Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

ELTON DOS SANTOS ANDRÉ

APLICATIVO WEB NA ÁREA DE EDUCAÇÃO COM ACESSO MÓVEL

Assis
2010

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

ELTON DOS SANTOS ANDRÉ

APLICATIVO WEB NA ÁREA DE EDUCAÇÃO COM ACESSO MÓVEL

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de Curso de Graduação.

Orientadora: Profa. Dra. Marisa Atsuko Nitto

Área de Concentração: Informática

Assis
2010

FICHA CATALOGRÁFICA

ANDRÉ, Elton dos Santos.
Aplicativo Web com Acesso Móvel na Área de Educação / Elton dos Santos André. Fundação Educacional do Município de Assis – FEMA – Assis, 2010.
59p.

Orientadora: Prof.ª Dr.ª Marisa Atsuko Nitto.
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Web. 2. Móbile. 3. Internet.

CDD: 001.6
Biblioteca da FEMA

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

APLICATIVO WEB NA ÁREA DE EDUCAÇÃO COM ACESSO MÓVEL

ELTON DOS SANTOS ANDRÉ

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientadora: Profa. Dra. Marisa Atsuko Nitto

Analisador: Prof. Dr. Almir Rogério Camolesi

Assis
2010

DEDICATÓRIA

Dedico este trabalho ao meu pai Pedro André, a minha mãe Luciana dos Santos a minha avó Malvina dos Santos e aos meus irmãos Anderson Luís S. André Letícia Santos André e a minha namorada Larissa Nogueira da Silva.

AGRADECIMENTOS

Agradeço, em especial, a Deus por toda a força e paz espiritual ao longo de toda minha caminhada e no decorrer do desenvolvimento deste projeto, muito obrigado Senhor;

A meus pais pela força e dedicação por tornar possível mais uma conquista e pela ajuda por todos esses anos de estudo;

A Profa. Dra. Marisa Atsuko Nitto minha amiga e orientadora pela paciência e dedicação na orientação para comigo e a este trabalho, muito obrigado;

Aos meus lideres de trabalho do Departamento Municipal de Educação de Paraguaçu Paulista em especial a Suzana Vieira Lopes e Maria de Fátima Gaspar pela compreensão nas necessidades focadas a esse trabalho;

A todos os meus colegas de Curso que de alguma forma deram força para a elaboração deste trabalho;

A todos os Professores do Curso de Ciência da Computação da FEMA pelos ensinamentos e dedicação para tornar possível o desenvolvimento deste trabalho;

A Fundação Educacional do Município de Assis – FEMA, por proporcionar o conhecimento e ter me capacitado para desenvolver este trabalho;

A Larissa Nogueira da Silva minha namorada e amiga na compreensão da importância deste trabalho;

Aos meus amigos, pelo apoio, amizade e demonstração de companheirismo;

A todos que direta ou indiretamente, contribuíram para a realização deste trabalho.

RESUMO

O objetivo deste trabalho é desenvolver um aplicativo *web* com acesso a relatórios no móbile. Este aplicativo desfruta das tecnologias *web* e para dispositivos móveis que estão cada vez mais presentes no cotidiano do humano. Este trabalho estará focalizado à área de educação onde usuários externos terão acesso às informações através de acesso web e a relatórios por dispositivos móveis que serão disponibilizadas através de um aplicativo web que será alimentado pelo administrador do aplicativo com o fim de mostrar informações em que os usuários externos têm acesso e privilégios.

Palavras Chaves: Web, Móbile, Internet.

ABSTRACT

The objective is to develop a web application with access to reports on mobile. This application enjoys web technologies and mobile devices that are increasingly present in everyday human. This work will be focused on the area of education where external users have access to information through web access and reports for mobile devices which are available through a web application that will be powered by the administrator of the application with information in order to show that external users have access and privileges.

Keywords: Web, Mobile, Internet.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 – INTRODUÇÃO

Cada vez mais a humanidade vem buscando meios mais fáceis, ágeis de se ter informações de assuntos de seus interesses. A *Web* e as tecnologias móveis vêm trazendo uma forma rápida e automática de disponibilizar esses dados para o usuário por meio da internet. Essas informações são disponíveis tendo em vista que o usuário já possua os dados de acesso para essas informações disponíveis via Aplicação web. A Aplicação para *Web* é a tecnologia ideal para comunicação de sistemas móbile. A comunicação das aplicações é facilitada pela sua padronização de dados não se prendendo a plataformas e linguagens de programação (JAVA FREE, 2010).

A linguagem de programação que será utilizada é Java, pois apresenta uma série de características que a torna muito atrativa para o desenvolvimento de aplicações destinadas a dispositivos móveis (DEXTRA, 2010). Essas características abrangem a orientação a objetos, independência da plataforma, robustez, segurança, confiabilidade e conectividade com a Web (DEITEL, 2005). Isto proporciona o desenvolvimento de projetos com maior qualidade, portabilidade e com menor tempo de desenvolvimento. As ferramentas que serão utilizadas no desenvolvimento do aplicativo e IDE é *Netbeans* 6.9.1 com os Frameworks JSF (*Javaserver Faces*), JSP (*Javaserver Pages*), Java SDK versão 1.5, Apache Tomcat versão 6.0, XML (*Extensible Markup Language*), AJAX, a ferramenta de mapeamento objeto-relacional *Hibernate* junto com o banco de dados HSQLDB e iReport versão 3.0 para a geração de relatórios. A vantagem da utilização destas ferramentas é que elas são *open source*, e isso é muito importante com relação ao custo-benefício de implantação (HENDRICKS, 2002). Certamente o mercado com mais oportunidades em que o Java está presente é o de *Web*. Trabalhando com Java as empresas fica mais independente dos fabricantes de vários softwares do *servlet* container, do banco de dados e até da própria fabricante da sua Máquina Virtual (INFOWESTER, 2010). A área de educação ainda tem se utilizado pouco dessas tecnologias, e pretende se desenvolver neste projeto um aplicativo que possa auxiliar e agilizar os serviços prestados neste segmento.

1.1 – OBJETIVOS

O objetivo principal é apresentar uma arquitetura de um sistema para a área de educação através da aplicação de dispositivos móveis e *Web* utilizando as tecnologias Java para desenvolvimento *Web*. Será desenvolvida uma arquitetura modular que permite integrar diversas plataformas de desenvolvimento, que pode ser facilmente estendida ou mesmo adequada a sistemas já existentes. Uma das vantagens da adoção deste sistema é a facilidade de troca de informações.

1.2 – JUSTIFICATIVAS

Com o forte avanço da conexão com a internet e mobilidade dos dispositivos é um dos grandes focos nos dia de hoje e a necessidade de novas aplicações em cima desses avanços se tornam necessárias. A tendência do mercado é investir cada vez mais em aplicativos para dispositivos móveis, abrindo um amplo mercado de trabalho para desenvolvedores com conhecimento dessa tecnologia.

1.3 – MOTIVAÇÃO

As motivações em matéria de benefícios relacionados ao tema são diversas. Entre elas, destacam-se a disponibilidade de ferramentas, que possibilitam o desenvolvimento deste trabalho, tendo em vista o fascinante mundo da web e de suas aplicações e da computação móvel que estão no topo nos dias de hoje. A aplicação desta tecnologia apresenta um ganho de qualidade e agilidade aos serviços prestados. Tudo que se referem os dispositivos móveis evolui a passos largos. E é essa evolução que abre caminho para aplicativos mais poderosos e tem aumentado dia-a-dia a presença desses dispositivos em nossas vidas. E na área educacional ainda não se tem utilizado muito as vantagens dessas tecnologias.

1.4 – ESTRUTURAS DO TRABALHO

O trabalho está organizado em quatro capítulos onde será descrito a estrutura do trabalho seguir.

O capítulo um apresenta uma introdução e descreve os objetivos, justificativas e as motivações para o desenvolvimento do trabalho.

O capítulo dois aborda toda fundamentação teórica necessária para compreender os conceitos específicos para execução do trabalho onde são abordados assuntos das tecnologias usadas.

No capítulo três é referente ao desenvolvimento do trabalho onde é apresentada a modelagem do problema bem como a especificação, por meio de diagramas, da implementação do sistema desenvolvido para uma melhor compreensão do que será trabalhado.

Por fim, no capítulo quatro são apresentadas as conclusões do trabalho.

2 – FUNDAMENTAÇÃO TEÓRICA BASICA

Neste capítulo será feita uma descrição de toda fundamentação teórica que será usada para o desenvolvimento do aplicativo.

2.1 - LINGUAGEM JAVA

Java é uma linguagem de programação orientada a objetos, desenvolvida pela equipe na Sun Microsystems. Inicialmente elaborada para ser a linguagem-base de projetos de software para produtos eletrônicos, Java explodiu em 1995, com o grande sucesso mundial da *World Wide Web*. Java é uma linguagem de alto nível, com sintaxe similar à linguagem do C++, e com diversas características herdadas de outras linguagens. E antes de tudo uma linguagem simples, fortemente tipada, independente de arquitetura, robusta, segura, extensível, bem estruturada, distribuída. (RICARTE, 2001). A figura 1 mostra o processo de compilação Java.

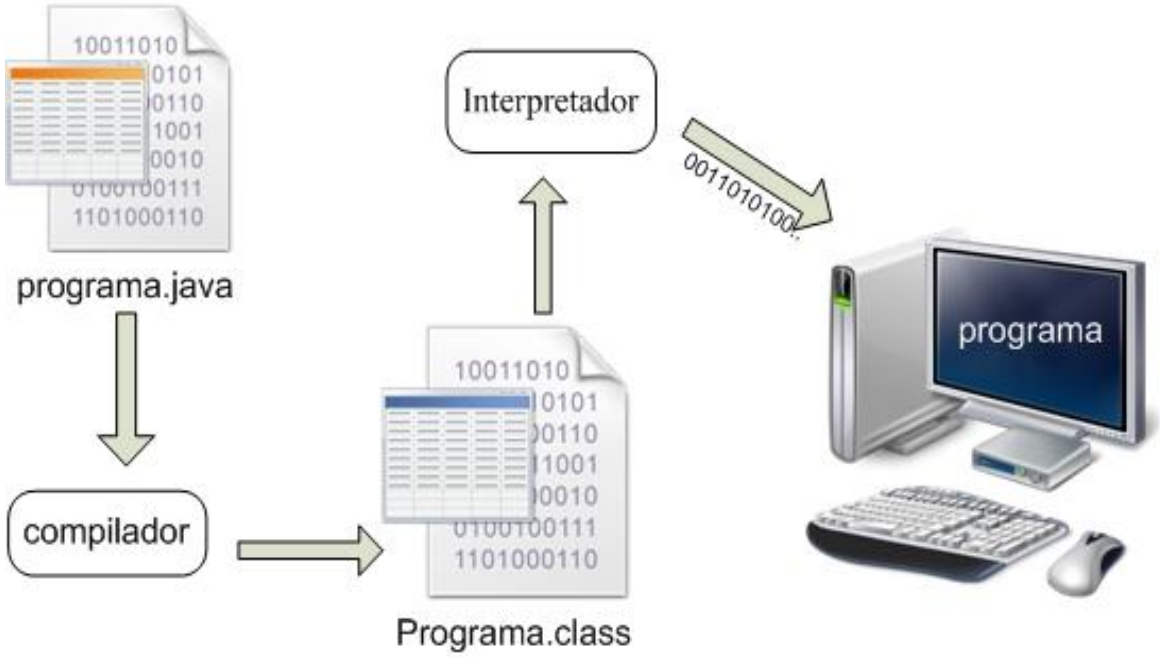

Figura 1: Processo de compilação Java (RICARTE, 2001).

2.1.1 – Ambiente Java

O ambiente de desenvolvimento de software Java, Java SDK (*Software Development Kit*) antigamente, denominado JDK, é formada principalmente pelas classes fundamentais da linguagem Java e um conjunto de aplicações que, entre outras atribuições, realizar a compilação e execução de programas escritos em Java. Há um ambiente de desenvolvimento integrado, oferecendo não editores ou ambientes de programação visual. No entanto, são as suas características que permitem a operação desses ambientes.

O Java SDK contém um amplo conjunto de APIs que compõem o núcleo de funcionalidades da linguagem Java. Uma API (*Application Programming Interface*) é uma biblioteca composta de código pré-compilado, pronto para ser utilizado no desenvolvimento de suas aplicações. A figura 2 mostra o processo de compilação no ambiente Java.

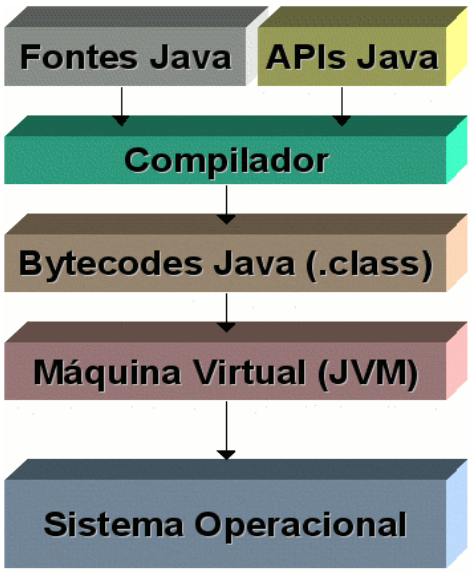

Figura 2: Processo de compilação no Ambiente Java (RICARTE, 2001).

2.1.2 – Ferramenta Java SDK

As ferramentas essenciais do ambiente de desenvolvimento de *software* Java são o compilador Java, *javac* o interpretador de aplicações Java e o interpretador de *applets* Java, *appletviewer*. Um programa fonte em Java pode ser desenvolvido usando qualquer editor que permita gravar textos sem caracteres de formatação. Um arquivo contendo código Java constitui uma unidade de compilação, podendo incluir comentários, declaração relacionadas a pacotes e pelo menos uma definição de classe ou interface. O resultado dessa execução, se o programa fonte estiver sem erros, será a criação de um arquivo .class contendo o *bytecode* que poderá ser executado em qualquer máquina. Segundo (RICARTE, 2001), além das ferramentas essenciais, o Java SDK oferece os aplicativos de desenvolvimento *javadoc*, um gerador de documentação para programas Java, jar, um manipulador de arquivos comprimidos no formato *Java Archive*, que opera juntamente com *extcheck*, o verificador de arquivos nesse formato jdb, um depurador de programas *Java javap*, um *disassembler* de classes Java e *javah*, um gerador de arquivos *header* para integração a código nativo em C.

Java oferece também aplicativos para o desenvolvimento e execução de aplicações Java em plataformas de objetos distribuídos. Há também ferramentas para permitir o desenvolvimento de aplicações distribuídas, incorporando também o conceito de assinaturas digitais. A ferramenta *keytool* gerencia chaves e certificados; *jarsigner* gera e verifica assinaturas associadas a arquivos Java; e *policytoo*l é uma interface gráfica para gerenciar arquivos que determinam a política de segurança do ambiente de execução (RICARTE, 2001). A figura 3 mostra as ferramentas disponíveis no Java SDK.

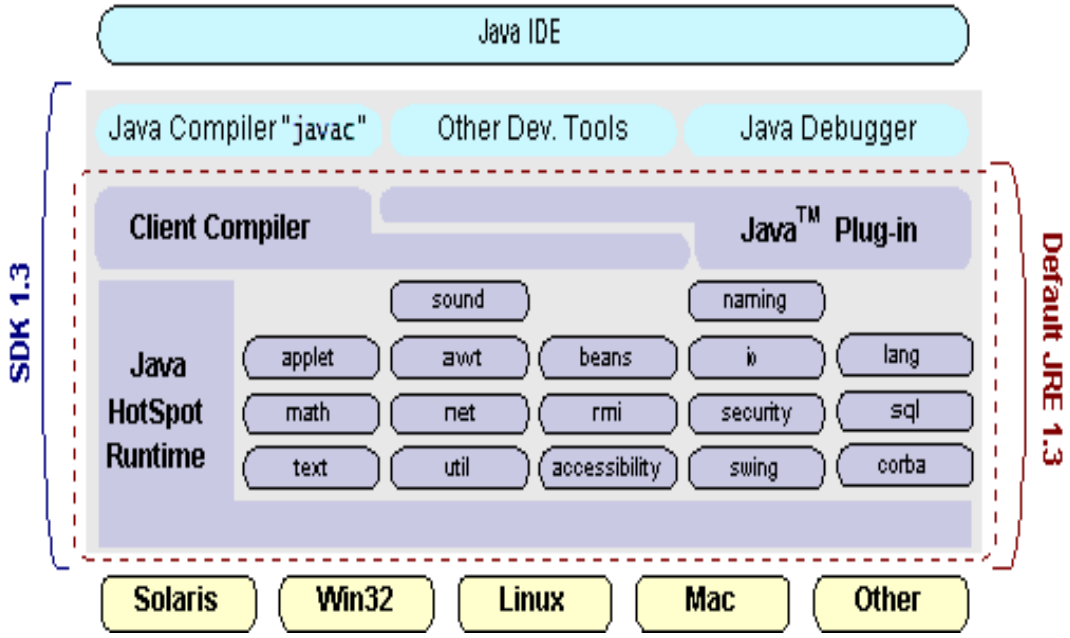

Figura 3: Ferramentas de Java SDK (RICARTE, 2001).

2.1.3 – Java 2 *Standard Edition*

É a tecnologia Java para computadores pessoais, *notebooks* e arquiteturas com poder de processamento e memória consideráveis. Várias APIs acompanham esta versão e tantas outras podem ser baixadas opcionalmente no site da Sun. Com elas que a maioria das aplicações é construída e executada. A figura 4, mostra a plataforma do Java 2 *Platform Standard Edition*. Segundo (RICARTE, 2001), o J2SE possui duas divisões:

- ***Java Runtime Edition*** **(JRE)**: uma versão mais leve da JDK, pois é preparada para o ambiente de execução, ou seja, é esta versão que executará os sistemas construídos com a SDK;
- ***Java Development Kit*** **(JDK) ou** ***Standard Development Kit*** **(SDK)**: um conjunto para desenvolvimento em Java e deveria ser instalado apenas pelos desenvolvedores por possuir ferramentas para tal tarefa.

2.1.4 – Principais Características da Linguagem Java

As principais características da linguagem Java serão descritas para facilitar o entendimento da programação.

- **Programação dinâmica e extensível**: a programação Java é baseada integralmente em POO (Programação Orientada á Objetos), ou seja, é baseada na descrição de classes. Para conceitos novos e tarefas novas criam-se novas classes ou, realizam-se a manutenção ou incrementam-se classes antigas;
- ***Write once, run everywhere***: escreva uma vez e execute em qualquer lugar. Este é o motor de Java, o que impulsionou a linguagem, a característica de compatibilidade do código Java com qualquer *hardware*. A promessa que navegadores *web*, telefones, PDAs e outros aparelhos eletrônicos pudessem executar um mesmo código fonte Java ajudou a popularizar a linguagem e atrair novos programadores;
- ***Web programming*:** a Sun fez um prognóstico arrojado para a época afirmando que a *web* era um computador logo, deveriam existir ferramentas para "programar" este novo dispositivo, a rede. A arquitetura cliente/servidor é algo quase que natural em Java;
- **Eficiência na codificação**: com a utilização de classes, a programação torna-se mais eficiente já que as chances de reescrever *software* inédito tornam-se menor, ou seja, sempre existirá uma classe que faz aquilo que o programador deveria implementar;
- **Recursos de Rede**: possui extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP;
- **Segurança**: permite que usuários utilizem *software*s seguros que foram copiados da *web*;
- **Internacionalização**: é o poder de escrever código fonte que trabalhe com caracteres dos mais diferentes alfabetos. Este foi um dos últimos recursos, dos citados aqui, a ser implementado em Java;

- **Carga Dinâmica de Código**: programas em Java são formados por uma coleção de classes armazenadas independentemente e que podem ser carregadas no momento de utilização;
- **Carga dinâmica**: as classes são armazenadas independentemente e carregadas somente quando são necessárias;
- **Concorrência**: permite o uso de threads múltiplos e mecanismos explícitos de concorrência;
- **Tratamento de eventos e exceções**: permite a programação orientada por eventos e a possibilidade de escrever programas que respondem a falhas de execução.

2.1.5 – Tecnologia J2ME

Os principais componentes da Plataforma Java 2, *Micro Edition* (plataforma J2ME) são o CDC (*Connected Device Configurations* - Configurações para dispositivos conectados), o CLDC (*Connected Limited Device Configurations* - Configurações para dispositivos com conexão limitada), o MIDP (*Mobile Information Device Profiles* - Perfis de informações de dispositivos móveis), além de muitas outras ferramentas e tecnologias que levam as soluções Java aos mercados de consumo e de dispositivos integrados.

A tecnologia Java ME foi originalmente criada a fim de lidar com as limitações associadas a criação de aplicativos para pequenos dispositivos. Com esse objetivo a Sun redefiniu os princípios para a tecnologia Java ME para caber em um ambiente tão limitado e tornar possível a criação de aplicativos Java rodando em pequenos dispositivos.

As tecnologias J2ME contêm um JRE altamente otimizado, especialmente desenvolvido para o mercado de grande consumo. Essas tecnologias abrangem uma ampla gama de produtos muito pequenos e habilitam programas utilitários úteis, de segurança e conectividade em *smart cards*, *pagers*, conversores de sinal digital (*set-top boxes*) e outros aparelhos de pequeno porte. As plataformas Java relacionadas incluem a Plataforma Java 2 *Standard Edition* (plataforma J2SE) e a

Plataforma Java 2 *Enterprise Edition* (plataforma J2EE) (JAVA, 2010). A figura 4 mostra a visão geral da tecnologia JME.

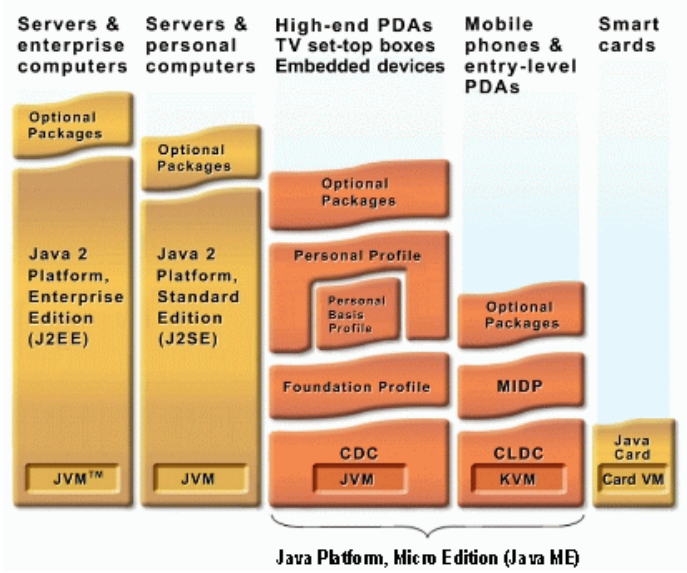

Figura 4: Visão geral da tecnologia JME (JAVA, 2010).

2.1.6 – Java Server Pages

A Tecnologia JSP é um componente chave na plataforma Java 2 *Enterprise Edition*, que é uma arquitetura altamente escalonavel da *Sun* para aplicações empresariais. Uma página JSP é uma página que possui uma estrutura fixa somada com algum tipo de linguagem de marcação para incluir outro tipo de texto ou lógica embarcada. Esta estrutura fixa normalmente é toda baseada em HTML. A linguagem de marcação que tem como função gerar algum tipo de conteúdo dinâmico pode apresentar-se das seguintes formas: scripts ou *tags* customizadas (MASDEVAL, 2007). As páginas JSP partilham as características da tecnologia do Java "*Write Once, Run Anywhere*". JSP é um acrônimo para *Java Server Pages* e consiste numa linguagem de *script* baseada em Java para criação de sites com conteúdos dinâmicos. Inicialmente as páginas na Web eram apenas páginas estáticas, isto é, seu conteúdo não variava a cada solicitação conforme algum parâmetro. Com a sofisticação dos serviços disponibilizados via Web, surgiu à necessidade de

disponibilizar informações com natureza dinâmica (lista de preços atualizados, compras on-line, etc). Isso exigiu que o servidor web fizesse algum processamento adicional da solicitação a fim de gerar uma resposta personalizada (SILVEIRA ET AL., 2009).

2.1.6.1 – *Common Gateway Interface* (CGI)

O CGI foi primeiro padrão para conteúdo da Web dinâmico. Ele é um protocolo de comunicação que o servidor HTTP utiliza para conversação com outro programa. Um CGI script é qualquer programa (PERL, C, Java) que se comunica com o servidor *Web* através do protocolo CGI. Um *script* CGI (como ficou conhecido este mecanismo) é um programa que atende às requisições enviadas por um cliente e a ele repassadas pelo servidor HTTP (SILVEIRA ET AL., 2009). A figura 5 mostra o processo do servidor para executar programa de CGI.

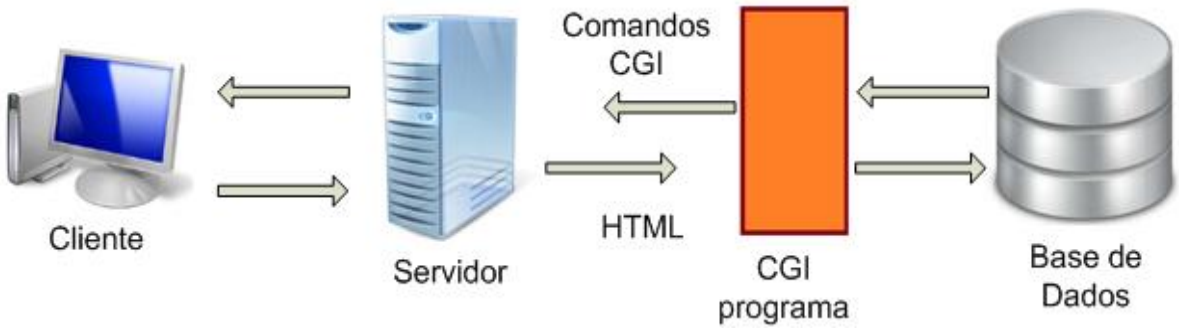

Figura 5 – Processo do servidor para executar programa de CGI. (SILVEIRA ET AL., 2009).

2.1.6.2 – Aplicações na Web

A utilização de *Servlets* e de páginas JSP oferece diversas vantagens em relação ao SO de outras tecnologias (PHP, ASP e CGI). As principais vantagens são herdadas da própria linguagem Java: Portabilidade: a aplicação desenvolvida pode ser implantada em diversas plataformas, como por exemplo, Windows, Unix e

Macintosh, sem que seja necessário modificar ou mesmo reconstruir a aplicação. Facilidade de programação: a programação é orientada a objetos, simplificando o desenvolvimento de sistemas complexos. Além disso, a linguagem oferece algumas facilidades, como por exemplo, o gerenciamento automático de memória (estruturas alocadas são automaticamente liberadas, sem que o desenvolvedor precise se preocupar em gerenciar esse processo).

Flexibilidade: o Java já se encontra bastante difundido, contando com uma enorme comunidade de desenvolvedores, ampla documentação e diversas bibliotecas e códigos prontos, dos quais o desenvolvedor pode usufruir. Além dessas vantagens, a arquitetura de *Servlets* e páginas JSP possibilitam alguns benefícios adicionais: Escalabilidade: na maior parte dos servidores de aplicações modernos, é possível distribuir a carga de processamento de aplicações desenvolvidas em diversos servidores, sendo que servidores podem ser adicionados ou removidos de maneira a acompanhar o aumento ou decréscimo dessa carga de processamento (MASDEVAL, 2007). A figura 6 mostra a arquitetura distribuída com *servlets* e páginas JSP.

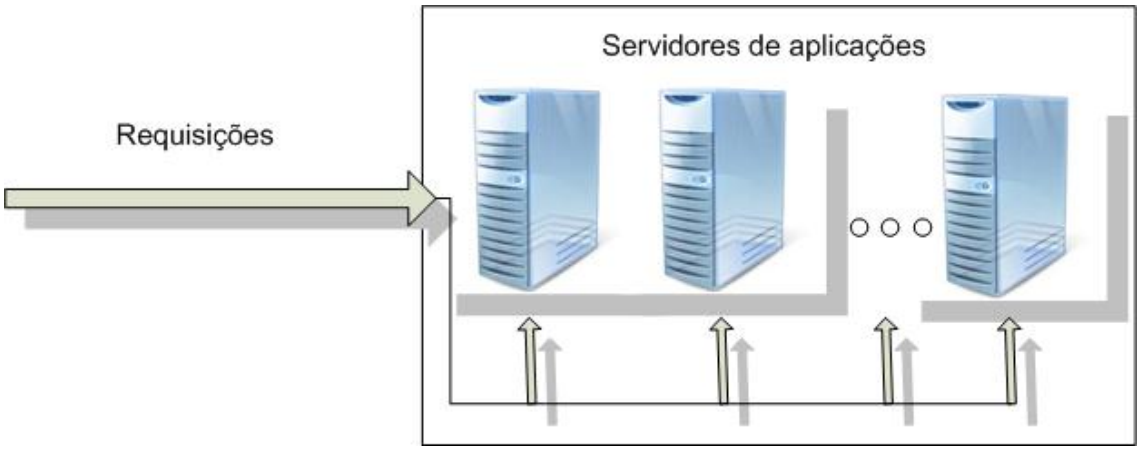

Figura 6 – Arquitetura distribuída com *Servlets* e Páginas JSP. (MASDEVAL, 2007).

2.1.6.3 – *Servlets*

Servlets são classes Java, desenvolvidas de acordo com uma estrutura bem definida, e que, quando instaladas junto a um Servidor que implementa um *Servlet Container* (um servidor que permita a execução de *Servlets*, muitas vezes chamado de Servidor de Aplicações Java), podem tratar requisições recebidas de clientes (TEMPLE ET AL., 2004). A figura 7 mostra o funcionamento de uma aplicação com *servlets*.

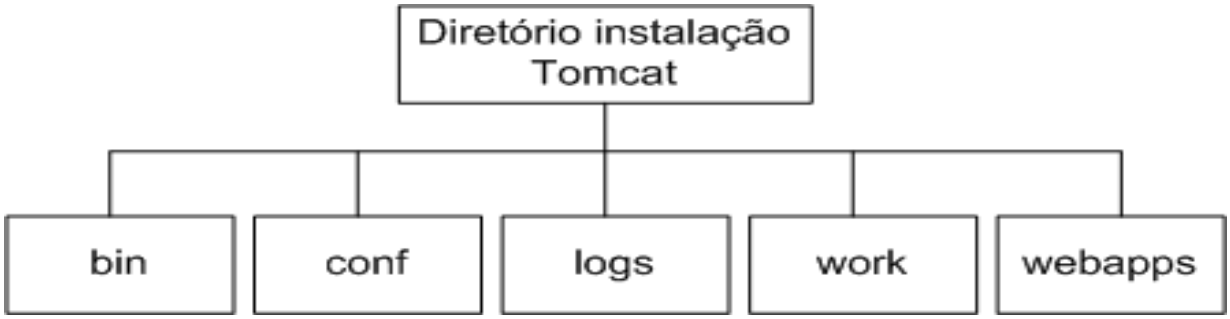

Figura 7 - Funcionamento de uma aplicação com *Servlets* (TEMPLE ET AL., 2004).

Ao receber uma requisição, um *Servlet* pode capturar parâmetros desta requisição, efetuar qualquer processamento inerente a uma classe Java, e devolver uma página HTML por exemplo. *Servlets* são alternativa Java para CGI Scripts e assim possuem todas as vantagens da plataforma à sua disposição como: API's, multiplataforma, *multithreading* e orientação a objetos. Sua desvantagem é que tanto o conteúdo estático quanto o dinâmico residem no código fonte do programa, o que acarreta sérios problemas de manutenção. JSP é a combinação de HTML com Java dentro de uma mesma página (como ASP e PHP), porém, usando-se *tags* especiais do tipo HTML que interagem com objetos Java no servidor, podemos introduzir conteúdo dinâmico em qualquer parte da página sem necessidade que código Java bruto apareça. Em princípio, todo o código fora dos *tags* é HTML puro. Isso possibilita gerar um código mais manutenível, pois permite a separação do código que se destina a apresentação (HTML) (TEMPLE ET AL., 2004).

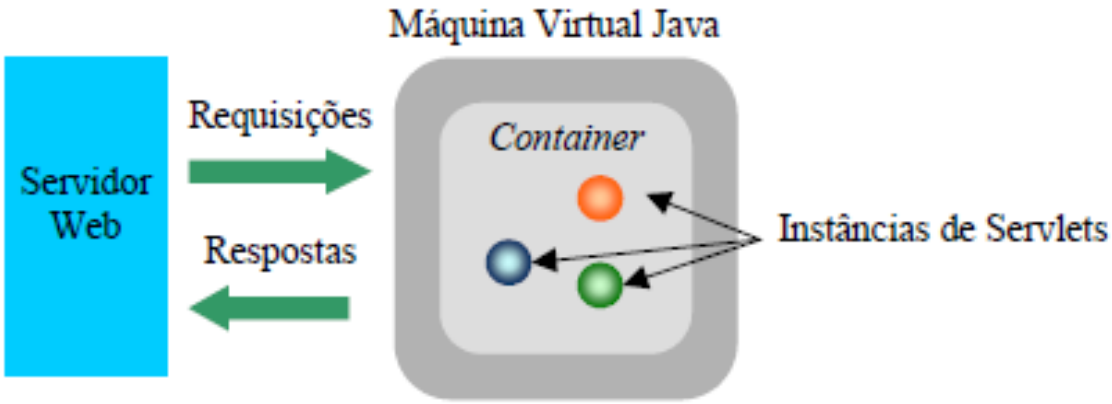

Figura 8 - Relacionamento entre *servlets*, *container* e *servidor web* (TEMPLE ET AL., 2004).

2.1.6.4 – Arquitetura de um *Servlet*

Todos os *servlets* implementam direta ou indiretamente a interface *Servlet*. O mais comum é o *servlet* dar *extends* na *HttpServlet* (que implementa a interface *Servlet*). interface *Servlet* fornece métodos para gerenciamento do *servlet* e sua comunicação com clientes. Quando um *servlet* aceita uma chamada do cliente, ele recebe dois objetos:

- ***ServletRequest* e *ServletResponse***

A classe *ServletRequest* encapsula a comunicação do cliente com o servidor, enquanto a *ServletResponse* encapsula a comunicação do servidor com o cliente. A *ServletRequest* permite que o *servlet* acesse informações como os nomes dos parâmetros passados pelo cliente, o protocolo usado pelo mesmo e o nome do host que fez o chamado ao servidor. Ela também permite que o *servlet* acesse um *inputstream*. O *Servlet InputStream* através do qual o *servlet* recebe dados do cliente. As subclasses da *ServletRequest* permitem ao *servlet* obter dados mais específicos, como informações do cabeçalho http. A *ServletResponse* fornece ao *servlet* métodos para responder ao cliente. Ela permite que o *servlet* defina o

tamanho do conteúdo e seu *mime type*, fornece uma *OutputStream* - a *ServletOutputStream* - e também um *Writer*, através dos quais o *servlet* poderá enviar respostas ao cliente. As subclasses da *ServletResponse* fornecem ao *servlet* mais capacidades específicas em relação ao protocolo, como manipular o cabeçalho HTTP da resposta (TEMPLE ET AL., 2004.).

2.2 – XML (EXTENSIBLE MARKUP LANGUAGE)

Extensible Markup Language (XML) é linguagem de marcação de dados que provê um formato para descrever dados estruturados. Isso facilita declarações mais precisas do conteúdo e resultados mais significativos de busca através de múltiplas plataformas. O XML também vai permitir o surgimento de uma nova geração de aplicações de manipulação e visualização de dados via internet. O XML permite a definição de um número infinito de *tags*. Enquanto no HTML, se as tags podem ser usadas para definir a formatação de caracteres e parágrafos, o XML provê um sistema para criar *tags* para dados estruturados. Um elemento XML pode ter dados declarados como sendo preços de venda, taxas de preço, um título de livro, a quantidade de chuva, ou qualquer outro tipo de elemento de dado. Como as *tags* XML são adotadas por intranets de organizações, e também via Internet, haverá uma correspondente habilidade em manipular e procurar por dados independentemente das aplicações onde os quais são encontrados. Uma vez que o dado foi encontrado, ele pode ser distribuído pela rede e apresentado em um *browser* como o Internet Explorer 5 de várias formas possíveis, ou então esse dado pode ser transferido para outras aplicações para processamento futuro e visualização (W3SCHOOLS, 2010).

2.2.1 – O Protocolo HTTP

Embora *Servlets* possam ser utilizados não só para o desenvolvimento de aplicações HTTP, a maior parte das aplicações desenvolvidas é destinada a esse

fim. Sendo assim, vale a pena estudar um pouco mais a fundo o funcionamento e características desse protocolo. O protocolo HTTP é utilizado na navegação nas páginas da Internet: quando você abre uma janela de um browser, acessa uma página Web e navega em seus links, você está, na verdade, utilizando esse protocolo para visualizar, em sua máquina, o conteúdo que está armazenado em servidores remotos.

O HTTP é um protocolo *stateless* de comunicação cliente - servidor: o cliente envia uma requisição para o servidor, este processa a requisição e devolve uma resposta para o cliente, sendo que, a princípio, nenhuma informação é mantida no servidor em relação às requisições previamente recebidas. Assim, quando digitamos o endereço de uma página em um browser Web, estamos gerando uma requisição a um servidor, que irá, por sua vez, devolver para o browser o conteúdo da página HTML requisitada. A requisição enviada por um cliente deve conter, basicamente, um comando (também chamado de método), o endereço de um recurso no servidor (também chamado de *path*) e uma informação sobre a versão do protocolo HTTP sendo utilizado.

Essa linguagem foi desenvolvida em 1992 por Tim Berners Lee e Robert Caillau no CERN, que é o Centro Europeu de Pesquisas de Física de Partículas. O html é um exemplo do SGML (*Standard Generalized Markup Language*). Originalmente o html definia estritamente a estrutura lógica de um documento, e não a sua aparência física. Mas, com a pressão dos usuários (principalmente da indústria), as versões posteriores do html foram forçadas a prover cada vez mais e mais controle da aparência do documento (W3SCHOOLS, 2010).

2.2.2 – Comparações Entre HTML E XML

HTML e XML são “primos”. Eles derivam da mesma inspiração, o SGML. Ambos identificam elementos em uma página e ambos utilizam sintaxes similares. Se você é familiar com HTML, também o será com o XML. A grande diferença entre HTML e XML é que o HTML descreve a aparência e a ações em uma página na rede

enquanto o XML não descreve nem aparência e ações, mas sim o que cada trecho de dados é ou representa! Em outras palavras, o XML descreve o conteúdo do documento! Como o HTML, o XML também faz uso de *tags* (palavras encapsuladas por sinais '<' e '>') e atributos (definidos com name="value"), mas enquanto o HTML especifica cada sentido para as *tags* e atributos (e frequentemente a maneira pela qual o texto entre eles será exibido em um navegador), o XML usa as *tags* somente para delimitar trechos de dados, e deixa a interpretação do dado a ser realizada completamente para a aplicação que o está lendo. Resumindo, enquanto em um documento HTML uma *tag* <p> indica um parágrafo, no XML essa *tag* pode indicar um preço, um parâmetro, uma pessoa, ou qualquer outra coisa que se possa imaginar (inclusive algo que não tenha nada a ver com um p como, por exemplo, autores de livros).

Os arquivos XML são arquivos texto, mas não são tão destinados à leitura por um ser humano como o HTML é. Os documentos XML são arquivos texto porque facilitam que os programadores ou desenvolvedores "debuguem" mais facilmente as aplicações, de forma que um simples editor de textos pode ser usado para corrigir um erro em um arquivo XML. Mas as regras de formatação para documentos XML são muito mais rígidas do que para documentos HTML. Uma *tag* esquecida ou um atributo sem aspas torna o documento inutilizável, enquanto que no HTML isso é tolerado. As especificações oficiais do XML determinam que as aplicações não possam tentar adivinhar o que está errado em um arquivo (no HTML isso acontece), mas sim devem parar de interpretá-lo e reportar o erro (MICROSOFT, 2010).

2.2.3 – Características da Linguagem XML

O XML provê uma representação estruturada dos dados que mostrou ser amplamente implementável e fácil de ser desenvolvida. Implementações industriais na linguagem SGML mostraram a qualidade intrínseca e a força industrial do formato estruturado em árvore dos documentos XML.

O XML é um subconjunto do SGML, o qual é otimizado para distribuição através da

web, e é definido pelo Word Wide Web Consortium (W3C), assegurando que os dados estruturados serão uniformes e independentes de aplicações e fornecedores.

O XML provê um padrão que pode codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações desde simples até as mais complexas, dentre elas:

- Um simples documento;
- Um registro estruturado tal como uma ordem de compra de produtos;
- Um objeto com métodos e dados como objetos Java ou controles ActiveX;
- Um registro de dados. Um exemplo seria o resultado de uma consulta a bancos de dados;
- Apresentação gráfica, como interface de aplicações de usuário;
- Entidades e tipos de esquema padrões;
- Todos os links entre informações e pessoas na web.

Uma característica importante é que uma vez tendo sido recebido o dado pelo cliente, tal dado pode ser manipulado, editado e visualizado sem a necessidade de reacionar o servidor. Dessa forma, os servidores têm menor sobrecarga, reduzindo a necessidade de computação e reduzindo também a requisição de banda passante para as comunicações entre cliente e servidor.

O XML é considerado de grande importância na Internet e em grandes intranets porque provê a capacidade de interoperação dos computadores por ter um padrão flexível e aberto e independente de dispositivo. As aplicações podem ser construídas e atualizadas mais rapidamente e também permitem múltiplas formas de visualização dos dados estruturados (MICROSOFT, 2010).

2.3 – WEB SERVICE

Um requisito básico de qualquer empresa é prover serviços, sejam os vendedores de uma empresa, o setor de custos e compras, os prestadores de serviço, etc. Cada empresa oferece serviços para a comunicação entre ela e outras pessoas, sejam pessoas físicas ou jurídicas, internas ou externas a empresa. Alguns desses serviços podem ser automatizados. Por exemplo, não é necessário existir um representante já que o usuário tem, em mãos, o privilégio e todos os outros dados relevantes para constituir um pedido de relatórios. Este pedido pode e, em muitos casos, já é feito, via interfaces computacionais. Isto é um serviço web, ou seja, um serviço que está publicado na web para que qualquer pessoa possa fazer uso dele sem ter que se importar com plataformas de sistemas.

Web Services foi criada para construir aplicações deste tipo, aplicações que são serviços na internet. Porém não faz parte do conceito de Web Service à criação de interfaces gráficas para os usuários, deixando esta parte para outras empresas ou pessoas desenvolver. Web Services disponibiliza serviços somente para desenvolvedores, ou que Web Services nada mais é do que chamada de métodos usando XML.

Web Services é a tecnologia ideal para comunicação entre sistemas, sendo muito usado em aplicações B2B (*Business to Business*). A comunicação entre os serviços é padronizada possibilitando a independência de plataforma e de linguagem de programação. Por exemplo, um sistema de reserva de passagens aéreas feito em Java e rodando em um servidor Linux pode acessar, com transparência, um serviço de reserva de hotel feito em .Net rodando em um servidor Microsoft.

Para comunicar com a Web Service, é necessário uma implementação do protocolo SOAP (*Simple Object Access Protocol*) definido no W3C. Este protocolo é o responsável pela independência que a *Web Service* precisa. Atualmente já se encontra várias implementações disponíveis em várias linguagens. A figura 9 mostra um diagrama de mensagens trocadas entre cliente e servidor em uma comunicação SOAP. Existem duas aplicações se comunicando, um *Client Wrapper* e um *Server*

Wrapper que estão disponibilizando a transparência para as aplicações. Entre eles só trafega XML, seguindo o protocolo SOAP sobre HTTP (ARMSTRONG, 2003).

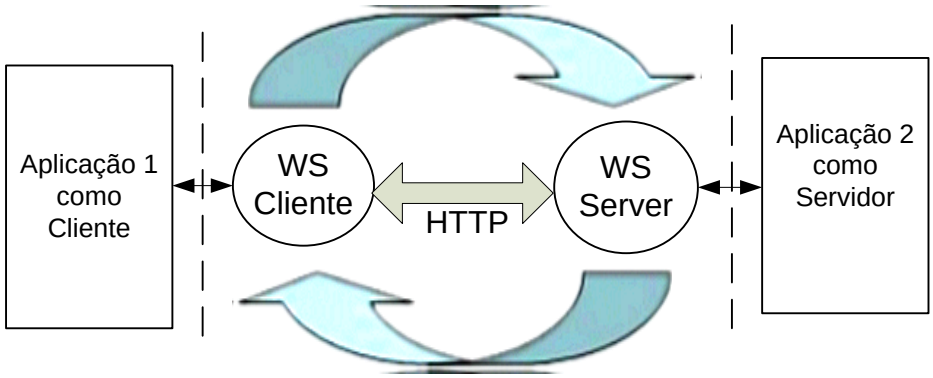

Figura 9: Diagrama de comunicação SOAP (ARMSTRONG, 2003).

2.4 – JAVA SERVER FACES (JSF)

Java Server Faces é uma tecnologia que incorpora características de um *framework* MVC (*Model View Controller*) para WEB e de um modelo de interfaces gráficas baseado em eventos. Uma de suas melhores vantagens é a clara separação entre a visualização e regras de negócio por ser baseada no padrão de projeto MVC, A idéia do padrão MVC é dividir uma aplicação em três camadas: modelo, visualização e controle. O modelo é responsável por representar os objetos de negócio, manter o estado da aplicação e fornecer ao controlador o acesso aos dados. A visualização representa a interface com o usuário, sendo responsável por definir a forma como os dados serão apresentados e encaminhar as ações dos usuários para o controlador. Já a camada de controle é responsável por fazer a ligação entre o modelo e a visualização, além de interpretar as ações do usuário e as traduzir para uma operação sobre o modelo, onde são realizadas mudanças e, então, gerar uma visualização apropriada (PITANGA, 2007). A figura 10 mostra uma visão geral do *Framework.*

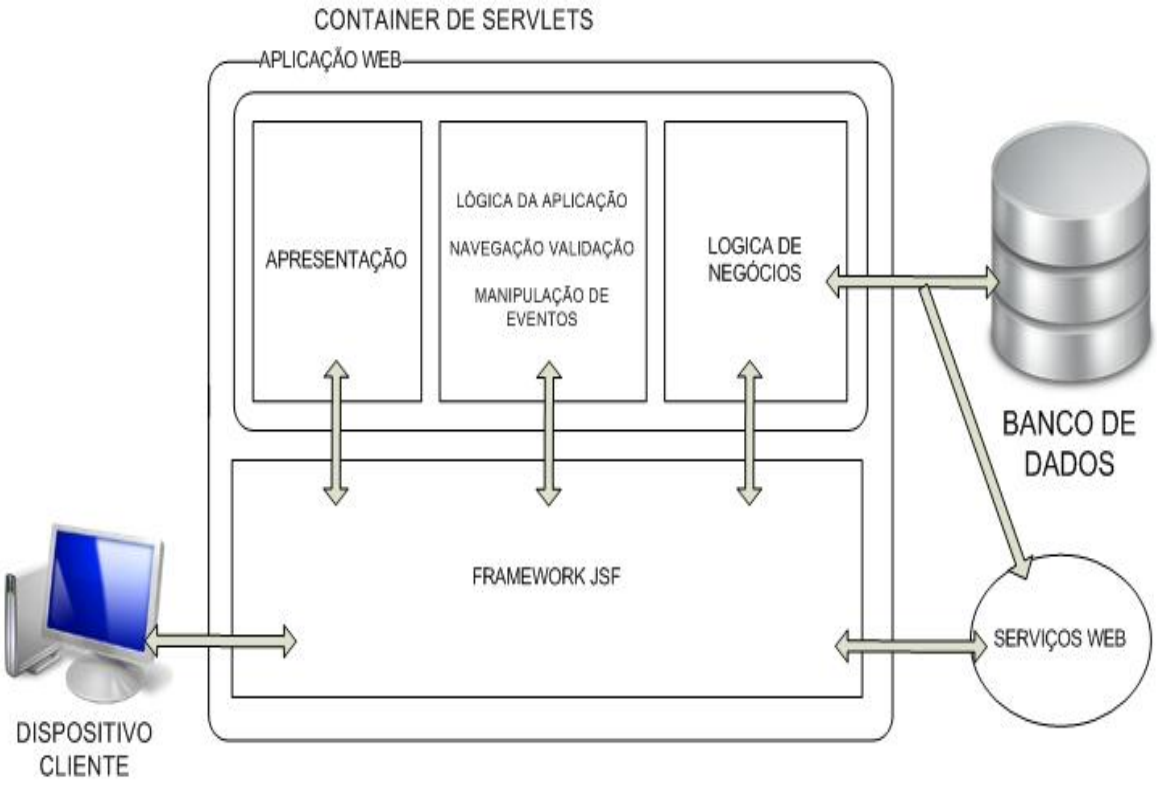

Figura 10: Visão geral do alto nível do Framework JSF (PITANGA, 2007).

2.4.1 – Padrão MVC Segundo JSF

No JSF, o controle é composto por um *servlet* denominado *FacesServlet*, que é responsável por receber requisições da WEB, redirecioná-las para o modelo e então enviar uma resposta. Os arquivos de configuração são responsáveis por realizar associações e mapeamentos de ações e pela definição de regras de navegação. Os manipuladores de eventos são responsáveis por receber os dados vindos da camada de visualização, acessar o modelo, e então devolver o resultado para o *Faces Servlet*.

O modelo representa os objetos de negócio e executa uma lógica de negócio ao receber os dados vindos da camada de visualização. Finalmente, a visualização é composta por *component trees* (hierarquia de componentes *User Interface* (UI)), tornando possível unir um componente ao outro para formar interfaces mais complexas. A figura 11 mostra a arquitetura JSF baseada no modelo MVC (SILVEIRA, 2009).

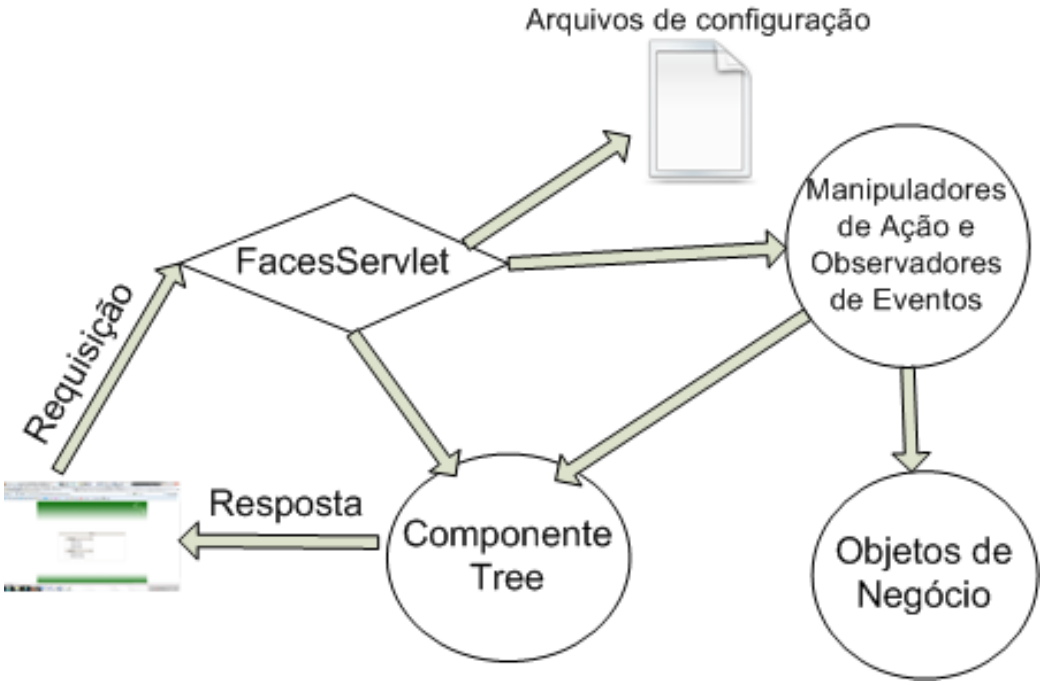

Figura 11 - Arquitetura JSF baseada no modelo MVC (SILVEIRA, 2009).

2.5 – HIBERNATE

Hibernate é uma ferramenta de mapeamento objeto-relacional para Java. Ela transforma os dados tabulares de um banco de dados em um gráfico de objetos definidos pelo desenvolvedor. Uso *Hibernate* libera o desenvolvedor de escrever um monte de código para acessar o banco de dados e SQL ele não escrever usando a ferramenta, acelerando o seu desenvolvimento de uma forma fantástica.

Mas o quadro não é uma boa opção para todos os tipos de aplicação. Sistemas que fazem uso estendido procedimentos armazenados, gatilhos ou que implementam a maior parte da lógica da aplicação em banco de dados, com um modelo de objeto "pobres" não vai beneficiar do uso de *Hibernate*. É mais adequado para os sistemas que dependem de um modelo rico, onde a maioria da lógica de negócio é em Java, dependendo de algumas funções específicas do banco de dados (LINHARES, 2007). A figura 12 mostra o diagrama de funcionamento do *Hibernate*.

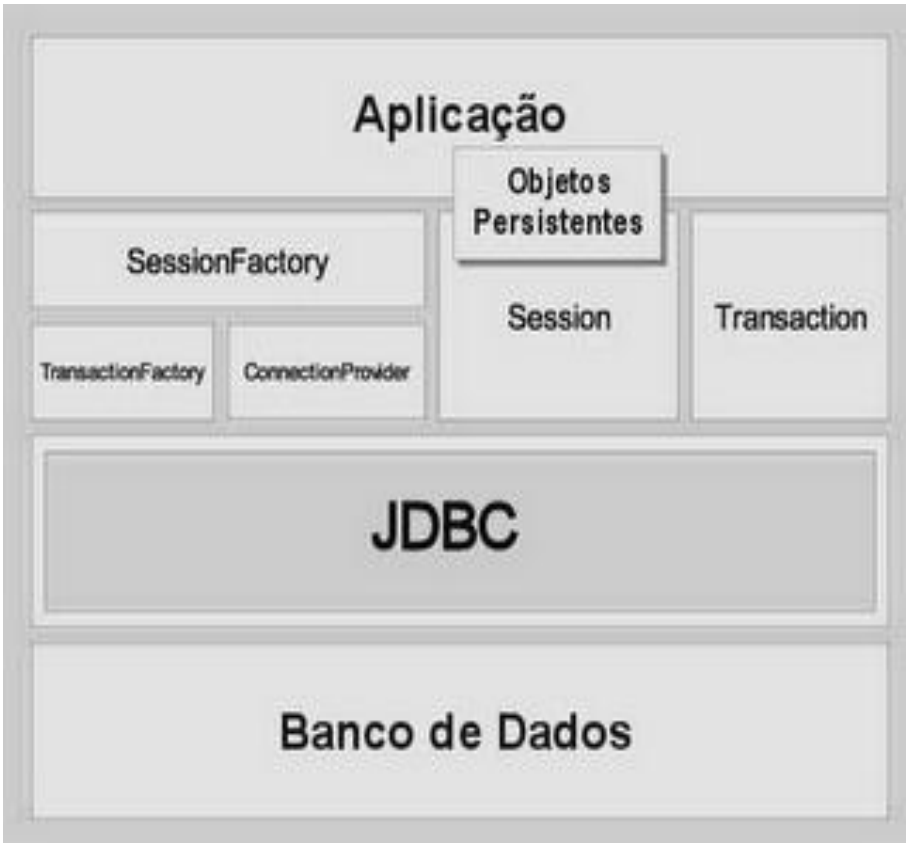

Figura 12: Diagrama de funcionamento do Hibernate. (LINHARES, 2007).

2.6 – HSQLDB

O HSQLDB (*Hypersonic SQL Database*) é um projeto de banco de dados livre, escrito em Java, que permite a manipulação de banco de dados em uma arquitetura cliente-servidor, ou *standalone*. Uma grande vantagem de utilização do HSQLDB é a possibilidade de agregarmos o banco de dados ao pacote de nossas aplicações. O banco é multiplataforma e ocupa um pequeno espaço em disco. Outra característica do banco é a possibilidade de manipularmos bancos de dados em disco, memória ou em formato texto. No núcleo do pacote estão o RDBMS e o driver JDBC que disponibilizam as principais funcionalidades do banco, que são o gerenciador de banco de dados relacional e o driver para conexão através de aplicações Java. Além disso, o pacote contém um conjunto de componentes e ferramentas para execução do SGBD. Através das ferramentas podemos criar estruturas de um banco de dados, acessar bancos de dados através de ferramentas para consulta, exportar e importar

esquemas entre bancos de dados distintos. Segundo (SEVERO, 2007), além de outras facilidades disponibilizadas para o desenvolvedor, apresenta os componentes:

- **HSQLDB JDBC Driver:** o pacote de distribuição disponibiliza um *driver* padrão JDBC para conexão de aplicações Java com o SGBD. Esta conexão com o banco de dados segue o padrão de protocolo proprietário e também realiza conexão via rede através de protocolos Internet;
- **Database Manager:** disponibiliza duas versões de ferramentas para gerenciamento de banco de dados são disponibilizadas: uma ferramenta escrita usando AWT e outra usando Swing. Elas são ferramentas gráficas para visualização do esquema do banco de dados, conjunto de tabelas e submissão de instruções SQL, sendo que a AWT pode ser executada como um *Applet* dentro de um navegador;
- **Transfer Tool:** é uma ferramenta utilizada para transferências de esquemas SQL ou dados de uma fonte JDBC para outra. Ela é uma ferramenta bastante útil para realizar uma migração de banco de dados, transferindo esquemas e o conjunto de dados entre duas tecnologias distintas;
- **Query Tool:** a finalidade dessa ferramenta é prover ao desenvolvedor um software para interação com o SGBD através do envio de instruções SQL a partir de uma linha de comando, ou através de um arquivo texto contendo um conjunto de instruções. A ferramenta apresenta uma Shell interativa ao usuário;
- **SQL Tool:** ferramenta do pacote para construção e submissão de instruções SQL ao banco de dados.

2.7 – APACHE TOMCAT

O Apache Tomcat é um servidor de aplicações Java para web que programa as tecnologias J*ava Servlets* e *Java Server Pages*. Ele também se comporta como um servidor web (HTTP) ou funciona integrado a um servidor web dedicado (como o

Apache ou o IIS). Este programa é um software livre (e uma aplicação de código aberto), nascido no Projeto Apache Jakarta e oficialmente autorizado pela Sun (desenvolvedora do Java) como a implementação de referência para as tecnologias *Java Servlet* e *Java Server Pages* (JSP). Ele cobre parte da especificação J2EE com tecnologias como *servlet* e JSP, e tecnologias de apoio relacionadas como *Realms* e segurança, JNDI *Resources* e JDBC *DataSources*, contudo, ele não implementa pacotes EJB (*Enterprise Java Beans*).O Apache Tomcat é inteiramente escrito em Java e, portanto, para ser executado em seu computador ele necessita de uma Java Virtual Machine (JVM) (Máquina Virtual Java) instalada. A instalação do servidor é simples, porém a configuração requer conhecimento prévio sobre o assunto ou uma leitura criteriosa nos manuais encontrados tanto na página do desenvolvedor como em vários sites na internet (DESTRO, 2006). A figura 13 mostra a arquitetura Tomcat.

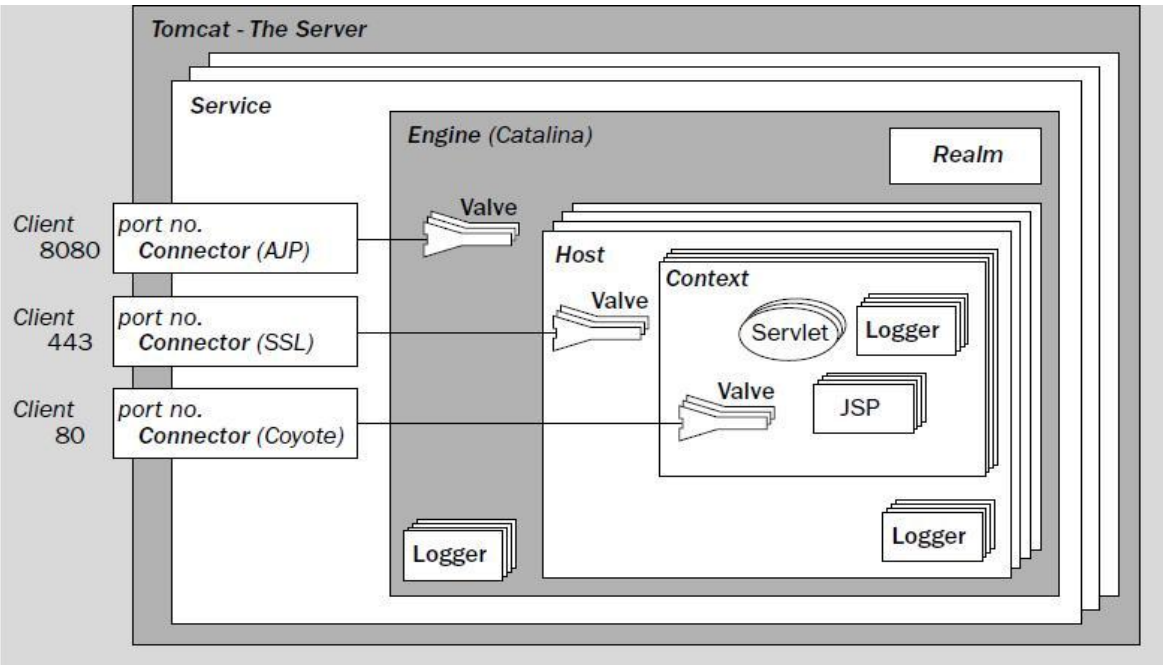

Figura 13: Arquitetura Tomcat (APACHE, 2010).

2.8 – ESTRUTURA BÁSICA DE APLICAÇÃO *WEB* DESENVOLVIDA EM JAVA

Aplicações *web* consistem de componentes *web* (Servlets, JSP, HTML, XML, etc), recursos estáticos (imagens, folhas de estilo, arquivos.js, etc) e classes e bibliotecas Java. Com raríssimas exceções, o desenvolvimento de aplicações web usando tecnologias baseadas em Java segue os seguintes passos:

1. **Desenvolver o código dos componentes Web**

   Esta fase envolve o desenvolvimento, depuração e teste dos vários *Servlets*, *Java Server Pages* e demais componentes que fazem parte do projeto. Em aplicações mais robustas e ou complexas, o uso de ferramentas de teste é muito empregado.

2. **Escrever o descritor de instalação da aplicação**

   Geralmente chamado de *web.xml* este arquivo é usado pelo container ou servidor durante o processo de instalação da aplicação. É aqui que descrevemos os componentes usados, as variáveis de ambiente e os requisitos de segurança. Na maioria das aplicações web este arquivo está localizado no diretório *WEB-INF.*

3. **Compilar os componentes e classes auxiliares**

   Este passo envolve a compilação de toda a componente web pertencente à aplicação e também as classes referenciadas por este componente.

4. **Empacotar a aplicação web**

   Aplicações *web* são distribuídas e instaladas em um container ou servidor após serem empacotadas em um arquivo com a extensão .war (*Web*

Aplication Archive). Embora seja possível efetuar a instalação sem passar por este processo, na prática o empacotamento é recomendado, pois facilita o processo de instalação.

5. Instalar a aplicação

Se a aplicação estiver empacotada em um arquivo .war, o processo de instalação se resume a copiar o arquivo war para o diretório *webapps* da instalação do Tomcat ou o diretório *autodeploy* do domínio desejado no Servidor de Aplicações da Sun. Em ambos os casos o container ou servidor fará a instalação automática da aplicação.

6. Acessar a URL que permite acessar a aplicação

Este passo final consiste em abrir o navegador e apontar para a URL da aplicação.

3 – DESENVOLVIMENTO DO APLICATIVO

Neste capítulo será apresentada a modelagem do problema, onde foram divididos em módulos para fácil entendimento de toda estrutura do trabalho e suas fases de desenvolvimento. Na confecção deste trabalho foram executados procedimentos de implementação buscando a padronização para o desenvolvimento dos objetivos onde é apresentada na definição do problema.

3.1 – DESCRIÇÕES DO PROBLEMA

Neste projeto, será desenvolvido um aplicativo *web* simples para a área de educação, que usará recursos do *framework* JSF, *hibernate* conectado ao banco de dados HSQLDB e do ambiente Java para dispositivos móveis JME. Este aplicativo *web* visa o acesso a relatórios através de um serviço de *web service* para o acesso do cliente móvel onde usuários externos com dispositivos suportados pelo Java e com conexão a internet, terão acesso a essas informações do aplicativo. A figura 14 mostra a visão geral do aplicativo *web*.

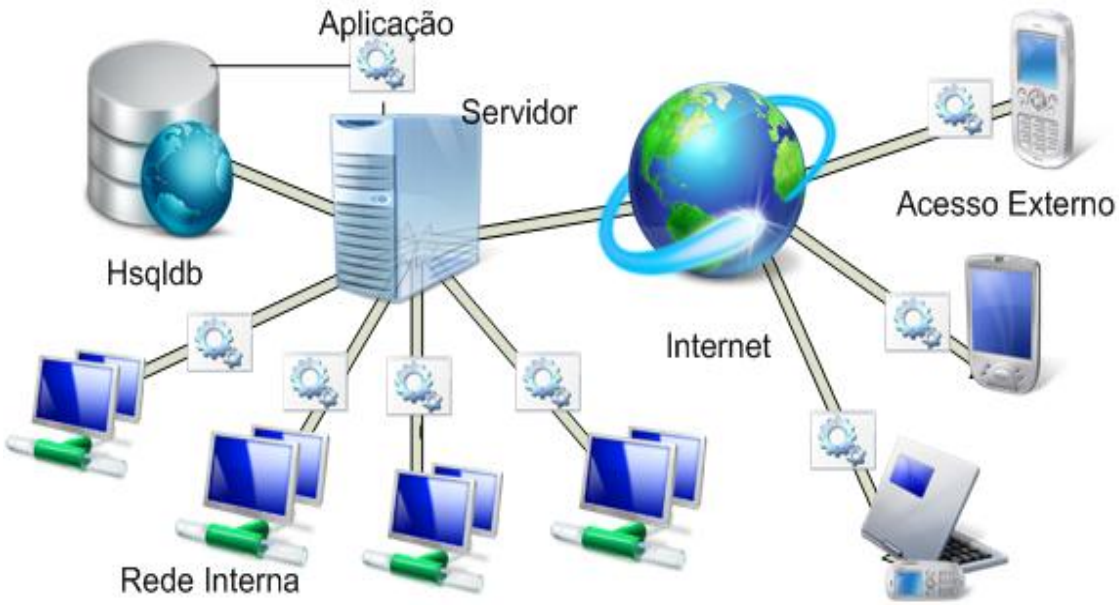

Figura 14 - Visão Geral do Aplicativo Web.

3.2 – MODELAGEM DO PROBLEMA

A modelagem do problema é um processo importante para estabelecer os passos a serem executados no desenvolvimento do aplicativo. Na figura 15 é mostrada a modelagem do problema para este projeto.

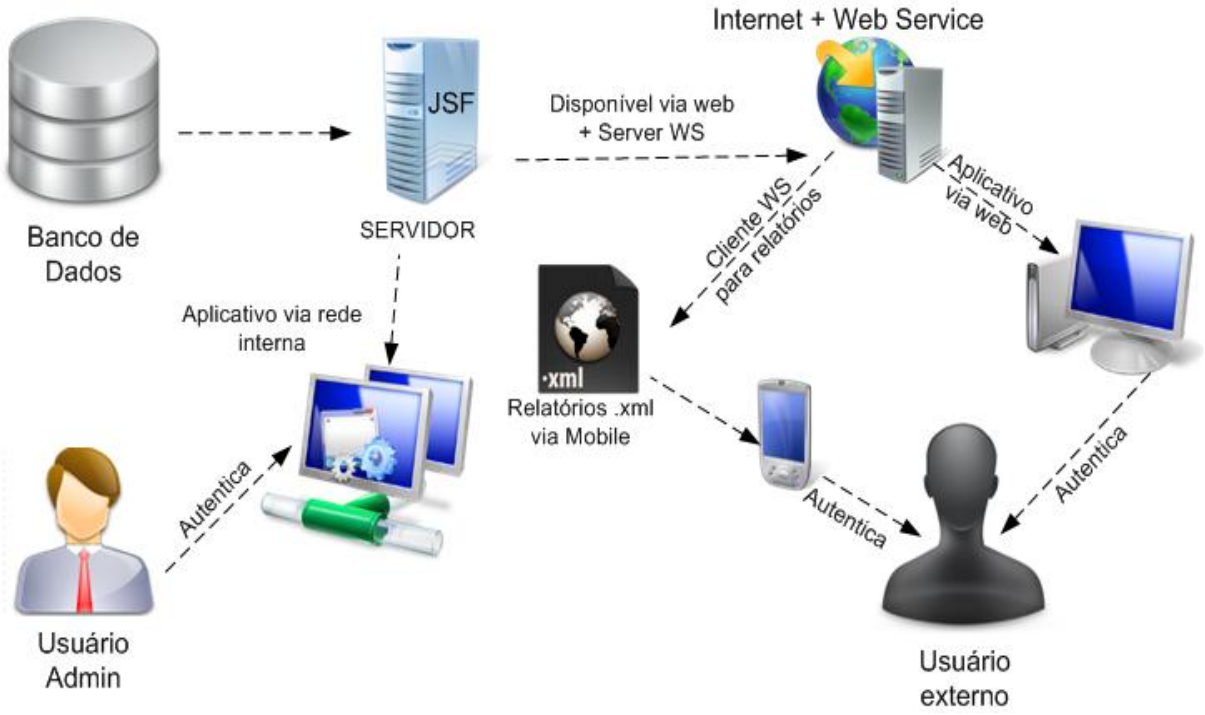

Figura 15 - Modelagem do problema

Este aplicativo web terá a função de fazer todo gerenciamento dos alunos e das escolas na rede de ensino do município. O aplicativo web também disponibiliza consultas a relatórios a partir de um dispositivo móvel suportado por Java e com conexão com a internet através da comunicação de um web service que disponibilizara esse serviço aos usuários externos.

- **Etapa 1: Implementação do Banco de Dados**

  Nesta etapa foi feito a criação do banco de dados “bancoweb” no banco de dados HsqlDB por apresentar um ótimo suporte a linguagem Java. Também foi configurada a conexão do banco com *Hibernate* junto ao aplicativo web.

- **Etapa 2: Implementação do aplicativo web**

  Nesta etapa foi desenvolvido o aplicativo web utilizando o framework JSF através do IDE Netbeans. Esse módulo é a parte principal do trabalho que sustentara os outros módulos para a finalização do trabalho. Com o aplicativo web o usuário administrador do aplicativo vai alimentar os dados com todas as informações necessárias. E através da parte móvel o usuário pode ter um relatório do que necessita.

- **Etapa 3: Servidor**

  Nesta etapa é a parte da execução do Apache Tomcat onde é feito o teste da aplicação e onde servira toda a execução da aplicação através do *container* servidor hospedando o aplicativo compactado em .war.

- **Etapa 4: Web Service**

  Nesta etapa é onde é a parte da implementação do web service para a disponibilização do serviço de relatório para o dispositivo móvel, desde que o mesmo tenha conexão com a internet.

- **Etapa 5: implementação do aplicativo do dispositivo móvel**

  Nesta etapa é a parte da implementação do aplicativo móvel que faz a requisição dos serviços para o web service onde é disponibilizado os relatório para o usuário externo.

3.3 – FUNCIONAMENTOS DO APLICATIVO

O funcionamento do aplicativo é representado com diagramas onde proporciona uma melhor representação de como deve ser desenvolvido o trabalho. Demonstrando como será o funcionamento do programa assim facilitando o entendimento de pessoas que não conhece sobre a parte de programação para possa ser entendimento o que será executado no projeto. Para a ilustração da figura 16 foi usado à ferramenta Bizz designer. A figura 16 mostra o diagrama de entidade do aplicativo.

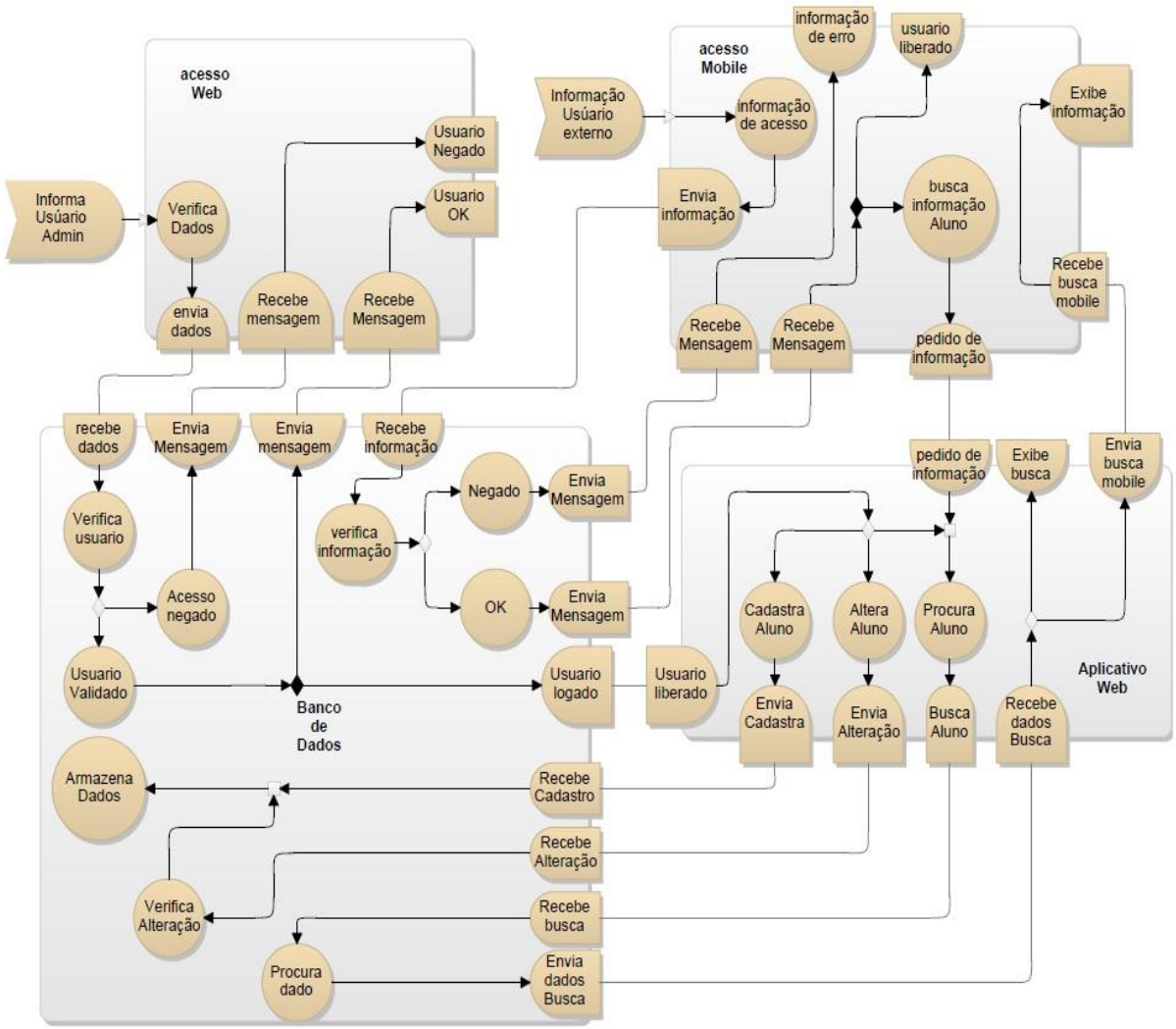

Figura 16 - Diagrama de entidade

Neste tópico será usado o aplicativo argoUML para diagramas para ajudar a visualização panorâmica do projeto, onde pessoas possam entender o projeto sem precisar olhar o código. Com os diagramas ajuda a planejar melhor o domínio da aplicação onde é a parte importante na fase de entendimento do sistema.

3.3.1 – Diagrama de Casos de Uso

- **Manter Aluno:** responsável por todas as operações relacionadas ao Aluno.
- **Manter Escola:** responsável por todas as operações relacionadas à escola.
- **Adicionar:** adiciona novos objetos no banco de dados.
- **Pesquisar:** recupera os objetos armazenados no banco de dados.
- **Atualizar:** atualiza o objeto pesquisado.
- **Remover:** Remove o objeto pesquisado.
- **Relatório:** responsável por gerar relatório do objeto pesquisado.

A figura 17 mostra o diagrama de caso de uso.

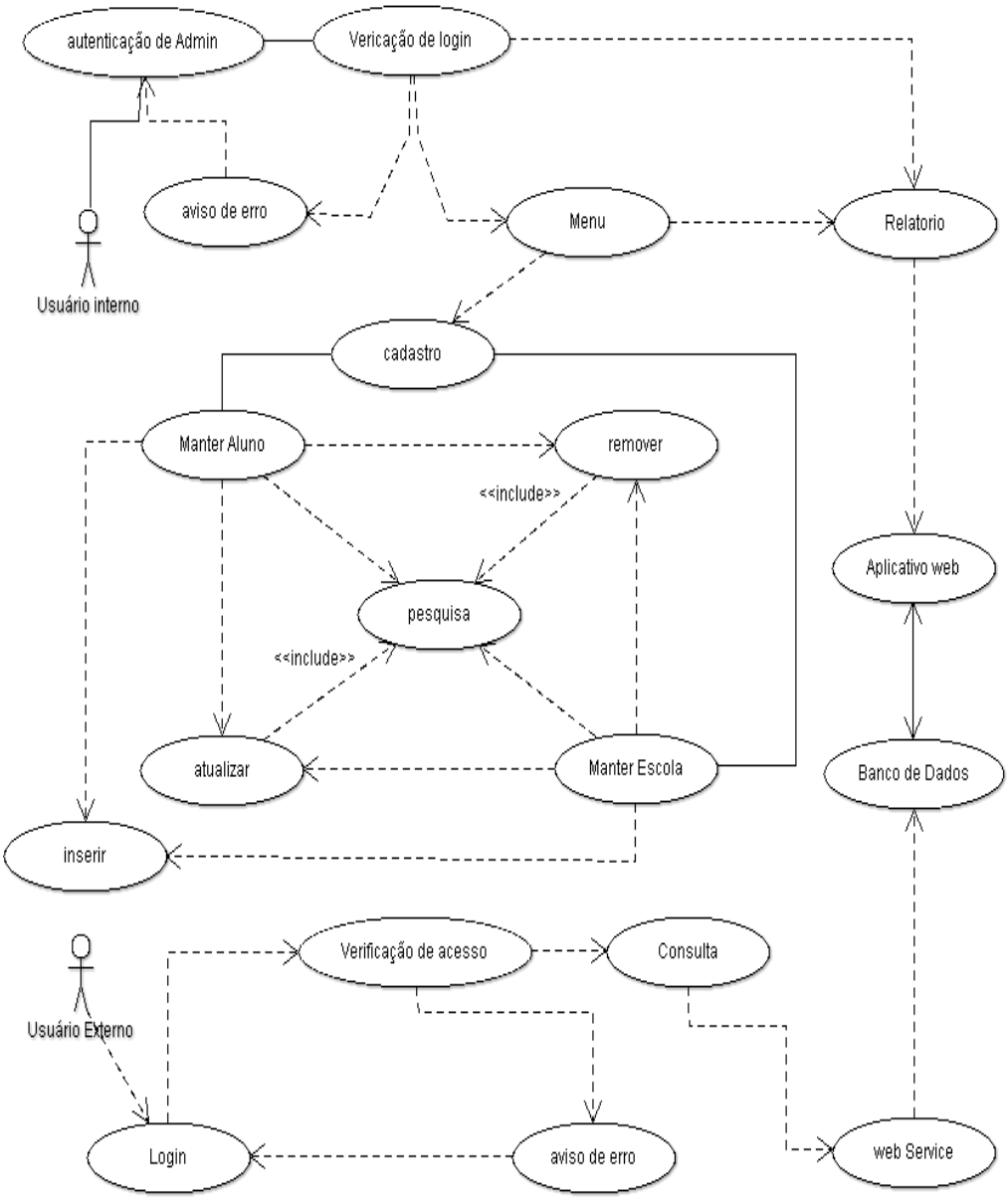

Figura 17 – Diagrama de casos de uso.

3.3.2 – Diagramas de Classes

As classes utilizadas no desenvolvimento deste aplicativo foram representadas por Diagrama de classes.

Pacote edu.web.beans

O pacote beans contem os beans classe que são responsáveis por definir o comportamento dos objetos através de métodos e atributos.

- **Aluno:** classe responsável por instanciar os Alunos do aplicativo.
- **Cidade:** classe responsável por instanciar as cidades.
- **Diretor:** classe responsável por instanciar os Diretores do aplicativo.
- **Escola:** classe responsável por instanciar as Escolas do aplicativo.
- **Estado:** classe responsável por instanciar os estados.
- **Login:** classe responsável por instanciar os login do aplicativo.
- **Pais:** classe responsável por instanciar os países.
- **Usuario:** classe responsável por instanciar os Usuários do aplicativo.

A figura 18 mostra o diagrama de classe do Pacote edu.web.beans.

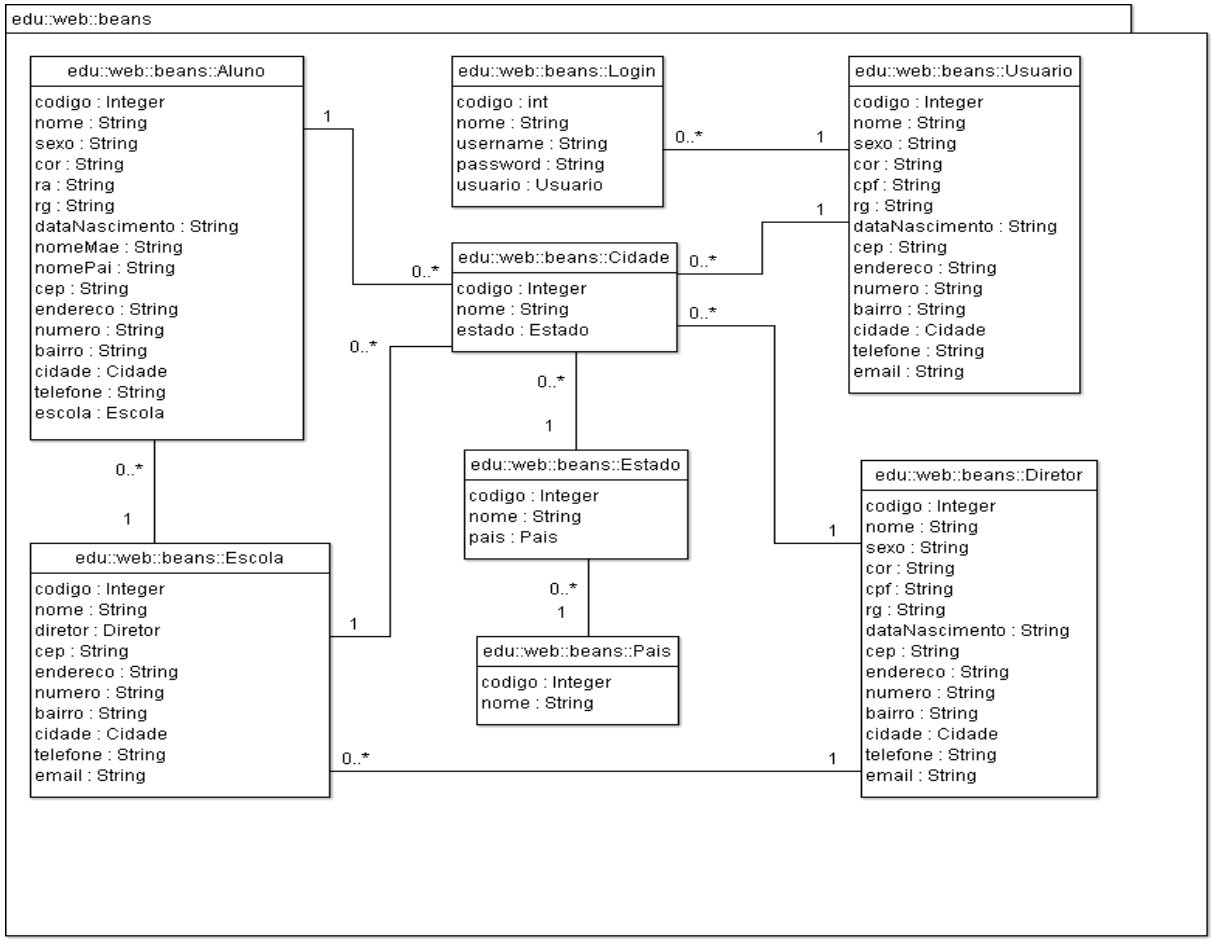

Figura 18 – Diagrama de Classes – pacote de beans.

Pacote edu.web.dao

O pacote Dao é o responsável pela movimentação no banco de dados é ele quem faz o acesso à requisição para a entrada e saída de informações no banco de dados.

- **Dao:** classe abstrata responsável pela comunicação entre *beans, os daos* para o banco de dados.
- **AlunoDao:** classe que estende a classe dao responsável em armazenar os alunos no banco de dados.
- **CidadeDao:** classe que estende a classe dao responsável em armazenar as cidades no banco de dados.
- **DiretorDao:** classe que estende a classe dao responsável em armazenar os Diretores no banco de dados.
- **EscolaDao:** classe que estende a classe dao responsável em armazenar os Escolas no banco de dados.
- **EstadoDao:** classe que estende a classe dao responsável em armazenar os estados no banco de dados.
- **LoginDao:** classe que estende a classe dao responsável em armazenar os logins de usuários no banco de dados.
- **PaisDao:** classe que estende a classe dao responsável em armazenar os países no banco de dados.
- **UsuarioDao:** classe que estende a classe dao responsável em armazenar os usuarios no banco de dados.

A figura 19 mostra o diagrama de classe do Pacote edu.web.dao.

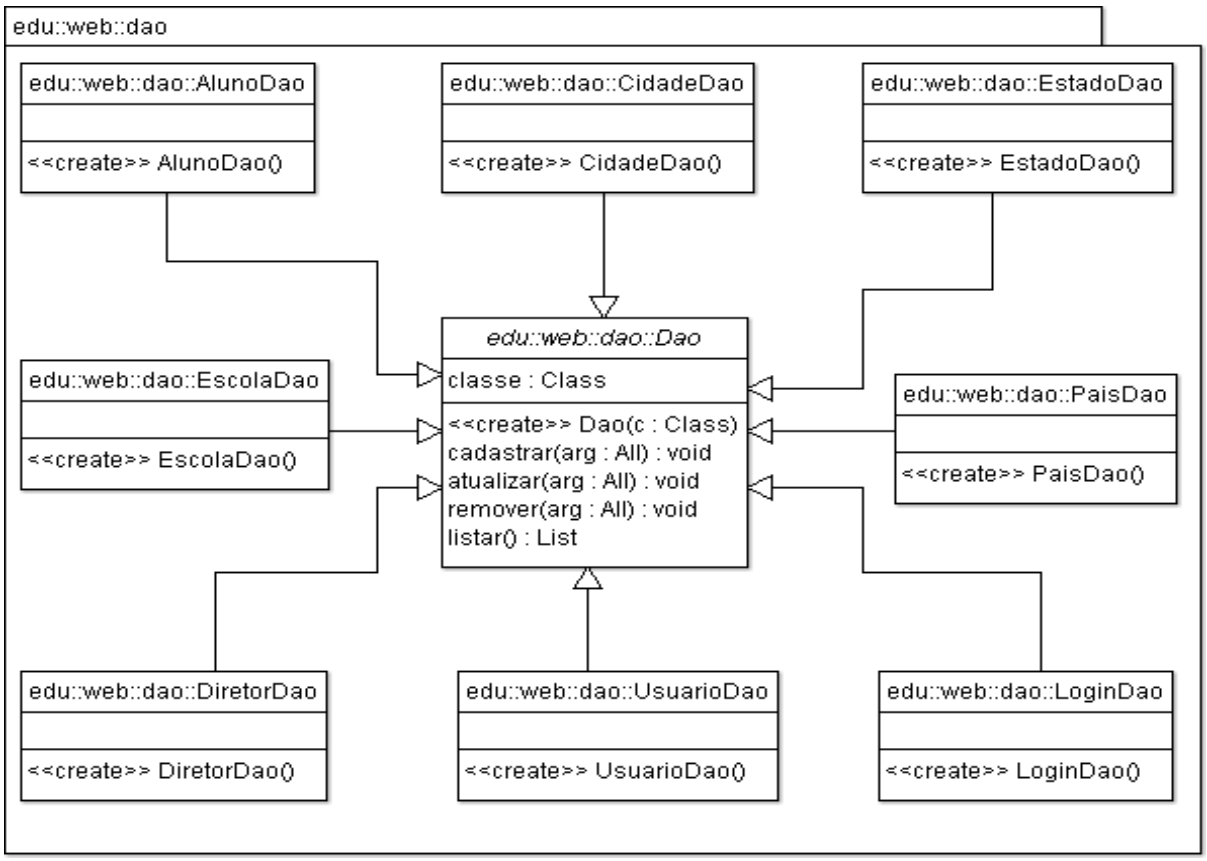

Figura 19 – Diagrama de Classes – pacote de dao.

Pacote edu.web.mb

Classes managed beans (mb) são as classes de regras de negócios são elas que fará a interação das paginas JSF com as classes Java.

- **CadastrarAlunoMB**: classe responsável pela comunicação entre a página de JSF com da classe de cadastro de aluno.
- **CadastrarDiretorMB**: classe responsável pela comunicação entre a página de JSF com da classe de cadastro de Diretor.
- **CadastrarEscolaMB**: classe responsável pela comunicação entre a página de JSF com da classe de cadastro de Escola.

- **CadastrarUsuarioMB**: classe responsável pela comunicação entre a página de JSF com da classe de cadastro de usuário.
- **RelatortioAlunoMB**: classe responsável pela comunicação entre a página de JSF com da classe de relatório de aluno.
- **RelatortioEscolaMB**: classe responsável pela comunicação entre a página de JSF com da classe de relatório de aluno.

A figura 20 mostra o diagrama de classe do Pacote Edu.web.mb.

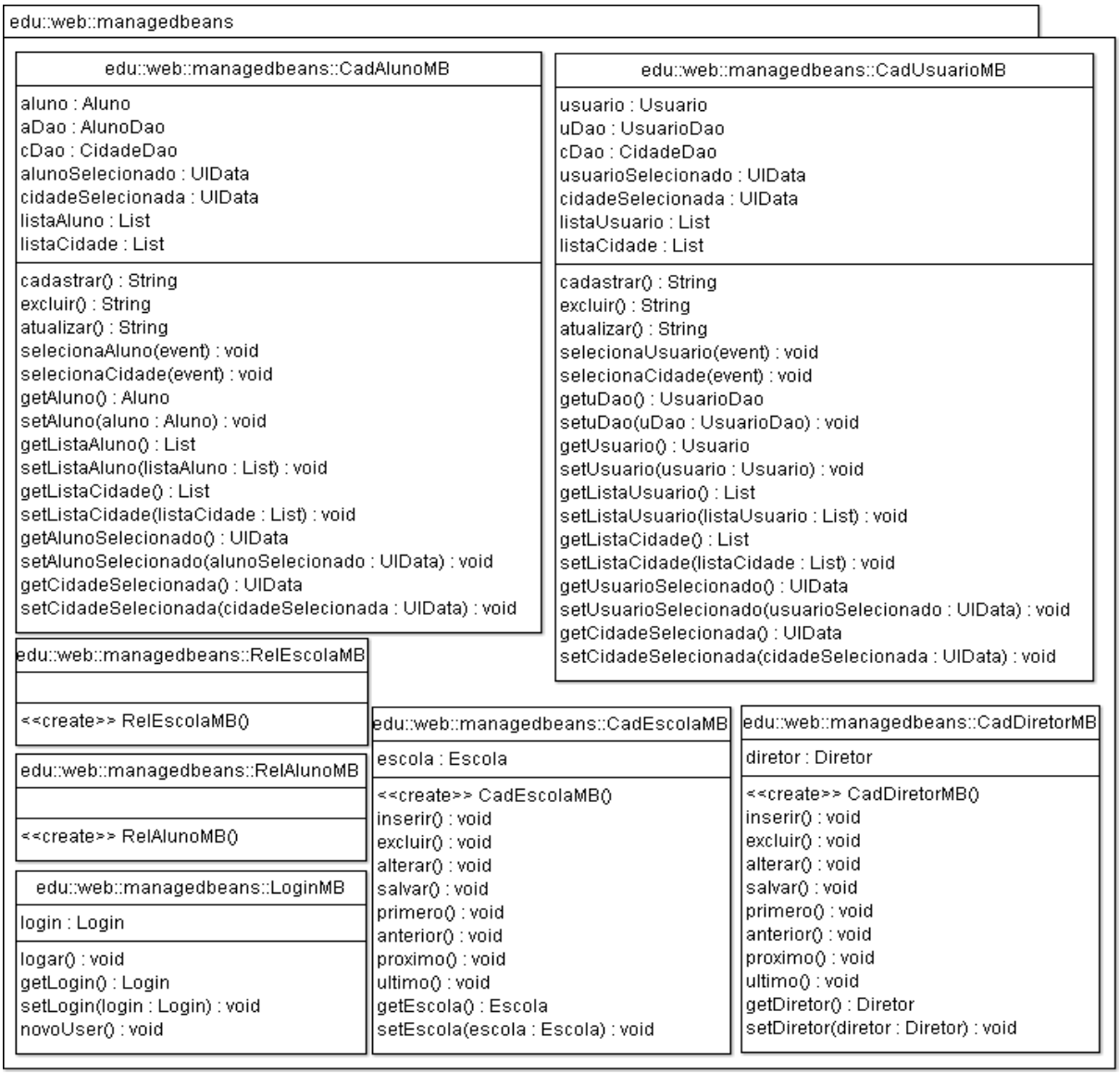

Figura 20 – Diagrama de Classes – pacote de managedbeans.

3.3.3 – Diagrama de Atividades

O diagrama de atividades faz uma representação das atividades de um sistema e do controle de fluxo das informações. Na figura 21, é apresentado o diagrama de atividades do aplicativo, destacando todos os procedimentos necessários para o acesso ao aplicativo.

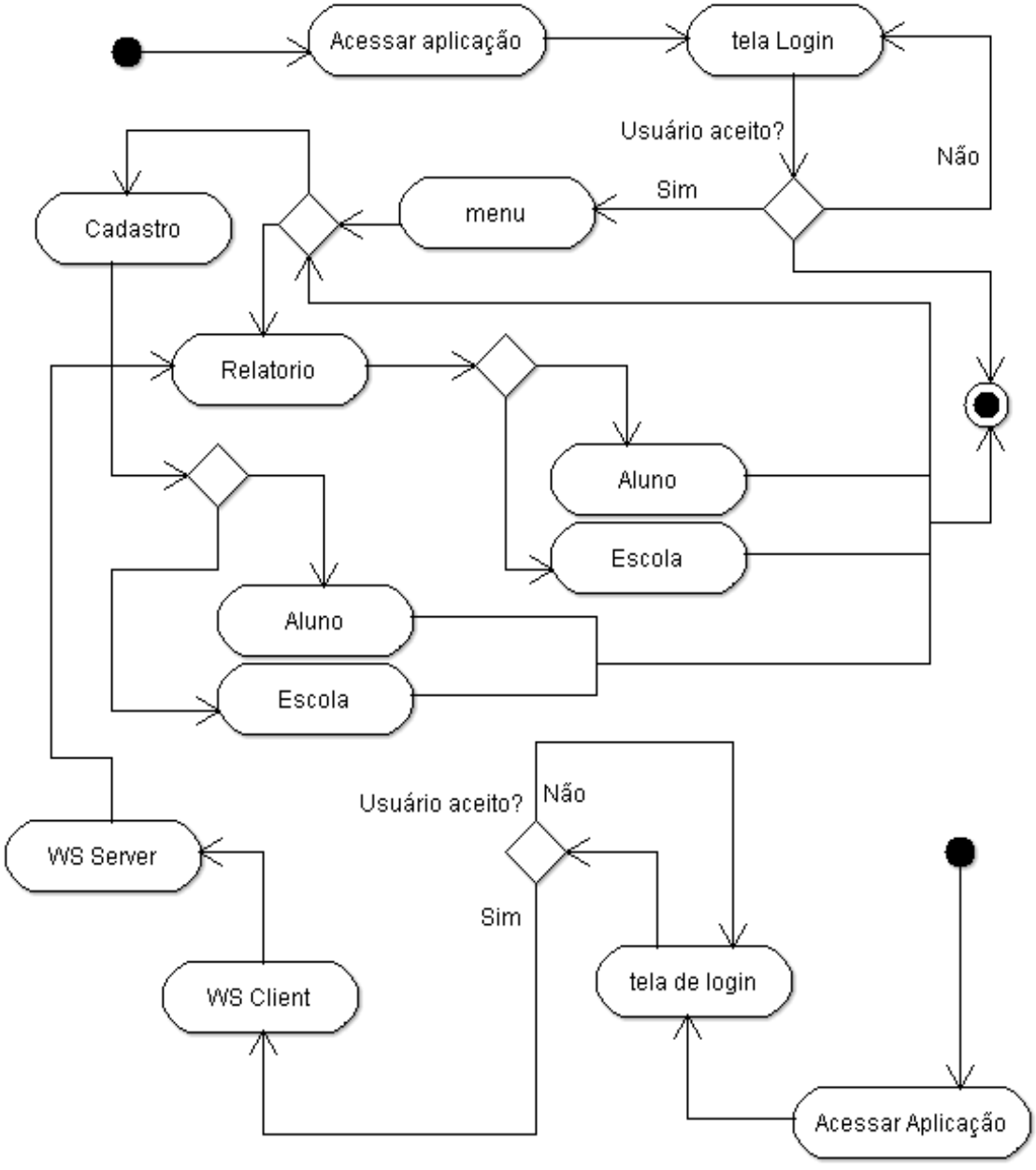

Figura 21 – Diagrama de atividades.

3.4 - IMPLEMENTAÇÃO DO APLICATIVO

Neste tópico será apresentado à implementação do aplicativo desenvolvido neste projeto, a utilizando o ambiente de programação *NetBeans* 6.9.1 torna possível a demonstração do aplicativo.

3.4.1 – Operação da Aplicação

O projeto desenvolvido neste trabalho é uma aplicação *web* capaz de cadastrar e gerar relatórios e personagens da área de educação, e disponibilizar os relatórios para dispositivos móveis. A figura 22 apresenta a interface da área de autenticação para entrada do aplicativo, onde o conteúdo da mesma pode ser dividida nas implementações dos casos de uso que seguem abaixo. Nesta interface o usuário cadastrado efetuará a autenticação para acessar o aplicativo.

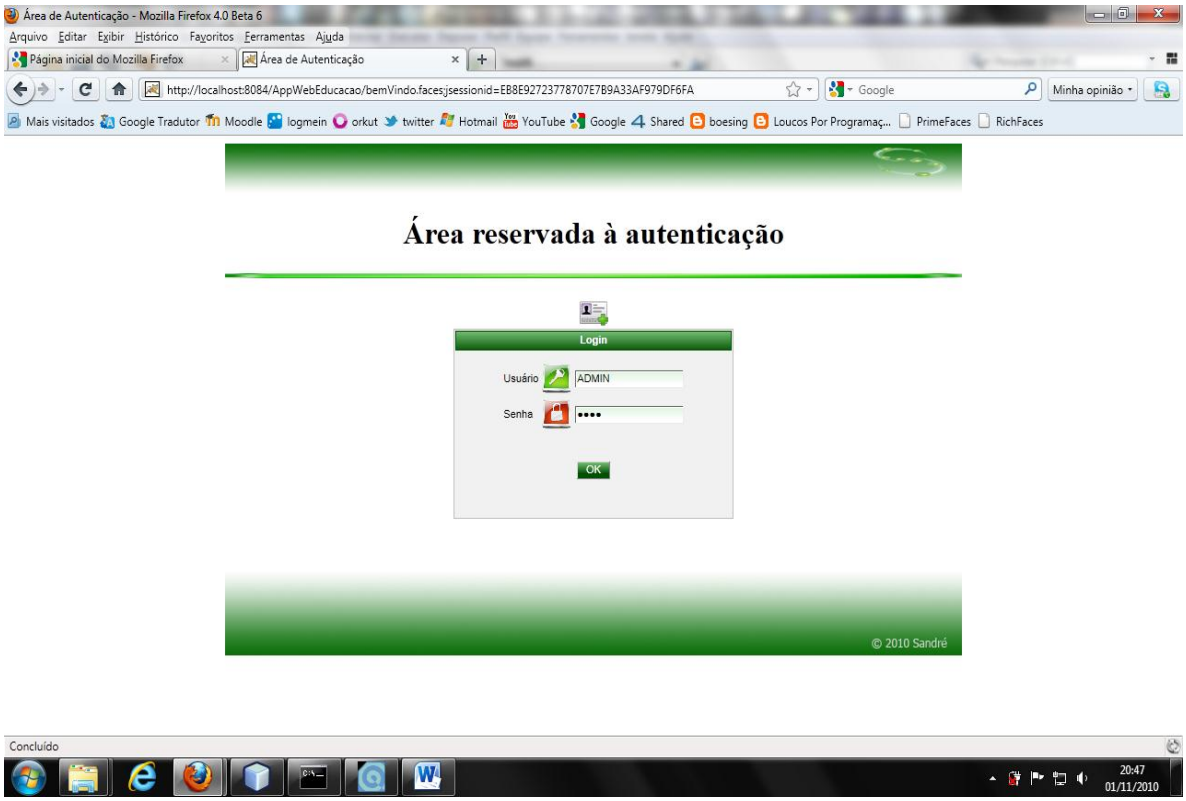

Figura 22 – Página de Login do aplicativo

A figura 23 mostra a interface com as opções que o usuário *web* tem para escolher a funcionalidade que será utilizada dependendo de sua necessidade.

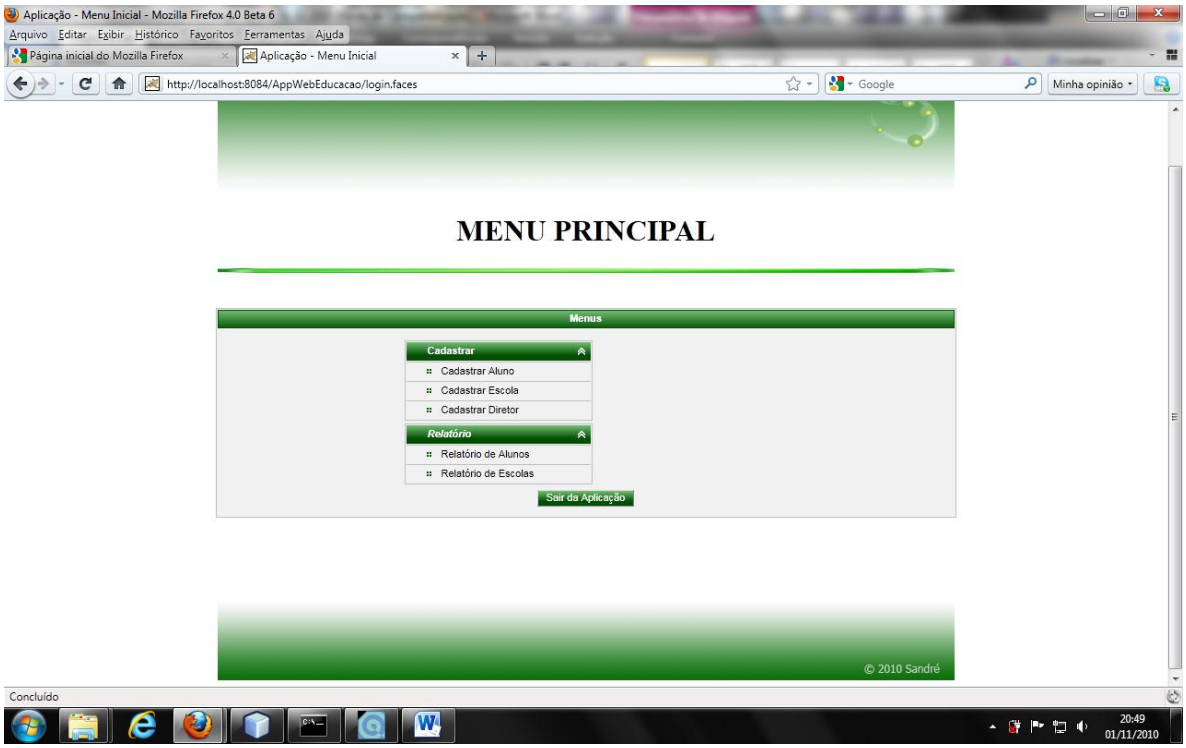

Figura 23 – Página de Menu do aplicativo

3.4.2 – Implementação do caso de uso “Manter Alunos”

O caso de uso “Manter Alunos” ocorre quando um usuário deseja incluir, pesquisar, alterar ou excluir um Aluno no aplicativo. Para a inclusão de um Aluno é necessário o preenchimento dos campos. Para a escolha da cidade é necessário clicar no ícone do combo box na frente do campo “Cidade” para pesquisar a cidade desejada, o mesmo para escolha de Escola. Após esses passos, basta confirmar a inclusão através do botão “Salvar” como mostra a figura 24, os demais cadastros são semelhantes.

Figura 24 – Página de Cadastro do aluno

Para excluir ou alterar um cliente, primeiro é necessário clicar pesquisar o Aluno desejado. Após a escolha do mesmo, serão carregadas todas as suas informações nos campos correspondentes, bastando escolher entre alterar os dados ou excluí-lo através dos botões “Atualizar” ou “Excluir” como mostra a figura 25.

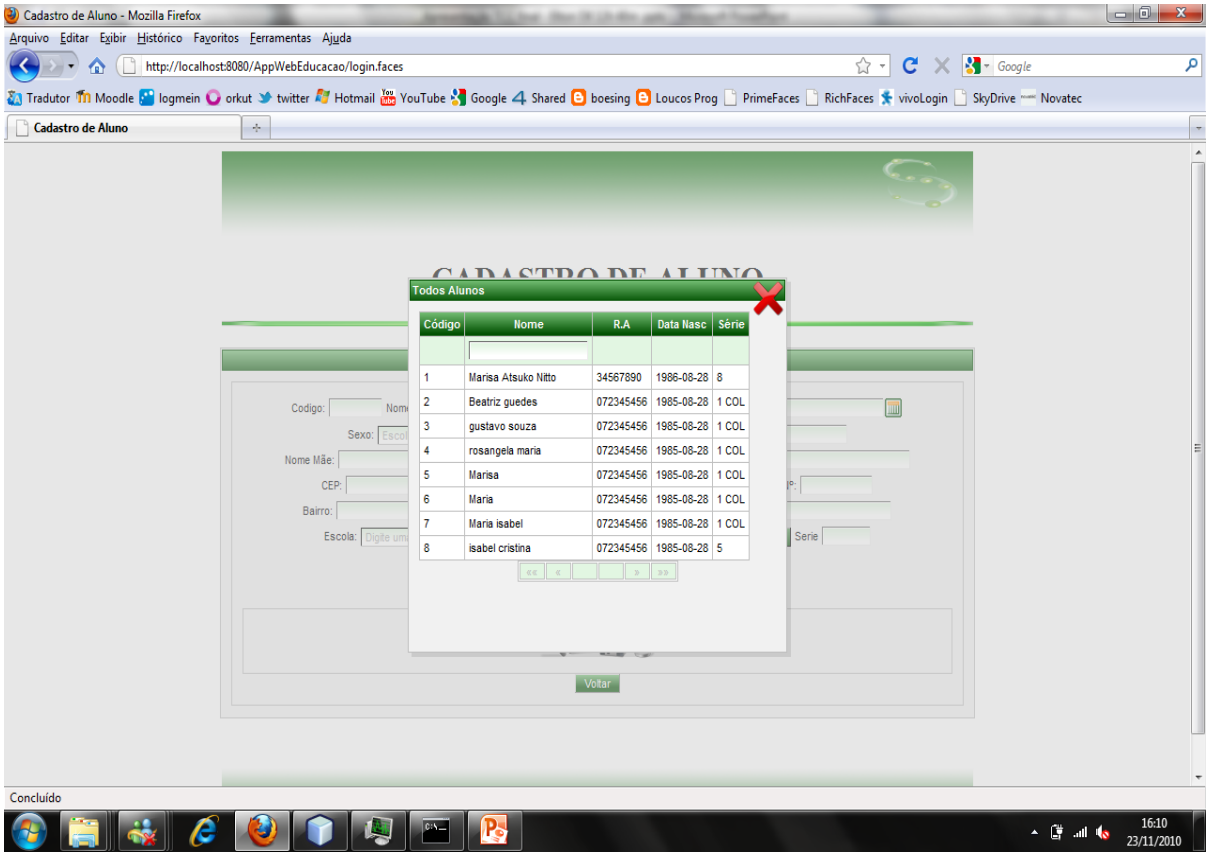

Figura 25 – Página de pesquisa

3.4.3 – Implementação do caso de uso "Acessar *WebService*"

O caso de uso "Acessar WebService" ocorrerá quando outro aplicativo implementar os serviços oferecidos por este aplicativo. Um aplicativo para celulares acessara a mesma base de dados para consumir estes serviços. A figura 26 mostra a interface de serviços contidos neste aplicativo.

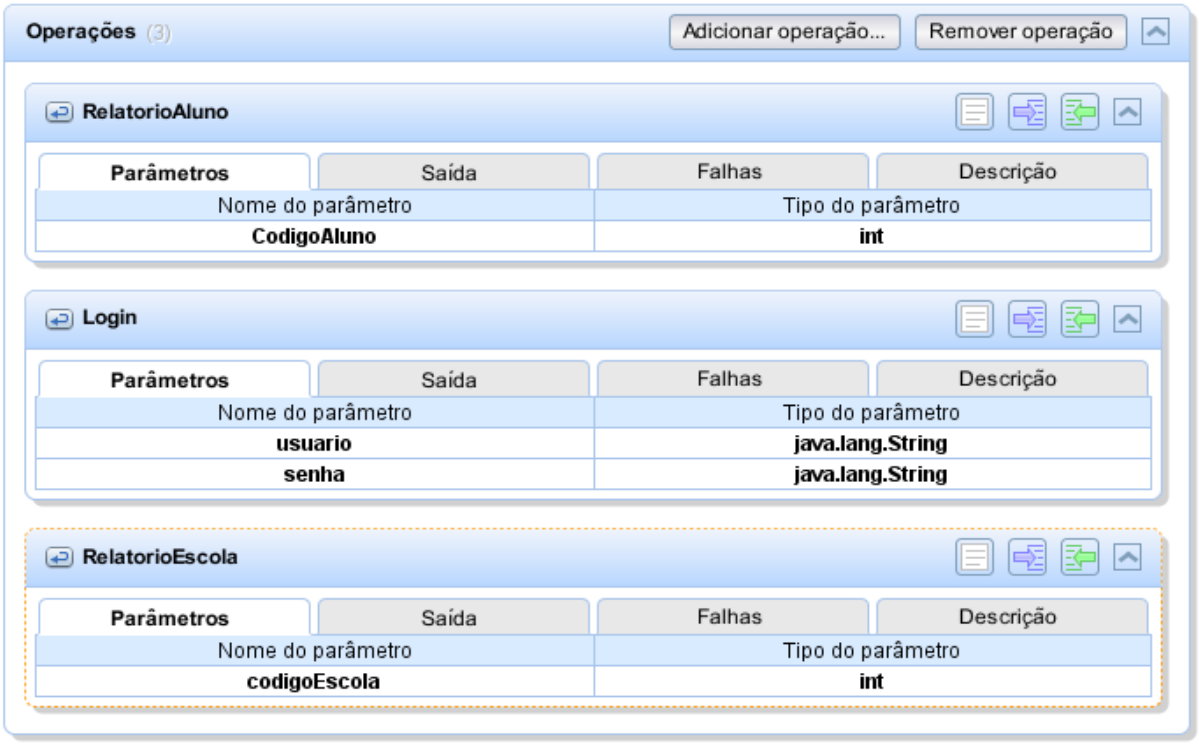

Figura 26 – Serviço de Web Service

3.4.4 – Implementação do aplicativo móvel

A implementação do aplicativo móvel fica responsável pela solicitação de relatório ao serviço de *web service.* Nesta tela o usuário cadastrado efetuara a autenticação para acessar o aplicativo. A figura 27 mostra a interface de autenticação do dispositivo móvel.

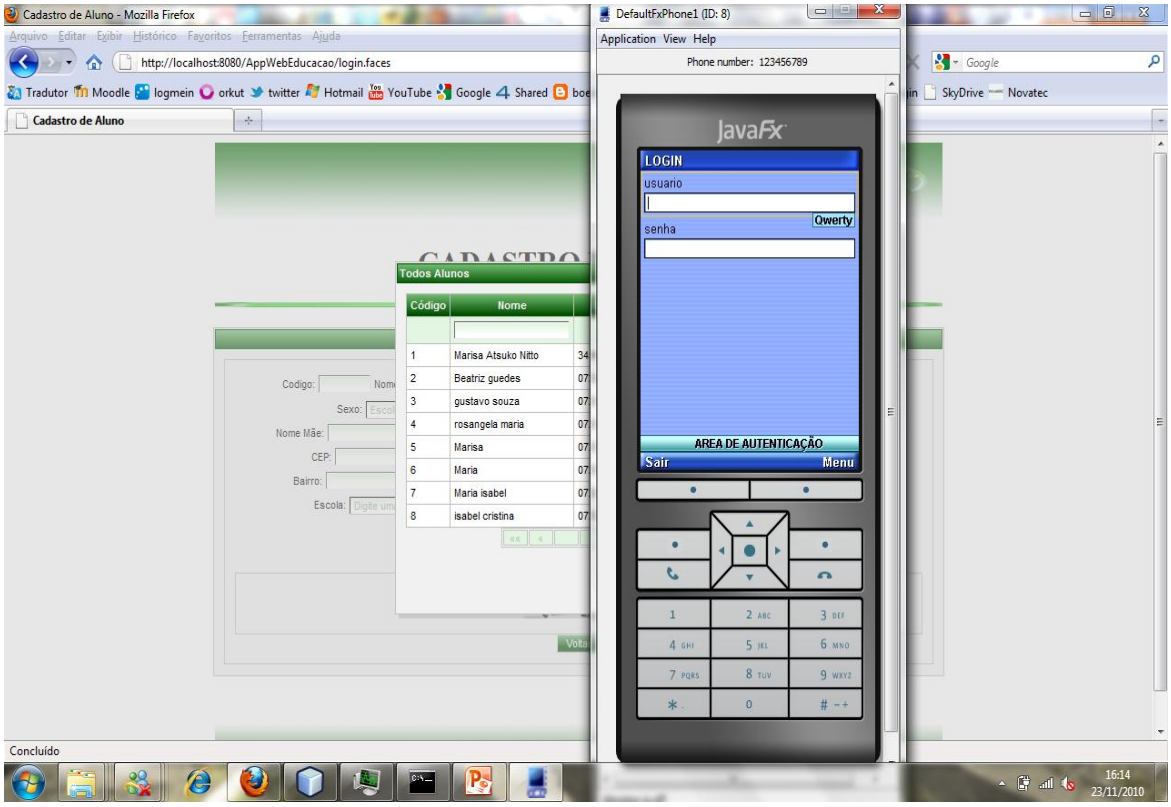

Figura 27 – Página de Login do aplicativo móvel

Após a autenticação o usuário externo tem acesso a relatório de alunos e escolas através do dispositivo móvel. Na figura 28 é mostrado o relatório de alunos via dispositivo móvel que foi requisitado o serviço a um *web service* do aplicativo web para a geração do relatório do mesmo.

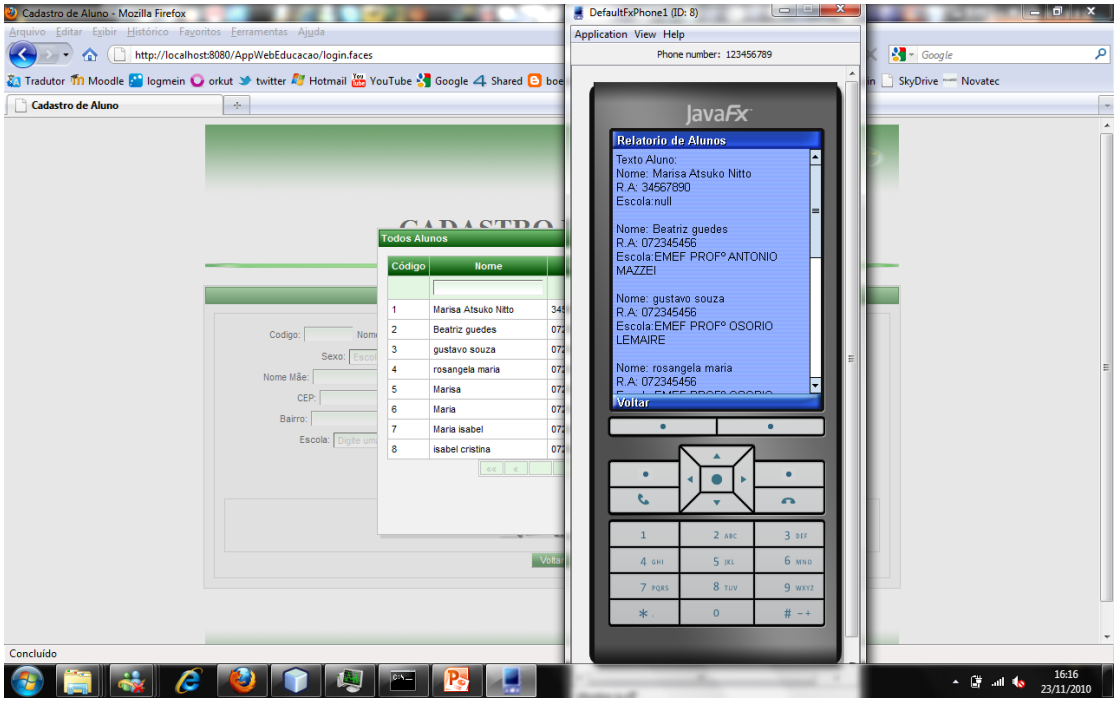

Figura 28 – Página de Relatório de Alunos no aplicativo móvel

Na figura 29 é mostrado o relatório de escolas via dispositivo móvel o mesmo passo semelhante ao de relatório de alunos mostrado na figura 28.

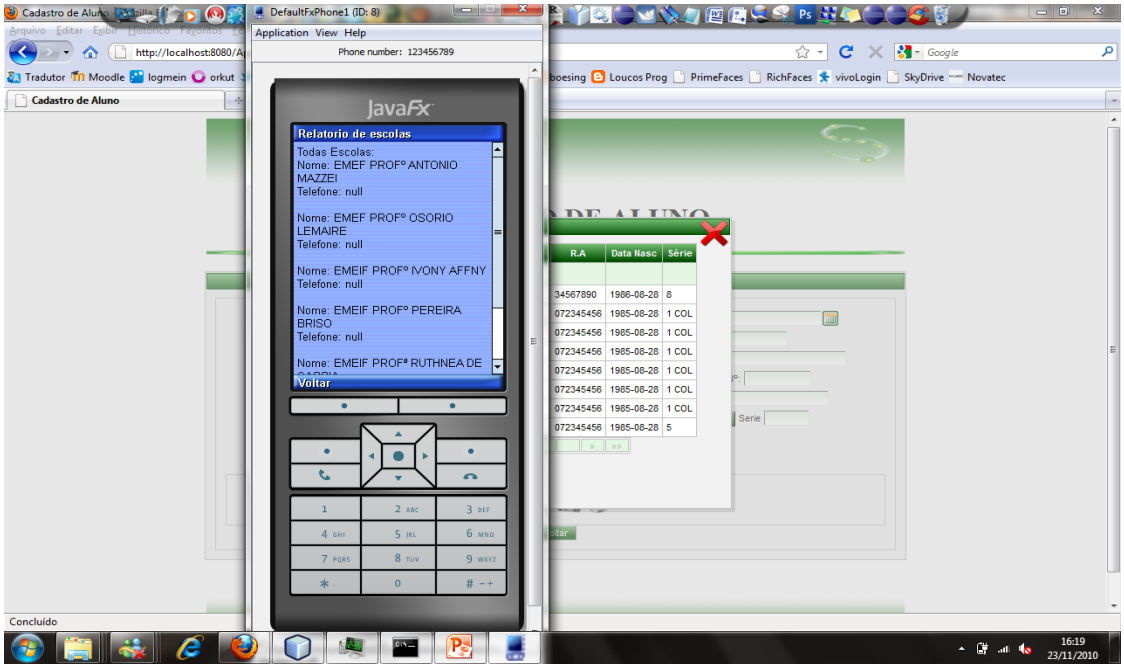

Figura 29 – Página de Relatório de escolas no aplicativo móvel

4 – CONCLUSÃO

Neste trabalho, foram levantadas e apresentadas novas tecnologias de desenvolvimento de aplicações *web* que surgiu no mundo da informática. Os paradigmas estão sendo modificados a uma velocidade muito rápida. Já se encontram disponibilizados elementos que nos permitem criar aplicações realmente orientadas a objetos e distribuídas.

As maiores vantagens da utilização desta tecnologia são a possibilidade de reutilização de código em larga escala, utilizando os objetos, e aproveitamento do poder computacional, que estão distribuídas em redes de computadores de todo o mundo. Neste ambiente, todo software desenvolvido tem a capacidade de ser executado e utilizado em múlti plataformas de *software* e *hardware*, com toda transparência para o usuário.

E com certeza usar essas novas tecnologias em áreas ainda pouco usadas engrandece cada vez mais a área de concentração da informática e tecnologia. Além da grande vantagem deste aplicativo ser baseado em plataformas de *hardware* de baixo custo e com uso de licença pública facilitando a instalações em escolas públicas.

REFERÊNCIAS

JAVA FREE, <http://javafree.uol.com.br/artigo/871485/>. Acesso em fevereiro de 2010.

DEXTRA - **Coding your Business.**
< http://www.dextra.com.br/empresa/artigos/webservices.htm>. Acesso em fevereiro de 2010.

DEITEL, H. M.: Java Como Programar, 6ª. Edição. Pearson Education. São Paulo. 2005.

HENDRICKS, M.: **Java Web Services**. Rio de Janeiro: Alta Books. 2002.
INFOWESTER: Disponível em <http://www.infowester.com/lingjava.php> Acesso em maio de 2010.

Ricarte,I - UNICAMP **Programação Orientada a Objetos: Uma Abordagem com Java <**http://www.dca.fee.unicamp.br/cursos/PooJava/Aulas/poojava.pdf>
. Acesso em maio de 2010.

JAVA - http://www.java.com/pt_BR/download/faq/whatis_j2me.xml Acesso em maio 2010.

SILVEIRA, P. E. A. e COSENTINO, R. A.: FJ-21 Java para desenvolvimento web. Caelum Ensino e Soluções em Java, 2009.

(MASDEVAL, C - Introdução a Java Server Pages, 2007.

TEMPLE, A.; **Programação Web com Jsp, Servlets e J2EE** - André Temple, Rodrigo Fernandes de Mello, Danival Taffarel Calegari, Maurício Schiezaro, 2004.

TEMPLE, A.; MELLO R. F.; CALEGARI D. T.; SCHIEZARO M. **– Tutorial – JSP, Servlets e J2EE** – 2004. www.students.ic.unicamp.br/~jugic/j2ee_primeiros_passos/livro-v03-figuras.pdf. Acesso em maio 2010.

W3SCHOOLS.: **XML Tutorial**. Disponível em: <http://www.w3schools.com/xml/>. Acesso em junho de 2010.

XMLDESIGNER.: **Backgrounder da Tecnologia XML**, Disponível em: <http://msdn.microsoft.com/pt-br/library/8ktdfywf4(VS.80).aspx>. Acesso em junho de 2010.

ARMSTRONG, E.; BALL, J.; BODOFF, S.; CARSON, D.; FISHER, M.; JENDROCK, E. - **The Java™ Web Services Tutorial**, 2003.

PITANGA, T. Grupo de Usuarios Java
< www.guj.com.br/content/articles/jsf/jsf.pdf>. Pitanga, T.: Acesso em maio de 2010.

SILVEIRA, P. E. A. e COSENTINO, R. A.: FJ-21 **Java para desenvolvimento web**. Caelum Ensino e Soluções em Java, 2009.

LINHARES, M.; **Introdução ao Hibernate 3** - grupo de usuários Java <www.guj.com.br/.../hibernate/intruducao_hibernate3_guj.pdf>. Maurício Linhares, Acesso em maio de 2010

GUJ http://www.guj.com.br/content/articles/hsqldb/hsqldb_guj.pdf. Carlos Emilio P. Severo, Acesso em maio de 2010

DESTRO, D**.; Tutorial do Apache Tomcat 1.3**, 2006.

APACHE - The Apache Software Foundation http://tomcat.apache.org. Acesso em maio de 2010.