas do dia a dia, por sua compreensão e extrema tolerância, mesmo quando eu me mostrava insuportável.

Ao papai e mamãe, agora eles podem pendurar o canudo na parede da sala.

Ao Núcleo de Informática pelo incentivo e apoio dispensado.

Ao meu amigo Maurício pela força, muito obrigado. Saudades da terrinha!

A todos que direta ou indiretamente me ajudaram neste trabalho.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	– *Application Program Interface*
ARPAnet	– *Advanced Research Projects Agency Network of the Department of Defense*
CERN	– *European Particle Physics Laboratory*
DBA	– *Data Base Administrator*
ETEVI	– *Escola Técnica do Vale do Itajaí*
FTP	– *File Transfer Protocol*
FURB	– *Fundação Universidade Regional de Blumenau*
HTML	– *Hypertext Markup Language*
HTTP	– *Hypertext Transport Protocol*
J2EE	– *Java 2 Enterprise Edition*
JBDC	– *Java DataBase Connectivity*
JDK	– *Java Development Kit*
JVM	– *Java Virtual Machine*
PL/SQL	– *Procedural Language*/*Structured Query Language*
SQL	– *Structured Query Language*
UML	– *Unified Modeling Language*
VAJ	– *Visual Age for Java*
WWW	– *World Wide Web*

RESUMO

Este trabalho tem por objetivo estudar e aplicar a tecnologia de *servlets* a um protótipo de consultas acadêmicas pela *internet*. Foram utilizadas neste trabalho as ferramentas *Visual Age for Java* para implementar os *servlets*, um servidor *Web Apache* integrado com o servidor de *servlets TomCat* acessando um banco de dados *Oracle*; na especificação foi utilizado o *Rational Rose* para modelar conforme a notação *UML*. Os resultados obtidos foram satisfatórios, principalmente no que diz respeito ao desempenho dos *servlets* em relação a aplicação atual que funciona em *PL/SQL*.

ABSTRACT

This work aims to study and to apply the servlets technology to a prototype of academic consultations for the internet. The tools used in this work are Visual Age for Java to implement servlets, Apache that is a Web server integrated with servlets server and TomCat that supports access a Oracle database. For specification was used UML notation in tool CASE Rational Rose. To results were satisfactories, mainly at servlets performance in comparison to PL/SQL runtimes.

1 INTRODUÇÃO

A Universidade Regional de Blumenau possui um sistema acadêmico de graduação que é responsável por todos os processos acadêmicos, tais como manutenção de alunos, professores, notas, boletins, currículos, planos de ensino, matrículas, deferimentos, enfim todos os processos pertinentes ao sistema acadêmico de graduação. Os usuários deste sistema são vários e distribuídos pelos diversos *campi*. Pode-se citar como exemplo destes usuários a divisão acadêmica, os alunos da universidade, os professores coordenadores de cursos e a Pró-Reitoria de Ensino.

Um componente importante do sistema acadêmico de graduação são as consultas acadêmicas, disponibilizadas na *Internet* e no quiosque eletrônico e que oferecem aos seus usuários consultas de notas, consultas financeiras, consultas de horários e consultas de turmas. Entretanto, a aplicação utilizada atualmente foi desenvolvida em *PL/SQL* e somente pode ser executada no servidor *Web* cujo fabricante seja o mesmo do banco de dados ou que o servidor possua *drivers* de integração com o banco de dados. Essa característica não é desejável, uma vez que torna a aplicação cara e dependente do fabricante do servidor *Web*, o que acarreta em aumento de custos pois são necessários licenças de uso para os *drivers* de integração e licença de uso para servidor *Web*.

Deste modo, neste trabalho propõe-se migrar o sistema atual para outra tecnologia de desenvolvimento, mantendo sua característica de operar em um ambiente *Web* e buscando aumentar sua performance.

Ao falar de sistemas para *Web* não se pode deixar de mencionar a linguagem de programação *Java*, que segundo Niemeyer (2000) foi projetada para ser uma linguagem de programação independente de máquina, que seja segura e ofereça todo o suporte necessário para o desenvolvimento de aplicações para internet. Este planejamento fez com que a linguagem *Java* se transformasse rapidamente numa importante plataforma para as aplicações no lado do servidor, pois possui diversas *Application Programming Interface* (*API*) que oferecem suporte a programação distribuída.

A linguagem *Java* vem ao encontro dos anseios de uma aplicação moderna, que seja, compacta, orientada a objetos e com suporte a concorrência. Um aspecto importante de *Java*,

é sua portabilidade e por ser multiplataforma um programa escrito nesta linguagem pode ser executado em qualquer hardware e sistema operacional sem a necessidade de alterações no código-fonte. Tal funcionalidade é possível através do processo de compilação que gera, a partir de um código-fonte, um código intermediário chamado *bytecode* que é interpretado em qualquer plataforma com suporte à *Java*. Devido ao vasto conjunto de recursos para programação distribuída, a linguagem *Java* tem sido bastante difundida na implementação e desenvolvimento de aplicações que exijam recursos de rede.

A execução das aplicações feitas em *Java* em um servidor *Web*, que interage com páginas em *HTML*, recebe o nome de *servlets*. Esta é a tecnologia proposta para a construção do protótipo de consultas acadêmicas.

Segundo Hall (2000) em função dos *servlets* serem escritos em *Java* o código da aplicação é portável e roda em qualquer servidor *Web* com suporte a *servlets*. Todo o processamento das informações é distribuído no servidor através dos vários objetos de classes que compõem a aplicação, restando ao navegador do usuário as diversas interações em código *HTML* que o *servlet* gerar, não havendo a necessidade portanto, de que seja instalado no cliente, qualquer tipo de *driver* ou *plug-in* ou ainda interpretadores de *Java* como a *Java Virtual Machine* (*JVM*).

Assim, a proposta deste trabalho é criar um protótipo de aplicação para consultas acadêmicas utilizando *servlets*, onde a troca de informações é feita em *HTML*. Desta forma, o usuário que possuir um navegador conectado à *Internet*, acessando um servidor *Web*, terá, à sua disposição, a aplicação no formato de páginas *Web*.

De um modo geral sistemas para *Web* são distribuídos, utilizando-se da arquitetura de multicamadas, que permite uma melhor distribuição do processamento em uma aplicação. Para o desenvolvimento do sistema de consultas acadêmicas será adotada a arquitetura de 3 (três) camadas, que compreende o navegador *Web*, o servidor *Web* com *servlets* e o servidor de banco de dados (Reese, 2000).

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma aplicação para consultas acadêmicas utilizando *servlets*.

Os objetivos secundários do trabalho são:

a) atender aos requisitos do sistema através de uma aplicação distribuída em um ambiente *Internet*, em conformidade com a situação atual onde os usuários do sistema também estão distribuídos geograficamente pelos *campi*.
b) abordar tecnologias abertas e emergentes em plataformas para o desenvolvimento de aplicações que executam no servidor, como *servlets*;
c) fornecer uma aplicação interativa, que não exija que o cliente tenha que instalar *plug-ins*, *drivers* ou qualquer máquina virtual *Java*.

1.2 ESTRUTURA

O presente 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 um breve histórico do desenvolvimento da plataforma *Web*, suas tecnologias e arquiteturas para o desenvolvimento de aplicativos, focando a proposta da utilização de uma arquitetura de 3 (três) camadas.

O terceiro capítulo será sobre *servlets* que é a tecnologia proposta para o estudo e desenvolvimento dos módulos protótipos de consultas acadêmicas. Contempla suas características e inovações, principalmente no que tange à implementação utilizando *Java* no servidor *Web*.

O quarto capítulo descreve a metodologia e especificação do protótipo, bem como detalhes de sua implementação em *Visual Age for Java*.

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

2 DESENVOLVIMENTO WEB

Neste capítulo serão abordados os conceitos para o desenvolvimento na *Web*, sendo dividido nos seguintes tópicos: introdução, arquiteturas para o desenvolvimento de aplicações para *Web*, *middleware*, *JDBC* e desenvolvimento de aplicações distribuídas em *Java*.

2.1 INTRODUÇÃO

A *Internet* começou no final dos anos 60 como uma experiência no projeto de redes de computador robustas. O objetivo era construir uma rede de computadores que poderia suportar a perda de diversas máquinas sem comprometer a habilidade das restantes de se comunicar. O financiamento veio do departamento de defesa dos Estados Unidos, que tinha interesse nas redes de informação que poderiam suportar um ataque nuclear. A rede resultante deste projeto, a *ARPAnet* (*Advanced Research Projects Agency Network of the Department of Defense*) se desenvolveu até o início dos anos 90 apenas no escopo acadêmico e militar. Com o fim da guerra fria e devido à vontade das universidades participantes em disponibilizar e popularizar o uso da *Internet*, o governo americano, que então financiava e controlava a rede, abriu-a para trafego irrestrito (Lucena, 2000).

A *World Wide Web* (*WWW*), que hoje é o coração da *Internet*, nasceu quase junto com a abertura da Internet em 1990 quando físicos pesquisadores do laboratório *CERN* (*European Particle Physics Laboratory*) tiveram a necessidade de distribuir documentos, gráficos, fotos e sons de maneira unificada e eficiente via *Internet* e necessitavam de algo mais sofisticado e inteligente que simplesmente transferência de arquivos via *FTP*. Assim nasceu a *Hypertext Markup Language* (*HTML*), a *WWW* e os primeiros navegadores *Web* (Lucena, 2000).

A *HTML* é uma linguagem de marcação muito simples limitada apenas à formatação de documentos. A característica mais importante da *HTML* é a possibilidade de criar ligações entre os documentos. Isto permitiu a criação de alguma organização aos documentos que até então se distribuíam de forma caótica. Contribuiu para a popularização da *HTML* a criação do primeiro navegador *Web* gráfico o *Mosaic*. Este navegador *Web* foi desenvolvido por um grupo de alunos e professores da Universidade de Illinois, Urbana-Champaign (Lucena, 2000).

Em 1994, baseado no sucesso do *Mosaic*, Marc Andressen, um dos alunos envolvidos no projeto do *Mosaic* fundou a *Netscape*, dando início à primeira geração de navegadores *Web* comerciais que trouxeram junto consigo a explosão da *World Wide Web*. Tão logo os documentos em *HTML* se tornaram populares em todo o mundo, começou a cogitar-se a possibilidade de expandir a sua utilidade tornando-os mais interativos. Em 1996, a versão 2.0 da *HTML* introduziu os *forms* que provêem um ambiente gráfico com os componentes de interação como *combobox*, *listbox*, *textbox*, *buttons*, etc que são exibidas dentro dos navegadores *Web*. A criação de um ambiente gráfico com recursos semelhantes aos disponíveis nos sistemas operacionais baseados em ambientes de janelas deu início ao desenvolvimento das primeiras aplicações para *World Wide Web* (Lucena, 2000).

2.2 ARQUITETURAS PARA DESENVOLVIMENTO DE APLICAÇÕES PARA WEB

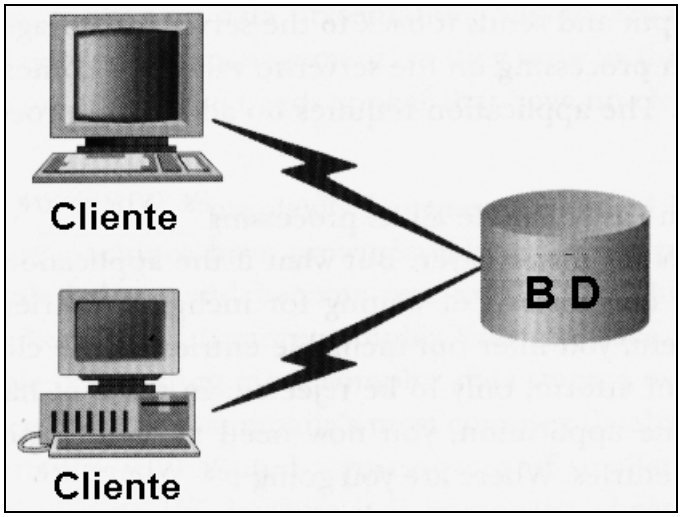
Há atualmente dois paradigmas básicos de arquiteturas para desenvolvimento de aplicações para *Web*, o primeiro paradigma é do tipo cliente/servidor tradicional em arquitetura de duas camadas e o segundo paradigma é conhecido como arquitetura de três camadas, também comumente chamada de arquitetura multicamadas (Reese, 2000).

2.2.1 ARQUITETURA DE DUAS CAMADAS

Uma simples arquitetura cliente/servidor tradicional é chamada de arquitetura de duas camadas. O termo “duas camadas” descreve a maneira como o processamento da aplicação pode ser dividido em uma parte cliente e outra servidora. Aplicações em duas camadas, providenciam, para os vários clientes, uma camada de apresentação uniforme que se comunica com uma camada centralizada de dados. A camada de apresentação geralmente é chamada de cliente, e a camada de armazenamento de dados é chamada de servidor. Pode-se citar como exemplo, uma série de aplicações simples de *Internet* tais como, e-mail, telnet, ftp e gopher (Reese, 2000).

A figura 1 apresenta um sistema de duas camadas, onde um cliente faz acesso a dados centralizados. Num sistema que utiliza a *Web*, por exemplo, o navegador *Web* no lado do cliente seria responsável por recuperar os dados armazenados em um servidor *Web*.

Figura 1 - Arquitetura cliente/servidor de duas camadas

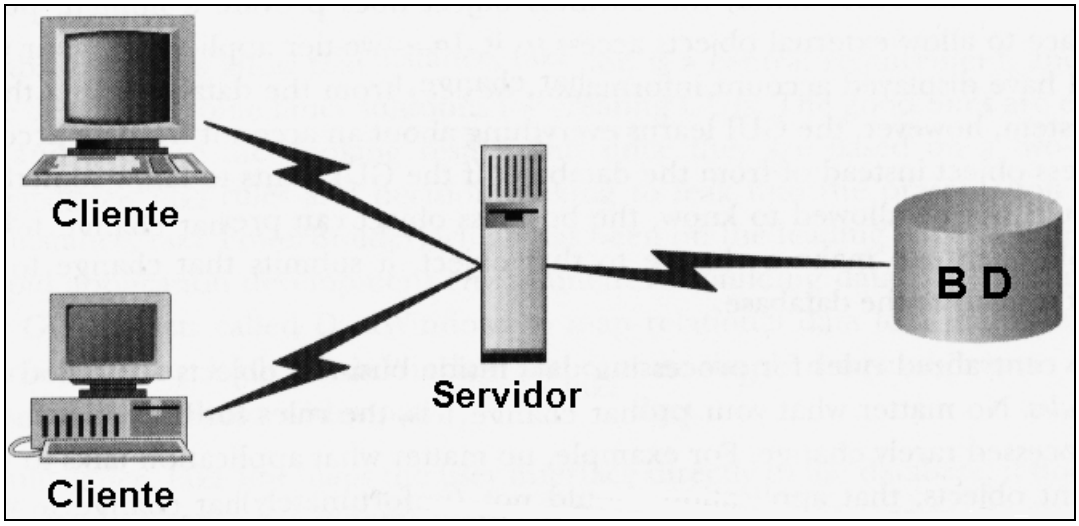


2.2.2 ARQUITETURA DE TRÊS CAMADAS

Com o aumento da demanda de aplicações que precisam ser acessadas por usuários que se encontram fora das instalações de uma empresa, velhas arquiteturas que implementam o tradicional modelo cliente/servidor (figura 1), começam a ficar limitadas quanto a atender um número maior de usuários em diversas plataformas e localidades. Uma aplicação tradicional cliente/servidor requer um administrador responsável pela instalação do programa cliente no *desktop* dos usuários. Ele se esforça muito para assegurar que cada *desktop* forneça um ambiente operacional semelhante, para que a aplicação possa funcionar conforme ela foi projetada. Quando uma mudança ocorre na aplicação, o administrador precisa instalar a atualização em todos os equipamentos.

A arquitetura cliente/servidor de três camadas é caracterizada pela utilização de uma camada chamada servidor de aplicações e é dividida da seguinte maneira: a camada cliente, a camada do servidor de aplicações, e a camada do servidor de banco de dados (ver figura 2). No servidor de aplicações é representada a lógica da aplicação, o código programado que faz acesso ao servidor de banco de dados e outros sistemas, entre a camada cliente e a camada servidora. Normalmente a lógica da aplicação (seus algoritmos, acesso a dados, etc.) são separados da apresentação para o cliente, que pode ser, por exemplo, uma página em *HTML* para apresentação e entrada de dados (Lucena, 2000).

Figura 2 - Arquitetura cliente/servidor de 3 (três) camadas



Segundo Luppi (1999) na arquitetura cliente/servidor em três camadas, todas as funções lógicas são implementadas em módulos. Os módulos são altamente portáveis e podem ser reusados, sendo facilmente integráveis, sem a necessidade de alterações nos componentes de apresentação e de dados. Como resultado, vários clientes podem usar os serviços oferecidos pela camada intermediária e, portanto, um servidor único é utilizado para as aplicações.

Com a introdução dos programas *middleware*, o modelo cliente/servidor de três camadas pode aumentar a interoperabilidade de dados residentes em diferentes bancos de dados ou sistemas operacionais. Quando uma aplicação em um cliente faz uma requisição de dados distribuídos em diferentes bancos de dados, a camada intermediária assume o papel de gerenciamento de seleção e integração dos dados. O fato de que cada camada pode ser desenvolvida independentemente, também implica em que cada uma das camadas trabalha independente da outra e pode funcionar de maneira cooperativa (Luppi, 1999).

Bonifácio (2000) entende que a arquitetura de três camadas cliente/servidor é um modelo interessante no que diz respeito a separação formal e explícita de uma aplicação em três camadas independentes e destaca as seguintes vantagens:

a) independência de código: é muito mais fácil trocar uma camada inteira quando

necessário. Uma vez que se tenham todas as interfaces entre as camadas bem definidas, trocar a camada de lógica do banco de dados é muito simples, não sendo necessário mexer no restante da aplicação;

b) independência de desenvolvimento e reutilização de código: a equipe pode se concentrar em apenas desenvolver a parte complicada da aplicação, que é a lógica de negócio, sem se preocupar em criar formulários e códigos *HTML* com *scripts* para validá-los. O administrador do banco de dados (*DBA*) pode cuidar do banco de dados de forma mais isolada e os programadores *HTML* (*webdesigners*) não precisam ter muito conhecimento de sistemas para desempenharem a sua parte, que é a interface com o usuário;

c) facilidade de gerenciamento de projeto: uma vez que se tem a aplicação bem projetada, com todas as interfaces definidas, é muito mais simples gerenciar grandes times de desenvolvimento trabalhando cada qual em um ponto da aplicação independentemente.

2.3 MIDDLEWARE

Conceitualmente define-se *middleware* como sendo qualquer tipo de software que controle e gerencie o fluxo de informação entre clientes e servidores. De acordo com o *Butler Group*, o objetivo do *middleware* é mapear as aplicações aos recursos como bancos de dados, *e-mail*, impressoras, entre outros. Através desta função o *middleware* protege as aplicações em seus ambientes de implementação, oferecendo subsídios para preservação dos investimentos feitos com as aplicações. Dentre os vários tipos de *middleware* existentes, pode-se destacar que a maioria deles oferece uma entre duas funções, serviços de acesso à bancos de dados ou serviços de sistemas distribuídos (Luppi, 1999).

Um aspecto relevante a respeito da funcionalidade do *middleware* são os serviços de acesso a bancos de dados, que neste trabalho é realizado através da biblioteca de classes *Java DataBase Connectivity* (JBDC).

2.4 JBDC

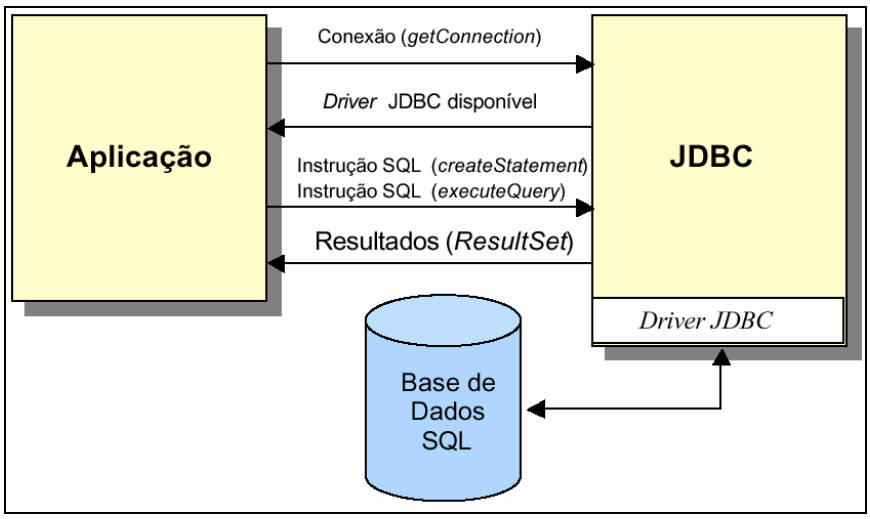
Em meados de 1995, a *SUN Microsystems*, tão logo foi feito o lançamento das primeiras versões da linguagem *Java*, passou a investigar o potencial da linguagem para o

acesso a bases de dados. Neste ponto, ao invés do desenvolvimento de um gerenciador de bases de dados baseado em *Java*, optou-se pelo desenvolvimento de uma biblioteca de programação que permitisse o acesso a bases de dados através da linguagem *SQL (Structured Query Language*), já bastante difundida na época. O resultado foi uma biblioteca, denominada *JDBC (Java DataBase Connectivity*), que se tornou um dos pontos de maior investimento da *SUN Microsystems* (Silva, 2000).

A biblioteca *JDBC* é uma *API* que permite aos programadores o desenvolvimento de programas pouco complexos, em termos da codificação Java, para executar instruções *SQL* em ambientes distribuídos. A combinação *Java - JDBC* permite o desenvolvimento de aplicações multiplataforma, visto que as bases de dados podem estar distribuídas em servidores remotos que utilizam diferentes ambientes operacionais. As versões mais novas da linguagem *Java* trazem esta biblioteca embutida no conjunto de bibliotecas da linguagem, não sendo mais necessária a sua instalação independente (Silva, 2000).

De modo geral, a arquitetura de software para uso do *JDBC* é simples. Inicialmente, uma aplicação faz um pedido ao *JDBC* para que seja criada uma conexão, via um *driver JDBC*, a uma base de dados. A partir do estabelecimento da conexão, um serviço de passagem de mensagens permite à aplicação o envio de *queries* (instruções *SQL*) ao banco de dados, que por sua vez executa os pedidos da aplicação e promove o envio dos resultados. A figura 3 ilustra a arquitetura de software utilizada para comunicação via *JDBC* (Silva, 2000).

Figura 3 - Arquitetura do ambiente de passagem de mensagem do JDBC



A classe *JDBC* fornece uma *API* padrão completa para desenvolvimento de ferramentas/banco de dados. Com o *JDBC*, não é necessário escrever vários programas para acessar gerenciadores de banco de dados de diferentes fabricantes. Um simples programa usando a *API JDBC*, será capaz de transmitir declarações para diferentes gerenciadores de banco de dados. Além disso, com o desenvolvimento de uma aplicação utilizando a linguagem de programação *Java*, não há necessidade de se preocupar com a execução desta aplicação em diferentes plataformas.

2.5 DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS EM JAVA

O desenvolvimento de aplicações distribuídas requer a utilização de linguagens de programação que forneçam bibliotecas de suporte à comunicação entre plataformas heterogêneas de hardware e software e que forneçam recursos de programação que isolem os programadores das complexidades dos aspectos de comunicação, tais como padrões de conexão, detalhes de protocolos, conversões entre formatos de dados e outros.

Java oferece, através do *J2EE* (*Java 2 Enterprise Edition*), uma especificação completa para desenvolvimento de *servlets* e aplicações distribuídas. *Servlets* comunicam-se com objetos distribuídos através de *EJB* (*Enterprise JavaBeans*) que é uma arquitetura de componentes projetada para distribuir objetos (Hunter, 2001).

No próximo capítulo serão discutidos alguns dos principais recursos e técnicas de programação da linguagem *Java* utilizando *servlets*.

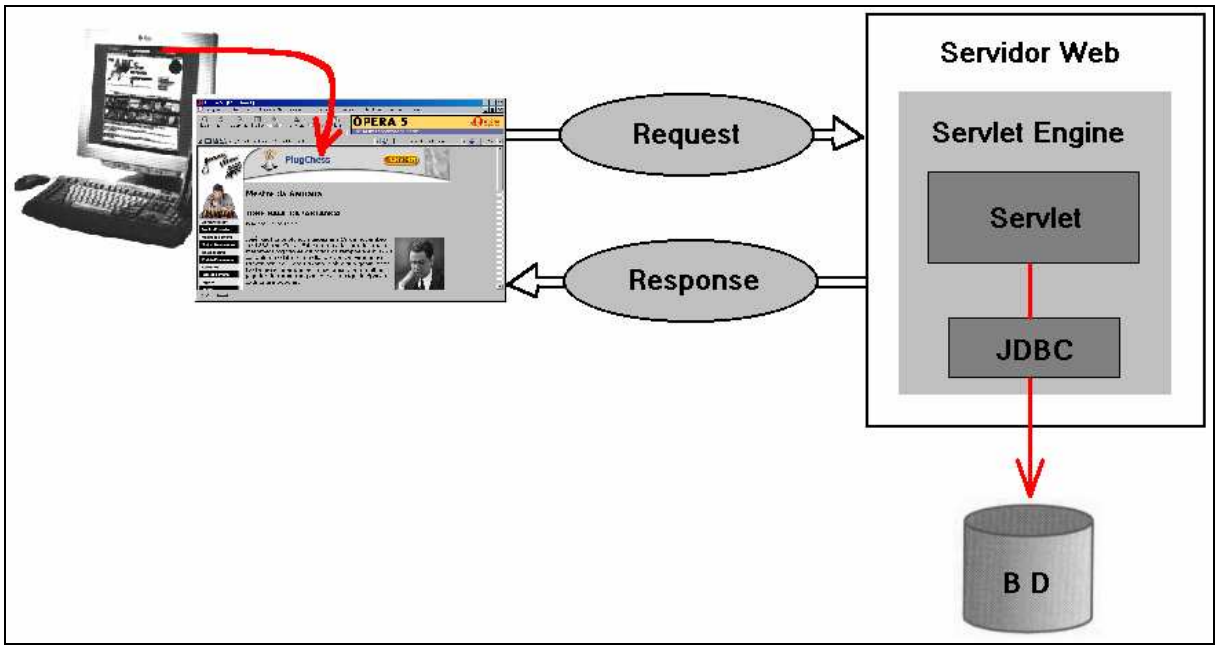
3 SERVLETS

Neste capítulo será abordada a tecnologia proposta para o desenvolvimento do trabalho; sendo dividido nos seguintes tópicos: introdução, arquitetura, a *API Java servlet*, funcionalidades, o ciclo de vida de um *servlet*, sessões e pool de conexões.

3.1 INTRODUÇÃO

Servlets são módulos baseados em solicitações/respostas que executam no lado do servidor, classificando-se como sendo uma *API* para servidores *Web* estendida. O *servlet* é responsável por pegar os dados pela ordem de entrada em uma página *HTML* e aplicá-los em uma lógica de negociação usada em um banco de dados de uma companhia. A figura 4 mostra o fluxo de informação em um *servlet* (Cornell, 1997).

Figura 4 - Esquema de solicitações/respostas



No contexto da arquitetura cliente/servidor da *Internet*, deve-se observar que o lado do cliente (*client-side*) é caracterizado pelo uso de uma interface universal de navegação *(browser*) através da qual se tem acesso aos recursos providos por um servidor. Algumas tarefas podem ser executadas na máquina do cliente. Por outro lado, existem tarefas que devem ser processadas pela máquina servidora *(server-side*) e os resultados enviados ao

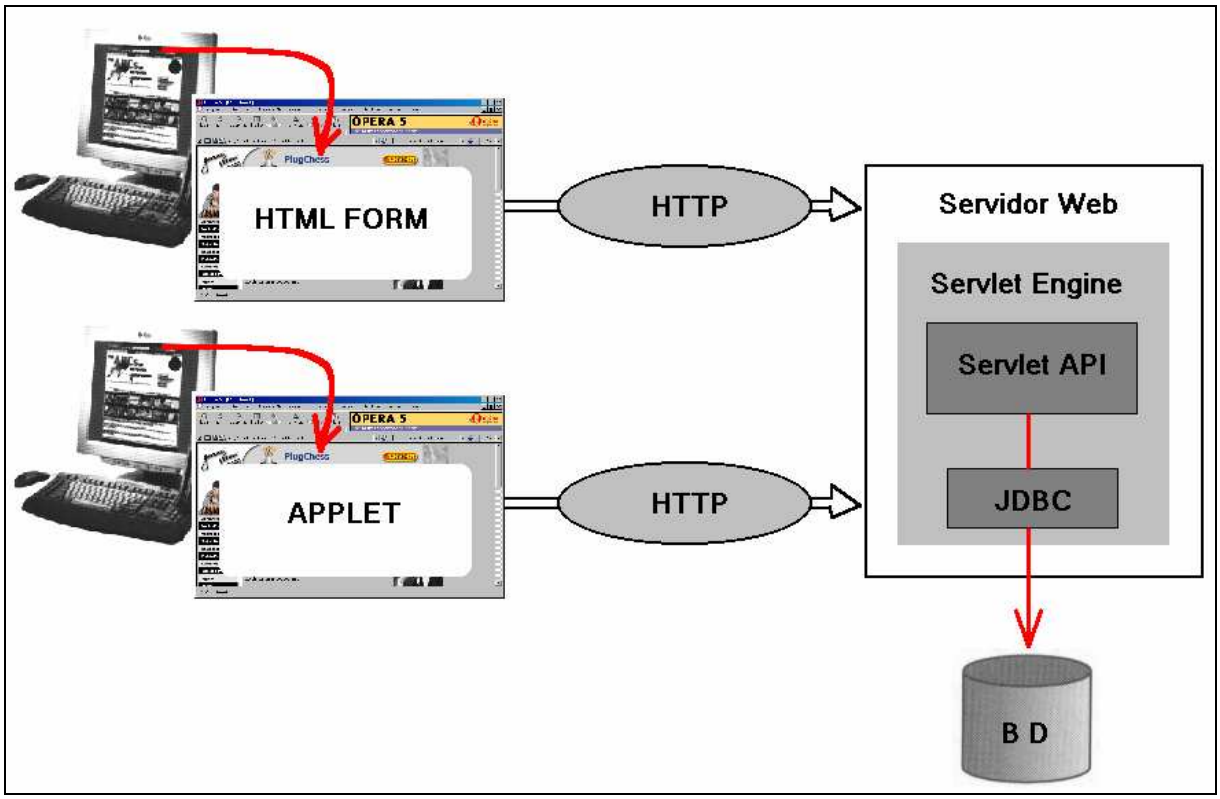
cliente. Os *servlets* estendem as capacidades de um servidor, permitindo o processamento otimizado de tarefas na máquina servidora (Colla, 1999).

A biblioteca *API Java servlet*, usada para escrever os códigos fontes do *servlets*, não assume nada sobre como o *servlet* é carregado, ou o ambiente no qual o *servlet* é executado, ou o protocolo usado para transmitir dados para o usuário. Isto permite que os *servlets* sejam embutidos em diferentes servidores *Web* (Cornell, 1997).

3.2 ARQUITETURA

A figura 5 apresenta uma arquitetura onde uma *API Java servlet* fornece mecanismos que gerenciam a comunicação com o cliente e o servidor. Quando um *servlet* aceita uma chamada de um cliente, ele recebe dois objetos: um objeto de solicitação que encapsula a comunicação entre o cliente e o servidor e um objeto de resposta que encapsula a comunicação do servidor com o cliente (Hunter, 2001).

Figura 5 - Aquitetura de um servlet



O objeto de solicitação permite ao *servlet* acessar informações tais como o nome do parâmetro passado pelo cliente, o protocolo que está sendo usado pelo cliente, o nome da máquina remota que fez a solicitação e o servidor que recebeu esta solicitação. O objeto de resposta fornece ao *servlet* a capacidade de resposta ao cliente (Hunter, 2001).

Os *servlets* são carregados e executados pelo servidor, aceitando assim, zero ou mais solicitações do cliente e retornando dados para o mesmo. O servidor também tem o potencial de remoção dos *servlets* inicializados. Quanto a manipulação de solicitações realizadas pelo cliente, somente após o carregamento do *servlet* isto é possível (Hunter, 2001).

3.3 API JAVA SERVLET

A *API Java servlet* contém várias interfaces *Java* que fornecem suporte para tratar a ligação entre o servidor remoto e os *servlets*. Esta *API* é uma extensão do padrão *JDK* (*Java Development Kit*) e está abaixo do pacote *javax*. A tabela 1 apresenta a *API Java servlet* com os pacotes: *javax.servlet* e *javax.servlet.http* (Jubin, 1999).

Tabela 1 - API Java Servlets

Pacote *javax.servlet*	
`Servlet`	Descreve os métodos de comunicação entre um servidor *web* e um *servlet*.
`ServletConfig`	Descreve os parâmetros de configuração para um *servlet*.
`ServletContext`	Descreve como o *servlet* pode obter informações sobre o servidor em que estiver sendo executado.
`ServletRequest`	Descreve como o *servlet* obtém informações sobre solicitações do cliente.
`ServletResponse`	Descreve como o *servlet* envia informações para os clientes.
`GenericServlet`	Utilizada para implementar *servlets* independentes de protocolo.
`ServletInputStream`	Uma especialização de `InputStream` utilizada para ler dados a partir de solicitações clientes.
`ServletOutputStream`	Uma especialização de `OutputStream` onde respostas são escritas para o cliente.

`ServletException`	Encarregada de derrubar o *servlet* quando um problema é encontrado.
`UnavailableException`	Encarregada de derrubar o *servlet* quando estiver indisponível por alguma razão desconhecida.
Pacote *javax.servlet.http*	
`HttpServletRequest`	Uma subclasse de `ServletRequest` que estabelece vários métodos de análise de cabeçalho e solicitações *HTTP*.
`HttpServletResponse`	Uma subclasse de `ServletResponse` responsável pelo acesso e interpretação de códigos de status e cabeçalho de informação *HTTP*.
`HttpServlet`	Uma subclasse de `GenericServlet` responsável por automaticamente separar uma solicitação *HTTP* através do seu tipo.
`HttpUtils`	Classe que fornece assistência para análise de solicitações (request)/respostas(*response*) *HTTP*.

3.4 FUNCIONALIDADES

Segundo Lucena (2000) *servlets* funcionam como um *plug-in* para o servidor *Web* (*webserver*) ou como um módulo de um servidor de aplicações. O *servlet plug-in* consiste em uma máquina virtual *Java* (*JVM*) que é executada dentro do servidor *Web* (*webserver*). O código dos programas *servlets* que rodam na máquina virtual *Java* instalada no servidor *Web* (*webserver*) são escritos em *Java* e tem acesso a todas as bibliotecas *Java* padrão (exceto as responsáveis por interface gráfica GUI como o *Swing*). Assim, o código do *Java servlet* é portável sem modificações para qualquer sistema operacional e qualquer servidor *Web* que possua um *plug-in servlet*. Os programas *servlets* são utilizados para representar a lógica do negócio: processamento dos dados, regras (algoritmos), computação matemática, acesso à dados através de *JavaBeans*, etc. Em prol do encapsulamento das aplicações, os programas *servlets* são responsáveis apenas pela execução da funcionalidade das aplicações. Por isto, os *servlets* não costumam possuir interface com o usuário.

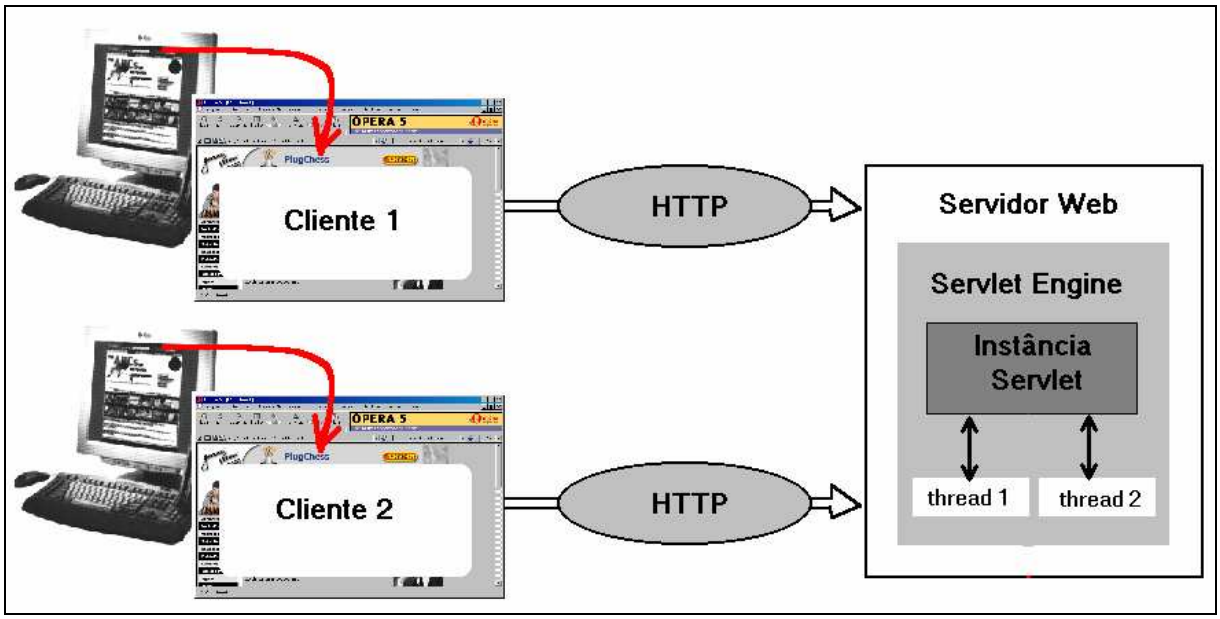
Colla (1999) cita algumas capacidades dos *servlets* que tornam esta tecnologia particularmente interessante:

a) geração dinâmica de páginas *HTML*: os *servlets* podem ser instalados em servidores *Web* para processarem informações transmitidas via *HTTP* a partir de formulários *HTML*, por exemplo. As aplicações podem incluir acesso a banco de dados ou comunicação com outros *servlet*s;
b) balanceamento de carga entre servidores: para entender como utilizar *servlets* para balanceamento de carga, deve-se considerar a infra-estrutura de um provedor de serviços via *Internet* composta de cinco servidores, dos quais quatro são capazes de executar as mesmas aplicações. O servidor restante poderia ficar responsável por monitorar as cargas dos demais e receber o acesso inicial de cada cliente às aplicações. Em seguida, poderia redirecionar os pedidos de acesso para um dos quatro servidores de aplicação, conforme a ocupação de cada um no momento em que o cliente tenta estabelecer uma conexão. Assim, o cliente passa a trocar informações somente com o servidor que foi alvo do redirecionamento;
c) modularização do código: um *servlet* pode executar outro *servlet*, mesmo que remotamente, sendo possível executá-los em "corrente". Essa característica possibilita a modularização dos aplicativos, criando *servlets* com funções específicas. Supondo que, para acessar um conjunto de aplicativos, o cliente deva ser autenticado. Neste caso, uma configuração possível seria criar um *servlet* responsável apenas pela tarefa de autenticação. Uma vez autenticado o cliente, este *servlet* o redirecionaria para outro *servlet*, não necessariamente instalado no mesmo servidor, e que executaria o aplicativo. A vantagem deste tipo de arquitetura é que, se por alguma razão for necessário modificar o procedimento de autenticação, por exemplo pela mudança do banco de dados de usuários, não será necessário reescrever toda a aplicação, mas apenas o *servlet* responsável pelo processo de autenticação.

Uma vez inicializado, o *servlet* estará apto a lidar com centenas de acessos simultaneamente, disparando, para cada acesso, uma nova *thread* (figura 6). As *threads* de um mesmo aplicativo utilizam um espaço de endereçamento de memória comum a todas, o que permite que elas compartilhem dados e recursos do sistema. As *threads* de um *servlet* podem usar uma única conexão estabelecida com um banco de dados no momento da inicialização do

servlet. Esta conexão permanece aberta até que o *servlet* seja desativado, saia da memória, ou seja recarregado.

Figura 6 - Várias threads utilizando a mesma instância

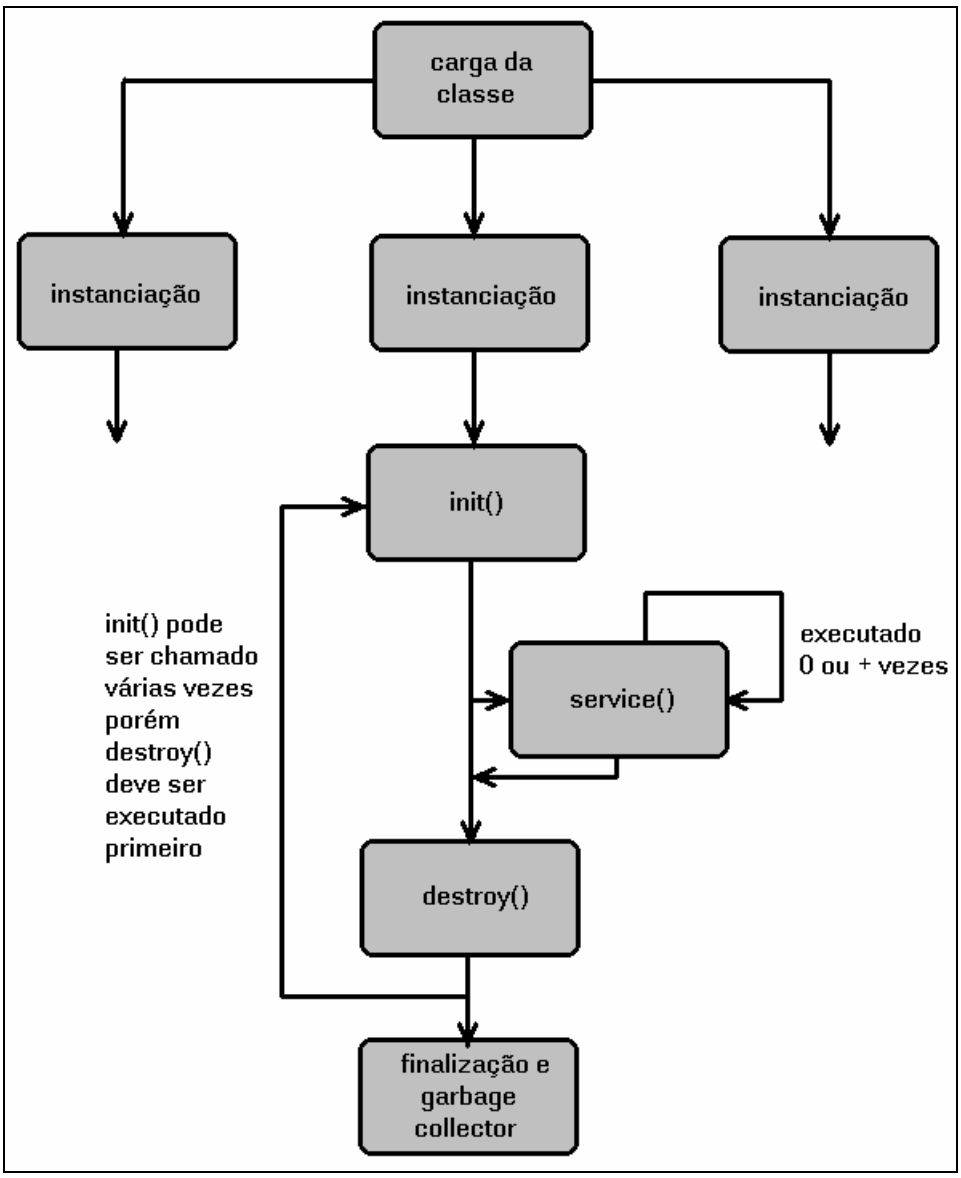


Soares (2000) diz que em função dos *servlets* serem escritos em *Java*, é possível escrever um *servlet* para um servidor *UNIX*, que poderá ser instalado em um servidor Windows NT, sem a necessidade de reescrever o código, ou recompilá-lo. A orientação a objetos, uma das características de Java, favorece a modularização dos aplicativos, o que os torna mais escaláveis.

3.5 CICLO DE VIDA

Os *servlets* são executados na mesma plataforma do servidor *Web*, como parte do mesmo processo que o servidor *Web*. O servidor *Web* é responsável pela inicialização, invocação e destruição de cada instância do *servlet* (Jubin, 1999). A comunicação do servidor *Web* com o *servlet* ocorre através da interface *javax.servlet.Servlet* e seu ciclo de vida consiste em três estágios: *init()*, *service()* e *destroy()*. A figura 7 apresenta o ciclo de vida de um *servlet*.

Figura 7 - Ciclo de vida do *servlet*



3.6 SESSÕES

O mecanismo pelo qual é possível persistir informações a respeito do cliente chama-se sessão. Através da utilização de sessões, é possível armazenar diversas informações do cliente

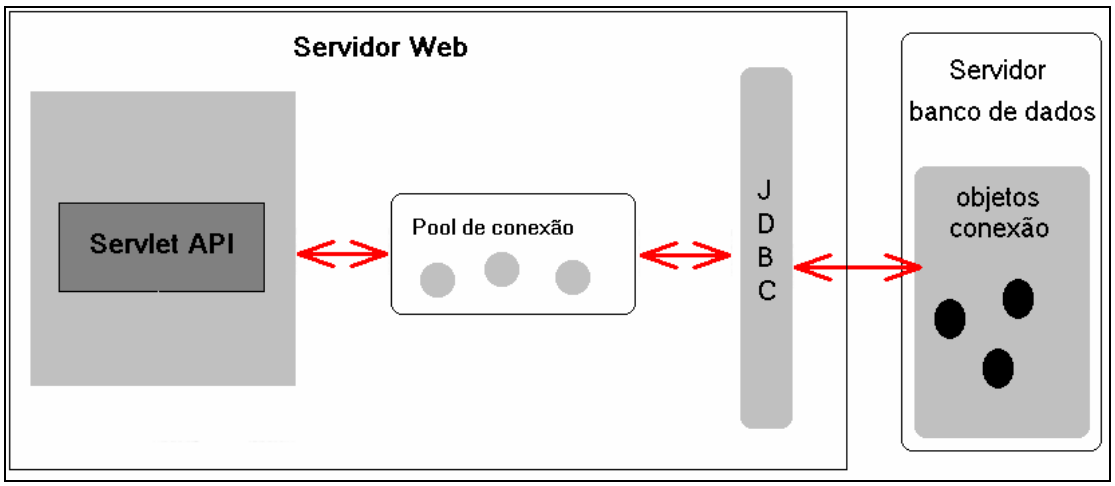
durante suas várias requisições. Uma sessão é inicializada quando o cliente faz a primeira chamada ao servidor, gerando um número de identificação. Este identificador é mantido durante as várias conexões do cliente, até que ele feche o navegador ou até que o seu tempo de vida expire (*time out*). O tempo de vida (*time out*) é configurado quando a sessão é criada (Jubin, 1999).

Uma sessão é mantida em memória no lado do servidor; os *servlets* permitem que sejam mantidas informações sobre a sessão através da classe *javax.servlet.HttpSession*. O pacote *javax.servlet.HttpSession* fornece uma coleção de métodos para criar, manipular e destruir sessões (Jubin, 1999).

3.7 POOL DE CONEXÕES

Pool de conexões é uma extensão da *API JDBC* para manipular *caches* de conexões a banco de dados. O pool de conexões reutiliza as conexões físicas ao banco de dados reduzindo o *overhead* de uma aplicação. A especificação da *API JDBC* providencia um serviço de acesso a banco de dados que utiliza a técnica de *pooling*, responsável por manipular múltiplos objetos de conexões em um *pool*, compartilhando-os de maneira transparente às requisições de clientes. Desta maneira um *servlet* pode utilizar um objeto de conexão sem sobrecarregar o *driver JDBC*. A aplicação que implementa um *pool* de conexões, o faz em um espaço de memória, otimizando recursos de conexão dinamicamente baseado na demanda, conforme ilustrado na figura 8 (Visveswaran, 2000).

Figura 8 - Pool de conexões



4 DESENVOLVIMENTO DO TRABALHO

Neste capítulo serão apresentadas as várias etapas do desenvolvimento do protótipo, tais como os requisitos do problema, a especificação, a implementação e resultados e discussão.

4.1 REQUISITOS DO PROBLEMA

Atualmente as consultas acadêmicas que estão disponíveis na *internet*, funcionam em um servidor *Web* proprietário com uma tecnologia fechada que dificulta a portabilidade e reusabilidade destas aplicações.

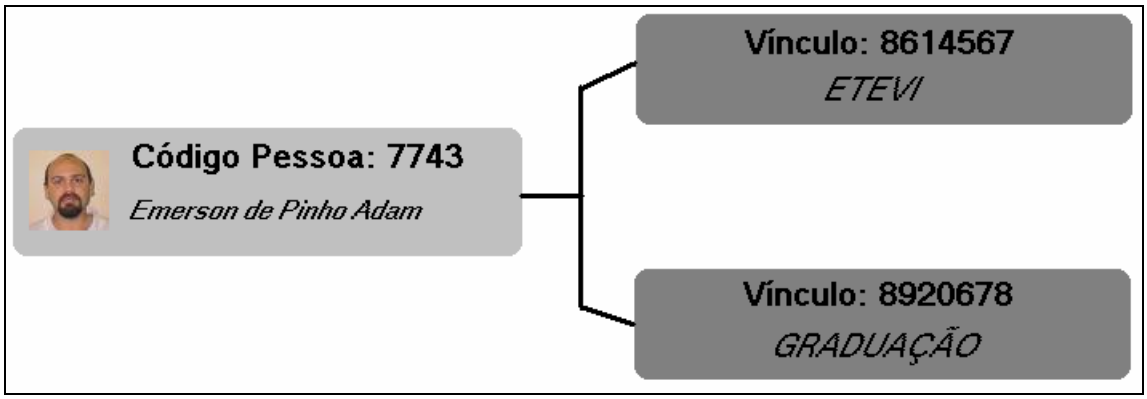
Deste modo, pretende-se com este trabalho fazer um protótipo de consultas acadêmicas para serem operadas pela *internet* e que ofereça aos seus usuários uma série de serviços como, consultas de notas, consultas financeiras, consultas de horários e consultas de turmas.

O objetivo deste trabalho é utilizar a tecnologia de *servlets* para o desenvolvimento do protótipo. A utililização de *servlets* se dá em função das principais características desta tecnologia que é aberta, funciona com qualquer servidor *Web*, é portável, rápida, orientada à objetos e reusável.

Qualquer pessoa que possua um vínculo com a universidade, seja este vínculo ativo ou não, poderá utilizar as consultas acadêmicas, desde que o usuário possua uma identificação pessoal.

A identificação pessoal é um código fornecido pela universidade no momento da matrícula da pessoa. Além do código de pessoa, o usuário necessitará de uma senha de autenticação no sistema, geralmente cadastrada também no ato da matrícula. A partir do código de pessoa e senha, o protótipo obtém todos os vínculos da pessoa. Podem existir vários tipos de vínculos relacionados com o código de pessoa, como pode ser visto na figura 9.

Figura 9 - Relacionamento do código de pessoa à vínculos



É importante ressaltar que as informações consultadas neste sistema já existem e são cadastradas através de outros sistemas. O banco de dados onde estas informações estão armazenadas é o *Oracle*.

4.2 ESPECIFICAÇÃO

Neste item será apresentada a especificação do problema segundo a notação da UML (*Unified Modeling Language*). Para a especificação do protótipo foram utilizados os seguintes diagramas:

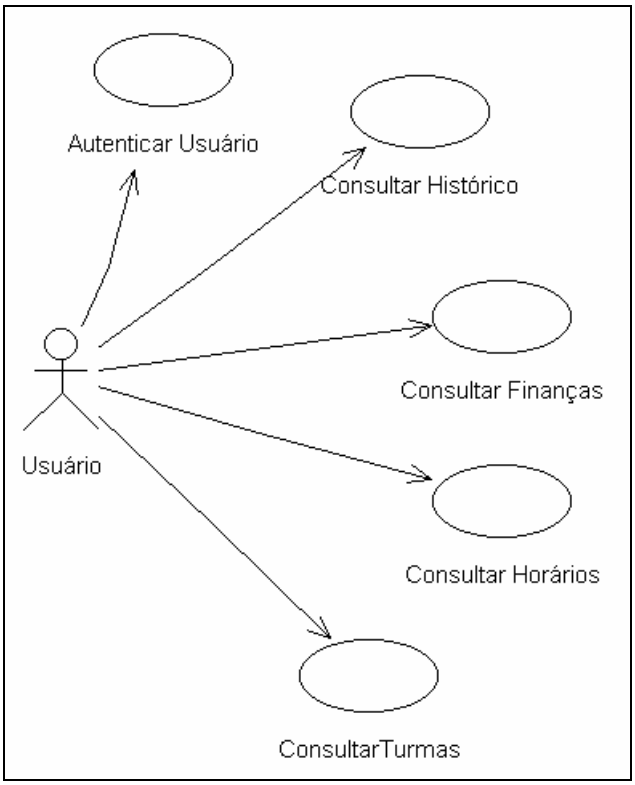
a) diagrama dos casos de usos: adotado para mostrar o conjunto de casos de uso e seus atores e relacionamentos;
b) diagrama de classes: adotado para mostrar o conjunto de classes, interfaces e seus relacionamentos;
c) diagrama de seqüência: adotado para dar ênfase à ordenação temporal de mensagens.

A ferramenta utilizada para fazer a especificação segundo a notação da UML (*Unified Modeling Language*) foi o *Rational Rose* (versão *student*).

4.2.1 DIAGRAMA DOS CASOS DE USO

Na figura 10 é apresentado o diagrama de casos de uso.

Figura 10 – Diagrama de casos de uso

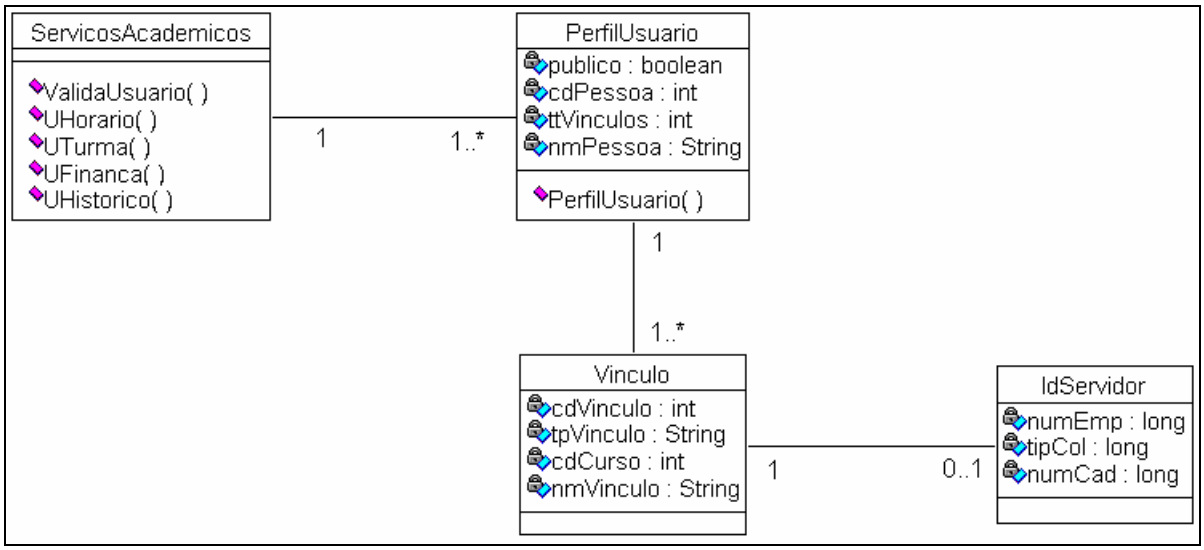


a) Autenticar Usuário: é o caso no qual o usuário é autenticado quando se conecta no sistema para fazer suas consultas. Após a autenticação, é ofertado um menu de serviços de acordo com o perfil do usuário;
b) Consultar Histórico: é o caso no qual o usuário quer informações sobre seu histórico escolar. O histórico do usuário apresenta informações sobre cada disciplina que o usuário cursou ou esteja cursando. Estas informações sobre as disciplinas são essencialmente sobre notas obtidas;
c) Consultar Finanças: é o caso no qual o usuário quer saber qual é a sua situação financeira. A situação financeira do usuário indica quais mensalidades foram pagas, a data de cada pagamento e o valor de cada pagamento;
d) Consultar Horários: é o caso no qual o usuário quer informaçõe sobre horário e sala onde acontecerão suas aulas. Na consulta de horários é informado também o dia da semana e a duração de cada aula;
e) Consultar Turmas: é o caso no qual o usuário quer obter informações sobre sua turma. Nas informações sobre turmas serão apresentados os professores daquela turma, os alunos e sala onde a turma tem aula.

4.2.2 DIAGRAMA DE CLASSES

Na figura 11 é representado o diagrama de classes.

Figura 11 – Diagrama de classes



A classe `ServicosAcademicos` é responsável por agrupar os diversos serviços ofertados pelo protótipo. Ela implementa os seguintes métodos:

a) `ValidaUsuario()`: faz a identificação e autenticação do usuário;

b) `UHorario()`: disponibiliza as consultas de horários;

c) `UTurma()`: disponibiliza as consultas de turmas;

d) `UFinanca()`: disponibiliza as consultas financeiras;

e) `UHistorico()`: disponibiliza as consultas de histórico.

Esta classe será implementada como um *servlet*.

A classe `PerfilUsuario` é a estrutura onde serão persistidas todas as informações pertinentes ao perfil do usuário.

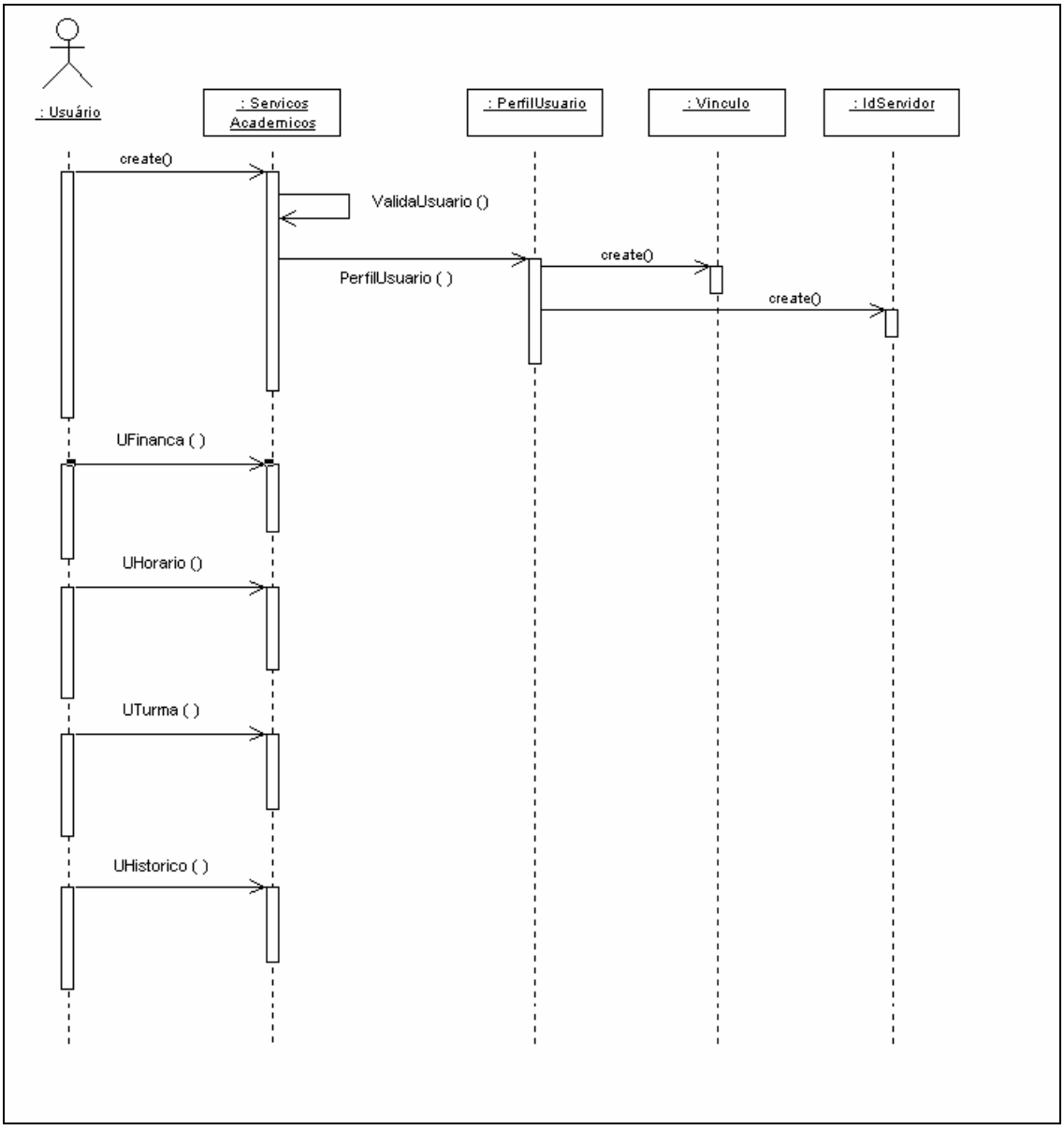
A classe `Vinculo` é onde serão mantidas as informações sobre ocorrência de diversos vínculos acadêmicos do usuário.

A classe `IdServidor` é onde serão mantidas informações sobre o caso de um usuário possuir somente um vínculo de servidor, ou seja, ele é um funcionário da instituição.

4.2.3 DIAGRAMA DE SEQÜÊNCIA

Na figura 12 é apresentado o diagrama de seqüência do protótipo. Este diagrama apresenta as operações que ocorrem em todos os casos de uso do sistema.

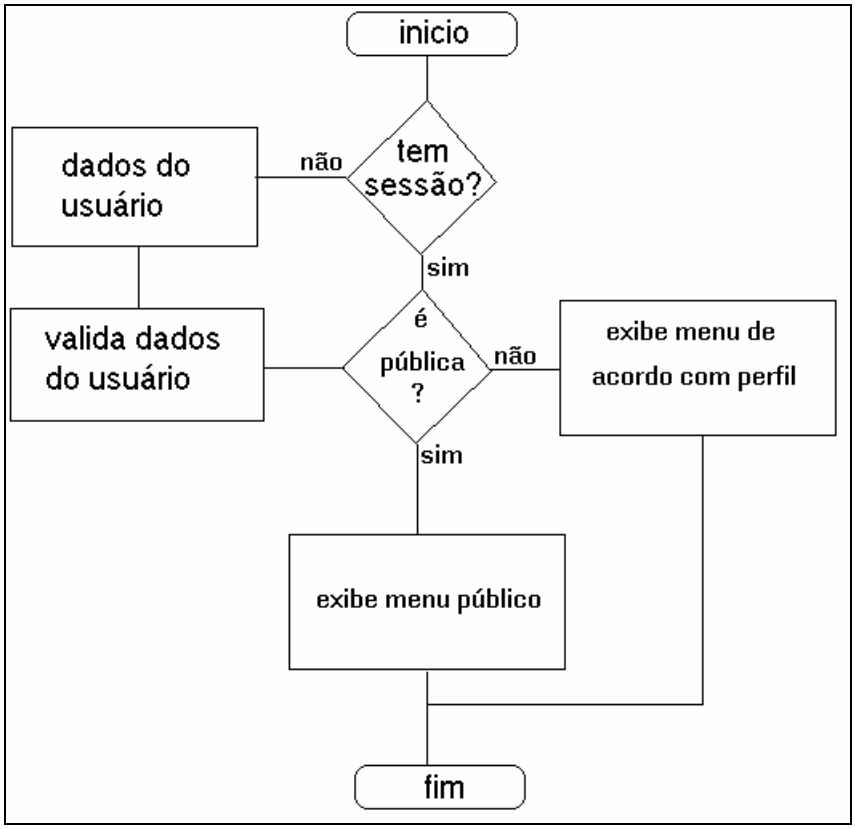
Figura 12 – Diagrama de seqüência



4.2.3.1 FLUXOGRAMA PARA VALIDAR USUÁRIO

Na figura 13 é apresentado o fluxograma do método `ValidaUsuario()`de validação de usuário.

Figura 13 - Fluxograma do método de validação de usuário



4.3 FLUXOGRAMA DAS CONSULTAS

Na figura 14 é apresentado um fluxograma genérico para os métodos:

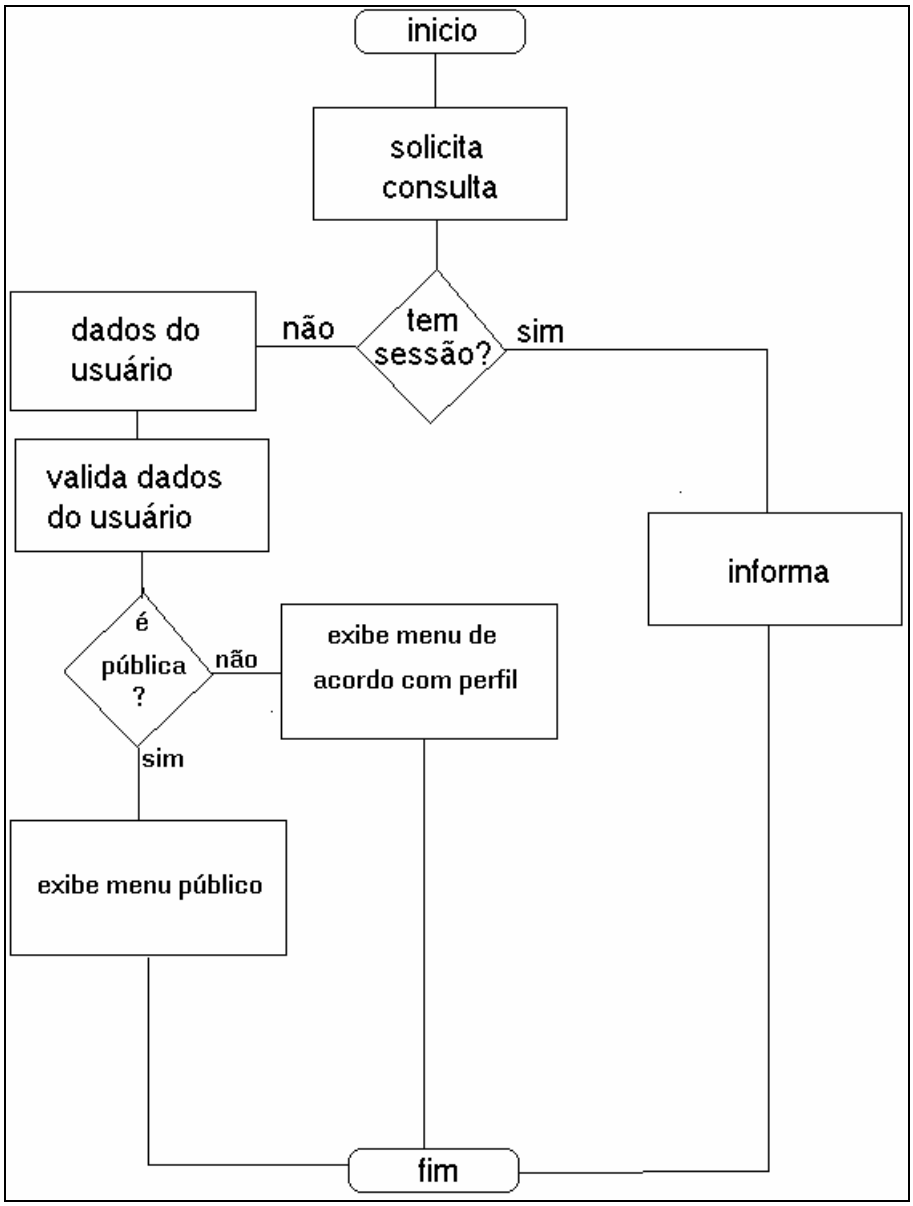
a) `UFinanca()`que dispara a consulta financeira;

b) `UHorario()`que dispara a consulta de horários de disciplinas e salas de aula;

c) `UTurma()`que dispara a consulta de turmas;

d) `UHistorico()`que dispara a consulta de notas.

Figura 14 – Fluxograma geral das consultas



É importante observar que independentemente da consulta que está sendo feita, sempre será necessário perguntar se existe uma sessão. Isso se deve ao fato do perfil do usuário estar vinculado a uma sessão. Quando uma sessão do usuário é criada, um *time-out* é configurado visando garantir a privacidade do usuário no caso de ele se esquecer de fechar sua sessão após realizar as consultas. Confirmada esta situação e passado o tempo configurado no *time-out*, a sessão é então invalidada.

4.4 MODELO DE DADOS DO SISTEMA ACADÊMICO

Como destacado anteriormente as consultas serão feitas sobre dados já existentes do sistema acadêmico de graduação da *FURB*.

As principais tabelas deste sistema utilizadas no trabalho estão descritas a seguir:

1) `curso_graduacao`: mantém informações do curso de graduação;
2) `curso_laboratorio_linguas`: mantém informações do curso do laboratório de línguas;
3) `curso_pos_graduacao`: mantém informações do curso de pós-graduação;
4) `disciplina_graduacao`: mantém informações sobre disciplina da graduação;
5) `historico_disciplina_graduacao`: mantém informações sobre os históricos de disciplinas da graduação;
6) `historico_graduacao`: mantém informações sobre o histórico da graduação;
7) `historico_laboratorio_linguas`: mantém informações sobre o histórico do curso do laboratório de línguas;
8) `historico_pos_graduacao`: mantém informações sobre o histórico da pós-graduação;
9) `horario_graduacao`: mantém informações sobre os horários de disciplinas da graduação;
10) `observacao_nota_graduacao`: mantém informações sobre observações de notas em disciplinas da graduação;
11) `parametro_curso_lblg`: mantém informações sobre parâmetros do laboratório de línguas;
12) `parametro_graduacao`: mantém informações sobre parâmetros da graduação;
13) `pessoa`: mantém informações sobre pessoas;
14) `r034fun`: mantém informações sobre funcionários;
15) `tipo_situacao_graduacao`: mantém informações sobre sobre o tipo de situação da graduação;
16) `tipo_situacao_lblg`: mantém informações sobre sobre o tipo de situação do laboratório de línguas;
17) `tipo_situacao_pgra`: mantém informações sobre o tipo de situação da pós-

graduação;

18) `titulo_receita`: mantém informações sobre receitas;
19) `titulo_receita_graduacao`: mantém informações sobre receitas da graduação;
20) `turma_curso_graduacao`: mantém informações sobre cursos de uma turma da graduação;
21) `turma_graduacao`: mantém informações sobre turmas da graduação;
22) `turma_horario_graduacao`: mantém informações sobre horários de uma turma da graduação;
23) `turma_professor_graduacao`: mantém informações sobre o professor de uma turma da graduação;
24) `ultimo_historico_graduacao`: mantém informações sobre o último histórico de um vínculo da graduação;
25) `vinculo`: mantém informações sobre os diversos vínculos que uma pessoa possa ter.

O diagrama do modelo de dados com atributos e relacionamentos foi omitido por ser muito extenso e não ser objeto de estudo e análise no trabalho.

4.5 IMPLEMENTAÇÃO

Neste item serão apresentadas as técnicas e ferramentas utilizadas no trabalho bem como a implementação do mesmo.

4.5.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para o funcionamento deste trabalho foram utilizadas as seguintes ferramentas:

a) servidor *Web Apache*: o *Apache* é um servidor *Web* que acompanha o sistema operacional *Linux*. Uma característica interessante que une o *Apache* ao *Linux* é que ambos possuem licença de uso com distribuição livre;
b) servidor de *servlets Jakarta-TomCat*: o *Jakarta-TomCat* é um servidor de *servlets* que roda integrado ao *Apache*. O *TomCat* também possui licença de uso com distribuição livre;
c) *VisualAge for Java (VAJ)*: é o ambiente selecionado para o desenvolvimento dos

diversos *servlets* que compõem o protótipo;

d) Banco de Dados *Oracle*: é o banco onde estão todas as tabelas do sistema acadêmico de graduação.

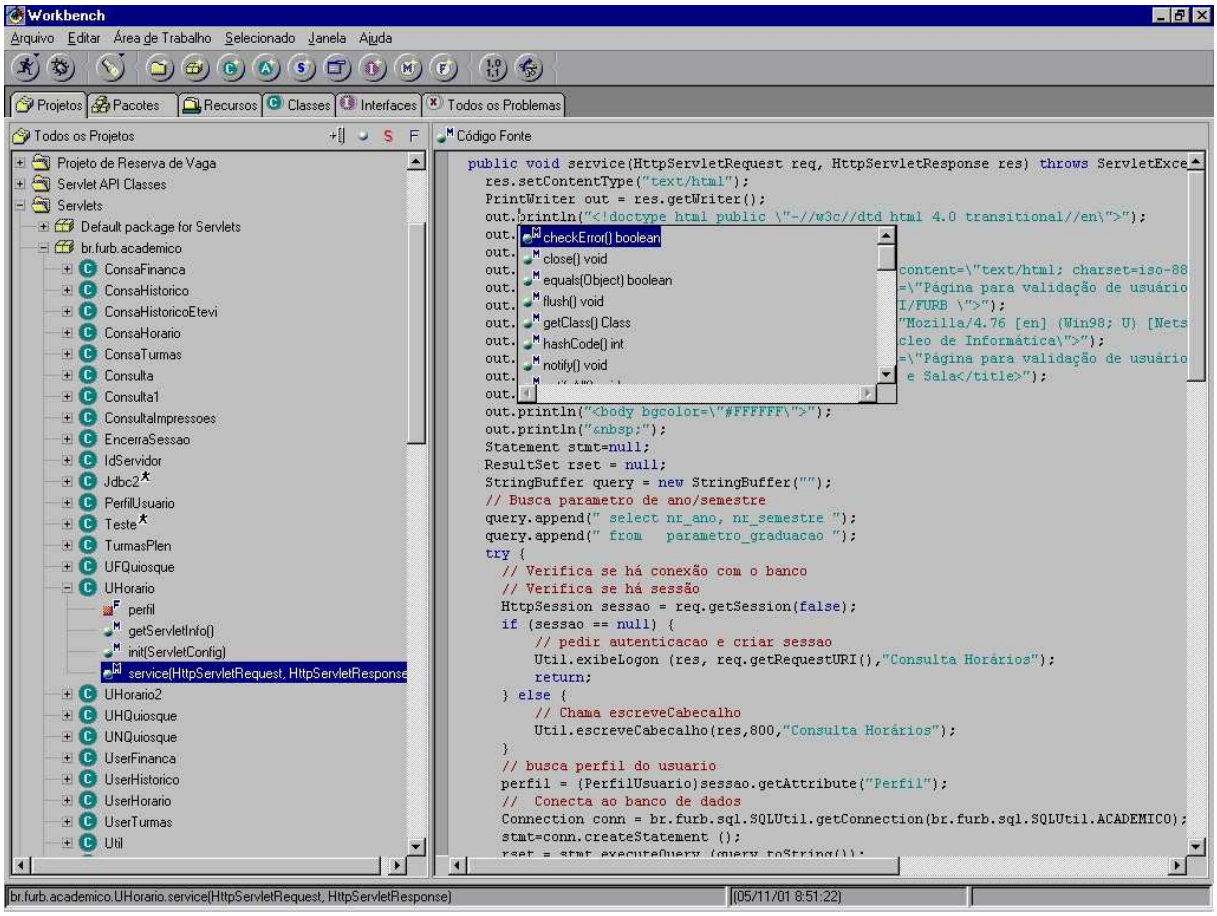
O servidor *Web* e o *TomCat* foram instalados em um *desktop* com sistema operacional *Linux* (petium 233mmx com 128MB de memória, com um disco rígido de 10GB). O banco de dados *Oracle* está instalado numa máquina servidora com sistema operacional *Windows NT*. O *VAJ* foi instalado num *desktop* de desenvolvimento, que roda o sistema operacional *Windows NT Workstation*.

A seguir serão apresentadas algumas telas do *VAJ*.

4.5.1.1 AMBIENTE DE DESENVOLVIMENTO VISUAL AGE FOR JAVA

O *Visual Age for Java* é um ambiente integrado e visual que suporta completamente o ciclo de desenvolvimento de programas em *Java*. Neste ambiente pode-se criar *applets Java* que rodam em navegadores *Web*, aplicações *Java stand-alone* e aplicações *servlets* que executam em um servidor. A figura 15 apresenta o ambiente de desenvolvimento *Visual Age for Java*. (IBM, 2000)

Figura 15 – Ambiente de desenvolvimento

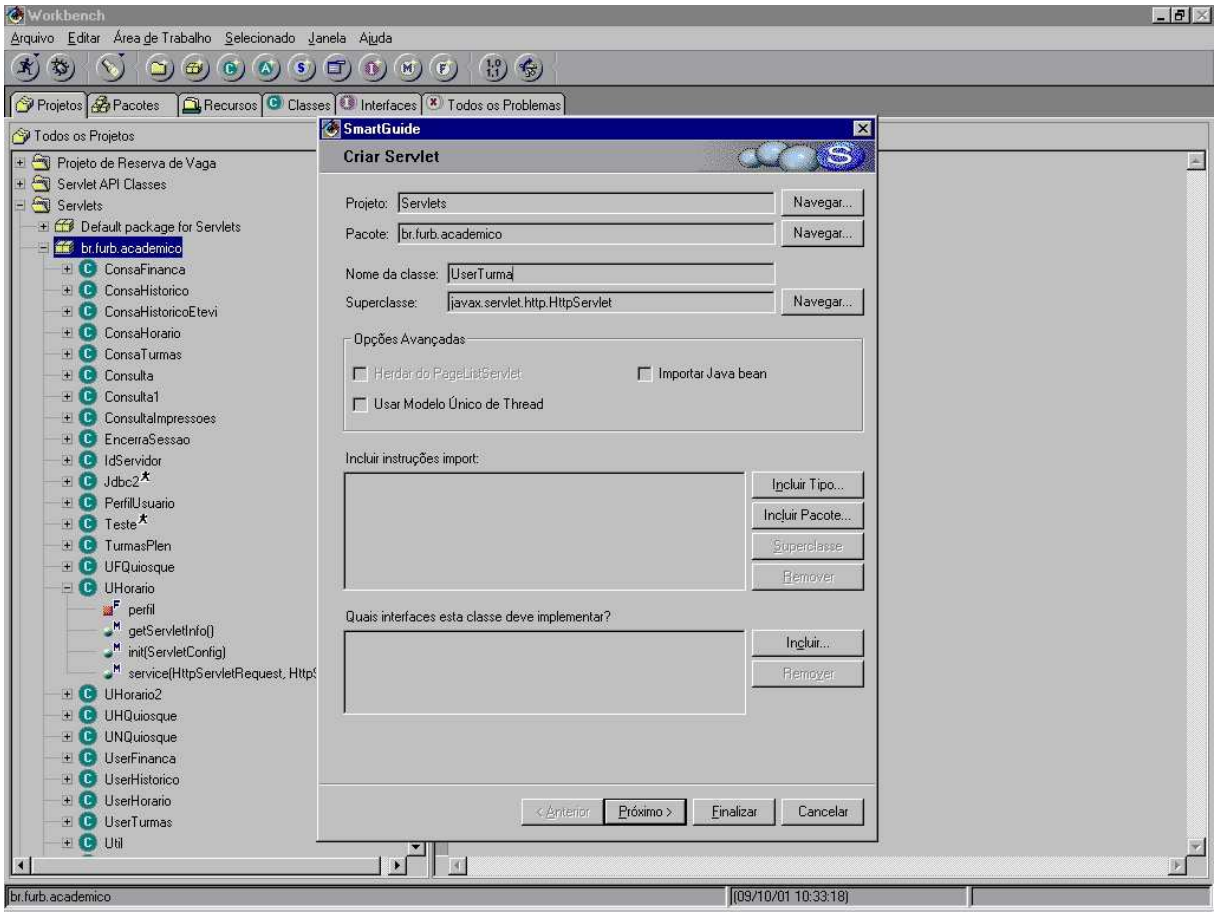


Toda atividade desenvolvida no *VAJ* é organizada dentro de uma área de trabalho (figura 15) onde estão todos os projetos, pacotes e arquivos. Dentro do ambiente do *VAJ* não se manipula arquivos, pois o *VAJ* gerencia todo o código escrito na área de trabalho em uma estrutura de banco de dados de objetos. Essa estrutura recebe o nome de repositório (IBM, 2000).

Qualquer código criado no *VAJ* é automaticamente armazenado no repositório, que mantém também todas as *API's* do *Java*. O *VAJ* proporciona também o gerenciamento de repositórios compartilhados, gerenciamento de versões, e diversas outras funcionalidades encontradas na maioria dos ambientes integrados de desenvolvimento. Dentre tais funcionalidades pode-se destacar vários *wizards* que auxiliam na criação de *applets*, *servlets*, *javabeans*, *enterprise javabeans*, aplicações e ajuda sensitiva ao contexto de classes.

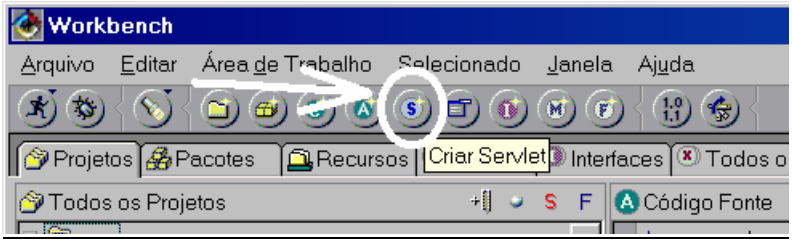
Na figura 16, pode ser visto o ambiente *VAJ* com suas ferramentas para o desenvolvimento de *servlets*.

Figura 16 – Criando um Servlet a partir do Visual Age for Java



Dentre os diversos *wizards* que o *VAJ* disponibiliza, um é de fundamental interesse para este trabalho, o de criar *servlets*. Para criar um *servlet* utilizando esta facilidade, basta acionar o botão criar *servlet* (figura 17). Após isso a janela criar *servlet* é aberta, e deve-se, então, informar o projeto, o pacote, o nome do *servlet* e qual a super classe a implementar.

Figura 17 – Barra de ferramentas com o botão criar servlet

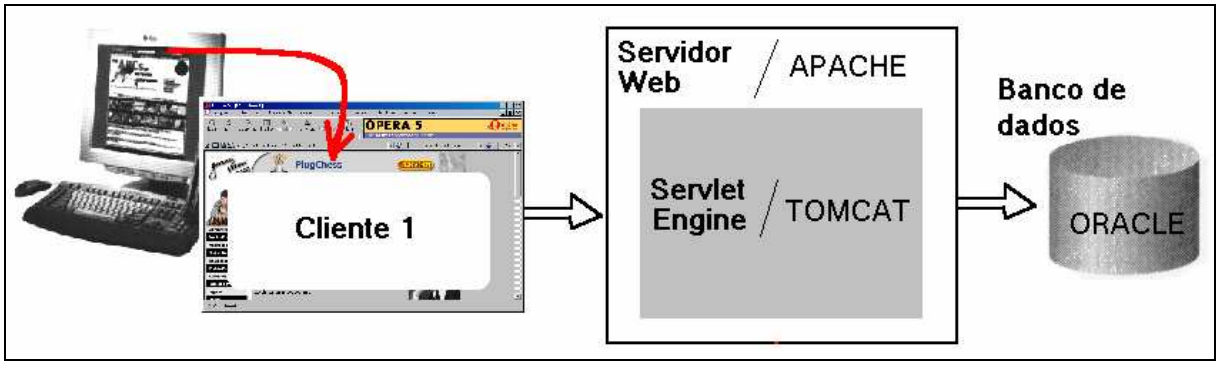


É interessante destacar que nas informações *default* da janela de criar *servlets*, o campo onde se especifica a super classe, já vem preenchido com a classe `javax.servlet.http.HttpServlet` que é uma subclasse de `GenericServlet` e é responsável por automaticamente separar uma solicitação *HTTP* através do seu tipo.

4.5.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Neste tópico serão apresentadas as telas do protótipo, bem como os detalhes da operabilidade do protótipo implementado. A arquitetura utilizada para a operabilidade do protótipo pode ser visualizado na figura 18, onde é apresentado o ambiente do usuário (qualquer navegador), o servidor *Web* (*Apache*) e o banco de dados (*ORACLE*).

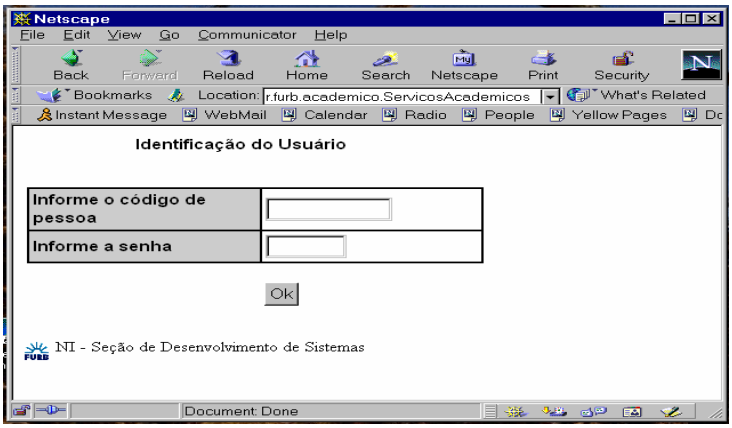
Figura 18 – Arquitetura da operabilidade da implementação



4.5.2.1 AUTENTICAÇÃO DO USUÁRIO

Na figura 19 é apresentada a tela onde o usuário deve fazer sua autenticação.

Figura 19 – Tela de autenticação do usuário

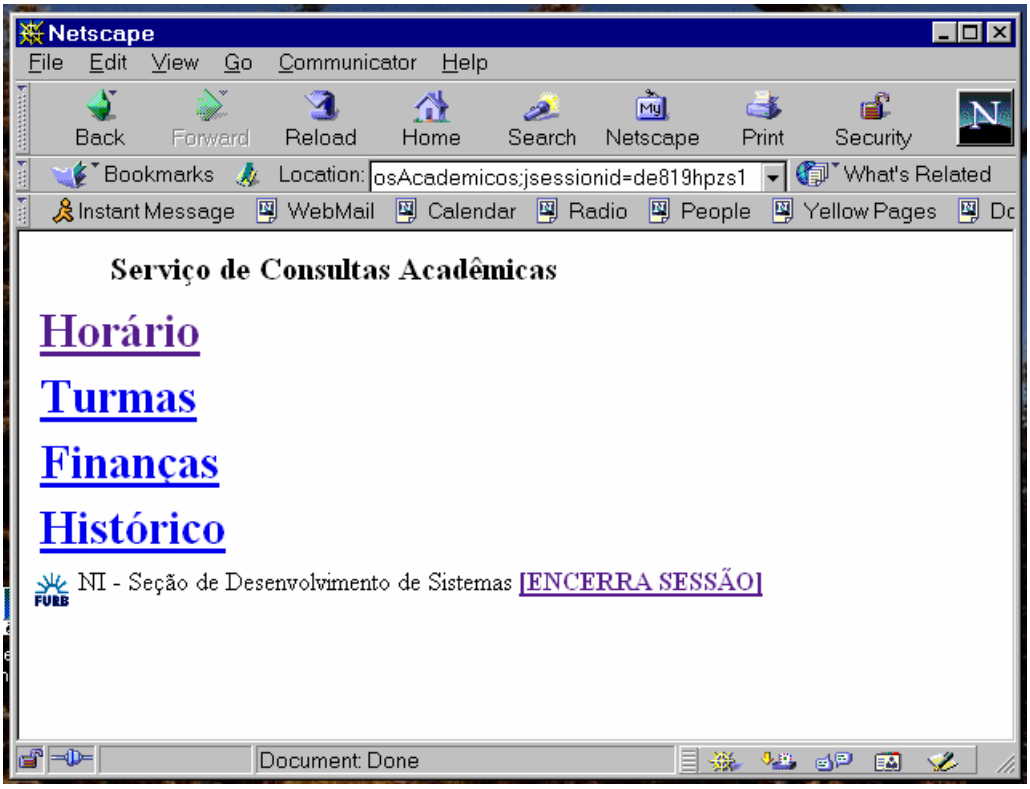


Ao clicar no botão *Ok* o usuário será validado pelo sistema, de acordo com o apresentado anteriormente no fluxograma do método `ValidaUsuario()`. O código desta implementação é acionado pela classe `ServicosAcademicos()`.

4.5.2.2 SERVIÇOS ACADÊMICOS

Na figura 20 é apresentada a tela de oferta de serviços acadêmicos.

Figura 20 – Tela de serviços acadêmicos



O quadro 1 apresenta o código fonte em *Java* responsável pela exibição do menu do usuário de acordo com seu perfil.

Quadro 1 – Código fonte que exibe o menu de serviços

```
package br.furb.academico;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class ServicosAcademicos extends HttpServlet {
  private PerfilUsuario perfil;

  public void init(ServletConfig config) throws ServletException {
```

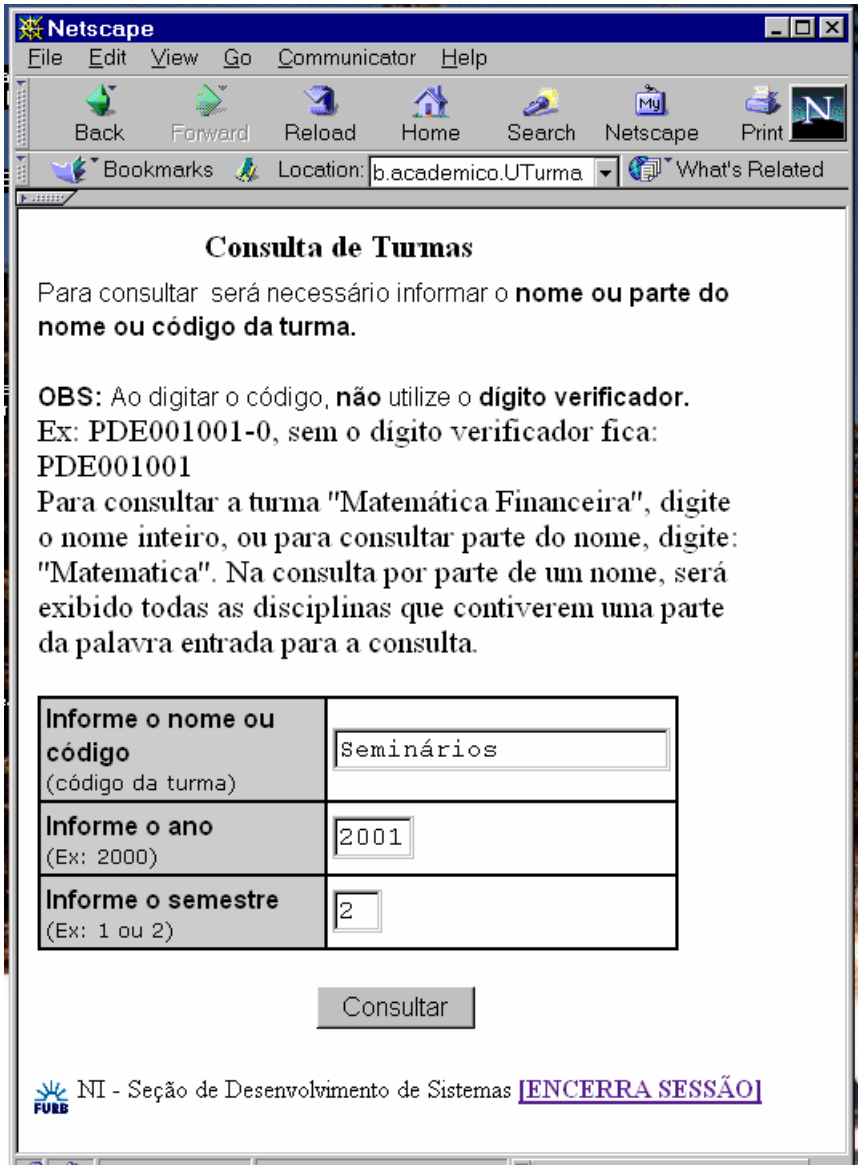
```
        super.init(config);
      }

      public void service(HttpServletRequest req, HttpServletResponse res) throws
                        ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        // Verifica se há sessão
        HttpSession sessao = req.getSession(false);
        if (sessao == null) {
           // pedir autenticacao e criar sessao
           Util.exibeLogon (res, req.getRequestURI(),"Identificação do Usuário");
           return;
        }
        // busca perfil do usuario
        perfil = (PerfilUsuario)sessao.getAttribute("Perfil");

        Util.escreveCabecalho(res, "377", "Serviço de Consultas Acadêmicas");
        out.println("<TABLE width=\"290\">");
        out.println("  <TBODY>");
        // exibe consultas publicas
        out.println("    <TR>");
        out.println("<TD><FONT size=\"+2\"><A href=\""+res.encodeURL(
                    "br.furb.academico.UHorario")+"\"><B>Horário</B></A></FONT></TD>");
        out.println("    </TR>");
        out.println("    <TR>");
        out.println("<TD><FONT size=\"+2\"><A ref=\""+res.encodeURL(
                    "br.furb.academico.Uturma ")+ "\"><B>Turmas</B></A></FONT></TD>");
        out.println("    </TR>");
        if (!perfil.publico) {
          // Consulta privada
          // Histórico, Financeira, Turmas e Horário
          out.println("    <TR>");
          out.println("<TD><FONT size=\"+2\"><A href=\""+res.encodeURL(
                      "br.furb.academico.UFinanca")+"\"><B>Finanças</B></A></FONT></TD>");
          out.println("    </TR>");
          out.println("    <TR>");
          out.println("<TD><FONT size=\"+2\"><A href=\""+res.encodeURL(
                      "br.furb.academico.UHistorico")+"\"><B>Histórico</B></A></FONT></TD>"
                      );
          out.println("    </TR>");
        }
        out.println("  </TBODY>");
        out.println("</TABLE>");
        // escreve rodape
        Util.escreveRodape(res);
        out.close();
      }


      public String getServletInfo() {
        return "br.furb.academico.ServicosAcademicos Information";
      }
    }
```

A classe `ServicosAcademicos()` é a primeira classe a ser instanciada no sistema e ela é responsável por disponibilizar diversos serviços acadêmicos de acordo com o perfil do seu usuário. Se existe uma sessão de usuário, então já existe um perfil e a partir disto é exibido um menu de serviços ao usuário. Caso não exista uma sessão, então o método `ValidaUsuário()` é acionado para pedir uma autenticação.

4.5.2.3 CONSULTA DE TURMAS

Na figura 21 é apresentada a tela de solicitação de consulta de turmas.

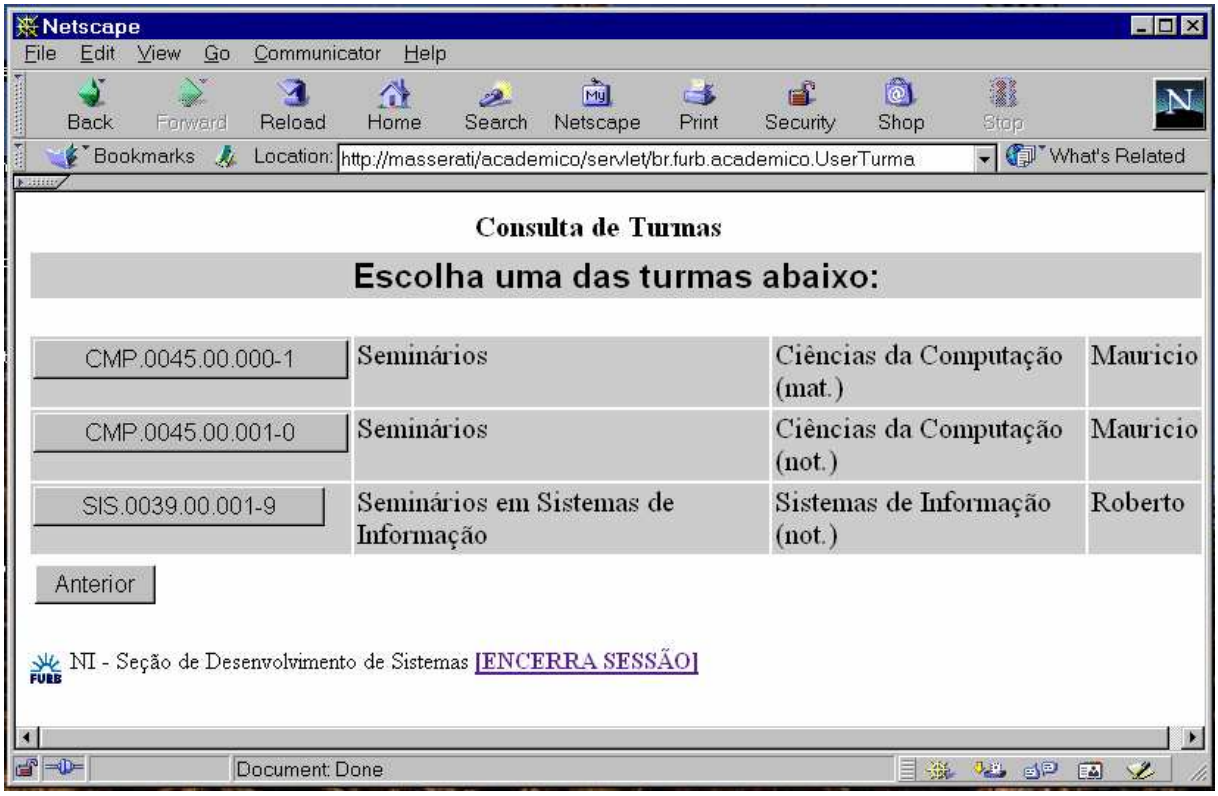
Figura 21 - Tela de solicitação de consulta de turmas



O método `UTurma()` que é um *servlet* dispara uma seqüência de *servlets* que ao final irá retornar o resultado desejado. Quando o usuário informar o nome ou código, ano, semestre da turma a ser pesquisado, e clicar no botão Consultar, o *servlet* `UTurma()` chama o próximo *servlet* que dará prosseguimento a consulta. A seqüência disparada pelo `UTurma()` será apresentada a seguir.

Na figura 22 é apresentada a tela de seleção de turmas para consultar.

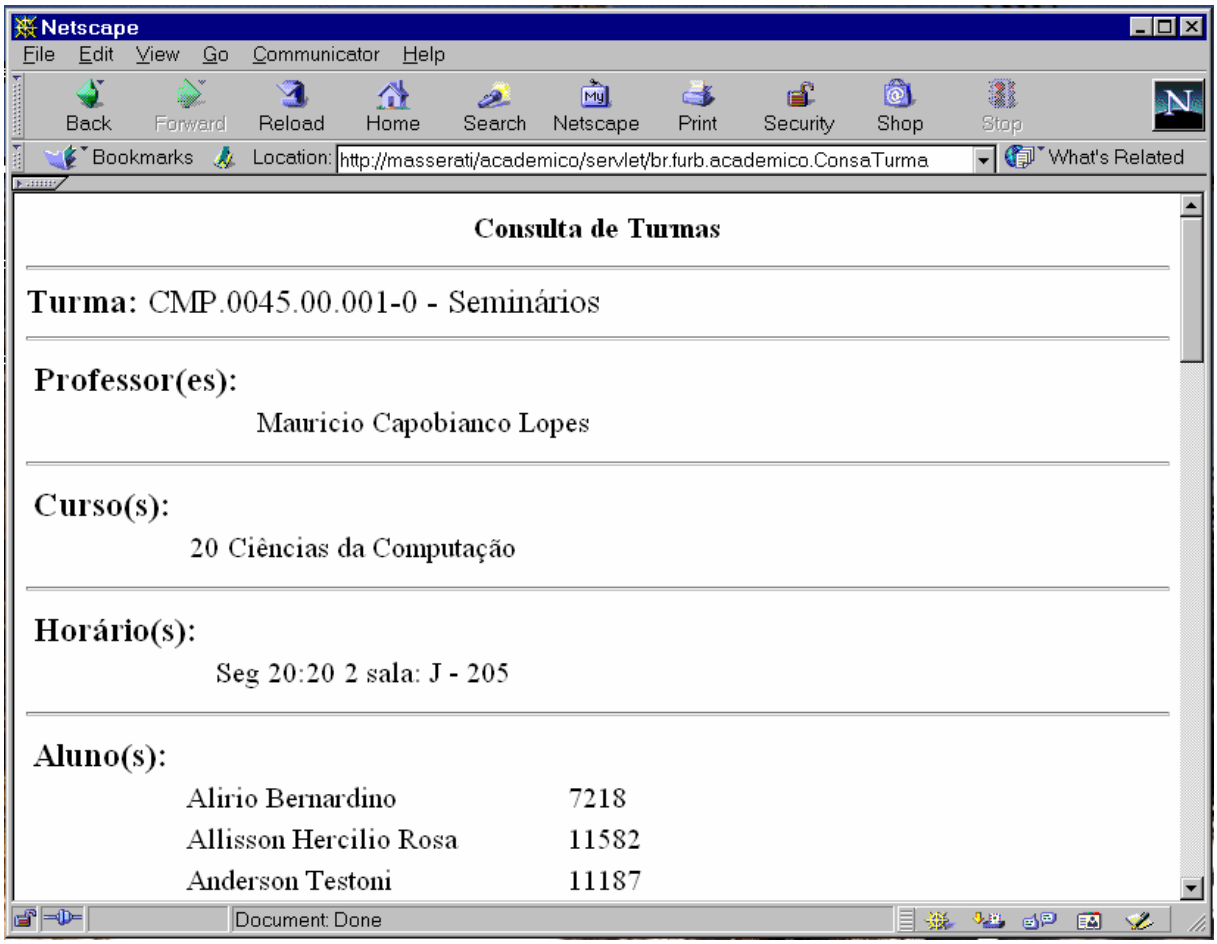
Figura 22 – Tela de seleção de turmas para consultar



O *servlet* `UTurma()`chama o *servlet* `UserTurma()`, que retorna a lista de turmas disponíveis para consultar. O próximo *servlet* será instanciado no momento em que o usuário pressionar o botão com o código da turma que deseja consultar.

Na figura 23 é apresentado uma tela com o resultado da consulta de uma turma específica.

Figura 23 – Tela do resultado da consulta de turma



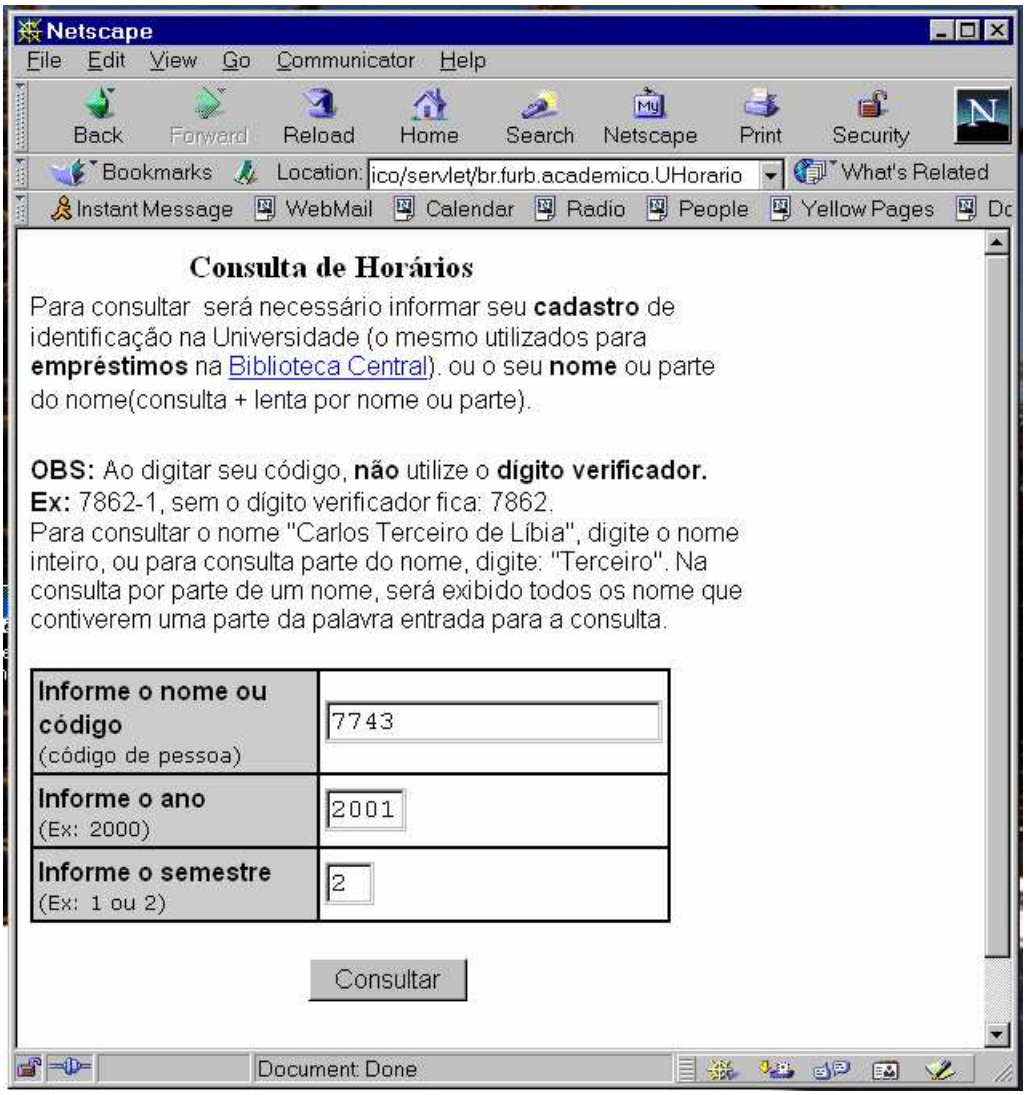
O *servlet* `UserTurma()` é quem chama o *servlet* `ConsaTurma()`, que retorna a consulta da turma desejada.

No anexo 1 é apresentado o código fonte em *Java* responsável pela exibição do resultado da consulta de turmas.

4.5.2.4 CONSULTA DE HORÁRIO

Na figura 24 é apresentada a tela de solicitação de consulta de horários.

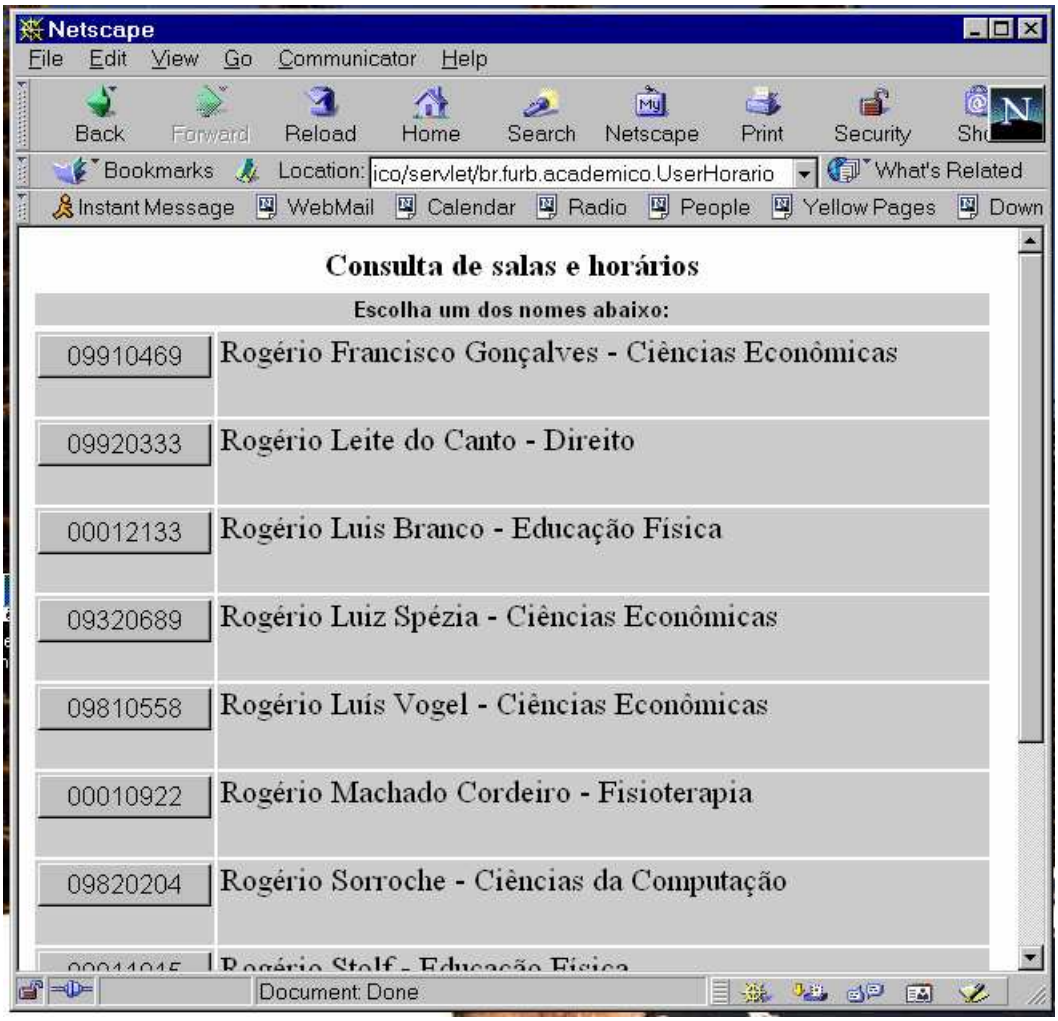
Figura 24 - Tela de solicitação de consulta de horários



O método `UHorario()`que é um *servlet* dispara uma seqüência de *servlets* que ao final irá retornar o resultado desejado. Quando o usuário informar o nome ou código, ano, semestre de pessoa a ser pesquisado, e clicar no botão Consultar, o *servlet* `UHorario()`chama o próximo *servlet* que dará prosseguimento a consulta. A seqüência disparada pelo `UHorario()`será vista a seguir.

Na figura 25 é apresentada a tela de seleção de nomes para consultar horário.

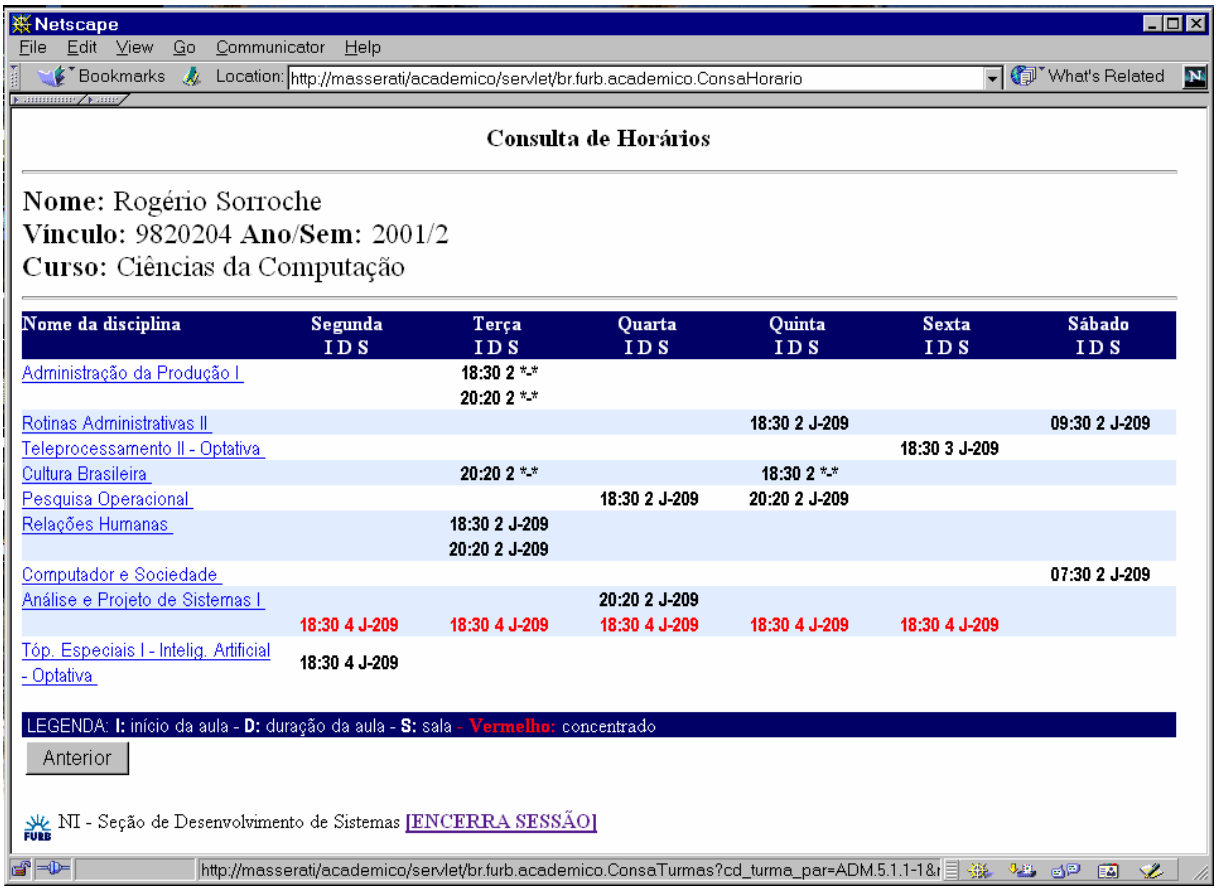
Figura 25 - Tela de seleção de nomes para consultar horário.



O *servlet* `UHorario()`é quem chama o *servlet* `UserHorario()`, que retorna a lista de nomes disponíveis para consultar um horário. O próximo *servlet* será instanciado no momento que o usuário pressionar o botão com o código do vínculo que deseja.

Na figura 26 é apresentado a tela com o resultado da consulta de horários.

Figura 26 – Tela de consulta de horários



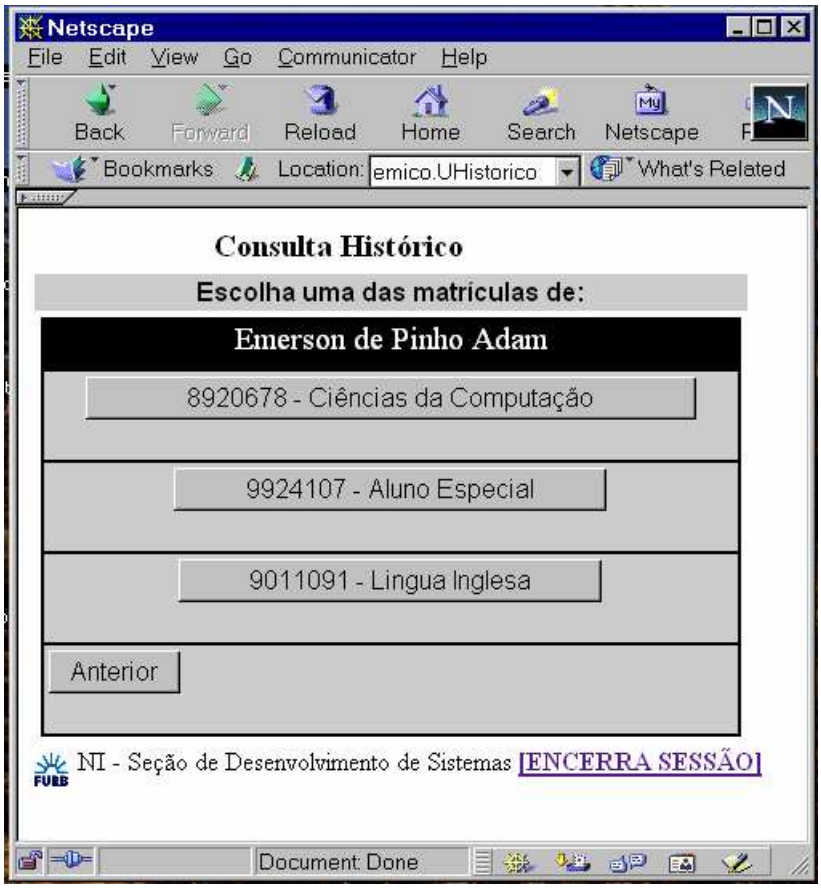
O *servlet* `UserHorario()` é quem chama *servlet* `ConsaHorario()`, que retorna a consulta de horário desejada.

No anexo 2 é apresentado o código fonte em *Java* responsável pela exibição do resultado da consulta de horários.

4.5.2.5 CONSULTA DE HISTÓRICO

Na figura 27 é apresentada a tela de solicitação de consulta de histórico.

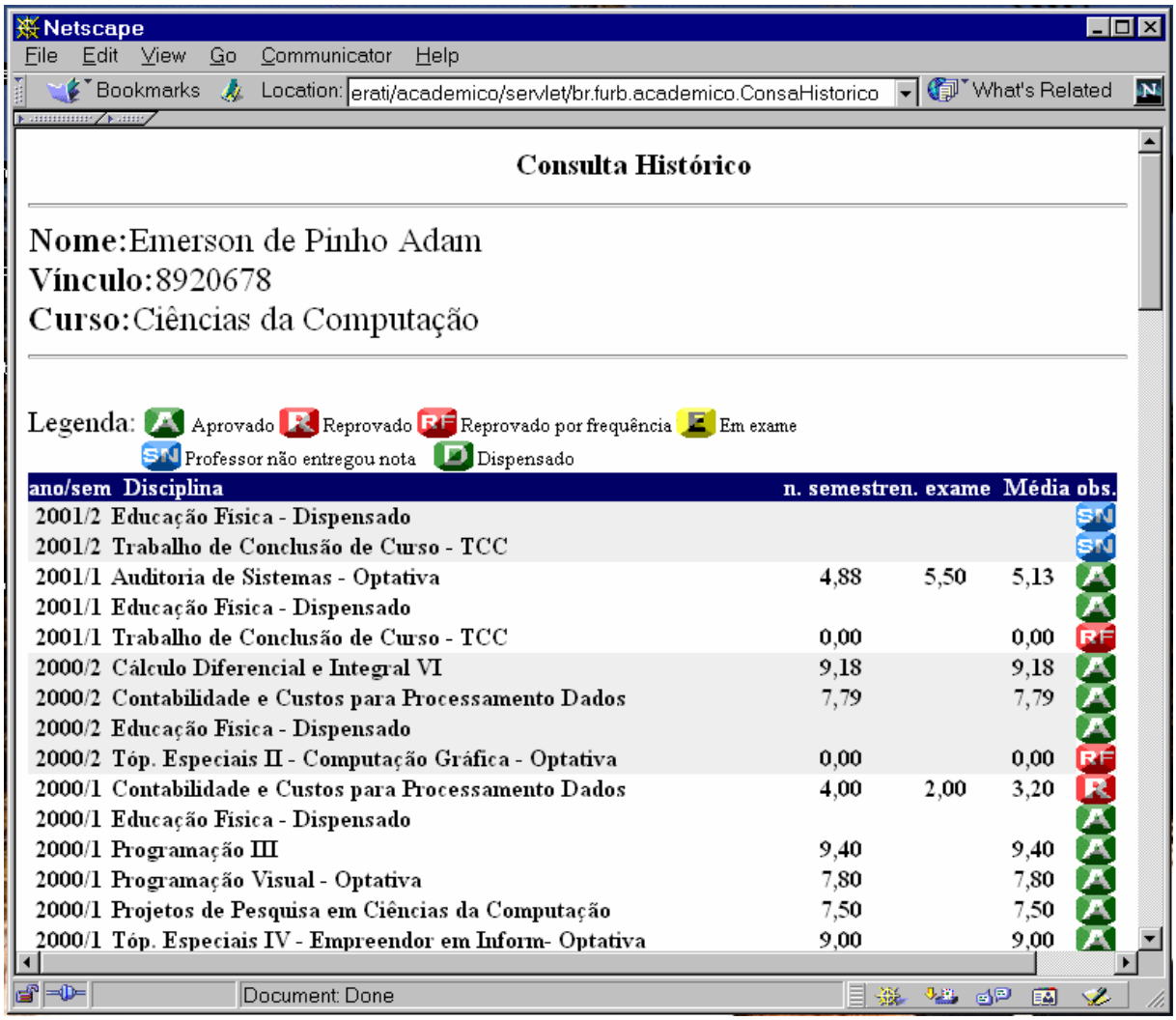
Figura 27 - Tela de solicitação de consulta de histórico



O método `UHistorico()`que é um *servlet* dispara uma seqüência de *servlets* que ao final irá retornar o resultado desejado. Quando o usuário clicar no botão com a descrição da matrícula que deseja consultar, o *servlet* `UHistorico()`chama o próximo *servlet* que dará prosseguimento a consulta. A seqüência disparada pelo `UHistorico()`será vista a seguir.

Na figura 28 é apresentado uma tela com o resultado da consulta de notas ou histórico.

Figura 28 – Tela de consulta de histórico



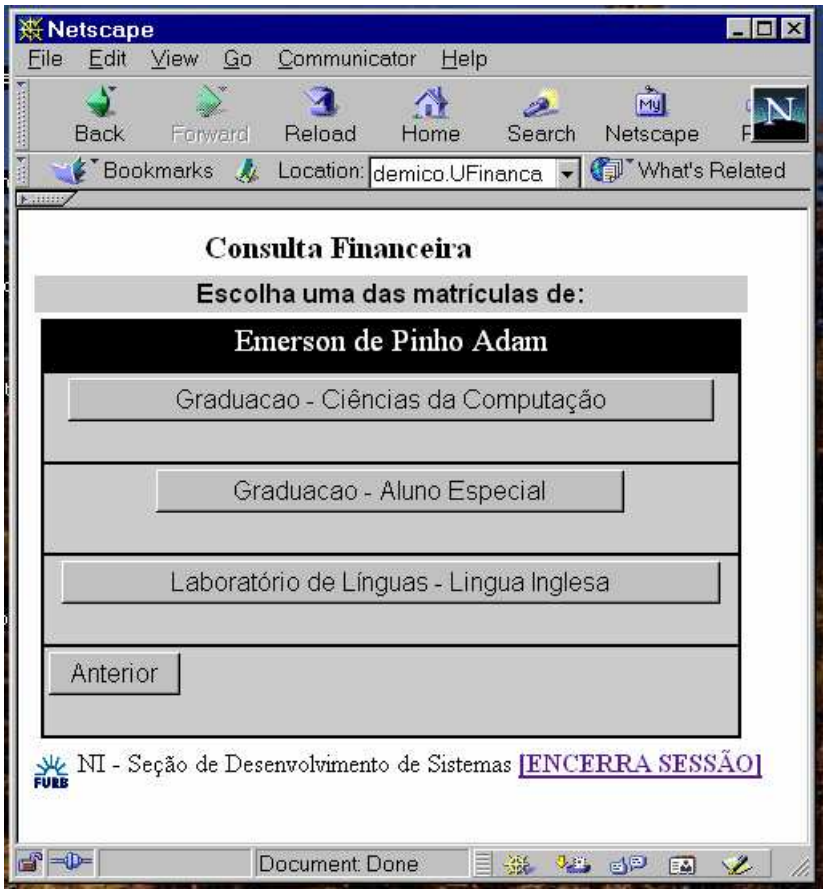
O *servlet* `UHistorico()` é quem chama o *servlet* `ConsaHistorico()`, que retorna a consulta de notas ou histórico desejada.

No anexo 3 é apresentado o código fonte em *Java* responsável pela exibição do resultado da consulta de histórico.

4.5.2.6 CONSULTA FINANCEIRA

Na figura 29 é apresentada a tela de solicitação de consulta financeira.

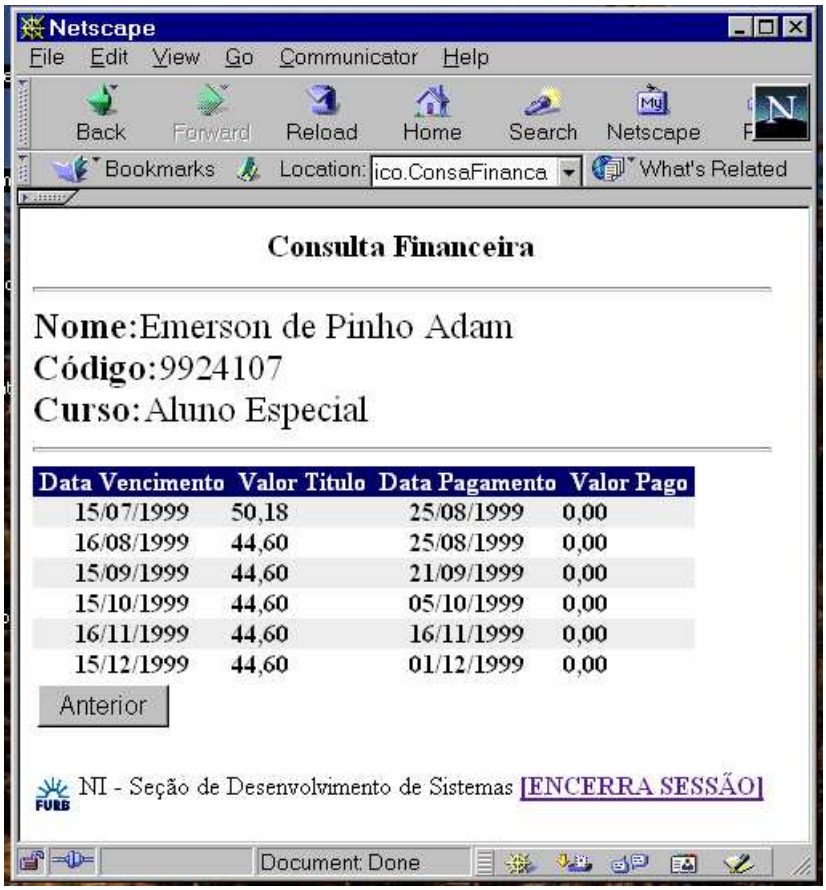
Figura 29 - Tela de solicitação de consulta financeira



O método `UFinanca()`que é um *servlet* dispara uma seqüência de *servlets* que ao final irá retornar o resultado desejado. Quando o usuário clicar no botão com a descrição da matrícula que deseja consultar, o *servlet* `UFinanca()`chama o próximo *servlet* que dará prosseguimento a consulta. A seqüência disparada pelo `UFinanca()`será vista a seguir.

Na figura 30 é apresentado uma tela com o resultado da consulta financeira.

Figura 30 – Tela do resultado da consulta financeira



O *servlet* `UFinanca()` é quem chama o *servlet* `ConsaFinanca()`, que retorna a consulta financeira desejada desejada.

No anexo 4 é apresentado o código fonte em *Java* responsável pela exibição do resultado da consulta financeira.

4.6 RESULTADOS E DISCUSSÃO

Na tabela 2 são apresentados alguns índices de comparação entre o protótipo desenvolvido e a aplicação atual que funciona em *PL/SQL*. Estes índices foram obtidos em simulações. Os valores referem-se a relação tempo de execução com a linguagem e tecnologia adotada. No entanto é difícil determinar com precisão esses números em função da quantidade de usuários que podem estar conectados em um momento específico.

Tabela 2 – Comparação de consultas PL/SQL X JAVA SERVLETS

Processo X tempo	**PL/SQL**	**JAVA SERVLETS**
Consulta de turmas	10 segundos	3 segundos
Consulta de horários	13 segundos	3 segundos
Consulta financeira	8 segundos	3 segundos
Consulta de histórico	14 segundos	3 segundos

O modelo inicial estava funcionando em um ambiente configurado com o servidor *Web* da *Oracle* numa máquina servidora acessando o banco de dados (*Oracle*), sendo as consultas implementadas em *PL/SQL*.

O protótipo desenvolvido neste trabalho está funcionando em um ambiente configurado com o servidor *Web APACHE* integrado com o *engine* de *servlets TOMCAT*, numa máquina servidora acessando o banco de dados (*Oracle*). As consultas do protótipo foram implementadas utilizando a tecnologia de *servlets*.

A obtenção dos valores de tempo de execução das consultas em *PL/SQL* deu-se através de acessos às consultas atuais cronometrando seu tempo de resposta. A obtenção dos valores de tempo de execução das consultas em *servlets* utilizando a proposta do protótipo, deu-se através de acessos às consultas do protótipo cronometrando seu tempo de resposta.

Apesar da dificuldade em precisar os índices de comparação, é perfeitamente visível que a resposta das consultas executadas em *servlets* foram no total em média 70% mais rápidas. Esse resultado induz a pensar que independentemente do número de usuários conectados ao servidor em um determinado momento, será de qualquer forma melhor do que os processos que rodam em *PL/SQL*.

5 CONCLUSÕES

Os objetivos deste trabalho foram atingidos uma vez que o protótipo para consultas acadêmicas foi desenvolvido utilizando a tecnologia de *servlets*. No que tange as expectativas secundárias, os requisitos foram plenamente satisfeitos, pois o protótipo pode ser acessado através da *internet* dispensando exigência de instalação de *plug-ins* e *drivers*, sendo necessário tão somente ao usuário, um navegador *Web*.

Java servlets demonstrou ser uma tecnologia adequada para a implementação de componentes que respondem pela lógica de apresentação e que sejam independentes de plataforma. Um aspecto não menos importante é o que diz respeito a reusabilidade do código gerado em *Java*, uma vez que em função disto, diversos outros serviços, como consulta de notas para laboratório de línguas e consulta de notas para o ensino médio (ETEVI) já estão sendo ofertados sem que se precisou escrever todo o código dessas consultas novamente.

Ao final deste trabalho observou-se que a proposta de utilizar a tecnologia de *servlets* para diversas consultas acadêmicas apresentou vantagens significativas sobre o modelo atual, principalmente no que tange ao desempenho de execução e portabilidade. No item portabilidade, o protótipo foi desenvolvido numa plataforma LINUX e posto em produção numa plataforma Windows NT sem que qualquer alteração fosse feita. Em relação ao desempenho, a análise de performance apresentada no tópico 4.6, que comparou o sistema atual com o protótipo desenvolvido, demonstrou que *servlets* possuem um tempo de resposta aceitável.

5.1 SUGESTÕES

Como sugestão para trabalhos futuros propõe-se migrar os sistemas atuais de consultas em *PL/SQL* que utilizam tecnologia proprietária para operarem com *servlets*. Não menos interessante seria um estudo mais abrangente a respeito da aplicação de metodologias para a especificação de *servlets*.

REFERÊNCIAS BIBLIOGRÁFICAS

BONIFÁCIO, José Maurício Di. As tecnologias por trás de uma aplicação web 3-camadas. **Developers' magazine**. Rio de Janeiro, v. , n. , p. 24–25, dez. 2000.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**: **guia do usuário**. Rio de Janeiro: Campus, 2000.

CORNELL, G., HORSTMANN, C. **Core Java**. São Paulo: Makron Books, 1997.

COLLA, E.C. : **Servlet, java do lado do servidor**. [2001?]. Disponível em: < http://www.insite.com.br/docs/develop-servlet95.html>. Acesso em: 10 out. 2001.

HALL, Marty. **Core servlets and Java server pages**. New Jersey: Prentice Hall PTR, 2000.

HUNTES, Jason. **Java Servlet Programming**. 2 ed. Sebastopol: O'Reilly & Associates, 2001.

IBM. **Getting Started: Visual Age for Java**. IBM, 2000.

JUBIN, Henri; FRIEDRICHS, Jürgen; TEAM, Jalapeño. **Enterprise javabeans by example**. New Jersey: Prentice Hall PTR, 1999.

LUCENA, Percival. **Tecnologias e arquiteturas para o desenvolvimento de aplicativos web**. São Paulo, [2000?]. Disponível em: <http://www.icmc.sc.usp.br/~percival/ download/aplicativosweb.pdf>. Acesso em: 18 nov. 2001.

LUPPI, Lupercio Fuganti. Sistemas distribuídos e a arquitetura em três camadas com middleware. **UNESC em revista**. Julho 1999.

NIEMEYER, Patrick; KNUDSEN, Jonathan. **Aprendendo JAVA**. Rio de Janeiro: Campus, 2000.

REESE, George. **Database programming with JDBC and Java**. 3 ed. Sebastopol: O'Reilly & Associates, 2000.

SILVA, Elaine Quintino da. **Agente gerenciador de cursos a distância via *Internet*.** São Carlos: Outubro, 2000. Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação, da Universidade de São Paulo - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional.

SOARES, Marinalva Dias. **Gerenciamento de versões de página *Web*.** São Carlos: Outubro, 2000. Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC, da Universidade de São Paulo - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional.

VISVESWARAN, Siva. **Connection pools: dive into connection pooling with J2EE.** , [2000]. Disponível em: <http://www.javaworld.com/javaworld/jw-10-2000/jw-1027-pool.html>. Acesso em: 01 abr. 2001.

ANEXOS

ANEXO 1 - Código fonte que exibe resultado da consulta de turmas.

```
       package br.furb.academico;

       import java.io.*;
       import javax.servlet.*;
       import javax.servlet.http.*;
       import javax.sql.*;
       import java.sql.*;
       import br.furb.sql.*;

       public class ConsaTurma extends HttpServlet {

         private PerfilUsuario perfil;


         public void destroy(){
         }


         public String getServletInfo(){
           return "br.furb.academico.ConsaTurma Information";
         }

         public void init(ServletConfig config) throws ServletException {
           super.init(config);
         }

         public void service(javax.servlet.http.HttpServletRequest req,
javax.servlet.http.HttpServletResponse res) throws ServletException, IOException
         {
           res.setContentType("text/html");
           PrintWriter out = res.getWriter();

           try {
             // Verifica se há sessão
             HttpSession sessao = req.getSession(false);
             if (sessao == null) {
                // pedir autenticacao e criar sessao
                Util.exibeLogon (res, req.getRequestURI(),"Consulta de Turmas");
                return;
             } else {
                // Chama escreveCabecalho
                Util.escreveCabecalho(res,"100%","Consulta de Turmas");
             }
             // busca perfil do usuario
             perfil = (PerfilUsuario)sessao.getAttribute("Perfil");
             //  Conecta ao banco de dados

             Connection conn =
br.furb.sql.SQLUtil.getConnection(br.furb.sql.SQLUtil.ACADEMICO);
             StringBuffer cd_turma_par=new StringBuffer(req.getParameter("cd_turma_par"));
             StringBuffer nm_turma_par=new StringBuffer(req.getParameter("nm_turma_par"));
             StringBuffer dt_anoati_par=new StringBuffer(req.getParameter("dt_anoati_par"));
             StringBuffer dt_semati_par=new StringBuffer(req.getParameter("dt_semati_par"));
             StringBuffer cd_aredis_par=new StringBuffer(req.getParameter("cd_aredis_par"));
             StringBuffer nr_discip_par=new StringBuffer(req.getParameter("nr_discip_par"));
             StringBuffer nr_seqdis_par=new StringBuffer(req.getParameter("nr_seqdis_par"));
             StringBuffer nr_turma_par=new StringBuffer(req.getParameter("nr_turma_par"));
             Statement stmt=null;
             // Query de consulta de professores
             StringBuffer consulta_prof = new StringBuffer("");
             consulta_prof.append(" select p.nm_pessoa nm_funcio ");
             consulta_prof.append(" from  r034fun fun, turma_professor_graduacao tur, pessoa
p ");
             consulta_prof.append(" where tur.nr_ano                     = "+dt_anoati_par+"
and ");
             consulta_prof.append("       tur.nr_semestre                = "+dt_semati_par+"
and ");
             consulta_prof.append("       tur.cd_area                    =
\'"+cd_aredis_par+"\' and ");
             consulta_prof.append("       tur.cd_numero_disciplina       = "+nr_discip_par+"
```

```
and ");
            consulta_prof.append("       tur.cd_sequencia_disciplina    = "+nr_seqdis_par+"
and ");
            consulta_prof.append("       tur.nr_turma                   = "+nr_turma_par+"
and ");
            consulta_prof.append("       fun.numcad                     = tur.cd_servidor
and ");
            consulta_prof.append("       fun.numemp                     = tur.numemp and
");
            consulta_prof.append("       fun.tipcol                     = tur.tipcol and
");
            consulta_prof.append("       p.cd_pessoa                    = fun.usu_codpes
");
            // Query de consulta de horarios da turma
            StringBuffer consulta_turma_horario = new StringBuffer("");
            consulta_turma_horario.append(" select
decode(to_char(nr_dia_semana),1,'Dom',2,'Seg',3,'Ter',4,'Qua',5,'Qui',6,'Sex',7,'Sáb')
nr_diasem, ");
            consulta_turma_horario.append(" to_char(nr_hora_inicio,'hh24:mi') nr_horini,
");
            consulta_turma_horario.append(" (tur.cd_horario_final - tur.cd_horario + 1)
nr_duracao, ");
            consulta_turma_horario.append("
decode(fl_concentrado,'S','(concentrado)','N',' ') fl_concen, ' sala: ' || cd_bloco ||' -'||
to_char(nr_sala,'000') nr_bloco");
            consulta_turma_horario.append(" from   turma_horario_graduacao tur,
horario_graduacao h ");
            consulta_turma_horario.append(" where tur.nr_ano                    =
"+dt_anoati_par+" and ");
            consulta_turma_horario.append("       tur.nr_semestre               =
"+dt_semati_par+" and ");
            consulta_turma_horario.append("       tur.cd_area                   =
\'"+cd_aredis_par+"\' and ");
            consulta_turma_horario.append("       tur.cd_numero_disciplina      =
"+nr_discip_par+" and ");
            consulta_turma_horario.append("       tur.cd_sequencia_disciplina   =
"+nr_seqdis_par+" and ");
            consulta_turma_horario.append("       tur.nr_turma                  =
"+nr_turma_par+"  and ");
            consulta_turma_horario.append("       h.cd_horario                  =
tur.cd_horario ");
            consulta_turma_horario.append(" order by nr_dia_semana, tur.cd_horario,
fl_concentrado ");
            // Query de consulta de alunos
            StringBuffer consulta_alu = new StringBuffer("");
            consulta_alu.append(" select p.nm_pessoa nm_aluno, alu.cd_vinculo cd_aluno,
v.dv_vinculo nr_digalu ");
            consulta_alu.append(" from  ultimo_historico_graduacao alu,
historico_disciplina_graduacao tur, ");
            consulta_alu.append("       vinculo v, pessoa p ");
            consulta_alu.append(" where tur.nr_ano                      = "+dt_anoati_par+"
and ");
            consulta_alu.append("       tur.nr_semestre                 = "+dt_semati_par+"
and ");
            consulta_alu.append("       tur.cd_area                     =
\'"+cd_aredis_par+"\' and ");
            consulta_alu.append("       tur.cd_numero_disciplina        = "+nr_discip_par+"
and ");
            consulta_alu.append("       tur.cd_sequencia_disciplina     = "+nr_seqdis_par+"
and ");
            consulta_alu.append("       tur.nr_turma                    = "+nr_turma_par+"
and ");
            consulta_alu.append("       tur.fl_cancelamento             != \'S\'
and");
            consulta_alu.append("       alu.cd_vinculo                  = tur.cd_vinculo
and ");
            consulta_alu.append("       v.cd_vinculo                    = tur.cd_vinculo
and ");
            consulta_alu.append("       p.cd_pessoa                     = v.cd_pessoa ");
            consulta_alu.append(" order by nm_pessoa ");
            StringBuffer consulta_curso = new StringBuffer("");
            consulta_curso.append(" select c.cd_curso cd_curso, nm_curso ");
            consulta_curso.append(" from  turma_curso_graduacao tc, curso_graduacao c ");
```

```
            consulta_curso.append(" where tc.nr_ano                     =
"+dt_anoati_par+"  and ");
            consulta_curso.append("       tc.nr_semestre                =
"+dt_semati_par+"  and ");
            consulta_curso.append("       tc.cd_area                    =
\'"+cd_aredis_par+"\'  and ");
            consulta_curso.append("       tc.cd_numero_disciplina       =
"+nr_discip_par+"  and ");
            consulta_curso.append("       tc.cd_sequencia_disciplina    =
"+nr_seqdis_par+"  and ");
            consulta_curso.append("       tc.nr_turma                   = "+nr_turma_par+"
and ");
            consulta_curso.append("       tc.cd_curso                   = c.cd_curso ");
            consulta_curso.append(" order by nm_curso ");
            ResultSet rset = null;

            stmt=conn.createStatement ();

            rset = stmt.executeQuery (consulta_prof.toString());

            // Informações da turma
            // Exibe o cd_turma e nm_turma
            out.println("<hr><font size=+1><b>Turma: </b>"+cd_turma_par+" -
"+nm_turma_par+"</font><hr>");
            // Informações do(s) professores
            out.println("<TABLE>");
            out.println("<TR NOWRAP>");
            out.println("<TD ALIGN=\"center\" NOWRAP><FONT SIZE=\"+1\"><B>Professor(es):
</B></FONT></TD></TR>");
            while (rset.next()) {
                out.println("<TR NOWRAP>");
                out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
                out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(1)+"</TD>");
                out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
                out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
                out.println("<TD ALIGN=\"left\" NOWRAP></TD>");
                out.println("</TR>");
            }
            out.println("</TABLE>");
            // Informações sobre os cursos

            rset = stmt.executeQuery (consulta_curso.toString());

            out.println("<HR>");
            out.println("<TABLE>");
            out.println("<TR NOWRAP>");
            out.println("<TD ALIGN=\"center\" NOWRAP><FONT SIZE=\"+1\"><B>Curso(s):
</B></FONT></TD></TR>");
            while (rset.next()) {
              out.println("<TR NOWRAP>");
              out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(1)+"</TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(2)+"</TD>");
              out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP></TD>");
              out.println("</TR>");
            }
            out.println("</TABLE>");
            // Informações sobre o sala e horários da turma

            rset = stmt.executeQuery (consulta_turma_horario.toString());

            out.println("<HR>");
            out.println("<TABLE>");
            out.println("<TR NOWRAP>");
            out.println("<TD ALIGN=\"center\" NOWRAP><FONT SIZE=\"+1\"><B>Horário(s):
</B></FONT></TD></TR>");
            while (rset.next()) {
              out.println("<TR NOWRAP>");
              out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(1)+"</TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(2)+"</TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(3)+"
```

```
"+rset.getString(4)+" "+rset.getString(5)+"</TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP></TD>");
              out.println("</TR>");
            }
            out.println("</TABLE>");
            out.println("<HR>");
            // Informações do(s) alunos

            rset = stmt.executeQuery (consulta_alu.toString());

            out.println("<TABLE>");
            out.println("<TR NOWRAP>");
            out.println("<TD ALIGN=\"center\" NOWRAP><FONT SIZE=\"+1\"><B>Aluno(s):
</B></FONT></TD></TR>");
            while (rset.next()) {
              out.println("<TR NOWRAP>");
              out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP>"+rset.getString(1)+"</TD>");
              out.println("<TD ALIGN=\"rigth\" NOWRAP>"+rset.getString(2)+"</TD>");
              out.println("<TD ALIGN=\"center\" NOWRAP></TD>");
              out.println("<TD ALIGN=\"left\" NOWRAP></TD>");
              out.println("</TR>");
            }
            out.println("</TABLE>");
            out.println("<HR>");
            rset.close();
            stmt.close();
          } catch (SQLException se) {
            System.out.println("Falhou!");
            out.println("<BR>Problemas em executar a consulta.");
            out.println("<BR>Erro:"+se.getMessage());
          }

          // Escreve mensagem
          out.println("<B>Aten&ccedil;&atilde;o: esta rela&ccedil;&atilde;o n&atilde;o deve
ser utilizada como lista de chamada. Em caso de d&uacute;vida procure a D.R.A.</B>");
          out.println("<P>");
          // Escreve o rodapé
          out.println("<table BORDER=0>");
          out.println("<tr>");
          out.println("<TD width=\"430\" valign=\"middle\" height=\"8\">");
          out.println("<FORM ACTION=\"javascript:history.go(-1)\" METHOD=\"POST\">");
          out.println("<INPUT type=\"submit\" value=\"Anterior\">");
          out.println("</FORM>");
          out.println("</TD>");
          out.println("</tr>");
          out.println("</table>");

          // Chama escreveRodape
          Util.escreveRodape(res);

          out.close();
        }
}
```

ANEXO 2 - Código fonte da consulta de horários.

```
       package br.furb.academico;
       import java.io.*;
       import javax.servlet.*;
       import javax.servlet.http.*;
       import javax.sql.*;
       import java.sql.*;
       import br.furb.sql.*;

       public class ConsaHorario extends HttpServlet
       {
         private PerfilUsuario perfil;

         public String colocaTabs(int qtd) {
           StringBuffer tb = new StringBuffer("");
           for (int i = 0; i < qtd; i++){
             tb.append('\t');
           }
           return tb.toString();
         }

         public void destroy()
         {
         }

         public String getServletInfo()
         { return " :P ";
         }

         public void init(ServletConfig config) throws ServletException
         {
          super.init(config);
         }

         public void service(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
         {
           res.setContentType("text/html");
           PrintWriter out = res.getWriter();
           Util.escreveCabecalho(res, "100%", "Consulta de Horários");
           StringBuffer nm_dis=new StringBuffer("");
           StringBuffer hor_ini = new StringBuffer("");
           int diasem, diasem_ant, diaanterior;
           diasem=diasem_ant = 2;
           StringBuffer duracao=new StringBuffer("");
           StringBuffer ds_bloco=new StringBuffer("");
           StringBuffer fl_con=new StringBuffer("");
           StringBuffer ds_sala=new StringBuffer("");
           String cd_aluno_par = req.getParameter("cd_aluno_par");
           String dt_anoati_par = req.getParameter("dt_anoati_par");
           String dt_semati_par = req.getParameter("dt_semati_par");
           String nm_aluno = req.getParameter("nm_nome_par");
           String nm_curso = req.getParameter("nm_curso_par");
           System.out.println("[ConsHorario] "+cd_aluno_par+"-"+dt_anoati_par+"-"+dt_semati_par+"-
"+nm_aluno+"-"+nm_curso);
           Statement stmt=null;
           StringBuffer consulta_his = new StringBuffer("");
           consulta_his.append(" select dis.nm_disciplina nm_discip, ");
           consulta_his.append("        nvl(to_char(h.nr_hora_inicio,'hh24:mi'),'*') nr_horini, ");
           consulta_his.append("        nvl(tuh.nr_dia_semana,2) nr_diasem,
nvl((tuh.cd_horario_final - tuh.cd_horario + 1),0) nr_duracao, ");
           consulta_his.append("        nvl(tuh.fl_concentrado,'*')  fl_concen,
nvl(tuh.cd_bloco,'*') nr_bloco, ");
           consulta_his.append("        nvl( to_char(tuh.nr_sala), '*') nr_sala, tuh.cd_area,
tuh.cd_numero_disciplina, tuh.cd_sequencia_disciplina, tuh.nr_turma, tg.dv_turma ");
           consulta_his.append(" from turma_horario_graduacao tuh, disciplina_graduacao dis, ");
           consulta_his.append("      historico_disciplina_graduacao hid, horario_graduacao h,
turma_graduacao tg ");
           consulta_his.append(" where ");
           consulta_his.append("      hid.nr_ano               = "+dt_anoati_par+" and ");
           consulta_his.append("      hid.nr_semestre          = "+dt_semati_par+" and ");
```

```
            consulta_his.append("      hid.cd_vinculo            = "+cd_aluno_par+"  and ");
            consulta_his.append("      dis.cd_area               = hid.cd_area   and ");
            consulta_his.append("      dis.cd_numero_disciplina  = hid.cd_numero_disciplina and ");
            consulta_his.append("      dis.cd_sequencia_disciplina = hid.cd_sequencia_disciplina and
");

            consulta_his.append("      tg.nr_ano                 = hid.nr_ano   and ");
            consulta_his.append("      tg.nr_semestre            = hid.nr_semestre and ");
            consulta_his.append("      tg.cd_area                = hid.cd_area   and ");
            consulta_his.append("      tg.cd_numero_disciplina     = hid.cd_numero_disciplina and
");
            consulta_his.append("      tg.cd_sequencia_disciplina  = hid.cd_sequencia_disciplina and
");
            consulta_his.append("      tg.nr_turma               = hid.nr_turma  and ");


            consulta_his.append("      tuh.nr_ano                (+) = hid.nr_ano   and ");
            consulta_his.append("      tuh.nr_semestre           (+) = hid.nr_semestre and ");
            consulta_his.append("      tuh.cd_area               (+) = hid.cd_area   and ");
            consulta_his.append("      tuh.cd_numero_disciplina  (+) = hid.cd_numero_disciplina
and ");
            consulta_his.append("      tuh.cd_sequencia_disciplina (+) = hid.cd_sequencia_disciplina
and ");
            consulta_his.append("      tuh.nr_turma              (+) = hid.nr_turma  and ");
            consulta_his.append("      hid.cd_area               != \'PDE\' and ");
            consulta_his.append("      h.cd_horario              (+) = tuh.cd_horario ");

            consulta_his.append(" order by tuh.cd_area, tuh.cd_numero_disciplina, ");
            consulta_his.append("          tuh.cd_sequencia_disciplina, tuh.fl_concentrado,");
            consulta_his.append("          tuh.nr_turma,  tuh.nr_dia_semana, ");
            consulta_his.append("          tuh.cd_horario ");
            ResultSet rset = null;
            boolean ap_concentrado=false;
            try {
              // Verifica se há conexão com o banco
              // Verifica se há sessão
              HttpSession sessao = req.getSession(false);
              if (sessao == null) {
                  // pedir autenticacao e criar sessao
                  Util.exibeLogon (res, req.getRequestURI(),"Consulta Horários");
                  return;
              }
              // busca perfil do usuario
              perfil = (PerfilUsuario)sessao.getAttribute("Perfil");
              //  Conecta ao banco de dados
              Connection conn = br.furb.sql.SQLUtil.getConnection(br.furb.sql.SQLUtil.ACADEMICO);
              stmt=conn.createStatement ();
              // Informações do horário
              // Consulta o histórico
              rset = stmt.executeQuery (consulta_his.toString());
              // Informações do aluno
              // Exibe o aluno e o curso
              out.println("<hr><font size=+1><b>Nome: </b>"+nm_aluno+"</font>");
              out.println("<br><font size=+1><b>Vínculo: </b>"+cd_aluno_par+" <b> Ano/Sem:
</b>"+dt_anoati_par+"/"+dt_semati_par+"</font>");
              out.println("<br><font size=+1><b>Curso: </b>"+nm_curso+"</font>");
              out.println("<br><hr WIDTH=\"100%\"><font color=\"#FFFFFF\"></font>");
              // Exibe o cabecalho e sub-cabecalho da grade de horario
              out.println("<table width=\"100%\" bordercolor=\"#993300\" cellspacing=\"0\"
cellpadding=\"0\" border=\"0\">");
              out.println("<tr bgcolor=\"#000066\" NOWRAP><!-- Cabecalho -->");
              out.println("<td width=\"150\"><font size=\"-1\" color=\"#FFFFFF\"><b>Nome da
disciplina</b></font></td>");
              out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\"><b>Segunda</b></font></td>");
              out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\"><b>Terça</b></font></td>");
              out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\"><b>Quarta</b></font></td>");
              out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\"><b>Quinta</b></font></td>");
              out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\"><b>Sexta</b></font></td>");
```

```
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\"><b>Sábado</b></font></td>");
             out.println("</tr>");
             out.println("<tr bgcolor=\"#000066\"  NOWRAP><!-- Sub-cabecalho -->");
             out.println("<td ALIGN=LEFT NOWRAP width=\"150\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;</font></td>");
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;&nbsp; <b> I   D   S </b> &nbsp;&nbsp;</font></td>");
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;&nbsp; <b> I   D   S </b> &nbsp;&nbsp;</font></td>");
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;&nbsp; <b> I   D   S </b> &nbsp;&nbsp;</font></td>");
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;&nbsp; <b> I   D   S </b> &nbsp;&nbsp;</font></td>");
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;&nbsp; <b> I   D   S </b> &nbsp;&nbsp;</font></td>");
             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font size=\"-1\"
color=\"#FFFFFF\">&nbsp;&nbsp; <b> I   D   S </b> &nbsp;&nbsp;</font></td>");
             out.println("</tr>");
             boolean sim = false;
             StringBuffer dis_ant=new StringBuffer("");
             StringBuffer cel=new StringBuffer("");
             boolean nome_ja_impresso = false;
             boolean primeiravez=false;
             boolean quebroulinha=false;
             boolean concentrado=false;
             while (rset.next()) {
                System.out.println(rset.getString(1)+" - "+rset.getString(2)+" -
"+rset.getString(3)+" - "+rset.getString(4)+" - "+rset.getString(5)+" - "+rset.getString(6)+" -
"+rset.getString(7));
                nm_dis = new StringBuffer(rset.getString(1));
                hor_ini = new StringBuffer(rset.getString(2));
                diaanterior=diasem;
                diasem = Integer.parseInt(rset.getString(3));
                duracao = new StringBuffer(rset.getString(4));
                ds_bloco = new StringBuffer(rset.getString(6));
                ds_sala = new StringBuffer(rset.getString(7));

                // prepara parametros para o link com a consulta de turmas
                StringBuffer cd_area = new StringBuffer(rset.getString(8));
                StringBuffer nr_disc = new StringBuffer(rset.getString(9));
                StringBuffer sq_disc = new StringBuffer(rset.getString(10));
                StringBuffer nr_turma = new StringBuffer(rset.getString(11));
                StringBuffer dv_turma = new StringBuffer(rset.getString(12));
                StringBuffer cd_disc = new
StringBuffer(cd_area.toString()+"."+nr_disc.toString()+"."+sq_disc.toString()+"."+nr_turma.toString
()+"-"+dv_turma.toString());
                if (rset.getString(5).equalsIgnoreCase("S")) {
                   fl_con = new StringBuffer("C");
                   concentrado = true;
                   ap_concentrado = true;
                } else {
                   fl_con = new StringBuffer("N");
                   concentrado = false;
                }
                if (concentrado||(diasem<diasem_ant)){
                   // Verifica se o concentrado ocorre dias anteriores ao atual
                   // Se sim pula linha(quebra) e ajusta o diaant para o dia atual
                   if ((diasem<diasem_ant)&&!quebroulinha) {
                       // Verifica se o concentrado mudou de nome
                       quebroulinha =  true;
                       if (!nm_dis.toString().equalsIgnoreCase(dis_ant.toString())) {
                           nome_ja_impresso=false;
                           quebroulinha=false;
                       } else {
                         // Quando quebra a linha
                         // completa o quadro com o fundo da cor
                         for (int i=diaanterior; i<7; i++){
                             out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font face=\"Arial
Narrow\" size=\"-1\" color=\"#000000\">&nbsp;</font></td>");
                         }
                         out.println("</tr>");
                         if (sim) out.println("<tr NOWRAP width=\"90\"><!-- Linha fundo branco --
```

```
><td ALIGN=LEFT NOWRAP><font face=\"Arial Narrow\" size=\"-1\"
color=\"#000000\"> </font></td>");
                        else out.println("<tr bgcolor=\"#E1EDFF\"  NOWRAP width=\"90\"><!-- Linha
fundo cinza --><td ALIGN=LEFT NOWRAP><font face=\"Arial Narrow\" size=\"-1\"
color=\"#000000\"> </font></td>");
                          //sim=!sim;
                        }
                        // Ajusta valores para a nova linha
                        diasem_ant = 2;
                     }
                  }
                  if (concentrado) {
                     cel = new StringBuffer("<b><font face=\"Arial Narrow\" size=\"-1\"
color=\"#FF0000\"> "+hor_ini.toString()+" "+duracao.toString()+" "+ds_bloco.toString
()+"-"+ds_sala.toString()+"</font></b>");
                  } else {
                     cel = new
StringBuffer(" "+hor_ini.toString()+" "+duracao.toString()+" "+ds_bloco.toString()+"
-"+ds_sala.toString());
                  }
                  // Imprime linha do gride fundo branco (sim/nao)
                  // verifica se o nome da disciplina ja foi impresso
                  if (nm_dis.toString().equalsIgnoreCase(dis_ant.toString())) {
                     nome_ja_impresso = true;
                     //sim = !sim;
                  } else {
                     if (primeiravez) {
                       // Caso ocorreu uma quebra no concentrado
                       // diasem_ant já foi atualizado
                       if (!quebroulinha) {
                         // Quando troca de nome
                         // completa o quadro com o fundo da cor
                         for (int i=diaanterior; i<7; i++){
                           out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font face=\"Arial
Narrow\" size=\"-1\" color=\"#000000\"> </font></td>");
                         }
                         nome_ja_impresso   = false;
                         diasem_ant = 2;
                         if (dis_ant.length()>2) {
                           out.println("</tr>");
                         }
                       }
                     } else primeiravez=true;
                     sim = !sim;
                  }
                  // Quando sim=true escreve com fundo branco
                  // quando sim=false escreve com fundo cinza
                  if (sim) {
                    // verifica se o nome da disciplina nao foi impresso
                    if (!nome_ja_impresso) {
                       out.println("<tr NOWRAP><!-- Linha fundo branco -->");
                       if (!quebroulinha) {
                         out.println("<td ALIGN=LEFT NOWRAP width=\"90\">");
                         out.println("<font face=\"Arial Narrow\" size=\"-1\" color=\"#000000\">");
                         out.println("<a href=\"br.furb.academico.ConsaTurmas");
                         out.println("?cd_turma_par="+cd_disc.toString());

out.println("&nm_turma_par="+Util.substituiAcentos(nm_dis.toString()).replace(' ','+'));
                         out.println("&dt_anoati_par="+dt_anoati_par);
                         out.println("&dt_semati_par="+dt_semati_par);
                         out.println("&cd_aredis_par="+cd_area);
                         out.println("&nr_discip_par="+nr_disc);
                         out.println("&nr_seqdis_par="+sq_disc);
                         out.println("&nr_turma_par="+nr_turma+"\">");
                         out.println(nm_dis);
                         out.println("</a></font></td>");
                       }
                    }
                    // procura o dia para  imprimir o horario
                    for (int i = diasem_ant; i <= 7; i++){
                      if ((diasem==i)) {
                         out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><b><font face=\"Arial
Narrow\" size=\"-1\" color=\"#000000\">"+cel.toString()+"</font></b></td>");
```

```
                    i=8;
                  } else out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><b><font face=\"Arial
Narrow\" size=\"-1\" color=\"#000000\"> </font></b></td>");
                }
              } else {
                // verifica se o nome da disciplina nao foi impresso
                if (!nome_ja_impresso) {
                 out.println("<tr bgcolor=\"#E1EDFF\"  NOWRAP><!-- Linha fundo cinza -->");
                 if (!quebroulinha) {
                   out.println("<td ALIGN=LEFT NOWRAP width=\"150\">");
                   out.println("<font face=\"Arial Narrow\" size=\"-1\" color=\"#000000\">");
                   out.println("<a href=\"br.furb.academico.ConsaTurmas");
                   out.println("?cd_turma_par="+cd_disc.toString());

out.println("&nm_turma_par="+Util.substituiAcentos(nm_dis.toString()).replace(' ','+'));
                   out.println("&dt_anoati_par="+dt_anoati_par);
                   out.println("&dt_semati_par="+dt_semati_par);
                   out.println("&cd_aredis_par="+cd_area);
                   out.println("&nr_discip_par="+nr_disc);
                   out.println("&nr_seqdis_par="+sq_disc);
                   out.println("&nr_turma_par="+nr_turma+"\">");
                   out.println(nm_dis);
                   out.println("</a></font></td>");
                   }
                }
                // procura o dia para  imprimir o horario
                for (int i = diasem_ant; i <= 7; i++){
                  if ((diasem==i)) {
                      out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><b><font face=\"Arial
Narrow\" size=\"-1\" color=\"#000000\">"+cel.toString()+"</font></b></td>");
                      i=8;
                  } else out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><b><font face=\"Arial
Narrow\" size=\"-1\" color=\"#000000\"> </font></b></td>");
                  }
                }
                quebroulinha=false;
                //sim = !sim;
                dis_ant = new StringBuffer(nm_dis.toString());
                diasem_ant = diasem+1;
              }
              rset.close();
              stmt.close();
              conn.close();
            } catch (SQLException se) {
              System.out.println("Ok!");
              out.println("<BR>Problemas em executar a consulta.");
              out.println("<BR>Erro:"+se.getMessage());
            }
            // completa o quadro com o fundo da cor
            for (int i=diasem_ant; i<=7; i++){
                out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font face=\"Arial Narrow\"
size=\"-1\" color=\"#000000\"> </font></td>");
            }
            out.println("</tr>");
            out.println("</table>");
            out.println("<table width=\"100%\" bordercolor=\"#993300\" cellspacing=\"0\"
cellpadding=\"0\" border=\"0\">");
            out.println("<tr NOWRAP><td ALIGN=LEFT NOWRAP><font face=\"Arial Narrow\" size=\"-1\"
color=\"#000000\"> </font></td></tr>");
            out.println("<tr BGCOLOR=\"#000066\"  NOWRAP><!-- Rodapé e legenda -->");
            if (ap_concentrado)       out.println("<td ALIGN=LEFT NOWRAP width=\"450\"><font
face=\"Arial Narrow\" size=\"-1\" color=\"#FFFFFF\"> LEGENDA: <b> I:</b> início da aula -
<b>D:</b> duração da aula - <b> S:</b> sala </font></b><b><font size=\"-1\" color=\"#FF0000\"> -
Vermelho:</font></b><font size=\"-1\" color=\"#FFFFFF\"> concentrado</font></td>");
            else out.println("<td ALIGN=LEFT NOWRAP width=\"450\"><font face=\"Arial Narrow\"
size=\"-1\" color=\"#FFFFFF\"> LEGENDA: <b> I:</b> início da aula - <b> D:</b> duração da aula
- <b> S:</b> sala </font></b></td>");
            // Quando termina
            // completa o quadro com o fundo da cor
            for (int i=6; i<=7; i++){
              out.println("<td ALIGN=CENTER NOWRAP width=\"90\"><font face=\"Arial Narrow\" size=\"-
1\" color=\"#000000\"> </font></td>");
            }
```

```
            out.println("</tr>");
            out.println("</table>");
            // Escreve o rodapé
            out.println("<table BORDER=0>");
            out.println("<tr>");
            out.println("<TD width=\"430\" valign=\"middle\" height=\"8\">");
            out.println("<FORM ACTION=\"javascript:history.go(-1)\" METHOD=\"POST\">");
            out.println("<INPUT type=\"submit\" value=\"Anterior\">");
            out.println("</FORM>");
            out.println("</TD>");
            out.println("</tr>");
            out.println("</table>");

            Util.escreveRodape(res);
            out.close();
          }
}
```

ANEXO 3 - Código fonte da consulta de histórico

```
package br.furb.academico;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.sql.*;
import java.sql.*;
import br.furb.sql.*;



public class ConsaHistorico extends HttpServlet
{
  private PerfilUsuario perfil;


  public void destroy() {
  }

  public String getServletInfo()
  { return " :P ";
  }

  public void init(ServletConfig config) throws ServletException
  {
         super.init(config);
  }

  public   void   service(HttpServletRequest   req,   HttpServletResponse   res)   throws
ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    StringBuffer anoEsem=new StringBuffer("");
    StringBuffer nt_semestre=new StringBuffer("");
    StringBuffer nt_exame=new StringBuffer("");
    StringBuffer nt_media=new StringBuffer("");
    StringBuffer ds_observacao=new StringBuffer("");
    StringBuffer nm_disciplina=new StringBuffer("");
    String cd_aluno_par = req.getParameter("vinculo");
    String nm_aluno = req.getParameter("nome");
    String nm_curso =req.getParameter("curso");
    String nm_grad = req.getParameter("graduacao");
    System.out.println("[ConsaHistorico_service()]
("+cd_aluno_par+")("+nm_aluno+")("+nm_curso+")("+nm_grad+")");
    Statement stmt=null;
    StringBuffer consulta_his = new StringBuffer("");
    if (nm_grad.trim().startsWith("Labor")) {
       consulta_his.append(" select hid.nr_ano dt_anoati, hid.nr_semestre dt_semati, ");
       consulta_his.append(" dis.nm_disciplina, ");
       consulta_his.append(" nvl(to_char(hid.nr_nota_semestre,'90D90'),' ') nr_notasem,
");
       consulta_his.append("   nvl(to_char(hid.nr_nota_exame,'90D90'),' ')   nr_notaexa,
");
       consulta_his.append("   nvl(to_char(hid.nr_media_final,'90D90'),' ')   nr_medfin,
");
       consulta_his.append(" nvl(obc.ds_observacao,' ') ds_observ ");
       consulta_his.append(" from  historico_disciplina_lblg hid, ");
       consulta_his.append(" observacao_nota_lblg obc, tipo_situacao_lblg tsl, ");
       consulta_his.append(" disciplina_lblg dis ");
       consulta_his.append(" where hid.cd_vinculo  = "+cd_aluno_par+" and ");
       consulta_his.append(" hid.cd_situacao = tsl.cd_situacao and ");
       consulta_his.append(" tsl.tp_situacao = 'I' and ");
       consulta_his.append(" dis.cd_area = hid.cd_area and ");
       consulta_his.append(" dis.nr_disciplina = hid.nr_disciplina and ");
       consulta_his.append(" dis.nr_sequencia_disciplina = ");
       consulta_his.append(" hid.nr_sequencia_disciplina and ");
       consulta_his.append(" obc.cd_observacao = hid.cd_observacao ");
       consulta_his.append(" order by hid.nr_ano desc, hid.nr_semestre desc, ");
```

```
              consulta_his.append(" dis.nm_disciplina ");
            } else {
              consulta_his.append(" select hid.nr_ano dt_anoati, hid.nr_semestre dt_semati, ");
              consulta_his.append("                                                    dis.nm_disciplina,
nvl(to_char(hid.nr_nota_semestre,\'90D90\'),\' \') nr_notasem, ");
              consulta_his.append(" nvl(to_char(hid.nr_nota_exame,\'90D90\'),\' \') nr_notaexa,
");
              consulta_his.append(" nvl(to_char(hid.nr_media_final,\'90D90\'),\' \') nr_medfin,
nvl(obc.ds_observacao,\' \') ds_observ ");
              consulta_his.append(" from  historico_disciplina_graduacao hid, ");
              consulta_his.append(" observacao_nota_graduacao obc, disciplina_graduacao dis ");
              consulta_his.append(" where hid.cd_vinculo  = "+cd_aluno_par+" and ");
              consulta_his.append(" hid.fl_cancelamento != \'S\' and ");
              consulta_his.append(" dis.cd_area = hid.cd_area and ");
              consulta_his.append(" dis.cd_numero_disciplina = hid.cd_numero_disciplina and ");
              consulta_his.append(" dis.cd_sequencia_disciplina = ");
              consulta_his.append(" hid.cd_sequencia_disciplina and ");
              consulta_his.append(" obc.cd_observacao = hid.cd_observacao ");
              consulta_his.append(" order by hid.nr_ano desc, hid.nr_semestre desc, ");
              consulta_his.append(" dis.nm_disciplina ");
            }
            ResultSet rset = null;
            try {
              // Verifica se há sessão
              HttpSession sessao = req.getSession(false);
              if (sessao == null) {
                  // pedir autenticacao e criar sessao
                  Util.exibeLogon (res, req.getRequestURI(),"Consulta Histórico");
                  return;
              }
              // Chama escreveCabecalho
              Util.escreveCabecalho(res,800,"Consulta Histórico");

              // busca perfil do usuario
              perfil = (PerfilUsuario)sessao.getAttribute("Perfil");
              //  Conecta ao banco de dados
              Connection conn = br.furb.sql.SQLUtil.getConnection(br.furb.sql.SQLUtil.ACADEMICO);
              stmt=conn.createStatement ();
              rset = stmt.executeQuery (consulta_his.toString());
              // Informações do aluno
              // Exibe o aluno e o curso
              out.println("<hr><font size=+1><b>Nome:</b>"+nm_aluno+"</font>");
              out.println("<br><font size=+1><b>Vínculo:</b>"+cd_aluno_par+"</font>");
              out.println("<br><font size=+1><b>Curso:</b>"+nm_curso+"</font>");
              out.println("<br><hr WIDTH=\"100%\"><font color=\"#FFFFFF\"></font>");
              // Escreve a legenda
              out.println("<br>Legenda: <img    SRC=\"/images/ap.gif\"    height=20    width=26
align=ABSCENTER>");
              out.println("<font    size=-2>Aprovado <img    SRC=\"/images/rp.gif\"    height=20
width=26 align=ABSCENTER>");
              out.println("Reprovado <img     SRC=\"/images/rf.gif\"     height=20     width=26
align=ABSCENTER> Reprovado");
              out.println("por  frequ&ecirc;ncia <img SRC=\"/images/ex.gif\" height=20 width=26
align=ABSCENTER>");
              out.println("Em exame</font>");

out.println("<br>             &nbs
p; <img SRC=\"/images/pnn.gif\" height=20 width=26 align=ABSCENTER><font size=-2>");
              out.println("Professor     n&atilde;o     entregou     nota    <img
SRC=\"/images/di.gif\" height=20 width=26 align=ABSCENTER>");
              out.println("Dispensado</font>");
              // Exibe o cabecalho do historico
              out.println("<table BORDER=0 CELLSPACING=0 CELLPADDING=0 >");
              out.println("<tr BGCOLOR=\"#000066\" NOWRAP>");
              out.println("<td    ALIGN=CENTER    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1>ano/sem</font></font></b></td>");
              out.println("<td    ALIGN=LEFT    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1>  Disciplina</font></font></b></td>");
              out.println("<td   ALIGN=CENTER   NOWRAP><b><font   color=\"#FFFFFF\"><font   size=-1>n.
semestre</font></font></b></td>");
              out.println("<td   ALIGN=CENTER   NOWRAP><b><font   color=\"#FFFFFF\"><font   size=-1>n.
exame</font></font></b></td>");
              out.println("<td    ALIGN=CENTER    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
```

```
1>  M&eacute;dia </font></font></b></td>");
              out.println("<td    ALIGN=LEFT    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1>obs.</font></font></b></td>");
              out.println("</tr>");
              // Consulta o histórico
              StringBuffer m_ano=new StringBuffer("");
              StringBuffer m_sem=new StringBuffer("");
              StringBuffer mn_legenda = new StringBuffer("");
              int ano, sem, ano_ant, sem_ant;
              ano=sem=ano_ant=sem_ant=0;
              boolean sim = false;
              while (rset.next()) {
                 m_ano = new StringBuffer(rset.getString(1));
                 m_sem = new StringBuffer(rset.getString(2));
                 nm_disciplina = new StringBuffer(rset.getString(3));
                 nt_semestre = new StringBuffer(rset.getString(4));
                 nt_exame = new StringBuffer(rset.getString(5));
                 nt_media = new StringBuffer(rset.getString(6));
                 ds_observacao = new StringBuffer(rset.getString(7));
                 sem = Integer.parseInt(m_sem.toString());
                 anoEsem=new StringBuffer(m_ano+"/"+m_sem);
                 ano = Integer.parseInt(m_ano.toString());
                 if   (ds_observacao.toString().trim().startsWith("Aprovado"))   mn_legenda   =   new
StringBuffer("/images/ap.gif");
                 if   (ds_observacao.toString().trim().startsWith("Reprovado"))   mn_legenda   =   new
StringBuffer("/images/rp.gif");
                 if    (ds_observacao.toString().trim().indexOf("por")>0)    mn_legenda    =    new
StringBuffer("/images/rf.gif");
                 if    (ds_observacao.toString().trim().startsWith("Em"))    mn_legenda    =    new
StringBuffer("/images/ex.gif");
                 if    (ds_observacao.toString().trim().startsWith("P"))    mn_legenda    =    new
StringBuffer("/images/pnn.gif");
                 if    (ds_observacao.toString().trim().startsWith("Dis"))    mn_legenda    =    new
StringBuffer("/images/di.gif");
                 if ((ano!=ano_ant) || (sem!=sem_ant)) {
                   ano_ant=ano;
                   sem_ant=sem;
                   sim=!sim;
                 }
                 if (sim) {
                   // Exibe linha da tabela com fundo destacado
                   out.println("<tr BGCOLOR=\"#EEEEEE\" NOWRAP>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+anoEsem+"</font></b></td>");
                   out.println("<td           ALIGN=LEFT            NOWRAP><b><font           size=-
1>"+nm_disciplina+"</font></b></td>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+nt_semestre+"</font></b></td>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+nt_exame+"</font></b></td>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+nt_media+"</font></b></td>");
                   out.println("<td   ALIGN=LEFT   NOWRAP><img   SRC=\""+mn_legenda+"\"   height=20
width=26></td></tr>");
                   out.println("</tr>");
                 } else {
                   // Exibe linha da tabela com fundo normal
                   out.println("<tr NOWRAP>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+anoEsem+"</font></b></td>");
                   out.println("<td           ALIGN=LEFT            NOWRAP><b><font           size=-
1>"+nm_disciplina+"</font></b></td>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+nt_semestre+"</font></b></td>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+nt_exame+"</font></b></td>");
                   out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+nt_media+"</font></b></td>");
                   out.println("<td   ALIGN=LEFT   NOWRAP><img   SRC=\""+mn_legenda+"\"   height=20
width=26></td></tr>");
                   out.println("</tr>");
                 }
              }
```

```
              rset.close();
              stmt.close();
              conn.close();
            } catch (SQLException se) {
              out.println("<BR>Problemas em executar a consulta. Tente novamente mais tarde.");
              System.out.println("Falhou");
              System.out.println("[ConsaHistorico_service()] ERRO: "+se);
              System.out.println("[ConsaHistorico_service()]     : "+se.getMessage());
            }
            out.println("</table>");
            // Escreve Legenda
            out.println("<br>Legenda: <img     SRC=\"/images/ap.gif\"     height=20     width=26
align=ABSCENTER>");
            out.println("<font  size=-2>Aprovado <img  SRC=\"/images/rp.gif\"  height=20  width=26
align=ABSCENTER>");
            out.println("Reprovado <img      SRC=\"/images/rf.gif\"      height=20      width=26
align=ABSCENTER> Reprovado");
            out.println("por  frequ&ecirc;ncia <img  SRC=\"/images/ex.gif\"  height=20  width=26
align=ABSCENTER>");
            out.println("Em exame</font>");

out.println("<br>             &nbs
p; <img SRC=\"/images/pnn.gif\" height=20 width=26 align=ABSCENTER><font size=-2>");
            out.println("Professor      n&atilde;o      entregou      nota    <img
SRC=\"/images/di.gif\" height=20 width=26 align=ABSCENTER>");
            out.println("Dispensado</font>");
            // Escreve o rodapé
            out.println("<table BORDER=0>");
            out.println("<tr>");
            out.println("<TD width=\"430\" valign=\"middle\" height=\"8\">");
            out.println("<FORM ACTION=\"javascript:history.go(-1)\" METHOD=\"POST\">");
            out.println("<INPUT type=\"submit\" value=\"Anterior\">");
            out.println("</FORM>");
            out.println("</TD>");
            out.println("</tr>");
            out.println("</table>");

            Util.escreveRodape(res);

            out.close();
          }
}
```

ANEXO 4 - Código fonte da consulta financeira

```
package br.furb.academico;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.sql.*;
import java.sql.*;
import br.furb.sql.*;

public class ConsaFinanca extends javax.servlet.http.HttpServlet implements SingleThreadModel
{
  private PerfilUsuario perfil;

  public void destroy() {
  }

  public String getServletInfo()
  { return " :P ";
  }

  public void init(ServletConfig config) throws ServletException
  {
         super.init(config);
  }

  public    void    service(HttpServletRequest    req,    HttpServletResponse    res)    throws
ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    Util.escreveCabecalho(res,"100%", "Consulta Financeira");

    StringBuffer anoEsem=new StringBuffer("");
    StringBuffer nt_semestre=new StringBuffer("");
    StringBuffer nt_exame=new StringBuffer("");
    StringBuffer nt_media=new StringBuffer("");
    StringBuffer ds_observacao=new StringBuffer("");
    StringBuffer nm_disciplina=new StringBuffer("");
    String cd_aluno_par = req.getParameter("vinculo");
    String nm_aluno = req.getParameter("nome");
    String nm_curso =req.getParameter("curso");
    System.out.println("[ConsFinanca_service()]
("+cd_aluno_par+")("+nm_aluno+")("+nm_curso+")");
    Statement stmt=null;
    StringBuffer consulta_fin = new StringBuffer("");
    consulta_fin.append("       select       nvl(to_char(dt_vencimento,'DD/MM/YYYY'),'       '),
nvl(to_char(dt_pagamento,'DD/MM/YYYY'),' '), ");
    consulta_fin.append("           to_char(sum(vl_bruto),'999G999G990D00')          vl_titulo,
to_char(sum(nvl(vl_cobrado,0)),'999G999G990D00') vl_pago ");
    consulta_fin.append(" from titulo_receita ");
    consulta_fin.append(" where nr_bloquete in (select nr_bloquete ");
    consulta_fin.append(" from titulo_receita_graduacao tit, titulo_receita tr ");
    consulta_fin.append(" where tit.cd_vinculo_historico  = "+cd_aluno_par);
    consulta_fin.append(" and tr.nr_titulo          = tit.nr_titulo) ");
    consulta_fin.append(" group by dt_vencimento, dt_pagamento ");
    consulta_fin.append(" order by dt_vencimento ");
    ResultSet rset = null;
    try {
        // Verifica se há sessão
        HttpSession sessao = req.getSession(false);
        if (sessao == null) {
          // pedir autenticacao e criar sessao
          Util.exibeLogon (res, req.getRequestURI(),"Consulta Financeira");
          return;
        }
        // busca perfil do usuario
        perfil = (PerfilUsuario)sessao.getAttribute("Perfil");
        //  Conecta ao banco de dados
        Connection conn = br.furb.sql.SQLUtil.getConnection(br.furb.sql.SQLUtil.ACADEMICO);
```

```
        stmt = conn.createStatement();
       // Informações do histórico
       rset = stmt.executeQuery (consulta_fin.toString());
        // Informações do aluno
        // Exibe o aluno e o curso
        out.println("<hr><font size=+1><b>Nome:</b>"+nm_aluno+"</font>");
        out.println("<br><font size=+1><b>C&oacute;digo:</b>"+cd_aluno_par+"</font>");
        out.println("<br><font size=+1><b>Curso:</b>"+nm_curso+"</font>");
        out.println("<br><hr WIDTH=\"100%\"><font color=\"#FFFFFF\"></font>");
        // Exibe o cabecalho da situacao financeira
        out.println("<table BORDER=0 CELLSPACING=0 CELLPADDING=0 >");
        out.println("<tr BGCOLOR=\"#000066\" NOWRAP>");
        out.println("<td    ALIGN=CENTER    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1> Data Vencimento </font></font></b></td>");
        out.println("<td    ALIGN=LEFT    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1> Valor Titulo </font></font></b></td>");
        out.println("<td    ALIGN=CENTER    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1> Data Pagamento </font></font></b></td>");
        out.println("<td    ALIGN=CENTER    NOWRAP><b><font    color=\"#FFFFFF\"><font    size=-
1> Valor Pago </font></font></b></td>");
        out.println("</tr>");
        // Consulta o histórico
        StringBuffer dt_pagto=new StringBuffer("");
        StringBuffer dt_vencto=new StringBuffer("");
        StringBuffer vl_bruto = new StringBuffer("");
        StringBuffer vl_cobrado = new StringBuffer("");
        java.text.SimpleDateFormat formatter = new java.text.SimpleDateFormat ("dd/MM/yyyy");
        java.util.Date sysdate   = new java.util.Date();
        java.util.Date dtv = new java.util.Date();
        boolean sim = false;
        while (rset.next()) {
           dt_vencto = new StringBuffer(rset.getString(1));
           dt_pagto = new StringBuffer(rset.getString(2));
           vl_bruto = new StringBuffer(rset.getString(3));
           vl_cobrado = new StringBuffer(rset.getString(4));
           /* Verifica se titulo esta em debito, acusando na
            data de pagamento */
           dtv = formatter.parse(dt_vencto.toString());
           if ((dtv.before(sysdate)) && (dt_pagto.length()<2)) {
               dt_pagto = new StringBuffer("Débito");
           }
           sim=!sim;
           if (sim) {
              // Exibe linha da tabela com fundo destacado
             out.println("<tr BGCOLOR=\"#EEEEEE\" NOWRAP>");
             out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+dt_vencto+"</font></b></td>");
             out.println("<td           ALIGN=rigth           NOWRAP><b><font           size=-
1>"+vl_bruto+"</font></b></td>");
             out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+dt_pagto+"</font></b></td>");
             out.println("<td           ALIGN=rigth           NOWRAP><b><font           size=-
1>"+vl_cobrado+"</font></b></td>");
             out.println("</tr>");
           } else {
             // Exibe linha da tabela com fundo normal
             out.println("<tr NOWRAP>");
             out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+dt_vencto+"</font></b></td>");
             out.println("<td           ALIGN=rigth           NOWRAP><b><font           size=-
1>"+vl_bruto+"</font></b></td>");
             out.println("<td           ALIGN=CENTER           NOWRAP><b><font           size=-
1>"+dt_pagto+"</font></b></td>");
             out.println("<td           ALIGN=rigth           NOWRAP><b><font           size=-
1>"+vl_cobrado+"</font></b></td>");
             out.println("</tr>");
           }
        }
        rset.close();
        stmt.close();
        conn.close();
    } catch (SQLException se) {
        out.println("<BR>Problemas em executar a consulta. Tente novamente mais tarde.");
```

```
        System.out.println("Falhou!");
        System.out.println("[ConsFinanca_service()] Erro:"+se);
        System.out.println("[ConsFinanca_service()]      "+se.getMessage());
    } catch (java.text.ParseException pe){
        System.out.println("[ConsFinanca_service()] Erro:"+pe);
        System.out.println("[ConsFinanca_service()]      "+pe.getMessage());
    }
    out.println("</table>");
    // Escreve o rodapé
    out.println("<table BORDER=0>");
    out.println("<tr>");
    out.println("<TD width=\"430\" valign=\"middle\" height=\"8\">");
    out.println("<FORM ACTION=\"javascript:history.go(-1)\" METHOD=\"POST\">");
    out.println("<INPUT type=\"submit\" value=\"Anterior\">");
    out.println("</FORM>");
    out.println("</TD>");
    out.println("</tr>");
    out.println("</table>");

    Util.escreveRodape(res);

    out.close();
  }
}
```