UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ELANNE CRISTINA OLIVEIRA DOS SANTOS

SISTEMAS JAVA DE EDUCAÇÃO A DISTÂNCIA NA INTERNET

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: CARLOS ANDRÉ GUIMARÃES FERRAZ

RECIFE, FEVEREIRO/2003

Agradecimentos

Agradeço a Deus.

A meus familiares, pela compreensão e carinho nos momentos mais difíceis.

Ao CEFET-PI (Centro Federal de Ensino Tecnológico do Piauí), que tornou possível esta caminhada.

A meu orientador, Carlos André Guimarães Ferraz, pelo incentivo.

A meu colega de mestrado, Fabio de Jesus, pelas dicas de Java. Este trabalho foi concluído com sucesso graças a ajuda de todos vocês. Obrigada!

Elanne Cristina Oliveira dos Santos

Resumo

A Educação a Distância (EAD) vem propondo novas possibilidades de aprendizagem mais de acordo com as necessidades dos dias atuais, se considerarmos que estamos em uma sociedade do conhecimento. Do mesmo modo que hoje demandamos por mais bens materiais, nessa nova sociedade deveremos demandar por mais conhecimento.

Se pensarmos na quantidade de pessoas para serem educadas, na estrutura disponível e no número de educadores com capacidade para facilitar o processo de construção de conhecimento, facilmente chegaremos à conclusão que a EAD é uma solução viável e pode, certamente, ser uma alternativa para corrigir as distorções educacionais no nosso país, por exemplo. Ela pode atender regiões que não dispõem de especialistas e atingir maior número de pessoas.

Basicamente a EAD consiste em se realizar ensino-aprendizagem em que o aluno e o professor não se encontram em um mesmo ambiente físico. Esta idéia vem sendo bastante discutida nos meios acadêmicos e defendida em aplicações na Internet atual. Duas razões têm contribuído particularmente para o desenvolvimento desta área: a proliferação de recursos de informática e o grande avanço na tecnologia de comunicação.

Este trabalho se propõe a levantar características de tecnologias Java aplicadas a desenvolvimento de sistemas de Educação a Distância para Internet. Esses pressupostos servirão de base para a especificação e implementação de um protótipo de um sistema EAD. Com isto o trabalho estará contribuindo para o avanço da linha de pesquisa sobre sistemas de Educação a Distância na Internet e para o bom aproveitamento desta rede para aplicações de educação a distância.

Abstract

Distance Education (DE) has been providing new possibilities of learning more accordingly to the daily needs, if we consider that we are in a society where knowledge is a must. The same way that today we demand for more possessions, in this new society we also demand for more knowledge.

If we think about the amount of people to be educated, the available structure and the number of educators enabled to make the process of knowledge building easier, we can easily conclude that DE is a possible solution, and certainly can be an alternative to correct education distortions in our country, for example. It can cover areas that do not have experts and reach a higher number of people.

Basically the DE consists of accomplishing the teaching-learning process where the student and the teacher are not in the same physical place. This idea has been very much discussed in the academic environment and present in current Internet applications. Two reasons specifically contributed to the development of this field: the proliferation of informatics resources and the great advance in the communication media technology.

This work has the purpose of enlisting characteristics of Java applied to the development of DE systems for the Internet. These conjectures will be the basis for the specification and implementation of a prototype of a DE system. Thus, the paper will be contributing for the advancement of the research about Distance Education systems in the Internet and for the good utilization of this web for the application of Distance Education.

Índice

Índice de Figura

Índice de Tabela

Capítulo 1 Introdução

Este capítulo apresenta uma introdução sobre os temas citados na dissertação, a motivação deste trabalho, seus objetivos gerais e específicos e a organização desta dissertação.

Capítulo 1 Introdução

A Educação a Distância (EAD) vem propondo novas possibilidades de aprendizagem mais de acordo com as necessidades dos dias atuais, se considerarmos que estamos em uma sociedade do conhecimento. Do mesmo modo que hoje demandamos por mais bens materiais, nessa nova sociedade deveremos demandar por mais conhecimento.

Se pensarmos na quantidade de pessoas para serem educadas, na estrutura disponível e no número de educadores com capacidade para facilitar o processo de construção de conhecimento, facilmente chegaremos à conclusão que a EAD é uma solução viável e pode, certamente, ser uma alternativa para corrigir as distorções educacionais no nosso país, por exemplo. Ela pode atender regiões que não dispõem de especialistas e atingir maior número de pessoas.

Basicamente a EAD consiste em se realizar ensino-aprendizagem em que o aluno e o professor não se encontram em um mesmo ambiente físico. Esta idéia vem sendo bastante discutida nos meios acadêmicos e defendida em aplicações na Internet atual. Duas razões têm contribuído particularmente para o desenvolvimento desta área: a proliferação de recursos de informática e o grande avanço na tecnologia de comunicação.

1.1 Motivação

Este trabalho se propõe a levantar características de tecnologias Java aplicadas a desenvolvimento de sistemas de Educação a Distância para Internet. Esses pressupostos servirão de base para a especificação e implementação de um protótipo de um sistema EAD. Com isto o trabalho estará contribuindo para o avanço da linha de pesquisa sobre

sistemas de Educação a Distância na Internet e para o bom aproveitamento desta rede para aplicações de educação a distância.

O estudo de um sistema baseado em objetos distribuídos é mais um ponto de interesse, visto as vantagens dos sistemas Orientados a Objetos. Esta oportunidade inclui o estudo do middleware Java RMI (*Remote Method Invocation*) e outros.

Este trabalho propõe duas arquiteturas de desenvolvimento para um sistema de Educação a Distância na Internet baseados em tecnologia Java. Está prevista também a construção de um protótipo para avaliar o uso das tecnologias envolvidas e para validar a arquitetura proposta.

1.2 Objetivos

1.2.1 Geral

Esse documento apresenta um plano de dissertação de mestrado que tem por objetivo principal pesquisar e analisar tecnologias Java para desenvolvimento de sistemas de Educação a Distância na Internet baseados em conceitos de orientação a objetos, apresentar modelos de arquitetura e implementar um protótipo de um sistema de Educação a Distância que deverá se utilizar da manipulação de informações multimídia.

Ao alcançar este objetivo, o trabalho estará contribuindo para a consolidação de metodologias de desenvolvimento e funcionamento de sistemas Java de Educação à Distância na Internet.

1.2.2 Específicos

- Especificar características de EAD;
- Especificar tecnologias Java disponíveis para desenvolvimento de ambientes para Web;
- Analisar "middleware" com relação ao seu funcionamento num sistema EAD para Internet;
- Adotar um modelo de desenvolvimento de um sistema EAD na Internet, propondo duas possíveis arquiteturas, uma baseada em *servlet* Java e a outra usando o

middleware Java RMI;

- Implementação de um protótipo para validação do modelo.

1.3 Organização da dissertação

Neste primeiro capítulo foram abordadas as motivações para este trabalho, bem como seus objetivos.

No segundo capítulo será abordado o tema Educação a Distância, incluindo características básicas, elementos envolvidos e alguns modelos já utilizados atualmente.

No terceiro capítulo serão abordadas as tecnologias disponíveis no desenvolvimento de sistemas para Internet, incluindo descrição para suporte a aplicações distribuídas como: DCOM, CORBA, JAVA/RMI, JAVA/SERVLETS, JSP, WEB SERVICE e JAVA/WEB SERVICE, incluindo uma análise comparativa.

No quarto capítulo será apresentada a modelagem de um sistema Java, onde para cada necessidade de um sistema EAD será proposta uma tecnologia, suas características e vantagens.

No quinto capítulo será descrito o protótipo desenvolvido, duas propostas de arquitetura para o desenvolvimento e analisadas suas características e limitações.

No sexto capítulo serão apresentados os comentários conclusivos e destacadas as contribuições deste trabalho e sugestões de trabalhos futuros.

Capítulo 2 Educação a Distância

Este capítulo apresenta características da Educação a Distância, tais como conceitos e fundamentos, mídias utilizadas, elementos de percepção em um ambiente para EAD e alguns modelos utilizados.

Capítulo 2 Educação a Distância

2.1 Conceitos e Fundamentos da EAD

A Educação a Distância (EAD) pode ser definida em função de alguma de suas características:

- Separação professor-aluno: docente não se faz presente, mas transmite conhecimentos ao aluno, suscita sua aprendizagem através do planejamento da instrução, do qual participou, e dos recursos didáticos que elaborou.

- Utilização de meios tecnológicos: atualmente não existem distâncias nem fronteiras para o acesso à informação e à cultura. Os recursos tecnológicos de comunicação (impressos, áudios, vídeos etc.), acessíveis a boa parte da população, têm possibilitado o grande avanço da educação a distância e se convertido em mais oportunidades de acesso ao conhecimento e em democratização das possibilidades da educação. A escolha e a utilização dos recursos didáticos em programas de EAD dependem do diagnóstico da população-alvo e do planejamento de instrução previamente estabelecido.

- Comunicação massiva: As novas tecnologias da informação e os modernos meios de comunicação tornaram inesgotáveis as possibilidades de recepção de mensagens educativas, eliminando fronteiras espaço-temporais e propiciando o aproveitamento destas mensagens por grande número de pessoas, dispersas geograficamente.

O interesse de se elaborar projetos de EAD na Internet vem de razões indiscutíveis: a proliferação de recursos de informática e o grande avanço na tecnologia de transmissão de dados.

Quando se fala na rede mundial atual não se pode ignorar as limitações existentes para a implementação da EAD. Deve ser possível oferecer, via rede, aos alunos a possibilidade de acessar serviços típicos de uma "escola via Web" [Moura2002], seja para ler uma apostila, assistir a um vídeo de uma aula, submeter uma tarefa atribuída pelo professor ou apenas consultar as notas e que permita a professores e alunos manter contato com diferentes níveis de interação, que podem ir do simples correio eletrônico à videoconferência, intermediando, assim, a relação ensino-aprendizagem. Para que essas opções se tornem uma realidade eficiente tem que se dar tratamento especial a dados de áudio e vídeo, pois não é conveniente, no caso da realização de uma vídeo-conferência, que as taxas de transmissão das duas mídias estejam em defasagem uma da outra. A estrutura da rede que vai transportar esses dados deve ser observada considerando-se algumas funções para o controle de Qualidade de Serviço (QoS). Também é de extrema importância uma resposta do sistema em tempo real, o que restringe a utilização de redes com banda estreita ou meios de comunicação estritamente limitados.

Diante dos problemas apresentados, novas tecnologias de redes e sistemas de comunicação estão surgindo, assim como a Internet 2, a nova geração da Internet. Assim, espera-se tornar possível a construção de software que atenda a requisitos estritos de escala, comunicação e desempenho.

2.2 Abordagens de Mídia para EAD

Há duas categorias de sistema de distribuição de EAD: a síncrona e a assíncrona. EAD síncrona requer a participação simultânea de estudantes e professores, tendo como vantagem o fato de a interação ser feita "em tempo real". Ela pode ser realizada através de TV interativa, teleconferência, vídeo-conferência e/ou os "chats" da Internet.

A modalidade assíncrona não requer a participação simultânea de todos os estudantes e professores. Assim, os estudantes não precisam se encontrar ao mesmo tempo. Em vez disso, eles podem escolher seu próprio ritmo para a aprendizagem e podem obter os conteúdos de acordo com a sua programação. Esta modalidade assíncrona é mais flexível do que a síncrona.

A modalidade assíncrona permite a disseminação de conhecimento para a comunidade através de vários meios de comunicação (como o correio eletrônico, listas de discussão, apresentação de vídeos etc).

As vantagens da distribuição assíncrona incluem a escolha do estudante quanto ao lugar e ao tempo. Entretanto, muitas vezes, os recursos disponíveis para comunicação assíncrona na Internet são subutilizados e o conteúdo de cursos à distância são disseminados em simples páginas *html* utilizando grande volume de material escrito [CCEAD2002].

Há quatro categorias de opções tecnológicas disponíveis para o educador a distância [CCEAD2002]:

- **Voz**

As ferramentas áudio-educacionais incluem as tecnologias interativas do telefone e de tele-conferência (de sentido único). As ferramentas-áudio passivas incluem CD-Rom e rádio.

- **Vídeo**

As ferramentas de vídeo incluem imagens imóveis (corrediças), imagens móveis pré-produzidas (videocassette) e imagens ativas em tempo-real combinadas com tele-conferência (vídeo de sentido único ou em dois sentidos com áudio em dois sentidos).

- **Dados**

Os computadores emitem e recebem a informação eletronicamente. Por esta razão, o termo "dados" é usado para descrever essa categoria abrangente de ferramentas educacionais. As aplicações de computador para EAD são variadas e incluem:

- **"Computer-Assisted Instruction" (CAI)** – uso do computador como mediador para apresentação de aulas individuais;

- **"Computer-Managed Instruction (CMI)** - o computador organiza registros do estudante e da trilha de seu progresso; o ensino propriamente dito não necessita ser distribuído através de um computador, embora CAI seja combinado, frequentemente, com CMI;

- "Computer-Mediated Education (CME) - descreve as aplicações do computador que facilitam a distribuição do ensino. Exemplos: correio eletrônico, fax, conferência via computador em tempo-real e aplicações World-Wide Web.

- **Impresso**

É um elemento fundamental dos programas de EAD, a partir do qual todos os sistemas de distribuição restantes evoluíram. Os vários formatos de impresso incluem livros-texto, guias de estudo, manuais de instrução, ementa do curso e estudos de caso.

2.3 Elementos de percepção em um Ambiente para EAD

Os ambientes virtuais de trabalho e de aprendizagem, através da interconexão de máquinas fornecida pelas redes de computadores e pela Internet, visam facilitar atividades em grupo. Portanto, estes ambientes devem prover elementos de percepção de forma a permitir a coordenação em tarefas cooperativas, principalmente onde a comunicação direta não ocorre. Além disso, devem prover elementos de percepção para que indivíduos possam interpretar eventos, prever possíveis necessidades e transmitir informações de maneira organizada. Perceber as atividades dos outros indivíduos é essencial para o fluxo e naturalidade do trabalho, e para diminuir as sensações de impessoalidade e distância, comuns nos ambientes virtuais [Gerosa2002].

A percepção dentro de um ambiente envolve vários aspectos cognitivos relativos à habilidade humana. Enquanto a interação entre pessoas e ambiente dentro de uma situação face-a-face parece natural, visto que sentidos como visão e audição estão disponíveis em sua plenitude, a situação fica menos clara quando há a tentativa de fornecer suporte à percepção em ambientes virtuais [Gerosa2002].

O projetista de ambientes virtuais deve prever quais informações de percepção são relevantes e como elas podem ser geradas, onde elementos de percepção serão necessários e de que forma apresentar estes elementos. Deve-se tomar cuidado para que os elementos realmente auxiliem a cooperação e não a dificultem. O excesso de informação pode causar sobrecarga e atrapalhar a comunicação.

A colaboração de pessoas com diferentes entendimentos, pontos de vista alternativos e habilidades complementares pode gerar resultados que dificilmente seriam encontrados individualmente. Resolver uma tarefa em um grupo de trabalho colaborativo é potencialmente mais proveitoso do que resolvê-la individualmente. Os membros do grupo podem auxiliar o pensamento individual, prover *feedback* e buscar em conjunto idéias, informações e referências para auxiliar na resolução dos problemas. Geralmente, o grupo tem mais capacidade de gerar criativamente alternativas, levantar as vantagens e desvantagens de cada uma, selecionar as viáveis e tomar decisões do que os indivíduos separadamente [Gerosa2002].

Apesar de suas vantagens, trabalhar em grupo demanda uma necessidade muito forte de coordenação de seus membros. Sem esta coordenação boa parte dos esforços de cooperação e de comunicação podem não ser aproveitados.

Para possibilitar a coordenação do grupo são necessárias informações sobre o que está acontecendo para que seja possível tomar decisões adequadas sobre os procedimentos a serem tomados para favorecer a cooperação. Estas informações são fornecidas através de elementos de percepção que capturem e condensem as informações coletadas sobre a interação dos participantes.

A seguir serão apresentados os elementos de percepção de um ambiente para EAD [Gerosa2002]:

- Comunicação: Comunicar é compartilhar informações. A comunicação pode se dar através de várias formas, maneiras e níveis. A utilidade da informação que está sendo comunicada pode ser atingida apenas se houver entendimento entre o comunicador e o receptor. Isto significa que a informação tem que ser percebida no contexto em que ela foi inserida e com base nas regras e linguagens estabelecidas para a comunicação.

- Coordenação: Comunicação para ação gera compromissos. Para garantir o cumprimento desses compromissos e para a organização do grupo é necessária a coordenação das atividades. Sem esta coordenação, boa parte do esforço da comunicação é perdido, não sendo revertido para cooperação. Deve-se disponibilizar elementos de percepção para prover informações sobre o que fazer e sobre o que os companheiros estão fazendo. Sem tal contexto, os indivíduos não podem medir a

qualidade de seu próprio trabalho com respeito aos objetivos e progressos do grupo. Outra situação de coordenação onde se fazem necessárias informações de percepção é quando o indivíduo precisa saber o que fazer para prosseguir seu trabalho. Com estas informações, a coordenação das ações nas atividades cooperativas torna-se possível, de forma que as ações sejam realizadas na ordem correta, no tempo correto e cumprindo suas restrições.

O gerente, líder ou facilitador do grupo também precisa de informações de percepção. Ele necessita saber, por exemplo, quem está e quem não está trabalhando, com quem está ocorrendo conflitos de interesse e as habilidades e experiências de cada um. Com base neste tipo de informação ele pode tomar as decisões adequadas para a coordenação do grupo.

Falha na coordenação ocorre quando há uma discordância entre as expectativas de um participante e as ações de outro. Possivelmente ocorre devido a um erro do dipositivo de comunicação ou de percepção, ou de diferenças da interpretação da situação ou de interesse. A coordenação deve atuar para que os participantes resolvam conflitos na tentativa de estabelecer novamente o canal de comunicação e de cooperação.

- Cooperação: É o trabalho conjunto dos indivíduos em torno de uma meta ou objetivo. Isto significa que os indivíduos usarão seus conhecimentos para apoiar o desenvolvimento do trabalho compartilhado, aproveitando as novas informações obtidas para aperfeiçoar o seu próprio conhecimento.

A interação entre os indivíduos, ou entre um indivíduo e os artefatos de um ambiente de trabalho, em geral tem como meta alcançar o objetivo do trabalho em grupo. Como resultado destas interações há um série de novos acontecimentos que implicarão em um conjunto de informações que, por sua vez, irão gerar uma estrutura cognitiva onde os indivíduos buscarão conhecimentos para planejar e coordenar interações posteriores. A finalidade dos elementos de percepção na cooperação é fornecer este contexto que possibilite ao participante agir na direção do objetivo comum complementando as atividades de seus companheiros. Portanto, a cooperação requer fundamentalmente coordenação das atividades e compartilhamento das informações. Os elementos de percepção também possibilitam antecipar ações e necessidades e conhecer

as intenções dos companheiros do grupo, de forma a tornar possível prestar assistência ao trabalho deles quando for possível e necessário.

2.4 Modelos para Ensino a Distância

Existem vários experimentos de utilização da Internet para EAD. Um exemplo simples de uma efetiva prática de utilização da Internet para fins de Educação à Distância é o uso da infra-estrutura da *web* para divulgação de material didático através de páginas HTML (*Hypertext Markup Language*), comunicação de forma assíncrona através de *e-mail*, e uso de *newgroups* para esclarecimento de dúvidas [Trinta2000]. Porém, tal forma de utilização subutiliza o potencial que a Internet pode oferecer para a Educação a Distância. Uma utilização eficaz da Internet para fins educacionais deve ser realizada através de ambientes, sistemas desenvolvidos por instituições acadêmicas ou empresas privadas, onde são divulgados cursos, realizadas aulas e interações entre professores e alunos. Atualmente existe um bom número destes sistemas e ambientes que se dispõem a realizar EAD através da Internet, como por exemplo o Aulanet™ da Puc-Rio [AulaNET2003] e o Projeto Virtus [Virtus2003] da Universidade Federal de Pernambuco (UFPE).

Embora tais ambientes tenham obtido sucesso na utilização da Internet para EAD, muito ainda há para se evoluir. A maior parte dos ambientes e sistemas que utilizam Internet como meio para EAD, estão baseados num modelo centralizado de *Website*, com material didático exposto na forma de páginas HTML e comunicação assíncrona entre aluno e professor [Trinta2000]. No entanto, a visão mais adequada para Educação à Distância utilizando a Internet seria a de “comunidades virtuais”, onde através das quais, grupos de professores cooperam entre si para produção de cursos, materiais didáticos, etc. para grupos de alunos [Baron97] [Carver99].

A seguir serão apresentados alguns modelos de Ensino a Distância:

2.4.1 AulaNet

O AulaNet é um ambiente baseado numa abordagem groupware para a criação, aplicação e gerenciamento de cursos pela Internet. Ele vem sendo desenvolvido desde Junho de 1997 pelo Laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Sua abordagem *groupware* facilita a colaboração entre alunos e instrutores , o que favorece o aprendizado [Gerosa2002].

Os serviços AulaNet são organizados baseados nos conceitos de elementos de percepção já vistos anteriormente. Eles são divididos em serviços de comunicação, de coordenação e de cooperação. Os serviços são colocados à disposição do docente durante a criação e atualização do curso, permitindo a ele selecionar e configurar quais ficarão disponíveis aos participantes do curso.

Os serviços de comunicação fornecem as facilidades que permitem a troca e o envio de informações [Gerosa2002]. Estes serviços incluem um mecanismo de discussão textual assíncrona no estilo de fórum (Grupo de Interesse), de conferência síncrona textual no estilo de *chat* (Debate), de troca instantânea de mensagens com participantes simultaneamente conectados (Contato com os Participantes), e de correio eletrônico individual com o instrutor (Contato com os Docentes) e com toda a turma (Grupo de Discussão).

Um indivíduo trabalhando sozinho lida com sua própria base de conhecimento e de fontes de informação, e tem que se organizar para resolver a tarefa. Ao trabalhar em grupo, diversos problemas complexos de coordenação aparecem. Os serviços de coordenação visam minimizar estes problemas, organizando o grupo para possibilitar a cooperação, com mecanismos de gerenciamento da agenda do grupo e da competência, entre outros. No AulaNet os serviços incluem uma ferramenta de notificação (Avisos), uma ferramenta de coordenação básica do fluxo do curso (Plano de Aulas), ferramentas de avaliação (Tarefas e Exames) e uma ferramenta de acompanhamento da participação do grupo (Relatórios de Participação).

Os serviços de cooperação fornecem meios para a aprendizagem cooperativa, para a resolução de problemas e para a co-autoria de cursos. Mesmo sabendo que o desempenho do grupo na resolução de tarefas é extremamente dependente de fatores contextuais, como composição do grupo, características e habilidades dos membros,

tipo do problema e suporte tecnológico, os mecanismos de cooperação suportam a interação entre os participantes de forma a minimizar as dificuldades contextuais [Gerosa2002]. No AulaNet, os serviços de cooperação incluem uma lista de referências do curso (Bibliografia e Webliografia), uma lista de conteúdos transferíveis para consumo desconectado (Download) e facilidades de co-autoria, tanto de docentes (Co-autoria de docentes) quanto de aprendizes (Co-autoria de Aprendiz).

Para navegar no curso, o participante do AulaNet tem a sua disposição um menu que fornece uma facilidade de navegação construída através da seleção prévia, feita pelo docente, dos mecanismos de comunicação, coordenação e cooperação. Na parte superior do menu, é disponibilizado para o usuário o código da disciplina em questão, oferencendo um mecanismo de percepção de localização. Os itens do menu oferecem a percepção de quais são as opções disponíveis no momento para o participante. Alguns dos itens são:

- Mensagem aos Docentes
- Contato com Participantes
- Grupo de Discurssão
- Grupo de Interesse
- Debate
- Plano de Aulas
- Tarefas
- Bibliografia
- Webliografia
- Documentação
- Relatórios de Participação

Sempre que é apresentada uma lista de temas que o participante pode escolher, como no caso dos temas das aulas no Plano de Aulas ou do fórum no Grupo de Interesse, são mostrados, além do nome do tema, entre parênteses, a quantidade de itens não lidos e o total de itens daquele tema. Com isso, o participante pode tomar

antecipadamente a decisão se vale à pena acessar o tema e tem uma idéia do volume de trabalho que ainda tem pendente.

Ao listar as mensagens dos serviços de comunicação assíncronos do ambiente, na opção de “Mensagem aos Docentes”, são oferecidos informações de percepção que ajudam o participante a contextualizar a mensagem, a decidir se vai acessá-la no momento ou a localizar alguma informação que esteja procurando. Algumas das informações são extraídas automaticamente, como a data do envio e o autor, mas outras como o título da mensagem e sua categoria, precisam ser fornecidas explicitamente. Estas informações dão idéia de tempo, de autoria e do conteúdo da mensagem [Gerosa2002].

O AulaNet oferece um serviço denominado “Relatórios de Participação”. Estes relatórios visam favorecer a percepção do grupo sobre as atividades dos participantes. Há relatórios que sumarizam a quantidade e a qualidade das contribuições. A informação de quantidade pode ser extraída automaticamente pelo ambiente, mas não é possível avaliar qualitativamente as contribuições. Esta informação tem que ser fornecida pelos instrutores do curso. Cada contribuição – mensagens, participação em debates, submissão de conteúdos e resolução de tarefas – é conceituada pelo docente. O coordenador do curso escolhe os nomes, quantos são e a que faixa de notas correspondem os conceitos, que podem ser diferentes para eventos síncronos e assíncronos. Um dos objetivos dos relatórios é ajudar os participantes a se conhecerem melhor e a escolherem seus companheiros para formação de grupos. Também fornecem subsídios para que o instrutor organize o grupo, motivando participantes que não estejam contribuindo, e cobrando tarefas pendentes.

2.4.2 Um *Framework* para Ensino a Distância via Web

Uma das áreas da Engenharia de Software procura melhorar a qualidade dos processos e produtos de software, bem como a redução dos esforços e custos de produção, com a reutilização de recursos. Um destes recursos são os *frameworks* construídos usando padrões de projeto. Um *framework* é descrito por um conjunto de classes abastratas e concretas relacionadas, que são estendidas para desenvolver aplicações do mesmo domínio [Sanches2002]. A utilização de padrões de projeto em

sistemas complexos de software, suporta o reuso de soluções, previamente testadas, tornando mais fácil desenvolvimento e manutenção do software.

Um *framework* proporciona ao desenvolvedor funcionalidade pré-fabricada e extensibilidade das suas classes para as aplicações que o reutiliza. A figura abaixo mostra a arquitetura de um *framework*, que integra diferentes tecnologias: um *browser*, Protocolo HTTP, Servidor Web, o FEDW (*Framework* para Ensino a Distância via Web), o Servidor de Objetos, o SGBD e o Sistema Operacional:

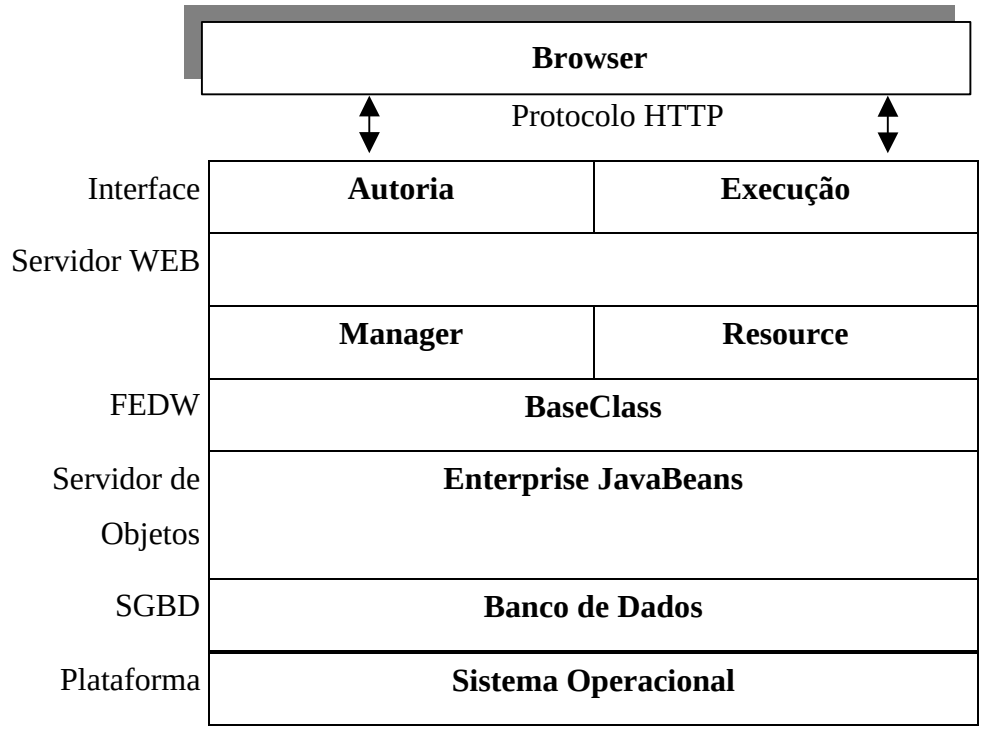

Figura 2.1 Arquitetura de um *Framework* de EAD via Web [Sanches2002]

Um *Framework* para Ensino a Distância via Web estruturado no pacote FEDW é composto pelos pacotes *Manager*, *BaseClass* e *Resource*, que representam o domínio das funções de negócio do *framework*, como mostra a figura 2.2 a seguir.

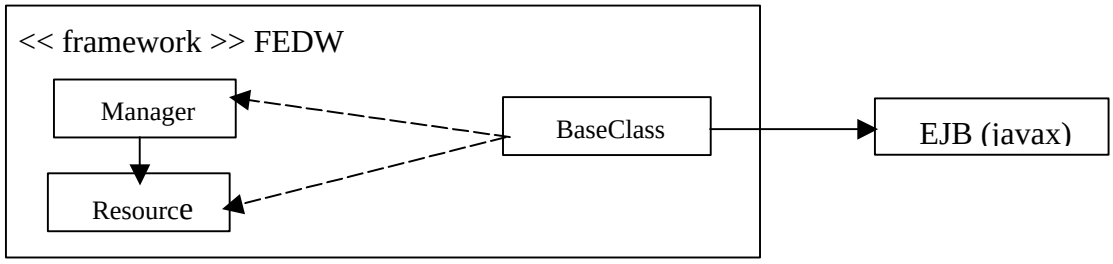

Figura 2.2 Estrutura do Framework para Ensino a Distância via Web

O pacote *BaseClass* contém as classes Interfaces que especificam as classes básicas com a declaração dos métodos comuns que tratam a manipulação e identificação dos objetos de negócio do *framework*.

O pacote *Manager* compõe as classes: *User, Register, Qualification, Structure, Content, Historic e Planning.*

O tipo *User* mantém características que podem representar, na aplicação, objetos Usuários como: Aluno, Professor e Autor. O tipo *Register* generaliza as características de um objeto Matrícula, como outro objeto que representa o registro de um aluno no Curso. O tipo *Qualification* descreve as características que representam um objeto Curso, composto de uma *Structure*, que generaliza a estrutura, as aulas do objeto Curso, podendo ter objetos *Content*, que representam os objetos Arquivo de uma Aula. *Structure* pode estar relacionada com objetos *Planning* que representam o Cronograma de um Curso, para compor as características dos objetos do tipo *Historic*. *Historic* generaliza objetos do tipo Horário de aulas de uma aplicação EAD.

O pacote *Resource* mantém as classes: *TypeResource, Service, Skill, Task, Response, TaskDescription, List* e *Catalogue* para representarem os objetos: Agenda, Avaliação, Questionário, Lista de Discussão e Serviços.

As ferramentas para desenvolvimento de cursos pela Web são emergentes no mercado e refletem um forte domínio de sistemas de softwares para investimento, face a evolução das tecnologias de comunicação.

Diferentes técnicas podem ser integradas para compor um ambiente que facilita a construção de aplicações para EAD via Web, através do reuso e com base nos princípios da Orientação a Objeto.

A seguir veremos algumas técnicas de desenvolvimento para a construção de um *framework* para Ensino a Distância via Web.

Catalysis

Catalysis [Sanches2002] é um método de desenvolvimento de software Orientado a Objetos que utiliza Componentes Distribuídos, Padrões e *Frameworks* para projetar e construir Sistemas de Negócio, que devem suportar tecnologias orientadas a objetos recentes como Java, CORBA e DCOM. Sua notação é baseada na *Unified Modeling Language (UML)* e fundamenta-se em três princípios básicos: abstração,

precisão e componentes “plug-in”. O princípio de *abstração* orienta o desenvolvedor na busca dos aspectos essenciais do sistema, dispensando detalhes que não são relevantes para o contexto do sistema. O princípio *precisão* tem como objetivo descobrir erros e inconsistências na modelagem e o princípio *componentes “plug-in”* suportam o reuso de componentes para construir outros componentes [Sanches2002].

O processo de desenvolvimento de Sistemas de Negócio em *Catalysis* é dividido em três níveis: Domínio do Problema, Especificação dos Compontes e Projeto Interno dos Componentes. No nível *Domínio do Problema* é especificado “o quê” o sistema deve fazer para solucionar o problema. No nível *Especificação dos Componentes* são descritas especificações precisas do comportamento dos objetos, e no nível do *Projeto Interno dos Componentes*, define-se como serão implementados os requisitos dos objetos para componentes e com será feita a distribuição física desses componentes.

O processo de desenvolvimento do *framework* em *Catalysis* envolve as etapas de construção do Modelo do *Framework* e do Modelo das Colaborações do *Framework:*

- Na etapa de desenvolvimento do *Modelo do Framework* é dada ênfase na especificação estática do *framework*, que representa apenas os tipos com os seus atributos.
- Na etapa de desenvolvimento do *Modelo das Colaborações do Framework* é dada ênfase na especificação dinâmica do *framework*, que descreve as colaborações que interagem entre os grupos de objetos.

Linguagem Java e Enterprise JavaBeans

O *framework* para o Ensino a Distância via Web, sendo específico para o domínio de Ensino a Distância via tecnologia de comunicação Web, pode ser implementado na linguagem Java que dispõe de facilidades para o desenvolvimento de aplicações que são executadas via Internet em multiplataforma. A tecnologia específica de Java, denominada *Enterprise JavaBeans* (EJB), pode ser escolhida como Servidor de Objetos, na medida em que facilita o desenvolvimento de aplicações *multi-tier*, utilizando componentes orientados a transação e baseado em servidores escritos em Java [Sanches2002].

Um programa Cliente faz remotamente uma chamada ao Servidor EJB e se comunica, por intermédio das *Interfaces Home* e *Remote* residentes no *container* do Servidor EJB, com os objetos do Servidor EJB.

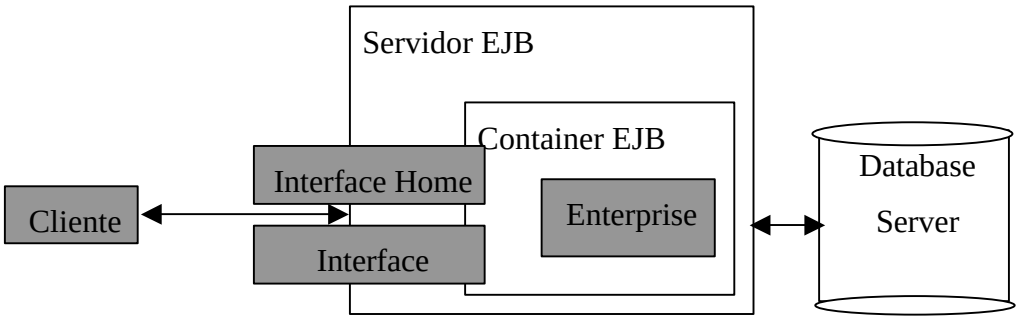

Figura 2.3 Arquitetura Lógica do sistema EJB

O Servidor EJB mantém os componentes EJB na camada intermediária, dentro de um *container*. O *container* fornece serviços de conexão com a rede, disponibilizando os *beans* (componentes Java que agem no servidor e proporcionam suporte embutido para serviços de aplicações como transações, segurança e conectividade de banco de dados) e seus serviços de suporte às transações, ao gerenciamento de múltiplas instâncias, à persistência dos objetos e à segurança.

O servidor de BD reside na terceira camada. *Beans* EJB podem acessar diretamente a Base de Dados, via *Java Database Connectivity* (JDBC), ou indiretamente pelo *container* [Sanches2002]. O JDBC é um conjunto de classes para integração da linguagem Java com banco de dados relacionais. O JDBC é uma API que possibilita a adição de comandos SQL em uma aplicação Java.

2.4.3 Co-Autoria Distribuída de Aulas para Ensino a Distância

A Educação a Distância na Internet tem como uma de suas principais vantagens em relação a outras formas de ensino a distância, a aprendizagem cooperativa, através da qual estudantes reúnem-se em grupos, ajudando uns aos outros no processo de aprendizagem, atuando como parceiros entre si, e também com os professores, com o objetivo de adquirir conhecimento sobre um dado domínio [Trinta2000]. De fato, a idéia de trabalho cooperativo é mais vinculada a grupos de alunos, onde tarefas eram divididas entre os integrantes de um grupo, e no final o trabalho de todos era reunido,

formando com isso uma solução mais completa a um problema. Porém, com a Internet, há a possibilidade que o mesmo conceito possa ser estendido também a um grupo de professores permitindo que os mesmos possam interagir em grupos para a construção, e até apresentação de aulas e cursos à distância para grupos de alunos.

A possibilidade de cooperação pode ser bastante proveitosa para os professores, visando uma troca de experiências e abordagens educacionais, possibilitando a produção de materiais de aulas com conteúdo ainda mais rico, além de se apresentar como uma excelente oportunidade para a reciclagem entre professores. Embora tal possibilidade apresente tais benefícios, são poucos os ambientes desenvolvidos atualmente de EAD na Internet que apresentam algum tipo de ferramenta que permita tal interação entre professores.

Na maioria dos atuais ambientes de EAD na Internet, ocorre até uma certa colaboração entre autores de material didático, onde, por exemplo, um professor associa *links* ao seu *site* para outros *sites* que contêm material que servirão de apoio a suas aulas, objetivando alcançar um conteúdo mais abrangente para seu próprio material. Porém, neste caso, não existe real colaboração entre professores [Trinta2000].

A idéia da co-autoria é distribuir recursos de produção de materiais de cursos entre instrutores e alunos, com ou sem distinção ou restrições. Como resultado, teríamos uma constante renovação dos materiais dos cursos e o aluno se sentiria estimulado a participar de um ambiente que se utilize desta característica de autoria.

O conteúdo das aulas também é um fator que merece atenção para autores. Atualmente, os recursos utilizados por autores de cursos na Internet baseiam-se em páginas HTML. A maior parte do material divulgado nas formas citadas anteriormente são baseados em textos, gráficos e imagens estáticas. A utilização de mídias contínuas, como vídeo e áudio, é quase inexistente. Deve ser permitido ao autor de um curso inserir em suas aulas, trechos de áudio e vídeo, pois tais recursos facilitam e muito o aprendizado.

Co-autoria de aulas utilizando CORBA

Em [Trinta2000] são apresentadas três arquiteturas propostas, onde cada elemento é apresentado como um objeto distribuído CORBA. As arquiteturas apresentam também elementos definidos pela arquitetura CORBA, como serviço de

nomes, no qual todo objeto CORBA deve se registrar, de forma a poder ser referenciado por outro objeto. Outro objeto que faz parte de todas as arquiteturas apresentadas é o *Internet Trader* [Macêdo2000], que representa a implementação de um serviço de *trading* entre objetos distribuídos. No paradigma de objetos distribuídos, vários objetos podem oferecer o mesmo tipo de serviço, através de replicação de instância da classe deste objeto. O uso de um *trader* faz com que os objetos clientes possam encontrar objetos servidores baseando-se nos serviços que um objeto oferece, e não pelo nome do objeto. Com isso, um objeto cliente pode escolher dentre os possíveis objetos servidores de um determinado serviço, qual melhor satisfaz sua necessidades em dado instante. O objeto que oferece um serviço se cadastra no *trader*, e a partir deste momento, os demais objetos que necessitam dos serviços oferecidos por este tipo de objeto podem, consultando o *trader*, conseguir a referência a este objeto e utilizar seus serviços. O *Internet Trader* é um serviço específico para encontrar serviços na Internet.

Todas as arquiteturas são baseadas no uso de três elementos principais [Trinta2000]:

Autor: Representa cada usuário do ambiente. Este componente tem, basicamente, como principais funções: o envio e o recebimento de mensagens para/de outros autores, assim como, de um slide trabalhado por um editor e emissão de opiniões sobre edições efetuadas.

Coordenador: Este objeto é responsável pelo controle da consistência da aula. Ele guarda informações como o nome de todos os autores presentes no ambiente, o editor da aula e a fila dos autores que estejam requerendo o acesso à edição da aula em dado instante. O coordenador gerencia também o processo de votação sobre a edição dos slides, emitindo telas para voto dos demais autores, contabilizando o resultado e realizando a ação do editor no caso de aprovação da mesma pelos autores.

Servidor de slides: Representa o conjunto de slides que compõem a aula. Suas funções basicamente refletem a navegação e edição sobre este conjunto, como ir para o próximo slide, ir para o último slide, inserir novo slide, etc. O servidor de slides deve fornecer ainda informações sobre a quantidade de slides cadastrados, assim como qual é o slide que está sendo editado em um dado instante.

Arquitetura com o Coordenador Centralizado

Baseia-se na centralização de toda a comunicação e transmissão de informações entre autores e/ou servidores de slides pelo elemento coordenador. Sendo assim, quando por exemplo, um autor deseja enviar uma mensagem no *chat*, esta mensagem é repassada para o coordenador, que distribui a mesma entre os demais autores presentes.

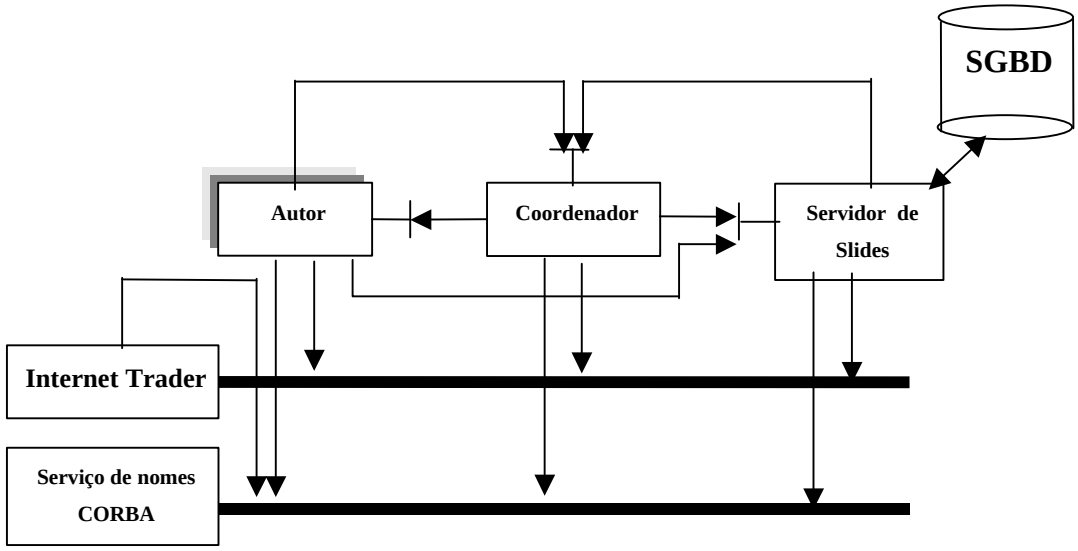

Figura 2.4 Arquitetura com centralização do objeto coordenador

A vantagem desta arquitetura é a facilidade de seu desenvolvimento em relação às demais. A centralização de ações pelo coordenador facilita o tratamento de possíveis erros de conexão entre autores, através do isolamento do autor com problemas dos demais. A desvantagem é a sobrecarga de funções por parte do coordenador, que pode causar problemas de desempenho e escalabilidade.

Arquitetura com o Coordenador Descentralizado

Nesta arquitetura é retirado do coordenador o papel de elemento centralizador da comunicação entre autores e o servidor de slides, fazendo com que as comunicações ocorram diretamente entre estes elementos. Para isto, cada autor possui uma lista local, contendo o nome e a referência de outros usuários. Ao entrar no ambiente, esta lista é preenchida com os dados referentes aos autores presentes no ambiente naquele dado

instante, notificando também a estes autores a sua presença e fazendo com que cada um destes autores insira o novo autor em sua lista local de autores presentes [Trinta2000].

A vantagem desta arquitetura está na diminuição de tráfego de informações pelo coordenador. As principais interações entre os autores (troca de mensagem e navegação entre os slides) são feitas sem o intermédio do coordenador, deixando para este elemento as tarefas de gerenciamento dos processos de votações relativos à edição de aulas e de entrada de novos autores, além de também monitorar as possíveis falhas de comunicação com autores.

A desvantagem desta arquitetura está na complexidade de sua implementação.

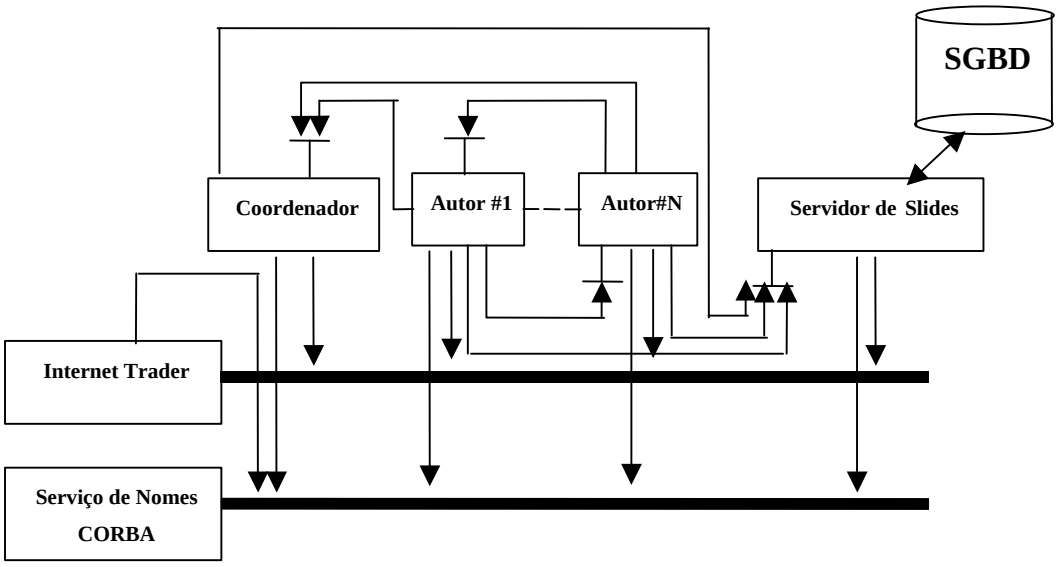

Figura 2.5 Arquitetura com descentralização do objeto coordenador

Arquitetura usando o Serviço CORBA de Eventos

Esta arquitetura utiliza um dos *CORBAservices*, o serviço CORBA de eventos, para fazer a comunicação entre os elementos. Neste serviço, um evento é definido como uma ocorrência em um determinado objeto que é de interesse de outro(s) objeto(s). Quando um evento ocorre, objetos que estão particularmente interessados nesta ocorrência devem ser avisados através de uma notificação enviada pelo objeto que gerou o evento [Trinta2000].

Através do serviço de eventos CORBA, objetos também podem enviar dados junto com a própria notificação [OMG2003].

Através do serviço de eventos CORBA, objetos assumem dois pápeis: produtores *(suppliers)* e consumidores *(consumers)*. Objetos produtores produzem eventos, enquanto objetos consumidores registram-se para ser notificados quando da ocorrência destes eventos. O serviço CORBA de eventos utiliza um modelo de comunicação produtor-consumidor que permite vários objetos produtores de eventos enviem notificações a objetos consumidores através de um chamado *canal de eventos*. Este canal de evento é um objeto CORBA padrão que está situado sobre o ORB, onde acontece a comunicação entre os objetos.

No sistema de Co-Autoria [Trinta2000] são utilizados quatro canais de eventos para transmissão de notificações e dados entre os elementos da arquitetura.

Um primeiro canal de eventos chamado *chat*, para comunicação de mensagens; um segundo canal de eventos chamado *slide*, para transmissão do slide que deve ser apresentado a todos autores; um terceiro canal de eventos chamado *token*, para atualização da lista de usuários presentes e aqueles que estão na lista de acesso ao *token*; e por fim, um quarto canal de eventos chamado *votação*, por onde são realizadas as votações sobre edições de slides e entrada de novos usuários.

Nestes canais, os elementos da arquitetura assumem diferentes papéis de produtores e consumidores em cada um deles. No caso do canal de eventos *chat*, todo objeto autor é produtor e consumidor das notificações geradas no canal. Cada notificação leva consigo uma *string*, que representa a mensagem enviada por um autor. Sendo todo autor consumidor deste canal, cada usuário receberá a notificação e consequentemente a mensagem, que será então mostrada numa área específica para o recebimento das mesmas. O coordenador também é um produtor do canal de eventos *chat*, pois este elemento também envia mensagens para serem mostradas como mensagens de *chat*, como por exemplo, a entrada de um novo usuário [Trinta2000].

O coordenador é também produtor de eventos para os canais *token* e *votação*. No canal *token*, o coordenador envia periodicamente, a lista de todos os usuários presentes no ambiente. Os consumidores deste canal são os objetos autor, fazendo com que, todo autor seja notificado quando da entrada ou saída de um autor do ambiente. No canal

votação, o coordenador envia notificações pedindo aos autores seus votos a respeito das ações do editor de aulas, quando da realização de uma edição por parte deste usuário. Neste caso, como todo autor é um consumidor deste canal, o editor da aula também será notificado, devendo haver uma filtragem no recebimento da notificação por parte deste usuário, para que o mesmo não seja questionado sobre o seu voto. Neste caso, o editor deve ser bloqueado e esperar o resultado da votação, para então continuar sua edição. O último canal, o de *slide*, serve para que o servidor de slides funcione como o produtor de eventos, repassando para os autores, que são os consumidores do canal, o slide que está sendo editado pelo dono do *token* [Trinta2000].

A vantagem desta arquitetura é a utilização de um serviço pronto, eficaz e fácil de se usar para a comunicação entre os elementos da arquitetura. A forma como o serviço de eventos trata os produtores e consumidores de eventos de cada canal permite que possíveis erros de comunicação entre um único autor e um canal não afetem os demais autores.

2.5 Considerações Finais

É iminente a intensa utilização da Internet para fins de educação a distância, através da gama de novas possibilidades educacionais que a rede traz para a EAD.

A utilização da especificação CORBA apresenta incompatibilidades no diz respeito ao seu funcionamento na Internet propriamente dita, pois o uso de esquemas de segurança baseados em *firewalls* dificulta a comunicação entre os componentes de aplicações baseadas em objetos distribuídos CORBA na Internet. A OMG [OMG99] busca soluções através de uma especificação sobre o uso de aplicações CORBA mesmo quando da presença de *firewalls* [Trinta2000], e que possibilite a interação entre objetos situados em diferentes redes. Nas situações onde clientes externos a um certa rede necessitam acessar objetos CORBA localizados em *hosts* protegidos por *firewalls*, ainda é necessário o uso de abordagens baseadas em soluções proprietárias.

Tendo em vista que sistemas de distribuição de EAD síncronos, operando vídeo conferência e teleconferência por exemplo, são mais adequados à redes de alta velocidade e têm dificuldade de funcionamento na Internet atual pelo fato de exigirem

interação em "tempo real", este trabalho se propõe a apresentar tecnologias Java disponíveis para desenvolvimento de sistemas EAD para Internet, resultando no desenvolvimento de um protótipo que visa a utilização de mídias contínuas assíncronas, como áudio e vídeo.

Capítulo 3 Tecnologias envolvidas

Neste capítulo serão apresentadas as tecnologias envolvidas no desenvolvimento de um sistema para a WEB, tais como *middlewares*, objetos distribuídos, linguagem Java, bem como uma ánalise comparativa entre as tecnologias que servem de suporte ao desenvolvimento de aplicações distribuídas: DCOM, CORBA, Java RMI, Java *Servlets*, JSP, Web Service e Java/Web Service.

Capítulo 3 Tecnologias envolvidas

3.1 Desenvolvimento para Web

Nas aplicações para Web, muitos dos aplicativos atuais são *aplicativos distribuídos de três camadas* [Deitel2001], que consistem em:

- Uma interface com o usuário
- Uma lógica do negócio
- Acesso a banco de dados

A interface com o usuário é freqüentemente criada utilizando HTML (*Hypertext Markup Language*) , HTML Dinâmico e/ou linguagem Java (*applets Java*, por exemplo).

A HTML é o mecanismo preferido para representar a interface com o usuário em casos onde a portabilidade é uma questão importante. Como a HTML é suportada por todos os navegadores, projetar a interface com o usuário para ser acessada por um navegador da Web garante portabilidade entre todas as plataformas que têm navegadores [Deitel2001]. Através da Web, a interface com o usuário pode comunicar-se com a lógica do negócio na camada intermediária. A camada intermediária então pode acessar o banco de dados para manipular os dados. Todas as três camadas podem residir em computadores separados conectados a uma rede.

Em arquiteturas de múltiplas camadas, os servidores da Web são cada vez mais utilizados para construir a camada intermediária. Eles fornecem a lógica do negócio que manipula dados do banco de dados e comunicam-se com navegadores da Web clientes [Deitel2001]. Em Java, por exemplo, os *servlets*, por meio de JDBC, podem interagir

com sistemas de banco de dados conhecidos. Os desenvolvedores não precisam conhecer as especificações de cada sistema de banco de dados. Em vez disso, os desenvolvedores utilizam consultas baseadas em SQL e no driver JDBC para tratar questões específicas da interação com cada sistema de banco de dados.

A utilização de *servlets* juntamente com JDBC baseado em consultas em SQL atende a requisitos de desenvolvimento de sistemas colaborativos, como reutilização de experiência e conhecimentos anteriores com o intuito de reduzir o tempo necessário para que um novo membro se integre à equipe de desenvolvimento [Fuks2003].

3.2 Middleware

O desafio de implementação de uma aplicação distribuída é justamente definir precisamente as camadas lógicas (camada de apresentação, de aplicação e de dados) e, a seguir, concretizá-las em componentes físicos. Em uma segunda etapa, o projeto deve se preocupar com o modo em que esses componentes se comunicarão, em outras palavras, com o *middleware.* Os produtos projetados para o desenvolvimento de *middleware* têm como função básica isolar o desenvolvedor dos detalhes de baixo nível envolvidos nos protocolos de comunicação em uma rede [Riccioni2000].

Para cada das camadas lógicas, defini-se um *middleware* correspondente. Um exemplo de *middleware de apresentação* é o trio navegador, protocolo HTTP e servidor Web, que divide a responsabilidade da comunicação para mostrar uma página através da Internet. Já um *middleware de dados* é, em geral, fornecido pela desenvolvedora do Banco de Dados e tem como função fazer as consultas SQL no servidor através da rede, retornando o resultado à estação-cliente.

Já o *middleware de aplicações* [Riccioni2000] difere das outras classes por apresentar um propósito mais genérico que, em geral, aborda as regras de comunicação propriamente ditas. Nessa classe a decisão básica é que tipo de comunicação utilizar. Existem alguns padrões no mercado. Obviamente cada um deles possui suas vantagens e desvantagens. Vamos analisar algumas das possibilidades disponíveis nos itens posteriores deste trabalho.

3.2.1 Middleware de Aplicações

Middleware é definido [Goulart1999] como um conjunto de serviços disponibilizados aos desenvolvedores de aplicações através de interfaces de programação para aplicações, APIs (*Aplication Program Interface*), com o objetivo de reduzir a complexidade do processo de desenvolvimento.

O aspecto inovador da abordagem do *middleware* está relacionado com a disponibilização estruturada de serviços, de tal forma que possibilite a abstração dos detalhes relativos à infra-estrutura de rede, ambientes operacionais, arquiteturas de hardware e software inerentes ao desenvolvimento de aplicações distribuídas em ambientes heterogêneos. Com essas características, os *middleware* trazem vantagens para o desenvolvimento de aplicações como: transparência, portabilidade e reusabilidade.

A utilização de um *middleware* oferece abstrações de alto nível facilitando a implementação de componentes distribuídos fornecendo uma visão uniforme do ambiente operacional. Outras características dos *middleware* que tornam o processo de desenvolvimento mais modular e incremental são as interfaces públicas e bem definidas dos componentes, o que permite facilidades de evolução, mobilidade, extensibilidade, tratamento de heterogeneidades e autonomia dos componentes.

3.2.2 Middleware JAVA

Middleware ajuda a integrar software que tem que ser escrito em diferentes sistemas e em diferentes linguagens de programação – ajuda a construir uma ponte entre os limites de software. Por várias razões há várias formas de *middleware*, e às vezes há uma escolha natural sobre qual usar. CORBA foi o primeiro padrão e forma popular de middleware de alto nível, e seu escopo engloba os modelos de programação de muitas outras formas de middleware. Sem representar ainda decisão de projeto ou preferência, aqui destacamos um “middleware Java” de forma genérica.

Entre os atrativos de Java está a facilidade que essa linguagem oferece para desenvolver aplicações para execução em sistemas distribuídos. Já em sua primeira versão, Java oferecia facilidades para o desenvolvimento de aplicações cliente-servidor usando os mecanismos da Internet, tais como os protocolos TCP/UDP/IP [Marques2000].

Se o cliente na aplicação distribuída precisa acessar um servidor de banco de dados relacional, Java oferece uma API específica para tal fim, JDBC. Através das classes e interfaces desse pacote é possível realizar consultas expressas em SQL a um servidor de banco de dados e manipular as tabelas obtidas como resultado dessas consultas [Marques2000].

Em termos de desenvolvimento voltado para a World-Wide Web, Java oferece o já clássico mecanismo de *applets*, código Java que executa em uma máquina virtual no lado do cliente (tipicamente um navegador) Web. O mecanismo de servlets permite associar o potencial de processamento da plataforma Java a servidores Web, permitindo construir assim uma camada *middleware* baseada no protocolo HTTP e em serviços implementados em Java [Marques2000]. Suporta implementações para diferentes plataformas, podendo estar embutida em outros programas, como o caso de *browsers* como o Netscape ou Explorer.

Desenvolvendo o protótipo de um sistema de Educação a Distância acessado via Internet, considera-se que qualquer usuário faça acesso ao ambiente através de qualquer plataforma de hardware ou software. O problema de portabilidade da aplicação para o cliente é resolvido através da utilização de applets Java, carregados a partir de uma URL e processados localmente. Consegue-se assim independência de plataforma e distribuição - como o applet é carregado pela rede, a disponibilização e as atualizações precisam ser feitas apenas no servidor de onde os programas (as classes) são carregadas [Sun2002].

No protótipo EAD, acesso a banco de dados e a outros recursos servidores específicos podem ser feitos via a utilização de servlets. Um servlet é um programa Java carregado e executado por um servidor Web e usado para negociar requisições do cliente. Os servlets se comunicam com os clientes através de requisições-respostas (*requests-responses)* modelado pelo comportamento do HTTP. Um *browser* ou outro programa cliente capaz de fazer conexões através da Internet acessa o servidor Web e faz uma requisição. A requisição é processada por um serviço de rede de um computador servidor que a transfere para o servlet. O servlet responde adequadamente a requisição [Davidson98].

Um sistema EAD na Web atualmente utiliza uma arquitetura de três camadas:

- O browser cliente será a primeira camada;

- O servidor Web, juntamente com os servlets, executam o processamento de aplicações designadas para a camada central. Nela o servlet pode transpor restrições de segurança impostas pela plataforma Java para applets e funcionar como um proxy, já que o applet só pode estabelecer conexões com o próprio servidor do qual foi carregado. Um servlet requisitado por um applet pode acessar um banco de dados localizado em um servidor diferente do servidor Web e retornar o resultado da consulta para o applet [MageLang98].

- O servidor de banco de dados exerce o papel da terceira camada [Darby98].

Uma proposta para construção de *middleware* é a utilização de Objetos Distribuídos.

3.3 Objetos Distribuídos

Objetos Distribuídos são componentes de software criados com os conceitos da orientação a objetos [Goulart1999], podendo estar distribuídos através de uma rede heterogênea, mas vistos pelas aplicações como se fossem componentes locais, fornecendo com isso vários tipos de transparências desejadas em sistemas distribuídos, podendo tornar os sistemas mais eficientes, flexíveis e menos complexos. A próxima geração de sistemas cliente/servidor deverá ser construída usando objetos distribuídos. Enquanto a tecnologia cliente/servidor apenas divide seu problema em duas partes, uma rodando no cliente e outra no servidor, os objetos ajudarão a subdividir as aplicações em componentes auto-gerenciáveis que podem executar sozinhos ou entre redes e sistemas operacionais. Estes componentes representam a última forma de distribuição cliente/servidor e nos prepara para um futuro onde milhares de máquinas serão clientes e servidores [Goulart1999].

Objetos distribuídos são projetados e construídos utilizando o paradigma da orientação a objetos aplicado a sistemas distribuídos. A utilização da orientação a objetos agrega inúmeras características ao processo de desenvolvimento dos objetos distribuídos, das quais merecem destaque o encapsulamento, comportamento, polimorfismo, herança e identidade de objetos. Já os sistemas distribuídos agregam novas funcionalidades:

- Processamento distribuído: Os serviços estando em várias máquinas balanceiam melhor o sistema; réplicas de um serviço diminuem a chance de indisponibilidade do mesmo.

- Escalabilidade: O crescimento do número de usuários não implica na queda de qualidade, pois a escalabilidade pode ser alcançada replicando o serviço em outras máquinas. A característica de escalabilidade vem atender a um requisito de desenvolvimento de sistemas colaborativos que diz respeito ao desempenho do sistema, que não deve se degradar na medida que novos usuários se conectam [Fuks2003].

- Interoperabilidade: Objetos executando em ambientes heterogêneos podem interagir.

- Tolerância a falhas: A replicação de um serviço diminui as chances do mesmo não ser oferecido, pois caso a máquina que o serve não esteja disponível, uma de suas réplicas poderá atender o chamado.

- Portabilidade: O objeto pode executar em diferentes plataformas. A característica de portabilidade atende ao requisito de usuário de sistemas colaborativos no que diz respeito à disponibilização de acesso ao ambiente independente da estação de trabalho [Fuks2003].

- Coexistência: Coexistir junto a aplicações legadas.

- Transparência: O cliente não percebe a distribuição dos serviços. Utilizando objetos distribuídos, os componentes podem ser utilizados localmente e liberados para acesso remoto somente quando desejado, atendendo assim um requisito de usuário de sistemas colaborativos que se refere à necessidade de existência de um espaço privativo e público e a transição entre eles [Fuks2003].

Objetos distribuídos são componentes de software acessados por clientes remotos através da invocação de métodos desses objetos. No entanto, os clientes não precisam saber onde os objetos distribuídos foram criados, onde estão sendo executados, ou em que plataforma podem ser executados. Uma outra característica importante dos objetos distribuídos é o encapsulamento. Os clientes solicitam serviços através dos métodos descritos nas interfaces, mas não sabem como os objetos distribuídos implementam ou executam os serviços solicitados.

Um outro termo utilizado para denominar objetos distribuídos é *Object Web* [Goulart1999]. Esse termo inicialmente foi utilizado para o conjunto CORBA e Java, mas atualmente é utilizado para denominar sistemas que implementam a tecnologia de objetos distribuídos para Web. A figura 3.0 mostra graficamente a evolução dos sistemas implementados para Web de acordo com sua funcionalidade e sua interatividade.

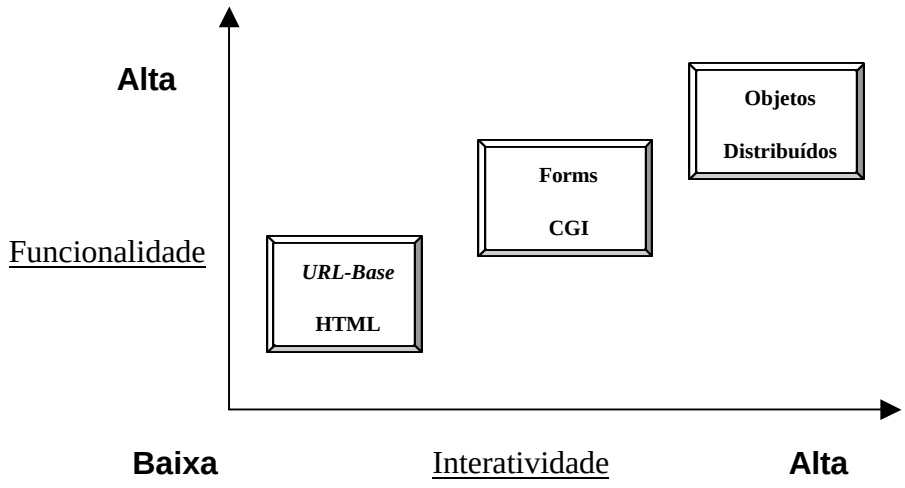

Figura 3.1 Evolução das Tecnologias Web

O primeiro componente mostrado na figura 3.0 são as tradicionais *Home Pages* que possuem somente hipertextos [Goulart1999]. O segundo componente já possui uma maior interatividade através do preenchimento de formulários, que podem ser feitos através de HTML (*Hyper Text Markup Language*) e DHTML (*Dynamic Hyper Text Markup Language*), usando *cookies*, CGI (*Common Gateway Interface*), *Tables* e ASP (*Active Server Pages*) inclusive podendo acessar e armazenar informações usando bancos de dados, mas ainda com pouca funcionalidade e interatividade. O último componente, os objetos distribuídos, são os mais adequados para construção de novas aplicações de negócio para Web. Exemplificando, as tecnologias para objetos distribuídos podem ser utilizados controles *Active* X / DCOM (*Distributed Component Object Model*), Java e CORBA (*Common Object Request Broker Architecture*), Java / RMI (*Remote Method Invocation*) e Java / *Servlets* e mais recentemente *Web Services*.

Alguns padrões com suporte a criação de objetos para sistemas distribuídos serão estudados: DCOM da Microsoft [COM99], CORBA da OMG [OMG99] (*Object Management Group*) e Java/RMI [RMI99] e Java / *Servlets* e *Web Services.*.

3.4 A linguagem Java

A linguagem Java surgiu da necessidade de desenvolvimento de aplicativos para Web. Os primeiros programas, com base em formulários, não possuiam os recursos e respostas interativas que os usuários passaram a esperar dos aplicativos modernos, gerando frustação tanto a desenvolvedores quanto a usuários Web. A necessidade era descobrir uma maneira de tornar os programas Web mais interativos. Foi aí que Java entrou em cena. Java tinha sido desenvolvida na premissa de aplicativos desenvolvidos em CGI, na Web, contudo, oferendo mais: acrescentar animações a aplicativos Web; e rodar programas independente da plataforma de hardware, usando o mesmo código.

Java é uma linguagem de programação [Riccioni2000] caracterizada por sua portabilidade, robustez, segurança, suporte a programação distribuída e amplos recursos para o desenvolvimento de aplicações multimídia. Em sistemas heterogêneos, uma das principais vantagens da linguagem se refere à portabilidade de suas aplicações.

A característica de portabilidade existente em Java atende ao requisito do usuário de sistemas colaborativos que diz respeito a acesso ao ambiente independente da estação de trabalho [Fuks2003]. O suporte à programação distribuída garante requisitos de usuário de sistemas colaborativos como a existência de espaço privativo e público no ambiente EAD e a transição entre eles, requisitos de desenvolvimento de sistemas colaborativos como compartilhamento transparente dos dados, suporte a dados locais e compartilhados e escalabilidade [Fuks2003]. Os recursos para o desenvolvimento de aplicações multimídia em Java podem facilitar o atendimento do requisito de usuário de sistemas colaborativos que diz respeito ao sistema prover informações de percepção para os usuários através do ambiente [Fuks2003].

A habilidade de se carregar *applets* e a sua integração com o Web *browser* tornam-na ferramenta ideal para a Internet e WWW. A portabilidade é alcançada porque

Java é uma linguagem interpretada, isto quer dizer que, quando compilamos um programa escrito em Java é gerado um código intermediário e não um código de máquina, como ocorre na *maioria* das linguagens. Existe assim, independência de plataforma de software e hardware. A execução do código Java é realizada sobre qualquer plataforma através da máquina virtual Java (JVM), que pode ser descrita como um interpretador de código intermediário (*Byte-Code* Java), para a plataforma em que se deseja executar a aplicação.

Java possui muitas interfaces de programação para uma ampla faixa de aplicações, são as APIs Java. Além do núcleo API Java, que consiste da linguagem básica e o sistema de janelas, Java oferece várias outras APIs que facilitam o desenvolvimento de aplicações [Riccioni2000], tais como:

- Java Enterprise:

Oferece suporte para as aplicações Java interagirem, com dados e aplicações distribuídas, obtendo o suporte de três componentes: Java IDL; Java RMI (Remote Method Invocation); Java JNDI (Java Naming and Directory Interface).

- Java IDL:

Oferece um modo para objetos Java cliente e servidor interagirem com outros servidores e clientes, compatíveis com CORBA. Ele oferece um mecanismo direto, pelo qual programas Java podem interagir com outros serviços em uma linguagem neutra;

- Java RMI:

Classe que permite desenvolvimento de aplicações distribuídas em Java. É similar ao mecanismo de RPC (Remote Procedure Call) e adequado a construção de aplicações distribuídas, em ambientes homogêneos, escrita totalmente em Java;

- Java JNDI:

Oferece mecanismo para os programas Java interagirem com diretórios distribuídos e com ambientes heterogêneos e ainda com serviços de nomes;

- Java Database Connectivity (JDBC):

Classe que permite ao cliente Java interagir com Banco de Dados Relacional compatível com *Microsoft Open Database Conectivity (OBDC);*

➢ Java Security:

Oferece um *framework* para escrever programas Java com mecanismo de segurança, tais como autenticação, assinatura digital e técnicas de criptografia;

➢ Java Media Framework:

Oferece um conjunto de facilidades ao desenvolvimento de aplicações Java que usam gráficos e multimídia. O pacote contém o *Java 2D, o Java Media, Java Management, Java Collaboration, Java Telephony, Java Speech, Java Animation e Java 3D;*

➢ Java Collaboration:

Oferece uma *framework* para escrever aplicações *collaboration-aware*, bem como aplicações *collaboration-unaware* para trabalhos coorporativos. Oferece mecanismo para compartilhar quadro-negro eletrônico (*blackboard),* sistema de compartilhamento de aplicações, etc.

Além destas, existem várias outras APIs disponíveis na Internet para aplicações gerais e específicas.

3.5 Suporte para aplicações distribuídas

Nesta seção serão abordadas as principais tecnologias para construção de aplicações distribuídas. Os mecanismos de utilização, dependendo do tipo e da natureza da aplicação, devem ser considerados (plataforma operacional, linguagem de programação). Após a descrição de cada tecnologia, será feita uma análise comparativa visando a escolha da solução mais adequada para construção de um sistema independente de plataforma, desenvolvido para Web cujo protótipo é descrito nos próximos capítulos.

3.5.1 DCOM

A proposta DCOM (Distributed Component Object Model) começou guando a Microsoft anunciou, em março de 1996, o *ActiveX*. O *ActiveX* é uma combinação do OLE nos serviços de desktop com a *World Wide Web*. Ou seja, o *ActiveX* combina *Web browsers* e *applets Java* com os serviços oferecidos pelo desktop, como por exemplo: documento MS-Word, suporte à transações, planilhas, *scripting*, documentos compostos e outros serviços. Com isso a Microsoft deu um grande passo para objetos distribuídos [Riccioni2000].

O *ActiveX* usa DCOM para oferecer comunicação entre objetos remotos *ActiveX* e outros serviços. E, neste sentido, o DCOM tornou-se um ORB (*Object Request Broker*) para o *ActiveX*.

O DCOM oferece, basicamente, um serviço de localização do *ActiveX*, APIs e de objetos através de invocação estática e dinâmica [Riccioni2000]. O DCOM utiliza DCE RPC (Chamada de Procedimentos do *Distributed Computing Environment*) para interações entre objetos. O modelo de objetos DCOM é limitado, porque o DCOM não suporta herança múltipla. Em outras palavras, DCOM suporta herança através de ponteiros que ligam várias interfaces. Além disso, o DCOM oferece ainda as seguintes facilidades:

- Interface Definition Language (IDL): DCOM usa interfaces muito parecidas com as do CORBA. Uma interface define um conjunto de funções relacionadas.

- Object Definition Language (ODL) and Type Libraries: O DCOM suporta uma linguagem de definição de objetos (ODL) que é usada para descrever Metadados. As especificações de interfaces e Metadados são armazenadas em um repositório, que é chamado de *Type Library*. *Type Library* equivale ao Repositório de Interface CORBA.

- Object Services: Oferece serviços como mecanismo de licença; serviço de diretório local, baseado no *Registry* do Windows; serviços de persistência de sistema e um serviço de eventos, muito simples, chamado *Connectable Objects*.

O DCOM é baseado na filosofia de orientação a objeto, utiliza o conceito de interface [Riccioni2000], através de IDL; a chamada do cliente para o servidor pode utilizar invocações estáticas e dinâmicas; um repositório de interfaces chamado de *Type Library* é destinado a invocar e localizar objetos.

O DCOM é uma tecnologia da proprietária Microsoft. Isto significa dizer que a idéia de sistema heterogêneo fica prejudicada, e que a interoperabilibadade entre as máquinas existirá, mas somente se estiver sendo utilizado um sistema operacional Microsoft. O requisito de usuário de sistemas colaborativos que diz respeito a acesso ao ambiente independente da estação de trabalho [Fuks2003] fica prejudicado devido DCOM ser uma tecnologia proprietária.

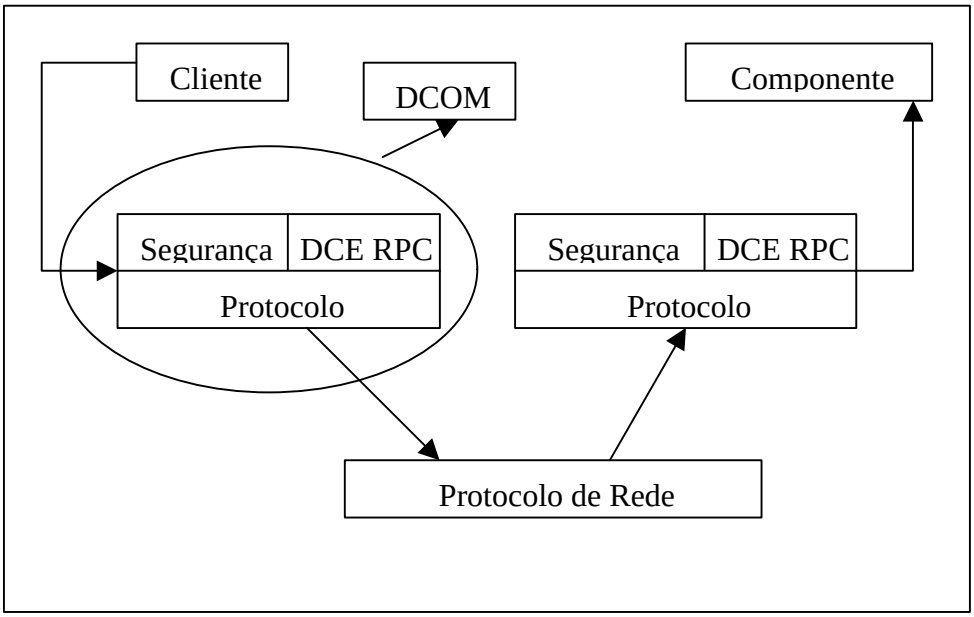

Figura 3.2 Arquitetura DCOM

3.5.2 CORBA

CORBA (*Common Object Request Broker Architecture*) é uma arquitetura que começou a ser definida em 1989 e, em sua versão 1.1, foi inicialmente implementada em 1991, como sendo um produto intelectual do *Object Management Group* (OMG). A OMG é um consórcio formado pelas maiores empresas de informática e desenvolvimento de software. Resumidamente, o objetivo da OMG é fornecer especificações de gerenciamento de objetos em ambientes distribuídos e heterogêneos com características de reusabilidade, portabilidade e interoperabilidade [Goulart1999].

Em vez de aplicações, a OMG produz especificações que tornam a computação orientada a objeto possível. O modelo CORBA baseado em objetos permite que

métodos de objetos sejam ativados remotamente, através de um elemento intermediário chamado ORB (*Object Request Broker)* situado entre o objeto propriamente dito (que encontra-se na camada de aplicação do modelo *Open System Interconnection - OSI)* e o sistema operacional, acrescido de funcionalidades que o permitam comunicar-se através da rede [Riccioni2000].

A fim de padronizar o protocolo de comunicação e o formato das mensagens, garantindo assim a interoperabilidade entre objetos de diferentes implementações de ORB, foram definidos os seguintes protocolos [Riccioni2000]:

- Protocolo Inter-ORB Geral (GIOP): Especifica um conjunto de formatos das mensagens e dados para comunicação entre ORBs;

- Protocolo Inter-ORB Internet (IIOP): A especificação CORBA possui, desde dezembro de 1994, como parte do CORBA 2.0, um protocolo chamado IIOP (*Internet Inter-Orb Protocol*) para objetos remotos, que especifica como mensagens GIOP são transmitidas numa rede TCP/IP. Antes do IIOP, a especificação CORBA definia somente interação entre objetos distribuídos criados pelo mesmo fornecedor. Os objetos tinham de ser definidos por uma implementação específica. Usando-se IIOP, a especificação CORBA 2.0 tornou-se a solução para interoperabilidade de objetos que não ficam presos a uma plataforma ou padrão específico [Riccioni2000]. Nesta especificação tudo depende de um ORB (*Object Request Broker*), que funciona como um meio de comunicação sobre o qual os objetos CORBA interagem transparentemente com outros objetos CORBA locais ou remotos.

- Protocolo Inter-ORB para Ambiente Específico (ESIOPs): É uma especificação para permitir a interoperabilidade do ORB com outros ambientes.

Em 1990 o OMG criou a OMA (*Object Management Architecture*) com o objetivo de fomentar o crescimento de tecnologias baseadas em objetos e fornecer uma infra-estrutura conceitual para todas especificações da OMG. O OMA é formado pelos seguintes elementos principais:

- ORB (*Object Request Broker*): Manipula requisições entre objetos;

- Serviços de Objeto (*CORBA Services*): Definem serviços a nível de sistema que ajudam a gerenciar e manter objetos;

- Facilidades Comuns *(CORBA Facilities*): Definem facilidades horizontais e interfaces no nível de aplicação;
- Interfaces de Domínios (*Domain Interfaces*): facilidades verticais;
- Objetos de aplicação: os objetos de aplicação propriamente ditos.

Na figura abaixo é mostrada graficamente a arquitetura OMA.

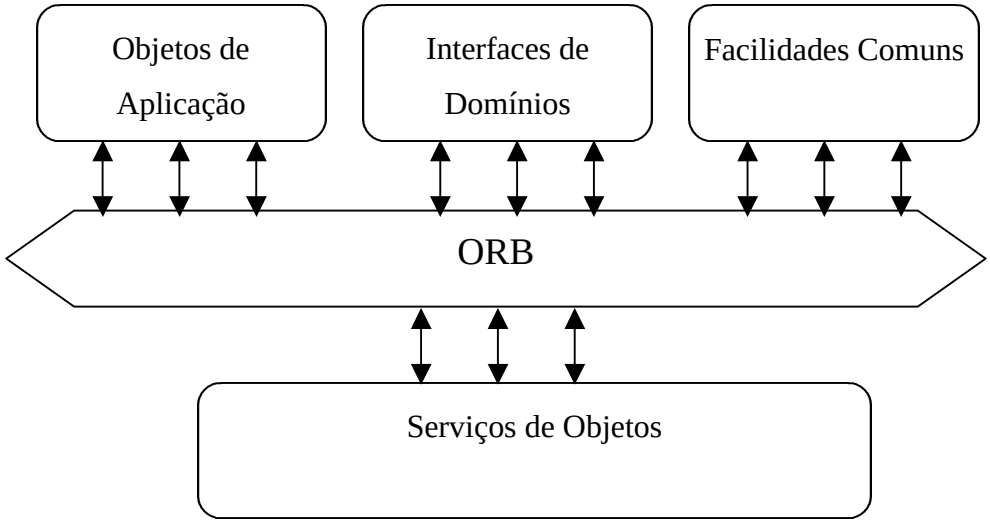

Figura 3.3 Arquitetura Geral (OMA)

O ORB é o componente mais importante da arquitetura OMA. Ele permite que objetos façam e recebam requisições de métodos transparentemente em um ambiente distribuído e heterogêneo [Riccioni2000].

Cada objeto CORBA servidor deve possuir uma interface, definida através da IDL (*Interface Definition Language*), linguagem de definição de interfaces que apresentam os métodos públicos (serviços) de cada objeto servidor. Do lado cliente um objeto CORBA adquire uma referência de um objeto CORBA servidor e faz uma solicitação para o ORB, que é responsável por encontrar a implementação do Objeto e preparar o objeto CORBA para receber a requisição. CORBA fornece mapeamento para diversa linguagens de programação [OMG99]: Ada, C, C++, Cobol, Java e Smalltalk. Assim, sistemas distribuídos CORBA podem ser heterogêneos não apenas quanto a sistemas

operacionais, mas também quanto a linguagens de programação, atendendo assim, tanto ao requisito de usuário de sistemas colaborativos que diz respeito ao acesso do ambiente independente da estação de trabalho, como também ao requisito de desenvolvimento de sistemas colaborativos que diz respeito ao reuso de experiência e conhecimento anteriores por suportar mapeamento para diversas linguagens de programação [Fuks2003].

CORBA ainda atende requisitos de desenvolvedor de sistemas colaborativos como compartilhamento transparente dos dados, suporte a dados locais e compartilhados e escalabilidade [Fuks2003].

Entretanto, CORBA não atende com eficiência requisitos de usuário de sistemas colaborativos como o de acesso ao ambiente independente da estação de trabalho [Fuks2003] pelo fato de que seu protocolo de comunicação ter dificuldade de atravessar *firewalls* na Internet.

3.5.3 Java RMI

O suporte ao desenvolvimento de aplicações distribuídas na primeira versão do JDK (Java Development Kit) era através de programação de *sockets* TCP/IP, bem rudimentar. Posteriormente, a JavaSoft apresentou seu primeiro suporte para sistemas distribuídos, denominado RMI (Remote Method Invocation).

O RMI pode ser visto como um ORB nativo Java, isto é, RMI é um ORB (não compatível ao CORBA) que suporta invocações de métodos em objetos Java distribuídos.

O RMI é uma extensão do núcleo da linguagem Java e depende de muitos aspectos da linguagem, tais como: serialização, portabilidade, implementação de objeto carregável (*down-loadable)*, definições de interface Java, entre outros.

A limitação do RMI está no que diz respeito ao seu uso em ambientes heterogêneos, a exemplo da interação com objetos desenvolvidos em outras linguagens.

Na figura 3.4 é apresentada a arquitetura Java RMI, constituída de três camadas [Riccioni2000]:

- A camada de stub/skeleton: Para cliente e servidor, respectivamente.
- A camada de referência remota: Responsável por conduzir as semânticas da invocação (ex.: determinar se o servidor é um objeto simples ou replicado); Abstrair os diferentes modos de referenciar os objetos (ex.: servidores que já estão executando em algumas máquinas, ou servidores que só executam quando são invocados);
- A camada de transporte: Trata da configuração da conexão, gerenciamento da conexão e de manter ligação do *dispatching* com os objetos remotos (o alvo da chamada remota), residentes no espaço de endereçamento.

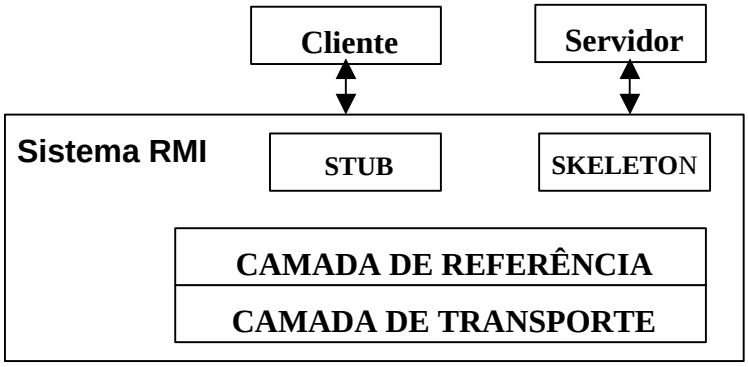

Figura 3.4 Arquitetura Java RMI.

Uma das características principais de RMI [Goulart1999] é sua habilidade de carregar o código (bytecodes) da classe de um objeto se a classe não for definida na máquina virtual do receptor. Os tipos e comportamento de um objeto, antes só disponíveis em uma única máquina virtual, podem ser transmitidos a outra máquina virtual, possivelmente remota. RMI passa objetos pelo seu tipo, assim os comportamentos desses objetos não se alteram quando os mesmos rodam em outra máquina virtual. Isto permite que novos tipos possam ser introduzidos numa máquina virtual remota.

A tabela 3.1 a seguir mostra alguns componentes RMI e suas funcionalidades:

Componente	Funcionalidade
Java.rmi	Organiza as classes, exceções e interfaces do lado cliente RMI
Java.rmi.server	Organiza as classes, exceções e interfaces do lado servidor RMI
Java.rmi.registry	Organiza as classes para o serviço de nomes RMI
Java.rmi.activation	Organiza as classes ativadas sob demanda em RMI
Rmic	Compilador RMI, que gera os *stubs* e *skeletons* usados na implementação de aplicações distribuídas
Rmiregistry	Utilitário do servidor que provê o serviço de nomes RMI
Rmiid	Utilitário do servidor que da suporte às classes RMI através de um identificador único

Tabela 3.1 Tabela de funcionalidades RMI

Os componentes RMI descritos acima podem ser utilizados em conjunto com outras APIs Java, por exemplo a API JDBC. A associação dessas duas APIs proporciona a junção das facilidade dos sistemas distribuídos fornecidos por RMI, aliadas à potencialidade das aplicações utilizando Java de acessar banco de dados.

3.5.4 Java Servlets

A World Wide Web torna a Internet fácil de utilizar e se beneficia da "onda" multimídia. As organizações consideram a Internet e a Web como cruciais para suas estratégias na área de sistemas de informações. Java fornece diversos recursos de rede predefinidos que tornam fácil desenvolver aplicativos para Web e para Internet. Java pode não apenas especificar paralelismo por meio de *multithreading*, como também pode permitir que os programas pesquisem o mundo buscando informações e colaborem com programas que executam em outros computadores internacionalmente, nacionalmente ou apenas dentro de uma organização. Java permite que *applets* e aplicativos executando no mesmo computador se comuniquem entre si, sujeitos às limitações de segurança.

Os recursos de rede de Java estão agrupados em vários pacotes. Os recursos fundamentais de rede são definidos por classes e interfaces do pacote *java.net*, por meio das quais Java oferece *comunicações baseadas em soquetes* que permitem ver as redes

como fluxos de dados - um programa pode ler de um *soquete* ou gravar em um *soquete* tão facilmente quanto ler de um arquivo ou gravar em um arquivo. As classes e interfaces do pacote *java.net* também oferecem *comunicações baseadas em pacotes* que permitem que *pacotes* individuais de informações sejam transmitidos.

As visualizações de nível mais alto de redes são fornecidas por classes e interfaces nos pacotes *java.rmi* para *Remote Method Invocation (RMI)* e os pacotes *org.omg* para *Common Object Request Broker Architecture (CORBA)* que são parte da API do Java 2. Os pacotes RMI permitem que os objetos Java sejam executados em *Java Virtual Machines* separadas para comunicar-se via chamada de métodos remotos. Essas chamadas de métodos parecem ser para um objeto no mesmo programa, mas na verdade usam recursos de rede embutidas (baseados nos recursos *java.net*) que comunicam as chamadas de método para outro objeto em um computador separado. Os pacotes CORBA fornecem funcionalidades semelhantes aos pacotes RMI. A diferença chave entre RMI e CORBA é que RMI pode apenas ser utilizada entre objetos Java, enquanto CORBA pode ser utilizada entre dois aplicativos quaisquer que entendem CORBA - incluindo aplicativos escritos em outras linguagens de programação.

No relacionamento cliente-servidor, o cliente solicita que alguma ação seja executada e o servidor realiza a ação e responde para o cliente. Esse modelo de comunicação pedido-resposta é o fundamento para a visualização do mais alto nível de redes em Java-servlets [Deitel2001]. O pacote *javax.servlet* e o pacote *javax.servlet.http* fornecem as classes e as interfaces para definir os *servlets.*

Os *servlets* aprimoram a funcionalidade de servidores da World Wide Web - a forma mais comum de *servlet* atualmente. A tecnologia *servlet* é projetada principalmente para utilização com o protocolo HTTP da World Wide Web, mas já sendo desenvolvidos *servlets* para outras tecnologias. Os *servlets* são eficientes para desenvolver soluções baseadas na Web que ajudam a fornecer acesso seguro a um site da Web, interagir com banco de dados em favor de um cliente, gerar dinamicamente documentos personalizados de HTML a serem exibidos por navegadores e manter informações para sessão exclusivamente de cada cliente.

Os *servlets* correspondem no lado do servidor aos *applets* do lado do cliente. Os *servlets* normalmente são executados como parte de um servidor Web. São suportados

por vários servidores da Web, como servidores da Web da Netscape, o *Internet Information Server (IIS)* da Microsoft, o servidor da Web Jigsaw do World Wide Web Consortium e o conhecido servidor da Web Apache.

3.5.5 JSP

Java Server Pages é uma tecnologia de *web-scripting* para desenvolvimento de aplicações Web [Ferreira2002]. Podemos dizer ainda, segundo a definição da Especificação JSP1.2 [JSP2003], que é "um documento baseado em texto que descreve como processar uma Solicitação (Resquest) para criar uma Resposta (Response)[1]." E ainda complementa: "A descrição mistura internamente dados modulares com ações dinâmicas e é alavancada pela plataforma Java 2."

A junção de dados estáticos (dados modulares) com códigos de uma linguagem de programação (ações dinâmicas) resulta em resposta a um usuário, na forma de um documento de texto, motivada por uma solicitação do mesmo usuário [Bomfim2002].

Os *JavaBeans* e as bibliotecas de *tags* personalizadas, que são componentes que encapsulam funcionalidade permitindo as aplicações baseadas em componentes, são partes ativas da tecnologia JSP [Bomfim2002].

Operando na camada de apresentação, as páginas JSP utilizam tecnologia Java do lado do servidor (*servlets*) para a criação de conteúdo dinâmico aliado com as *tags* HTML para manter o conteúdo estático ou modular.

Com a tecnologia JSP é possível, entre outras coisas, produzir aplicações que permitam o acesso a banco de dados, o acesso a arquivos-texto, a captação de informações a partir de formulários, a captação de informações sobre o visitante e sobre o servidor e o uso de variáveis e *loops*, entre outros [Ferreira2002].

O JSP não oferece nada que não possa ser criado com *servlets* puros. No entanto, ele oferece a vantagem de ser facilmente codificado, facilitando assim a elaboração e manutenção de uma aplicação Web. Além disso, a tecnologia JSP permite a separação da programação lógica (parte dinâmica) da programação visual (parte estática). Outra

[1] Na verdade, essa definição pode ser aplicada a outros sistemas destinados a gerar conteúdo dinâmico como ASP e PHP.

característica também é a possibilidade de produzir conteúdos dinâmicos que possam ser reutilizados.

As principais características JSP são [Ferreira2002]:

- Separação do conteúdo estático do dinâmico: em JSP, a lógica de geração de conteúdo é mantida separada das *tags* HTML responsáveis pela interface para o usuário. A parte lógica pode ser encapsulada em componentes *JavaBeans* externos, que podem ser utilizados pela página JSP através de *tags* especiais ou *scriptlets*.

- Interoperabilidade: sendo a tecnologia JSP baseada na plataforma Java, as páginas JSP podem ser "facilmente" portadas entre diferentes plataformas.

- Diversos formatos: qualquer formato atual pode ser utilizado numa página JSP, além de HTML. Podem ser utilizadas linguagens XML, DHTML, entre outras.

- Integração com a API Servlets: JSP pode ser considerado uma abstração de alto nível dos *servlets*, considerando que as páginas JSP são compiladas para gerarem *servlets*. Dessa forma, praticamente qualquer coisa feita em *servlets* pode ser feita em JSP.

- Utilização de código Java: é possível utilizar os chamados *scriplets*, que são nada mais que trechos de código Java puro inseridos dentro da página JSP.

Arquitetura do JSP

A arquitetura geral das páginas JSP tem muito em comum com os *servlets*, tendo em vista que a especificação JSP é definida como uma extensão da API Servlet [Ferreira2002].

As páginas JSP são submetidas a um processo de tradução e a uma fase de processamento de requisição [Ferreira2002]. A tradução é um processo que cria um arquivo .java e compila esse arquivo .java, gerando uma classe correspondente. A classe gerada se trata, basicamente, de uma classe *Servlet* correspondente.

Como uma classe Java será responsável por exibir o conteúdo da página JSP, o resultado é que as páginas JSP também desfrutam de todos os benefícios da linguagem e da plataforma Java [Bonfim2002].

A segunda fase, execução, é posta em funcionamento quando uma solicitação é recebida pela página. Quando uma página JSP é requisitada pelo cliente através de um

browser, a classe equivalente a esta página é executada pelo servidor, a partir daí será gerada uma página HTML que será enviada de volta ao *browser* do cliente. A figura 3.5 ilustra um funcionamento com componentes *JavaBeans*:

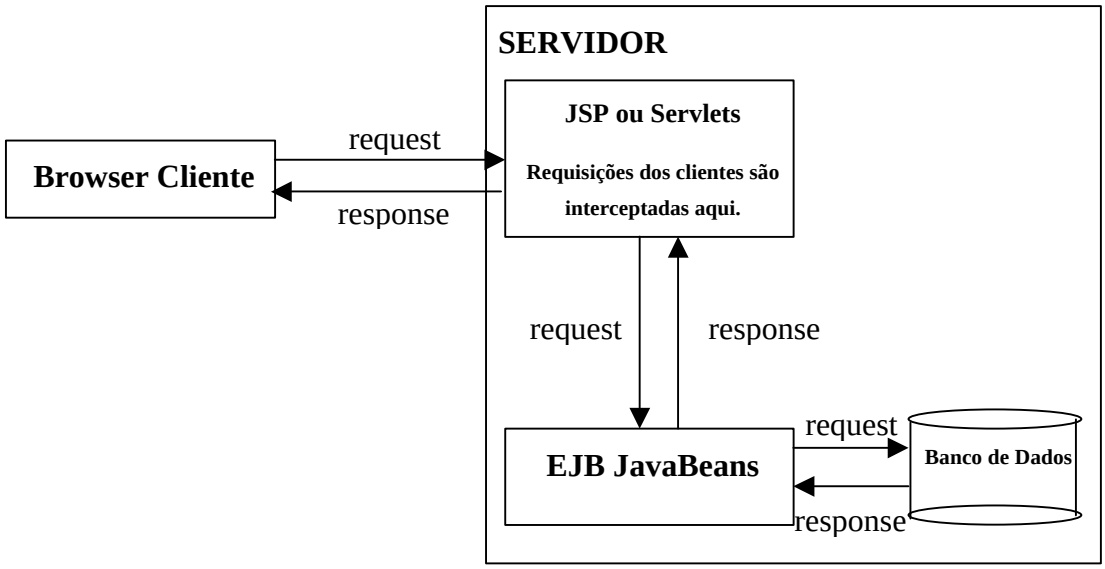

Figura 3.5 Arquitetura de uma requisição a uma página JSP usando JavaBeans

A especificação JSP 1.2 define ainda que a tecnologia JSP "herda da especificação Servlet os conceitos de *aplicação, contexto servlet, sessões, solicitações e respostas*".

O método de serviço da classe implementada a partir da página JSP que serve a requisição do cliente é *multithreaded* [Ferreira2002]. Por isso, é responsabilidade do autor da página garantir que o acesso a estados compartilhados seja sincronizado.

3.5.6 Web Service

A concepção de *Web Service* é a mais recente na evolução da computação distribuída. Ela torna capaz a comunicação entre aplicativos através da Web de uma forma eficaz, rápida e com baixo custo.

Um *Web Service* constitue uma arquitetura padrão de computação distribuída que faz computadores diferentes se comunicarem em rede [King2002]. Consiste num conjunto de padrões de desenvolvimento com a finalidade de implementar aplicações

distribuídas, cada padrão utilizado é responsável por determinadas tarefas, tais como transporte de mensagens entre aplicativos, codificação de padrão XML, entre outras.

Baseado na arquitetura *Web Service*, a Microsoft desenvolveu a plataforma .NET [Microsoft2003], que consiste numa iniciativa de software que tem como principal objetivo conectar informações, sistemas e dispositivos heterogêneos. A principal ferramenta de desenvolvimento da plataforma .NET é o Visual Studio .NET. Vários recursos para prover infra-estrutura à implementação de *Web Services* estão sendo desenvolvidos pela empresa de software, tais como o Windows .NET Server (indicado para o mercado corporativo como sucessor do Windows 2000) que facilita a criação de *Web Services* em XML devido a inclusão nativa do .NET *Framework* (plataforma voltada à construção, instalação e operação de Web services e aplicações no padrão XML).

A Sun desenvolveu o chamado Sun ONE - é o ambiente aberto Net da Sun (Sun Open Net Environment - Sun ONE) -, uma estrutura para suportar *Web Service* e cuja a plataforma Java 2 Enterprise Edition (J2EE) são fundamentais para seu funcionamento [King2002]. Isto significa que, no mundo de desenvolvimento da Sun, *Web Service* deve ser contruído usando *servlets*, páginas *JSP*, arquitetura *EJB* e outros padrões que fazem parte da tecnologia J2EE.

Empresas como Microsoft, Sun, IBM, HP, dentre outras, estão trabalhando para oferecer soluções *Web Services.* A Microsoft coloca-se de um lado com sua plataforma .NET, que roda sobre servidores Windows, enquanto as concorrentes apostam na plataforma aberta J2EE, baseada na linguagem Java. Mas as duas plataformas podem conversar entre si. A proposta é que *Web Services* possam ser utilizados por todos.

Três componentes de rede são evidentes para a evolução de *Web Service*: TCP/IP, HTTP/HTML e XML. São padrões que permanecem compatíveis até hoje.

Web Service reúne os seguintes requisitos de usuário [Fuks2003]:

- Acesso ao ambiente independente da estação de trabalho, já que uma de suas principais características é a interoperabilidade.
- Espaço privativo e público e a transição entre eles, já que é baseado em programação distribuída.

E também os requisitos do desenvolvedor:

- Reuso da experiência e conhecimento anteriores por parte dos desenvolvedores de *Web Service*, já que *Web Service* podem ser construídos em vários tipos de plataforma.
- Compartilhamento transparente dos dados, suporte a dados locais e compartilhados e escalabilidade, já que se baseia em programação distribuída.

XML: A chave da descrição Web Service

Web Service provê interfaces de transporte de componentes de dados e lógica de negócios através de HTTP. Uma enorme quantidade de dados esperam ser acessados por Web browsers e clientes de aplicação. XML está integrando dados em seus diversos ambientes de aplicação. Especificação de negócios e serviços podem ser encapsulados em documentos XML e apresentados como *Web Service*, facilitando assim transações de negócios e fornecendo uma cadeia de interação através da Web.

SOAP

SOAP (*Simple Object Access Protocol*) [SOAP2003] é uma tecnologia que deriva do padrão baseado em XML (XML-RPC) e do padrão chamado ebXML (Négocios Eletrônicos XML). EbXML é um trabalho em progresso, fornecendo uma definição detalhada de compartilhamento de mensagens de negócios através de parceiros comerciais [Clements2002].

O protocolo SOAP é uma chave importante na grande vantagem da tecnologia *Web Service*: a interoperabilidade. Através de SOAP, é permitido, por exemplo, que objetos Java e objetos COM conversem um com o outro, num ambiente baseado na Web, distribuído e descentralizado.

De uma maneira geral, SOAP permite objetos de qualquer espécie - em qualquer plataforma, em qualquer linguagem - se comunicarem. Atualmente, SOAP tem sido implementada em mais de sessenta linguagens em mais de vinte plataformas [Clements2002].

Em vez de invocar métodos através de um protocolo binário, um pacote SOAP usa XML, ou seja, uma sintaxe baseada em texto, para executar chamadas. Todas as chamadas entre requisição de aplicações e recebimento de objetos são feitas através de fluxo de dados XML sobre HTTP [Clements2002].

Mensagens e requisições de clientes SOAP são, portanto, tipicamente enviadas via HTTP. Como resultado, documentos SOAP são capazes de atravessar qualquer firewall, habilitando a troca de informações através de plataformas diferentes.

Um servidor SOAP é um simples código especial que escuta mensagens SOAP e age como um distribuidor e interpretador de documentos SOAP. O servidor SOAP recebe documentos através da conexão HTTP e os converte para uma linguagem que o objeto final seja capaz de entender.

Sem estudar a fundo a especificação de tipos de dados XML, a codificação SOAP pode ser descrita simplesmente como uma coleção de valores simples ou compostos. SOAP especifica regras para serialização de objetos, isto é, mecanismos para *marshalling* (converter dados em XML) e *unmarshalling* (extrair dados de XML) através da Internet.

SOAP estabelece um conjunto de regras que possibilita clientes e servidores a fazer invocação de procedimento remoto. SOAP, tendo como base um protocolo orientado a mensagem, pode trabalhar bem como um protocolo tipo RPC [Clements2002]. A serialização de objetos é o mecanismo principal do SOAP-RPC.

Padrão Web Service

A pilha de protocolos do *Web Service* é o conjunto de protocolos usados para definir e implementar *Web Services*. O núcleo da pilha de protocolos consiste de quatro camadas [Magic2002]:

- Camada de "Service Transport": Este inclui HTTP, SMTP, FTP, e mais novos como BEEP - Blocks Extensible Exchange Protolocol. Esta camada é responsável por transportar mensagens entre aplicativos.

- Camada "XML Messaging": Atualmente, este inclui XML-RPC (usa mensagens XML para realizar chamadas de procedimentos remotos) e SOAP - Simple Object Access Protocol. Esta camada é responsável por mensagens de codificação de padrão XML de forma que mensagens podem ser entendidas por qualquer plataforma.

- Camada de "Service Description": É manipulada via WSDL - Web Services Description Language. Esta camada é responsável por descrever a interface pública para o serviço específico.

- Camada "Service Discovery": É manipulada via o UDDI - Universal Description, Discovery and Integration. Esta camada é responsável por centralizar serviços em um registro do sistema comum, e prover a funcionalidade de “publish/find”.

O protocolo SOAP é combinado com o UDDI para prover registro e serviço de mensagens entre negócios. O SOAP, iniciando a conversação com o serviço UDDI, faz o acesso a objetos simples através de aplicações que invocam métodos de objetos, ou funções, em servidores residentes ou remotos [King2002].

A aplicação SOAP cria um bloco de requisição em XML, fornece dados necessários para os métodos remotos bem como a localização do objeto remoto.

Descrever o serviço disponível em um *Web Service* é função do padrão IDL proposto (linguagem de definição de interface) chamado WSDL (Web Service Description Language) [King2002]. A WSDL é referenciada pela UDDI como descrição de mensagens SOAP definidas para um Web Service particular. WSDL define *Web Service* como uma coleção de portas e operações.

Uma porta WSDL é análoga a uma interface e uma operação WSDL é análoga a um método específico. Assim, WSDL publica interfaces *Web Service* para partes interessadas se comunicarem através de plataformas heterogêneas [Clements2002].

UDDI age como um registro de publicação e localização *Web Service*.

	Java/RMI	CORBA	Web Services
Registro	RMI Registry	COS Naming	UDDI
Descrição de Serviços	Java	OMG IDL	WSDL
Transporte	JRMP[2]	IIOP	SOAP

Tabela 3.2 Comparação com outras soluções de RPC

[2] Java Remote Method Protocol

Implementação Web Service

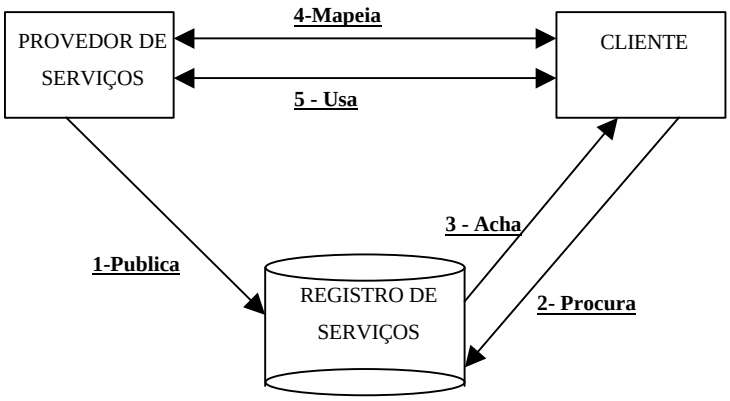

Figura 3.6 Funcionamento Web Service

a) Criação do *Web Service*: O provedor de serviços cria um *Web Service* e usa WSDL para descrever o serviço.

b) Publicação do *Web Service*: O provedor de serviços registra o serviço, através de UDDI.

c) Procura do *Web Service*: Outro serviço ou cliente localiza e solicita o serviço registrado através de uma consulta UDDI.

d) Ligação com o serviço registrado (mapeiar interface): O serviço de requisição ou cliente escreve uma aplicação para se ligar ao serviço registrado usando SOAP.

e) Uso do *Web-Service*: Dados e mensagens são trocados através de XML por HTTP.

3.5.7 Java/Web Service

De acordo com a especificação SOAP 1.1, SOAP é um "protocolo leve para transportar informações descentralizadas, em ambiente distribuído" [SOAP2003].

SOAP pode ser definido como uma linguagem que une linguagens de programação específicas. No contexto da linguagem de programação Java, foi definido o JAX RPC [JAX-RPC2003]. A tecnologia JAX [JAX-RPC2003] está habilitando a criação de *Web Services*, através da plataforma Java, usando a familiar tecnologia JSP e

componentes EJB. *Servlets* e *beans* são citados com as duas tecnologias Java mais adequadas para encapsular *Web Service*.

3.6 Análise comparativa

DCOM pode ser uma boa solução para aplicações que rodam exclusivamente no ambiente Windows. Entretanto o protótipo apresentado neste trabalho tem o objetivo de rodar em qualquer plataforma de software e hardware. Logo DCOM não atende os requisitos para o desenvolvimento do protótipo.

Ainda que fosse necessário o suporte à heterogeneidade na construção dos objetos em questão, a solução mais conviente seria uma implementação baseada em *Web Service*, pois a tecnologia *Web Service* trás as características de interoperabilidade existentes em CORBA aliadas a outras vantagens: enquanto CORBA é orientado a objeto, usando uma comunicação binária baseada no protocolo IIOP, carregado com stubs, skeletons e ORBs específicos, *Web Service* é leve, baseado em HTTP, XML e com isso capaz de atravessar qualquer *firewall*, e independente de plataforma e linguagem de programação.

Posto que o prótipo deve executar via Web, interoperabilidade combinada com comunicação baseada em HTTP seria a solução ideal para o caso de objetos heterogêneos.

As escolhas de Java/*Servlets*, Java/JSP e Java/RMI estão relacionadas com os requisitos para construção do protótipo, na disponibilidade e amadurecimento da tecnologia e na análise de custos financeiro e computacional envolvidos, ou seja Java RMI é uma API que vem em conjunto com o JDK, é gratuita, é mais eficiente que CORBA para aplicações Java-Java e precisa somente de uma Máquina Virtual Java (JVM) do lado do cliente, comuns aos navegadores mais conhecidos. No caso da escolha de *Servlets*, o servidor *Tomcat 4.0* é, também, um servidor gratuito do projeto Jarkata da APACHE [Bonfim2002].

Capítulo 4 Modelagem de um Sistema Java

Este capítulo descreve em detalhes um modelo para construção de um sistema Java, seus componentes, a iteração de cada componente com a linguagem de programação Java. Este modelo atuará como a base para construção do protótipo que será apresentado no próximo capítulo.

Capítulo 4 Modelagem de um Sistema Java

4.1 Introdução

O modelo básico do protótipo Web Aula (capítulo 5) pode ser visualizado como mostra a figura 4.1 abaixo. Ela é composta dos seguintes componentes:

- Clientes JSP (*Java Server Pages*) escritos na linguagem HTML (*HiperText Makup Language*) e na linguagem Java que podem acessar os objetos distribuídos através do protocolo HTTP (*HiperText Transfer Protocol*).
- Servidores escritos em Java *Servlets* que podem usar Java RMI como suporte a servidores distribuídos e JDBC (*Java DataBase Connectivity*) para acessar um bancos de dados MySQL.

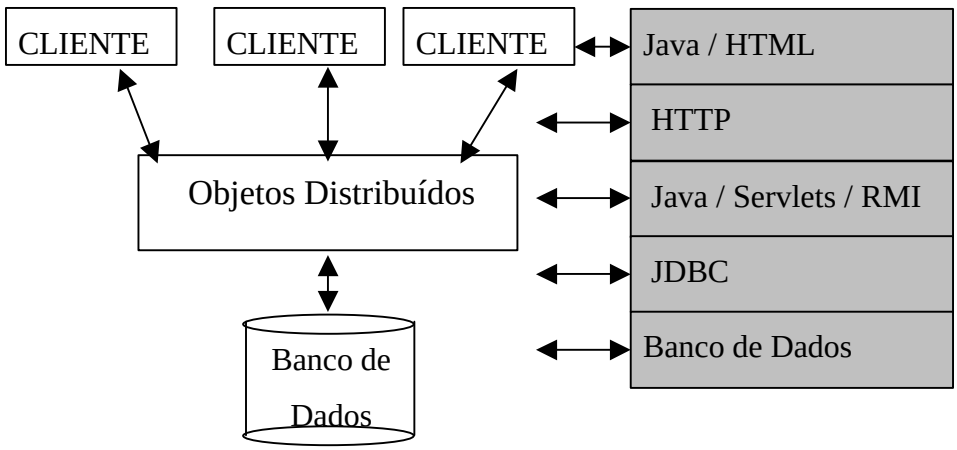

Figura 4.1 Modelo Geral

O modelo é baseado numa arquitetura cliente/servidor composto das seguintes camadas:

1° camada: É a interface com o usuário,

2° camada: Representada pelos objetos distribuídos, onde está contido a lógica do negócio.

3° camada: Representada pelo banco de dados.

Na primeira camada, a interface do usuário pode ser composta por vários clientes e podem estar rodando navegadores (browsers) em qualquer plataforma de hardware e sistema operacional que possua uma Máquina Virtual Java (JVM).

A interface do usuário se comunica com a camada dos objetos distribuídos através do protocolo HTTP com chamadas aos *Servlets*.

Nota-se a importância da linguagem Java para esta arquitetura através das camadas de interface do usuário, objetos distribuídos (*middleware*) e interface de acesso ao banco de dados. Todas as camadas possuem a linguagem de programação Java referenciada. Nos próximos tópicos serão abordados com maiores detalhes as características e vantagens de sua utilização, sua interface para aplicações distribuídas (Java / RMI), sua API para acesso a bancos de dados (Java / JDBC) e sua interface simplificada para Web (Java / *Servlets*).

4.2 Modelo Distribuído

Recursos de rede e computação distribuída em Java podem ser utilizados via invocação de método remoto (Remote Method Invocation –RMI). RMI permite que objetos Java executando no mesmo computador ou em computadores separados se comuniquem entre si via *chamadas de método remoto*.

RMI é semelhante a RPC, *chamada de procedimento remoto (remote procedure cals)*, desenvolvido nos anos 1980, que permite que um programa procedural (isto é, um programa escrito em C ou outra linguagem de programação procedural) chame uma função que reside em outro computador tão convinientemente como se essa função fosse parte do mesmo programa que executa no mesmo computador. Um objetivo de

RPC foi permitir aos programadores se concentrar nas tarefas exigidas de um aplicativo chamando funções e, ao mesmo tempo, tornar transparente para o programador o mecanismo que permite que as partes do aplicativo se comuniquem através de uma rede [Deitel2001]. A desvantagem de RPC é que ela suporta um conjunto limitado de tipos de dados simples. Portanto, RPC não é adequada para passar e retornar objetos Java.

RMI é a implementação de chamada remota por Java para comunicação distribuída de um objeto Java com outro. Uma vez que um método (ou serviço) de um objeto Java é registrado como sendo remotamente acessível, um cliente pode pesquisar ("lookup") esse serviço e receber uma referência que permita ao cliente utilizar este serviço (isto é, chamar o método). RMI oferece transferência de objetos de tipos e dados complexos via o mecanismo de *serialização* de objeto [Deitel2001]. A classe *ObjectOutputStream* converte qualquer objeto *Serializable* em um fluxo de bytes que pode ser transmitido através de uma rede. A classe *ObjectInputStream* reconstrói o objeto original para utilizar no método receptor. A transmissão de dados na rede é transparente. Usando RMI não é necessário aprender uma IDL porque todo código de rede é gerado diretamente a partir das classes existentes no programa. Além disso, uma vez que RMI suporta somente uma linguagem, Java, nenhuma IDL "neutra com relação à linguagem" é requerida; as próprias interfaces de Java são suficientes [Deitel2001].

Para criar objetos RMI são necessários os seguintes passos [Deitel2001]:

1. Definir uma interface remota que descreve como o cliente e o servidor se comunicam um com o outro:

   É feita uma descrição de métodos remotos que o cliente utilizará para interagir com o objeto servidor remoto por RMI. A interface remota estende a interface *Remote* (pacote *java.rmi*).

   Um objeto de uma classe que implementa a interface *Remote* direta ou indiretamente é um objeto remoto e pode ser acessado – com a devida permissão de segurança – a partir de qualquer máquina virtual Java que tenha uma conexão com o computador em que o objeto remoto executa.

   Se um problema de comunicação ocorre durante uma chamada de método remoto, um método remoto dispara uma *RemoteException* (um tipo de exceção verificada).

2. Definir o aplicativo servidor que implementa a interface remota:

A classe que implementa a interface remota estende a classe *UnicastRemoteObject* (pacote j*ava.rmi.server*) e deve ter no seu nome a terminação –Impl-.

A classe *UnicastRemoteObject* fornece a funcionalidade básica requerida por todos os objetos remotos: seu construtor exporta o objeto para torná-lo disponível para receber chamadas remotas. Exportar o objeto permite que o servidor remoto possa esperar conexões de cliente em um número de porta anônimo (o número é escolhido pelo computador no qual o objeto remoto executa). Isso configura o objeto para permitir comunicação *unicast* (comunicação ponto a ponto entre dois objetos via chamadas de método) utilizando conexões de soquete baseadas em fluxo padrão. Construtores para a classe *UnicastRemoteObject* podem permitir, também , informações adicionais como por exemplo o número de portas explícito em que um objeto remoto deve receber chamadas. Todos os construtores *UnicastRemoteObject* disparam *RemoteExceptions*.

O registro para objetos remotos é gerenciado pelo programa utilitário *rmiregistry* incluído no J2SDK.

O método *rebind* da classe *Naming* (pacote java.rmi) é chamado para vincular o objeto remoto ao *rmiregistry* e atribuir o nome usado pelo cliente para referenciar o objeto remoto no servidor.

3. Definir o aplicativo cliente que utiliza uma referência de interface remota para interagir com a implementação de servidor da interface, ou seja, um objeto da classe que implementa a interface remota:

O método *lookup* da classe *Naming* é usado para obter uma referência remota que permite ao cliente invocar métodos do objeto remoto.

4. Compilar e executar o servidor e o cliente:

A classe do servidor remoto deve ser compilada utilizando o compilador *rmic* (fornecido pelo J2SDK) para produzir uma classe *stub*. Um objeto de classe *stub* permite que o cliente invoque métodos remotos do objeto servidor. O objeto *stub* recebe cada chamada de método remoto e o passa para o sistema Java RMI, o qual

realiza as funções de rede que permitem que o cliente se conecte ao servidor e interaja com o objeto servidor remoto.

4.3 Componentes Multimídia

A maioria das linguagens de programação não têm capacidade de multimídia predefinida. Mas Java, por meio de pacotes de classes que são uma parte integralmente do seu mundo de programação, fornece instalações multimídia que permitem desenvolvimento de aplicativos multimídia poderosos.

Originalmente, Java suportava manipulações básicas de imagem e reprodução de clipes de áudio no formato de arquivo da Sun (arquivos com extensão .au). Hoje, existe uma extensão-padrão da Java API denominada –Java Media FrameWork (JMF), que fornece melhor processamento de imagens e reprodução de áudio aprimorada, além de suportar muitos formatos populares de áudio atuais. O JMF também inclui capacidade de reprodução de vídeo para vários formatos de vídeo. Novas versões incluem recursos como capacidade de gravar áudio e vídeo. O JMF 1.1 não faz parte do Software Development Kit do Java 2 (J2SDK). Portanto, é necessário descarregá-lo do site da Web da Sun Microsystems.

Para o caso específico de imagens, podem ser usadas duas classes: *Image* (pacote j*ava.awt*) e *ImageIcon* (pacote *javax.swing*) [Deitel2001].

A classe *Image* é uma classe *abstract*; portanto, não é possível criar um objeto da classe *Image* diretamente. Em vez disso, você deve solicitar que uma *Image* seja carregada e retornada para você. A classe *Applet* fornece um método chamado *GetImage* que tem a função de solicitar e carregar a imagem. Devem ser informados dois argumentos:

- O primeiro se refere a localização da imagem na Internet, ou seja, uma URL (pacote *java.net*).

- O segundo argumento especifica um nome de arquivo de imagem. Java, atualmente, suporta formatos de imagem como *Graphics Interchange Format (GIF)* e *Joint Photographic Experts Group (JPEG).*

A classe *ImageIcon* não é uma classe *abstract*; portanto, ela pode criar um objeto *ImageIcon.* O método *ImageIcon* não permite o redimensionamento de imagem.

Para o caso específico de áudio, Java fornece dois mecanismos para reproduzir sons – o método *Play* de *Applet* e o método *Play* da interface *Audio Clip* [Deitel2001]. O método *Play* de *Applet* reproduz o som apenas uma vez no programa.

O mecanismo de som que reproduz os clipes de áudio suporta vários formatos de arquivos de áudio, incluindo o formato de arquivo Sun (extensão .au), o formato Windows Wave (extensão .wav), o formato Macintosh AIFF (.aif ou .aiff) e o formato Musical Instrument Digital Interface (MIDI) (.mid ou .rmi). O Java Media Framework (JMF) suporta outros formatos adicionais.

O *AudioClip* é mais flexível. Permite que o áudio seja armazenado no programa de modo que possa ser reutilizado durante toda a execução do programa.

O Java Media Framework da Sun Microsystem oferece um aplicativo básico chamado *Java Media Player*. O player é capaz de reproduzir todos os formatos de áudio já anteriormente mencionados e uma variedade de outros formatos de áudio e vídeo como o AVI (.avi), GSM (.gsm), MPEG-1 (.mpg ou .mpeg), Apple QuickTime (.mov), RMF (.rmf), RTP (.rtp) e Vivo (.viv) [Deitel2001].

4.4 Banco de Dados

Um item relevante no desenvolvimento de aplicações Java no que diz respeito a banco de dados é o problema do "driver". As aplicações Java não operam diretamente com os drivers padrões da plataforma Windows, vale dizer ADO, OLE, ODBC. Os drivers compatíveis com aplicações Java seguem a API Java Database Connectivity, ou simplismente JDBC [Bonfim2002].

JDBC é basicamente um conjunto de classes e interfaces escritas na linguagem Java responsável por facilitar o envio de comandos SQL (Structured Query Language) para sistemas de banco de dados relacionais, com suporte a vários dialetos da linguagem SQL [Goulart1999].

A aproximação máxima que se pode obter com drivers não-JDBC é atingida por intermédio dos drivers que funcionam como "ponte" entre JDBC e ODBC *(Open DataBase Connectivity)*. Esses drivers são usados em conjunto com a grande variedade de produtos que possuem drivers ODBC disponíveis. Pontes JDBC/ODBC não serão utilizadas neste estudo.

A vantagem de usar tecnologia JDBC está em construir aplicações que podem acessar várias fontes de dados heterogêneos, podendo executar essa aplicação em qualquer plataforma que possua uma máquina virtual Java [Goulart1999].

A funcionalidade básica da API JDBC é prover basicamente [Goulart1999]:

- Conexão com banco de dados,
- Enviar comandos SQL e
- Processar resultados.

A conexão com banco de dados é feita em duas etapas: a primeira através da carga do driver *Class.forName ("package.DriverName")* e a segunda que cria uma conexão com um banco de dados específico que nada mais é do que uma instância da classe *Connection*, que através do método *getConnection()* da classe *DriverManager* recebe um objeto *Connection* que permite a conexão com um banco de dados específico.

Os comandos enviados para os bancos de dados podem ser de dois tipos, de consulta (*Select*) que retornam um objeto tipo *ResultSet*, que é um *array* com o resultado da consulta solicitada, e os comandos de atualização (*Insert, Delete, Update*), que retornam um valor inteiro como resultado. No caso de comandos de atualização, podem ser enviados também comando de DDL (*Data Definition Language*) como *Create Table e Drop Table*, sendo que nestes últimos dois casos é retornado o valor 0 (zero) caso o comando tenha sido executado com sucesso e um valor diferente de zero caso contrário.

Caso aconteça algum erro durante a execução de alguns desses comandos, tanto os de atualização quanto os de consulta, uma exceção (*SQLException*) é lançada e o erro é recuperado e tratado, sendo enviado, uma mensagem para o usuário.

O protótipo Web Aula (Capítulo 5) tomará como driver padrão JDBC o *org.gjt.mm.mysql.driver* para acessar o servidor de banco de dados MySQL. O driver *org.gjt.mm.mysql.*driver é um projeto "open source", portanto sua obtenção não implica em custo. No que diz respeito ao servidor de banco de dados, o MySQL tem código fonte aberto e gratuito. Junto com servidores Web Apache, o MySQL está formando a "tríade gratuita" mais utilizada na Internet, quando falamos em páginas dinâmicas que acessam banco de dados.

4.5 O Modelo na WEB

Uma característica importante desse modelo é o ambiente de execução dos clientes, com navegadores (*browsers*) acessando a Web. Uma das tecnologias que torna isso possível chama-se *Servlets.*

Os *Servlets* são programas escritos em Java que recebem pedidos e geram respostas para os clientes, normalmente através da linguagem HTML. Os *Servlets* rodam somente em servidores e rodam em conjunto com os servidores Web [Goulart1999]. Vários servidores Web já oferecem suporte a *Servlets.* No protótipo Web Aula será utilizado o *Tomcat 4.0* como servidor Web. O *Tomcat* é um servidor gratuito do projeto Jarkata da APACHE [Bonfim2002].

A API dos *Servlets* está dividida em dois pacotes, o *javax.servlet.http* e o *javax.servlet*. O primeiro pacote (*javax.servlet.http*) contém classes específicas para executar pedidos baseados no protocolo HTTP. O segundo pacote (*javax.servlet*) contém as classes genéricas que podem ser adaptadas para outros protocolos do tipo pedido / resposta [Goulart1999].

Todos os *Servlets* implementam a interface Servlet diretamente, ou através de outra classe que a implemente.

Os *Servlets*, no momento de sua invocação, fornecem dois objetos básicos que representam a ligação do cliente com o servidor. O *ServletRequest* encapsula o pedido do cliente para o servidor e o *ServletResponse* é responsável pela comunicação do servidor para o cliente, ou seja, é através desse objeto que o Servlet envia resposta do servidor para o cliente.

Os *Servlets* são para os servidores Web assim como os Applets são para os navegadores (browsers) [Goulart1999], ou seja, pequenos pedaços de código escritos em Java que implementam funcionalidades com segurança, robustez e independência de plataforma.

Numa aplicação *Servlet*, quando o usuário recebe resposta a uma solicitação, logo depois as informações a respeito deste usuário são perdidas. Na próxima solicitação que ele fizer, um novo processo será efetuado, pois o servidor não tem como obter informações a respeito de quem faz a solicitação. Isso ocorre devido ao fato de o protocolo de comunicação (HiperText Transfer Protocol) não manter informações no intervalo das solicitações [Bomfim2002]. Contudo, em muitas aplicações é fundamental a manutenção de informações que permitam a identificação do usuário. No protótipo Web Aula a ser desenvolvido nesse estudo quando um usuário iniciar um processo, uma **sessão** será estabelecida. Essa sessão conterá informações relevantes que serão a todo instante atualizadas. Para manter o vínculo entre a sessão e o usuário serão empregados mecanismos chamados **cookies**.

Um **cookie** é um objeto [Bomfim2002] no qual é armazenado um par nome-valor e enviado ao usuário por meio do cabeçalho HTTP. As informações são recebidas e armazenadas pelo navegador do usuário. Mais tarde, quando o usuário acessar a URL que remeteu um determinado cookie, o navegador reenviará esse cookie ao servidor que o enviou. Um cookie somente é reencaminhado ao URL de origem e a nenhum outro. Com isso, informações diversas podem ser armazenadas junto ao usuário para que ele possa ser posteriormente identificado no gerenciamento de uma sessão.

O gerenciamento de sessões em aplicações *Servlets* é diretamente dependente da interface *ServletContext*. A *ServletContext* trata do ambiente no qual está o servlet,. ou seja, toda aplicação está localizada em um servidor. Nesse servidor um processo específico e em separado chamado *container servlet* fornece o suporte necessário à funcionalidade dos servlets. Os métodos da *ServletContext* habilitam o servlet a se relacionar com esse ambiente no qual ele se encontra, o container servlet [Bomfim2002]. Por meio de um objeto *ServletContext*, um servlet pode definir e armazenar atributos que podem ser acessados por outros servlets.

4.6 Considerações Finais

A desvantagem na utilização de "*applets*" são as seguintes:

- A transferência dos "*applets*", do Servidor Web para o computador do usuário, vai depender do tamanho dos arquivos.

- A execução de "*applets*" é lenta - o código (chamado de "*byte-code*") precisa ser interpretado localmente por uma "máquina virtual" (interpretador de código Java). Com applets, a máquina cliente executa efetivamente parte da aplicação Java, portanto há necessidade de que ela tenha maior velocidade de processamento e capacidade de memória. O servidor se concentra apenas em receber e responder às solicitações do cliente. Clientes e Servidor executam a aplicação.

- O *applet* somente será visualizado pelo browser quanto todos os arquivos forem transferidos. Em *applets* o código Java é carregado a partir do servidor Web. Em uma rede local este tempo de carga não se mostra tão problemático, entretanto via Internet este tempo de *download* pode ser significativo. No entanto, a tecnologia atual permite que a aplicação seja carregada de forma compactada, reduzindo significativamente este tempo e, mais importante ainda, que a aplicação seja carregada apenas uma vez, eliminando o tempo de carga de todos os seus usos subsequentes, até que sofra alguma mudança.

Apesar das dificuldades em relação a execução de *"applets"*, no diz respeito ao tempo que ele leva para ser carregado, *"applet"* é bastante utilizado como recurso de Educação à Distância em razão das vantagens de interatividade que é capaz de propor. No protótipo apresentado neste trabalho, o *Java Media Framework* é utilizado em conjunto com um *"applet"* com a finalidade de visualização de vídeo.

A utilização do *Java Media Framework* envolve estratégias de resolução dos principais problemas encontrados no desenvolvimento de aplicações envolvidas no transporte, sincronização, controle e apresentação de mídias contínuas. Como benefício adicional, por se tratar de uma implementação em Java, ela se torna independente de plataforma. Além disso, são utilizados mecanismos difundidos para o desenvolvimento de conteúdo multimídia para o ambiente WWW, sem demandar a presença de *plug-ins*

adicionais na estação do cliente. Além disso, suporta os principais formatos padronizados de codificação (MPEG-1, MPEG-2, MIDI, AVI, AU e WAV, por exemplo).

A peça chave para a execução do vídeo é a interface *Player*. Um "*applet*" é responsável por chamar o *Player*, e assim, ler e processar fluxos de dados multimídia lidos a partir de uma determinada fonte, e exibi-los em um determinado intervalo de tempo.

Capítulo 5 WEB-AULA: PROTÓTIPO

Este capítulo apresenta as características do protótipo WEB-AULA, que foi desenvolvido durante o desenvolvimento desta dissertação e serve de exemplo para avaliar o uso das tecnologias abordadas no capítulo 3, bem como validar o modelo proposto no capítulo 4.

Capítulo 5 WEB-AULA: PROTÓTIPO

A Educação à Distância na Internet envolve informação digital, formas de sua definição, aquisição, organização, gerenciamento e disseminação através das redes de comunicação global. Um sistema EAD na Internet pode ser descrito como uma coleção de serviços e recursos, usualmente distribuídos, e que atuam sobre informações digitais.

O protótipo Web-Aula deve ser fácil de manipular e ter um baixo custo, evitando soluções caras como a videoconferência.

As tecnologias utilizadas foram amplamente discutidas nos capítulos anteriores e serão abordadas sempre que necessário no decorrer desta descrição, ficando claro que foi priorizado o uso de tecnologias de domínio público.

5.1 Proposta Inicial

O projeto Web Aula consiste na instalação de um sistema de cadastro e consultas de artigos-aula, fornecendo aos usuários/alunos interfaces baseadas em navegadores Web para acesso dos artigos-aula cadastrados e inclusão de comentários sobre os mesmos. Os alunos também podem consultar comentários inseridos sobre os artigos-aula consultados. Um artigo-aula é composto das seguintes informações: título, tema, autor, síntese, nome do artigo, um artigo escrito em html e um vídeo-aula.

Este capítulo descreve funcionalidades e duas possíveis arquiteturas de implementação propostas para o protótipo:

- Na primeira proposta, o banco de dados é centralizado em um único servidor e as pesquisas são feitas baseadas em servlets.

- Na segunda proposta, cada usuário professor terá seu próprio banco de dados local, um objeto servidor remoto deverá ser ativado pelo próprio usuário professor a fim de que seu banco de dados local possa ser pesquisado pelos alunos via rede.

5.2 Modelo da Interface do Usuário Professor

A interface do usuário professor, implementada via uma página Web, possui as características necessárias para que o usuário possa cadastrar seu artigo e o respectivo vídeo-aula relacionado.

A interface é composta pelos seguintes componentes:

(a) Formulário de autenticação

(b) Formulário de menu de cadastro

(c) Formulário de inclusão

(d) Formulário de atualização

(e) Formulário de exclusão

5.2.1 Formulário de autenticação

O formulário de autenticação é constituído de campos que permitem ao usuário definir atributos como “login” e “senha”, conforme mostra a figura a seguir.

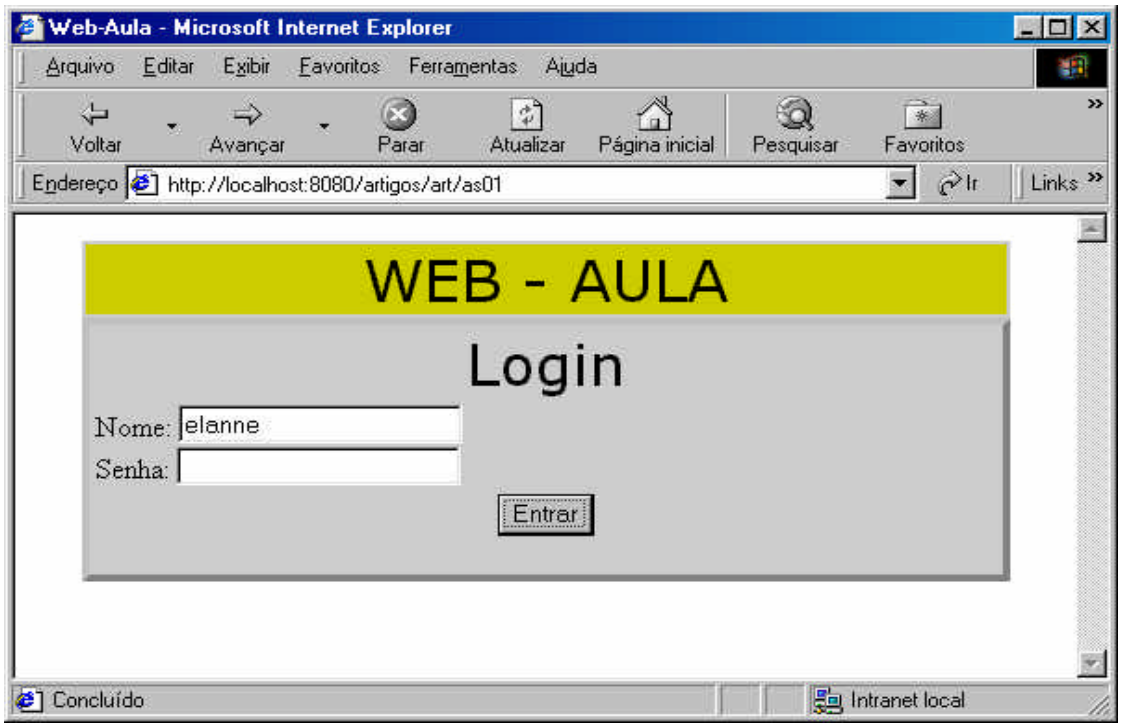

Figura 5.1 Formulário de autenticação.

5.2.2 Formulário de menu de cadastro

Se os atributos informados pelo usuário, no formulário de autenticação, forem validados, uma sessão automaticamente será criada para esse usuário a fim de que seja possível o seu acesso ao menu de cadastro.

Conforme é mostrado na figura 5.2, no menu de cadastro pode ser feitas:

(a) Inclusão de um Artigo-Aula

(b) Atualização de um Artigo-Aula

(c) Exclusão de um Artigo-Aula.

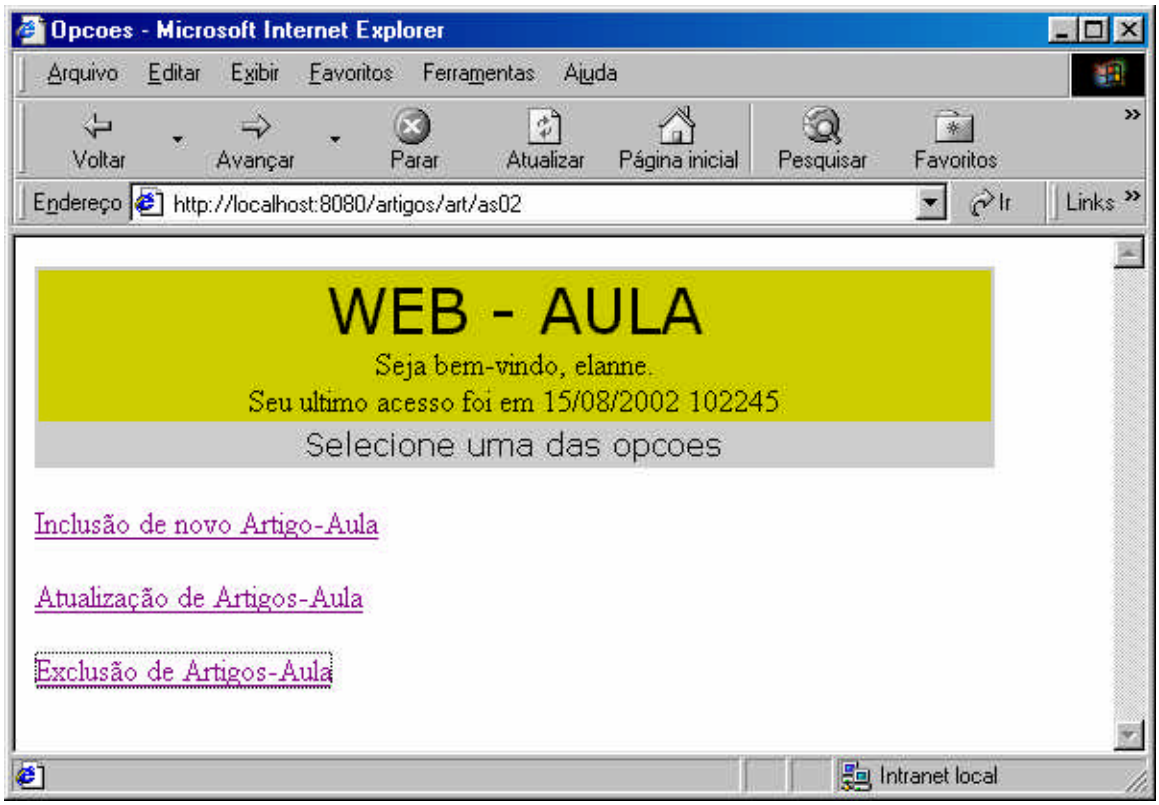

Figura 5.2 Formulário de Menu de Cadastro.

5.2.3 Formulário de inclusão

O formulário de inclusão é composto de campos permitindo ao usuário professor definir os atributos do objeto que o usuário aluno irá pesquisar. Os atributos são:

(a) título,

(b) tema,

(c) autor,

(d) síntese,

(e) nome do artigo,

(f) nome do arquivo do vídeo-aula

(g) data.

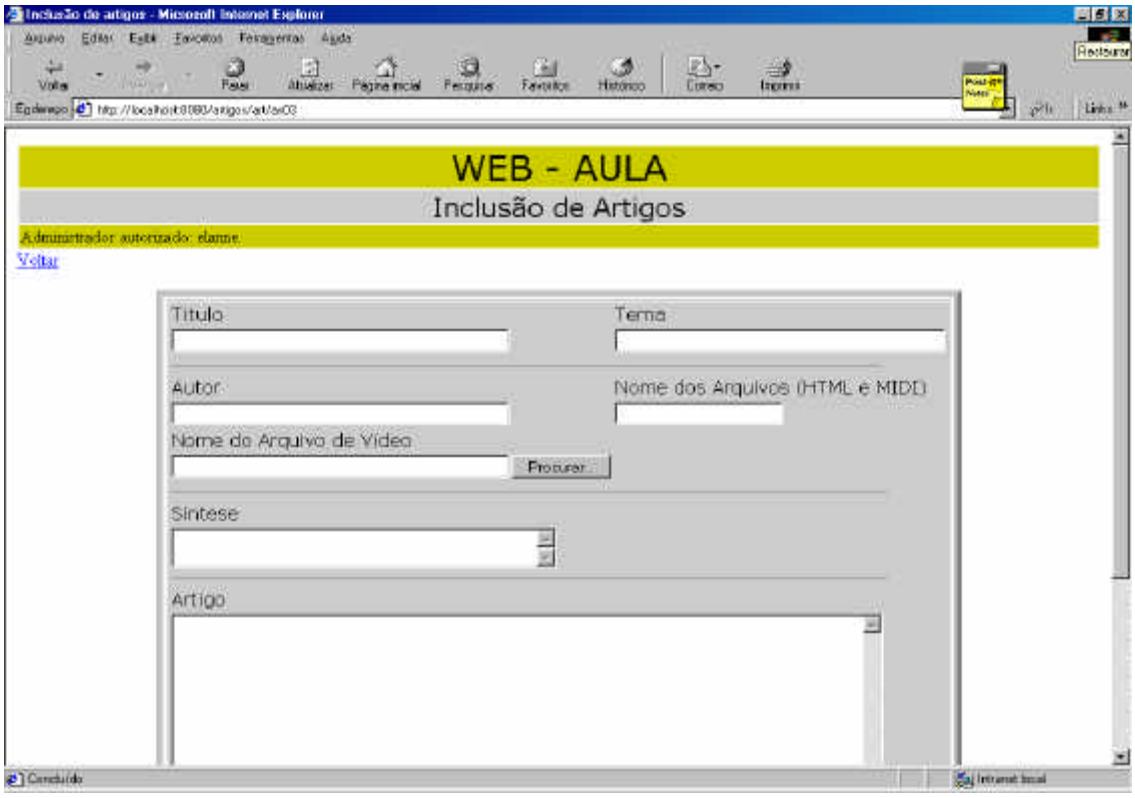

Figura 5.3 Formulário de Inclusão de Artigos.

5.2.4 Formulário de atualização

O formulário de atualização disponibiliza a alteração de um artigo-aula já existente, conforme mostra a figura 5.3.

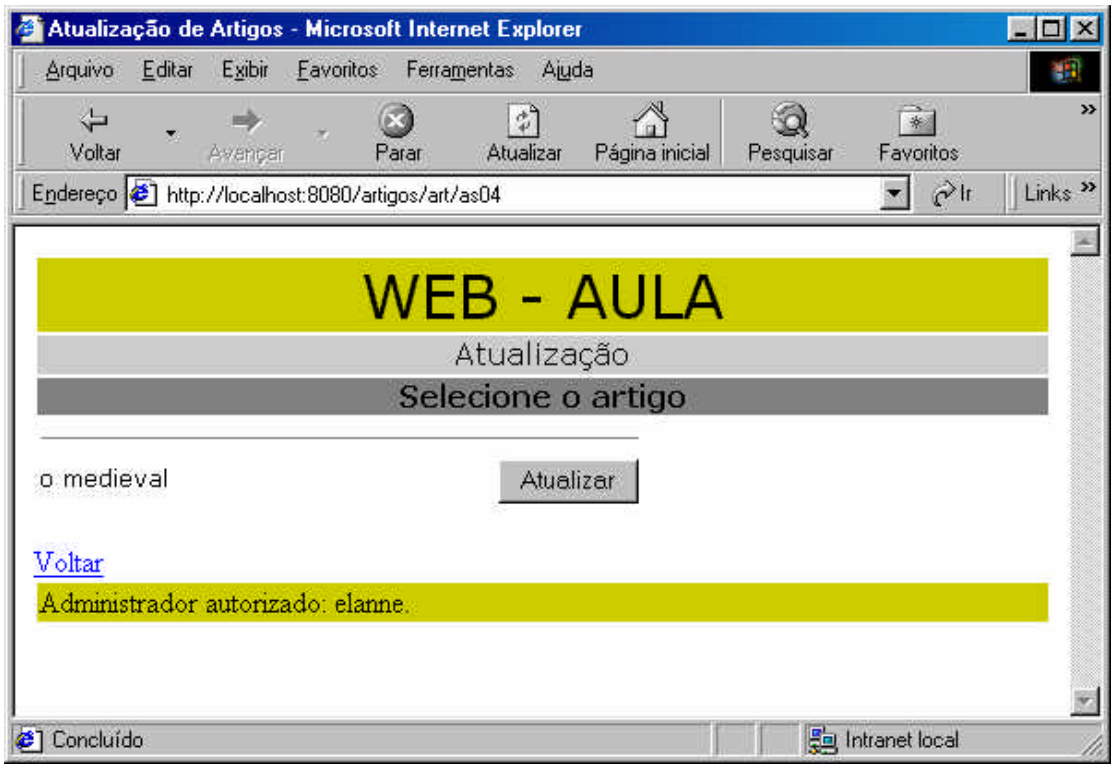

Figura 5.4 Formulário de Atualização.

5.2.5 Formulário de exclusão

O formulário de exclusão disponibiliza a exclusão de um artigo-aula já existente.

5.3 Modelo da Interface Administrador

A interface administrador é usada para cadastrar os usuários do Web-Aula. Esta funcionalidade só poderá ser acessada pelo (s) superusuário (s) do sistema.

5.4 Modelo da Interface Usuário Aluno

A interface do aluno é usada para solicitar uma determinada pesquisa, via Web, ao banco de dados Web-Aula.

A interface é composta pelos seguintes componentes:

(a) Formulário de autenticação

(b) Formulário de solicitação de pesquisa

(c) Formulário de apresentação de resultado de pesquisa

(d) Formulário de leitura do artigo

(e) Formulário de inclusão de comentário

(f) Formulário de acesso ao vídeo-aula

Similar à interface do professor, possui um formulário de autenticação constituído de campos que permitem ao usuário definir atributos como “login” e “senha”.

5.4.1 Formulário de solicitação de pesquisa

Se os atributos informados pelo usuário no formulário de autenticação forem validados, uma sessão automaticamente será criada para esse usuário a fim de que seja possível o seu acesso ao formulário de solicitação de pesquisa.

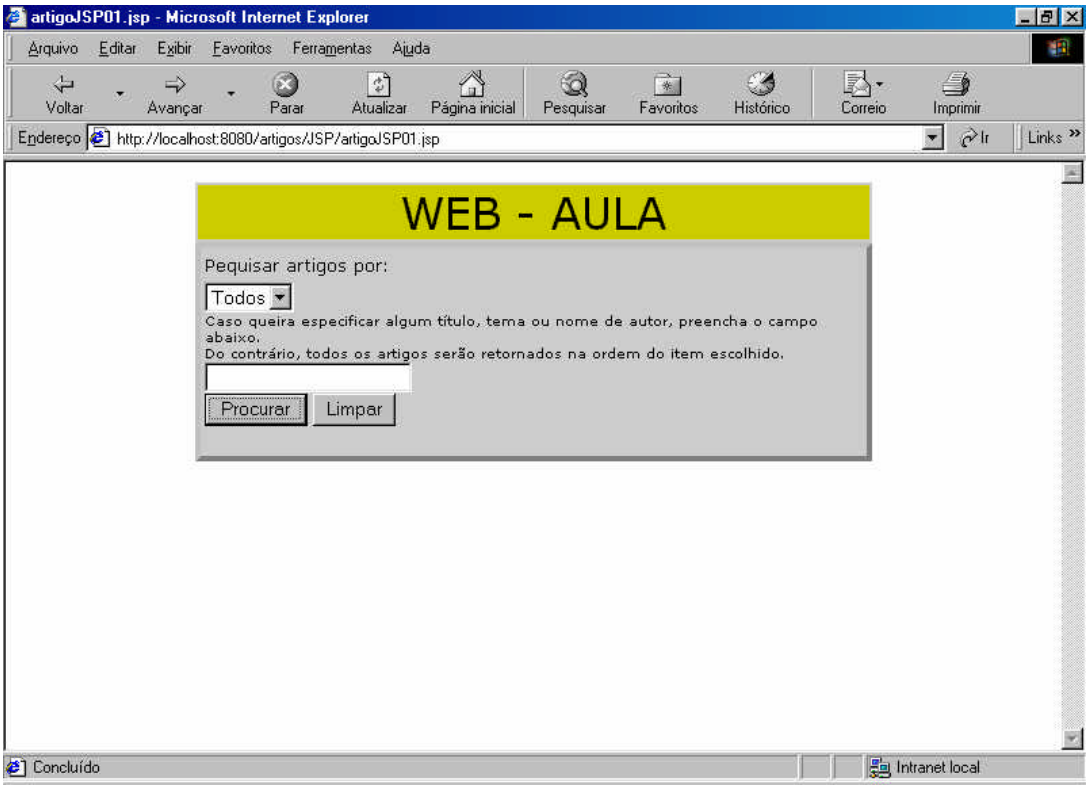

Figura 5.5 Formulário de solicitação de pesquisa.

5.4.2 Formulário de apresentação do resultado da pesquisa

No Formulário de apresentação de resultado da pesquisa serão exibidas informações sobre um determinado artigo-aula e disponibilizados três links:

- Um que permite ao usuário aluno ler o arquivo html relacionado ao artigo-aula pesquisado.
- Um que permite ao usuário aluno executar o arquivo de vídeo relacionado ao artigo-aula pesquisado.
- E outro que permite que o usuário aluno leia comentários sobre o artigo-aula.

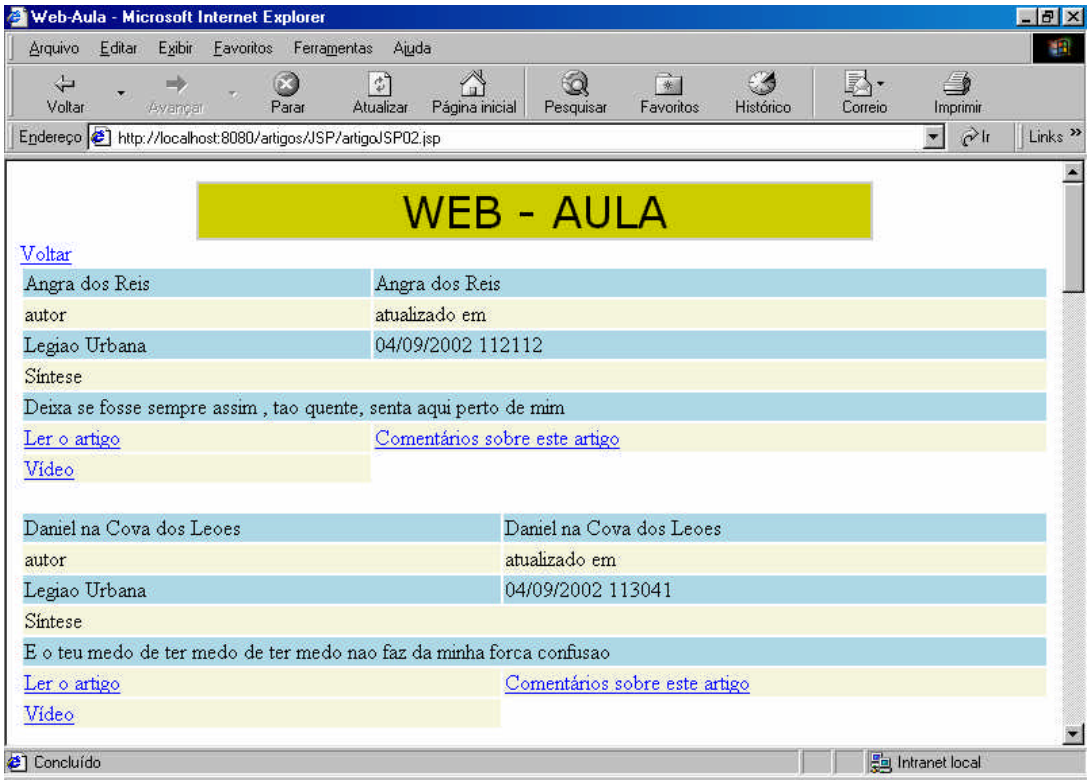

Figura 5.6 Formulário de apresentação do resultado da pesquisa.

5.4.3 Formulário de leitura do artigo

No formulário de leitura do artigo serão exibidos o artigo propriamente dito e disponibilizados dois links:

- um para inclusão de comentário sobre o artigo;
- outro para consultar os comentários sobre o artigo.

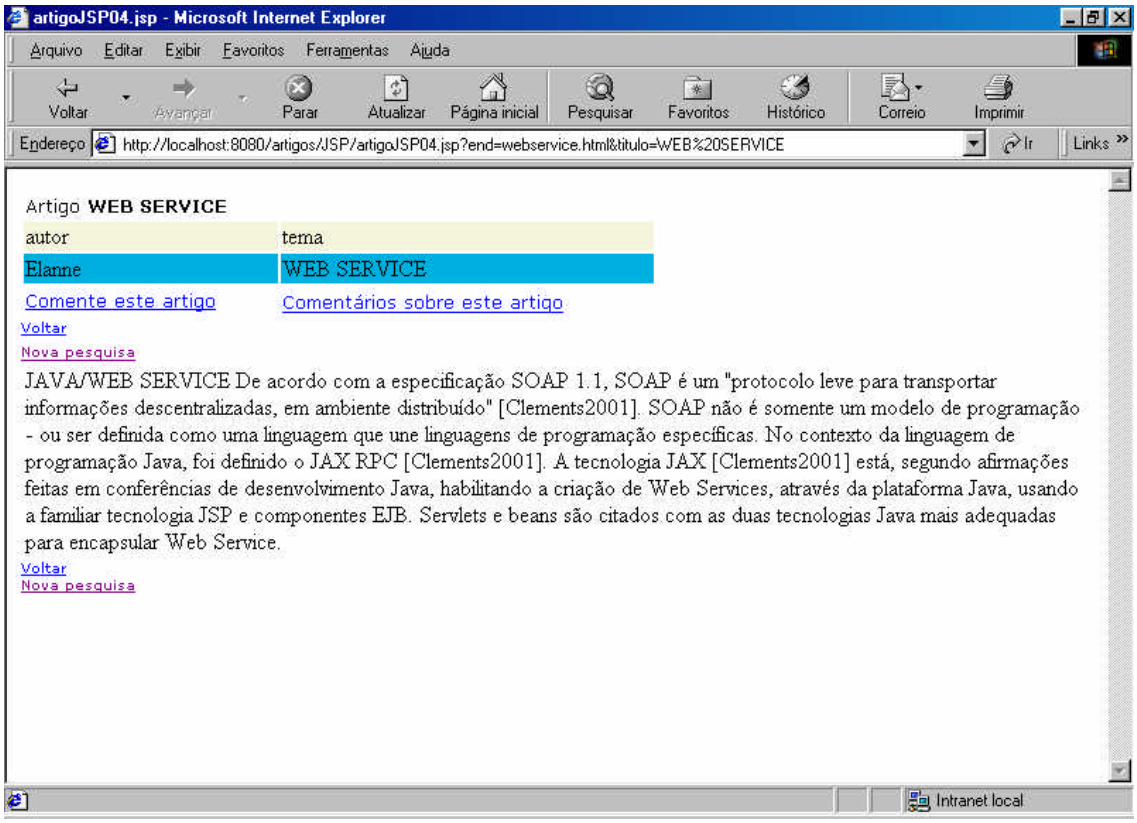

Figura 5.7 Formulário de leitura do artigo.

5.4.4 Formulário de inclusão de comentário

No formulário de inclusão de comentário, o aluno pode inserir um comentário a respeito do artigo lido e informações como e-mail para posterior obtenção de respostas no caso de dúvidas.

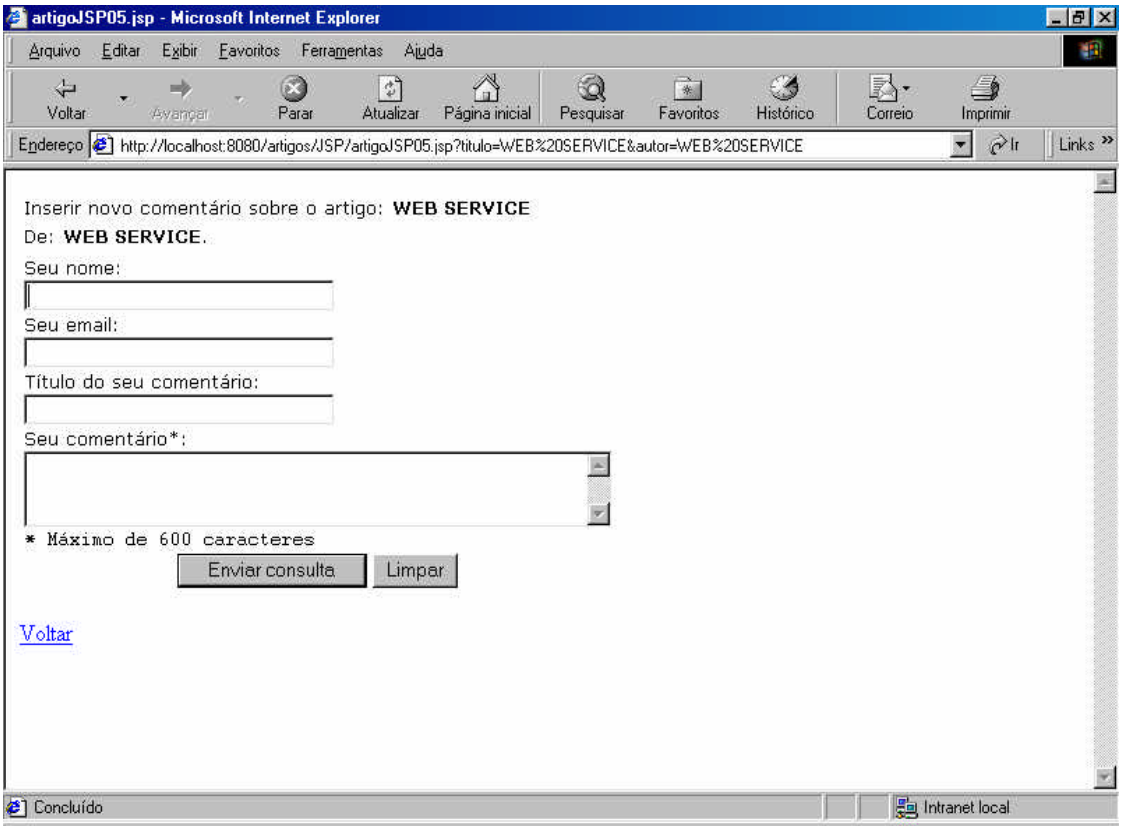

Figura 5.8 Formulário de inclusão de comentário.

5.4.5 Formulário de acesso ao vídeo-aula

No formulário de acesso ao vídeo-aula o aluno pode assistir o vídeo relacionado àquele artigo.

5.5 Arquitetura

Dois modelos de arquitetura foram estudados para a criação do protótipo e serão apresentados logo abaixo.

5.5.1 Arquitetura baseada em Servlet

No primeiro modelo de arquitetura, o banco de dados é centralizado em um único servidor e as pesquisas são feitas baseadas em servlets (**Figura 5.9**).

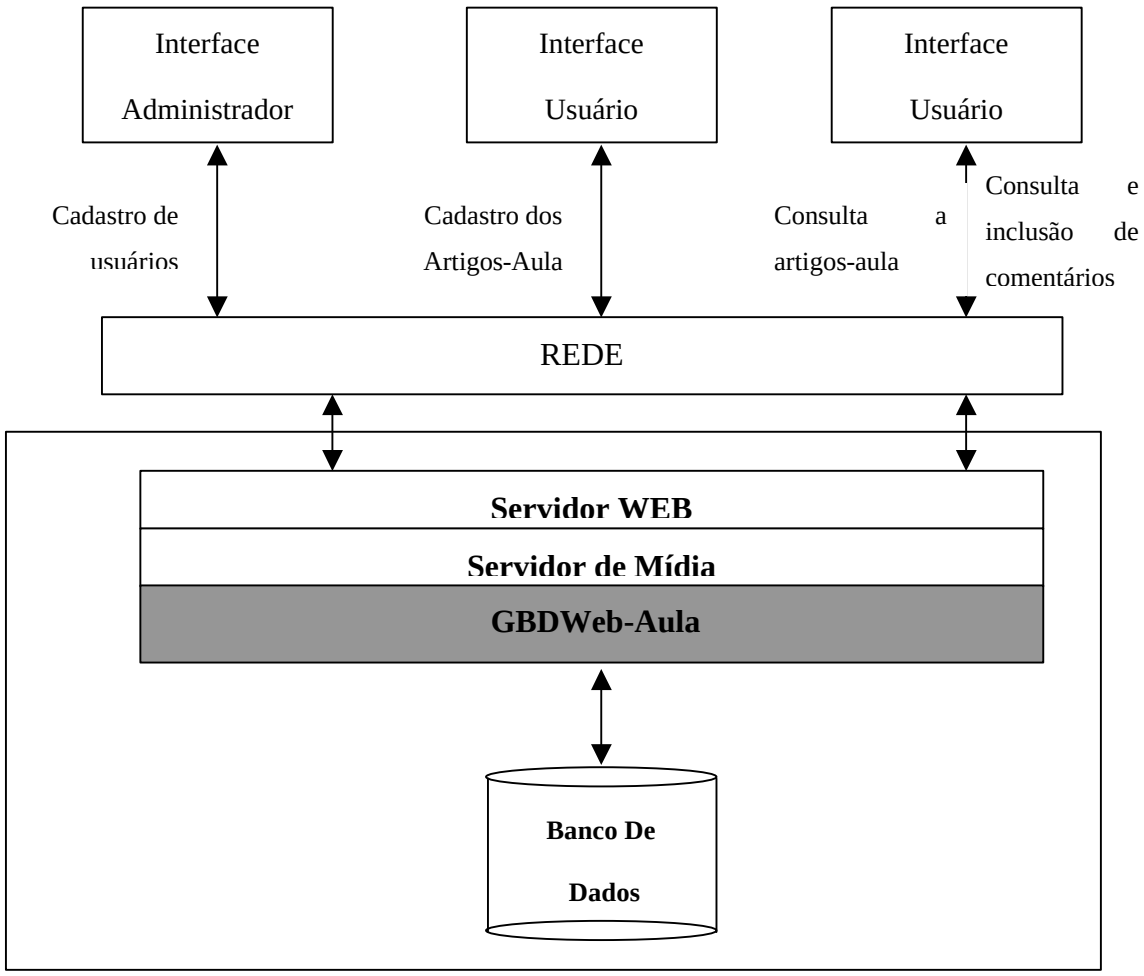

Figura 5.9 Arquitetura baseada em Servlets.

São três as interfaces propostas para o protótipo Web-Aula:

(a) Interface Administrador,

(b) Interface do Usuário Aluno e

(c) Interface do Usuário Professor.

Servidor WEB

O servidor WEB é usado com a finalidade de fornecer acesso aos usuários à sua interface Web-Aula correspondente e ao Banco de Dados Web-Aula através de navegadores Web.

GBDWeb-Aula

O Gerenciador de Banco de Dados Web-Aula é uma classe *servlet* que tem como tarefa implementar as funções disponibilizadas nas diversas interfaces realizando, através dessas funções, o acesso ao Banco de Dados.

Banco de Dados

O Banco de Dados tem o propósito de armazenar os dados que descrevem os artigos-aula para que eles possam ser manipulados pelos usuários. O SGBD adotado no protótipo foi o MySQL [MySQL].

As tabelas manipuladas no Web-Aula serão:

Usuários = Cadastro de usuários do Web-Aula

Artigos-aula = Cadastro dos artigos-aula pesquisados

Comentários = Cadastro de comentários sobre artigos-aula.

Servidor de Mídia

O servidor de mídia é utilizado pelo Web-Aula para armazenar as vídeo-aulas cadastradas. O servidor de mídia está localizado na mesma máquina que o servidor Web.

5.5.2 Arquitetura baseada em Servlet/RMI

No segundo modelo de arquitetura, cada usuário professor terá seu próprio banco de dados local, um objeto servidor remoto deverá ser ativado pelo próprio usuário (professor) a fim de que seu banco de dados local possa ser pesquisado pelos alunos via rede.

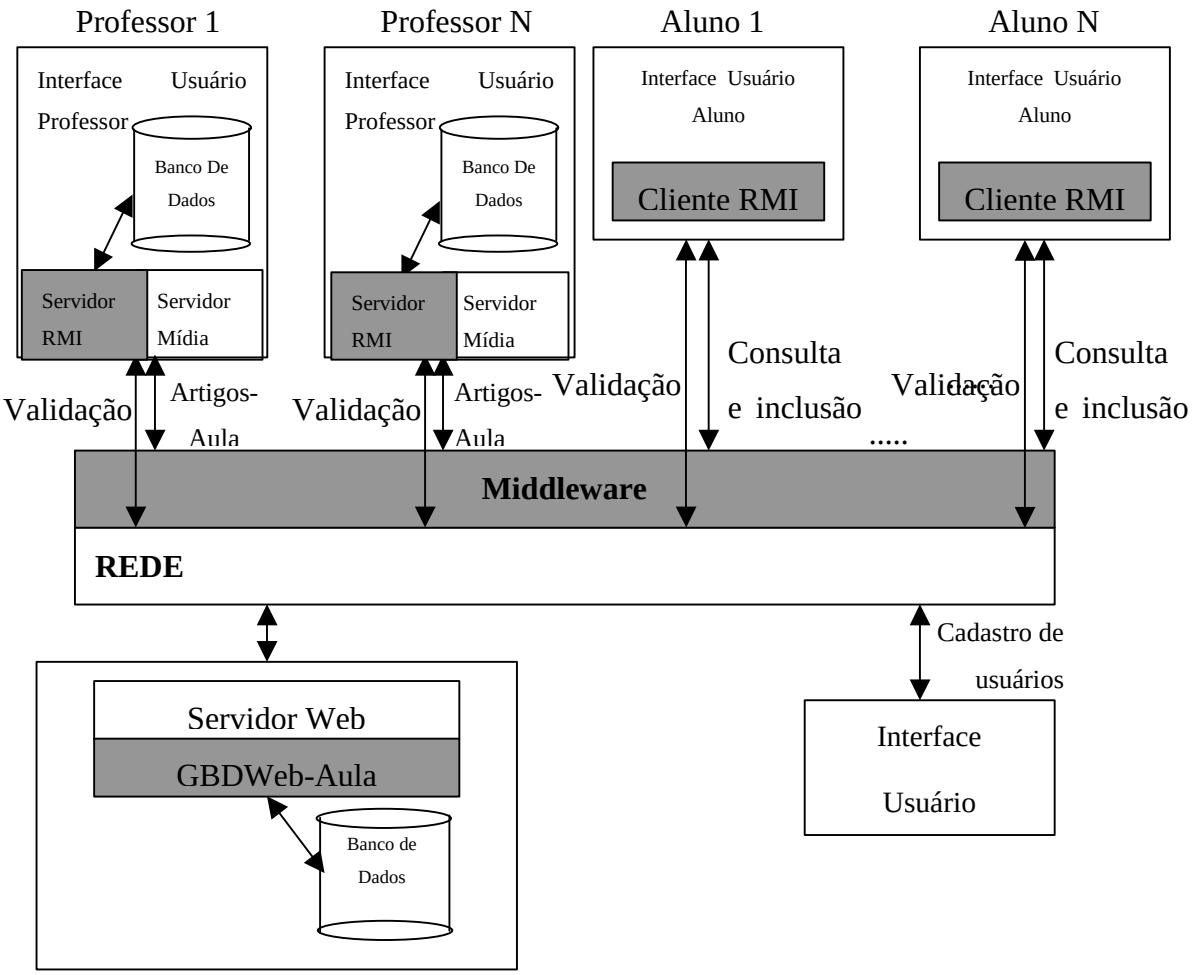

Figura 5.10 Arquitetura baseada em Java Servlets/RMI.

Servidor WEB

O servidor WEB é usado com a finalidade de fornecer acesso aos usuários à sua interface Web-Aula correspondente através de navegadores Web.

GBDWeb-Aula

O Gerenciador de Banco de Dados Web-Aula é uma classe *servlet* que tem como tarefa implementar as funções de validação de acesso de usuários disponibilizadas nas diversas interfaces. Somente depois que o usuário for devidamente validado pelo GBDWeb-Aula é que será permitido a ele acessar o ambiente Web-Aula.

Os professores validados no ambiente Web-Aula serão automaticamente cadastrados na lista de professores ativos na rede. Os alunos, ao acessarem o formulário

de solicitação de pesquisa, terão acesso à lista de professores ativos e poderão escolher qual o banco de dados onde desejam efetuar a pesquisa.

Banco de Dados

O SGBD foi o MySQL [MySQL] e há dois tipos de Banco de Dados:

- Um local na máquina de cada professor com o propósito de armazenar os dados que descrevem os artigos-aula cadastrados pelo usuário professor e consultados pelos usuários alunos.

  As tabelas manipuladas são:

  Artigos-aula = Cadastro dos artigos-aula pesquisados;

  Comentarios = Casdastro de comentários sobre artigos-aula.

- Outro remoto na máquina servidora com o propósito de armazenar os dados que descrevem os usuários do Web-Aula. Somente o usuário Administrador tem permissão de escrita neste banco de dados.

  A tabela manipulada é:

  Usuários = Cadastro dos usuários do Web-Aula.

Servidor de Mídia

O servidor de mídia é utilizado pelo Web-Aula para armazenar e recuperar as vídeo-aulas cadastradas.

Middleware

O middleware escolhido foi Java RMI, no qual um cliente solicita ao objeto remoto *"Pesquisa"* o resultado de algum tipo de consulta ao banco de dados.

Servidor RMI

O servidor RMI é uma classe *UnicastRemoteObject* que implementa as funções disponibilizadas na interface do usuário aluno, permitindo, através dessas funções, o acesso remoto do usuário (aluno) ao banco de dados de um usuário professor.

Cliente RMI

O cliente RMI tem a função de localizar o servidor remoto e efetuar as pesquisas no banco de dados remoto do usuário professor.

5.6 Conclusão

O trabalho trata do estudo de tecnologias Java com a finalidade de criação de um sistema EAD para Web que apresente interatividade entre alunos e professores mantendo baixo custo. Para tanto foram estudadas maneiras de aproveitar as características distribuídas do sistema, de libertar o usuário dos problemas de heterogeneidade de plataforma e fácil de utilizar.

Ao final do trabalho pode se observar um resumo do estado atual dos sistemas de EAD e das tecnologias Java, incluindo uma proposta de sistema e um estudo de arquiteturas para seu desenvolvimento. Dois protótipos, um para cada arquitetura, foram desenvolvidos para mostrar a viabilidade da proposta, resta testá-la a uma alta carga de trabalho (um número de alunos maior em uma hora onde há grande volume de dados trafegando na rede).

É necessário, ainda, desenvolver o sistema completo, incluindo módulos de comunicação síncrona, como sala de aula com chat, por exemplo, para alunos e professores se comunicarem num dado instante e discutirem os artigos-aula estudados.

Capítulo 6 Considerações Finais

Este capítulo apresenta as considerações finais, tais como contribuições desta dissertação, as limitações do protótipo e sugere os trabalhos futuros.

Capítulo 6 Considerações Finais

6.1 A Educação à Distância

A aplicação de novas abordagens de ensino-aprendizagem e estratégias pedagógicas têm sido possível graças ao uso, cada dia mais crescente, da Informática e de novas tecnologias de comunicação. A união da computação aos meios de comunicação facilita o acesso às informações, gerando assim novas perspectivas para o sistema educacional vigente.

Com o acesso à *Internet* facilitado pela WWW, tem-se um novo recurso tecnológico que, ao permitir a integração de texto, imagem, áudio e vídeo, disponíveis em qualquer parte do planeta, traz novas possibilidades à tecnologia da educação, no sentido da criação de materiais mais eficientes e novas técnicas de ensino, aonde a presença física de instrutores não é mais necessária [Bica1999].

Um aspecto importante a considerar é que o estudante é um indivíduo com características próprias, que devem ser respeitadas, merecendo portanto atenção ao ritmo de estudo individual. Deve-se levar em conta seu comportamento e os mecanismos que podem facilitar sua aprendizagem. Os alunos, geralmente, têm forte influência dos métodos presenciais tradicionais e, principalmente, são pouco educados a estudar a partir de seu próprio esforço individual. É fundamental que se oriente o estudante a estudar por conta própria, desenvolvendo suas habilidades e iniciativa [Bica1999].

Este projeto teve o objetivo de apresentar tecnologias Java disponíveis para a construção de ambientes de Educação a Distância na Internet e propor a utilização dessas tecnologias em dois tipos de arquiteturas diferentes. Espera-se com isto contribuir para que a Educação a Distância possa dar mais um passo para ser uma

alternativa viável no processo educacional, no sentido de que tecnologias de domínio público disponíveis hoje se mostram eficazes na construção de ambientes EAD para Internet.

6.2 As Tecnologias Envolvidas

A contribuição de computadores e da Internet nos estudos de Educação a Distância tem como um objetivo principal dinamizar o processo de aprendizagem, através de sistemas que implementem um ambiente ativo de cooperação entre os usuários envolvidos, tanto alunos como professores.

As tecnologias estudadas permitem a construção de ambientes que habilitam indivíduos a se engajarem na atividade de produção de conhecimento compartilhado.

Determinar o tipo de tecnologia a ser empregada em um ambiente depende dos objetivos educacionais apontados. No caso de ambientes EAD para Internet, *Servlets* e *JSP* são importantes aliados na hora de usar tecnologias para construção de aplicações distribuídas.

Dentre as tecnologias para aplicações distribuídas, cada uma se mostra mais adequada a um determinado tipo de situação. Por exemplo, CORBA em ambientes para Internet apresenta limitações relacionadas ao uso de esquemas de segurança baseados em *firewalls* que dificultam a comunicação entre os componentes de aplicações baseadas em objetos distribuídos CORBA na Internet. A limitação de Java RMI está no que diz respeito ao seu uso em ambientes heterogêneos, a exemplo da interação com objetos desenvolvidos em outras linguagens.

No caso da tecnologia *Web Service* pode-se obter as características de interoperabilidade existentes em CORBA aliadas a outras vantagens: onde CORBA usa uma comunicação binária baseada no protocolo IIOP, carregado com stubs, skeletons e ORBs específicos, *Web Service* é leve, baseado em SOAP/HTTP, XML e com isso capaz de atravessar firewalls, e independente de plataforma e linguagem.

Cada tecnologia, portanto, apresenta vantagens e desvantagens para determinados tipos de aplicações, de maneira que o conjunto de características de um ambiente EAD é que irá determinar qual ou quais tecnologias a serem empregadas.

6.3 Limitações do Protótipo

As limitações observadas neste trabalho foram as seguintes:

- A estratégia de ensino é simples, baseando-se na inclusão e leitura de artigos-aula, sem controle, por parte do professor, das atividades de seus alunos;
- Nenhum tipo de comunicação síncrona desenvolvido.

6.4 Trabalhos Futuros

Como trabalhos futuros têm-se os seguintes pontos:

- Agregar ao protótipo uma estrutura de *framework* que estabeleça informações como cursos, professores de cursos, alunos de cursos e aulas de cursos.
- Adicionar um módulo de aulas onde alunos e professores possam se comunicar de forma síncrona, via chat, em horários pré-estabelecidos de aula.
- Modificar a inclusão do vídeo-aula, que no protótipo está sendo armazenado no servidor Web, tendo no banco de dados somente uma referência por meio do nome do vídeo, de modo que ele seja gravado diretamente em banco de dados, evitando assim, possíveis acessos indesejados de fora do ambiente EAD e garantido maior segurança da informação.

Referências Bibliográficas

[AulaNET2003] AulaNet. Disponível em: http://guiaaulanet.eduweb.com.br/

Acesso em: fevereiro de 2003.

[Baron97] BARON, J. **Co-Authorship Via the Web for Distance Learners**. Australian Society for Computers in Learning in Tertiary Education Annual Conference, 1997.

Disponível em: http://www.ascilite.org.au/conferences/perth97/papers/Baron/Baron.html

[Bica1999] BICA, F. "Eletrotutor III – **Uma Abordagem Multiagente para o Ensino a Distância.** Porto Alegre – Universidade Federal do Rio Grande do Sul, 1999."

[Bomfim2002] BONFIM JÚNIOR, Francisco. **"JSP: A tecnologia Java na Internet"**. São Paulo: Érica, 2002.

[Carver99] CARVER, Carol. **Building a Virtual Community for a Tele-Learning Environment**. IEEE Communications Magazine. Março 1999. p 114-118.

[CCEAD2002] Coordenação Central de Educação a Distância. PUC, Rio de Janeiro.

Disponível em: http://www.ccead.puc-rio.br/tutorial/distribuicao.asp

Acesso em: outubro de 2002.

[Clements2002] CLEMENTS, Tom. **Web-Services – Technical Overviews**. 2001. Disponível em: http://dcb.sun.com/practices/webservices/overviews/overview_soap.jsp

Acesso em: outubro de 2002.

[COM99] **Microsoft COM Technologies**.

Disponível em: http://www.microsoft.com/com/

Acesso em: abril de 1999.

[Darby98] DARBY, C. **Developing 3-Tier Database Applications with Java Servlets**, 1998, SYS-COM Interactive –Java Developers Journal.

[Davidson98] DAVIDSON, J.D., **Java Servlet API Specification**, 1998, Sun Microsystems.

[Deitel2001] DEITEL, H.M. ,DEITEL, P.J. **Java:como programar** - trad. Edson Furnankiewicz. - 3.ed. – Porto Alegre: Bookman, 2001.

[Ferreira2002] FERREIRA, A. D.; MARTINS, L. A.; **Comparativo entre as tecnologias JSP (Java Server Pages) e ASP (Active Server Pages)**. UFRS.

Disponível em:

s.br/procpar/disc/cmp167/trabalhos/sem2001-1/T2/alex/

Acesso em: outubro de 2002.

[Fuks2003] Fuks, H., Raposo, A.B., Gerosa, M.A., **"Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas"**, in: Anais da XXI Jornada de Atualização em Informática, Capítulo 3.

Disponível em:

http://139.82.24.161/groupware/publicacoes/HTML/JAI2002.htm

Acesso em: março de 2003.

[Gerosa2002] GEROSA, Marco Aurélio. **Elemento de percepção em ambientes de aprendizagem como forma de facilitar a comunicação, coordenação e cooperação.**

Disponível em:

Http://www.asee.org/international/INTERTECH2002/600.pdf

Acesso em: outubro de 2002.

[Goulart1999] GOULART JÚNIOR, Fernando Silveira. **Arquitetura para Banco de dados heterogêneos:soluções usando objetos distribuídos e Java.** 1999. 94f.. Dissertação (Mestrado em Ciência da Computação) – UFPE, Recife.

[King2002] KING, Carla. **Getting Started on Developing Web Services**. 2001.

Disponível em: http://dcb.sun.com/practices/howtos/developing_webserv.jsp

Acesso em: outubro de 2002.

[JAX-RPC2003] **Java API for XML-based RPC.**

Disponível em: http://java.sun.com/xml/jaxrpc/

Acesso em: fevereiro de 2003.

[JSP2003] **JavaServer Pages 1.2**. Disponível em:

http://www.jcp.org/aboutJava/communityprocess/final/jsr053/

Acesso em:fevereiro de 2003.

[Macêdo2000] MACÊDO, R.C., **Internet Trader: Um Serviço para localização de objetos em um ambiente distribuído** 2000. Dissertação (Mestrado em Ciência da Computação) – UFPE, Recife.

[MageLang98] MageLang Institute. **Fundamentals of Java Servlets: The Java Applet API**, 1998.

[Magic2002] **MAGIC SOFTWARE BRASI.**

Disponível em:

http://www.magic-sw.com.br/Html/tec_ebu_web.html

Acessado em: outubro de 2002.

[Marques2000] MARQUES, Ivan Luiz Ricarde, **Programação Orientada a**

Objetos, www.dca.fee.unicamp.br/courses/PooJava/, 2000.

[Moura2002] MOURA, C. O. F., et al. **Um sistema IBW para Educação Tecnológica à Distância.** CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO CEARÁ, Fortaleza – CE.

Disponível em: http://www.edudistan.com/ponencias/Um%20Sistema%20IBW%20para%20a%20Educa%E7%E3o%20Tecnolgica%20%E0%20Dist%E2ncia.html

Acesso em: outubro de 2002.

[Microsoft2003] **Microsoft .NET.** Disponível em: http://www.microsoft.com/net/

Acesso em: fevereiro de 2003.

[OMG99] **Object Management Group**, http://www.omg.org/ , Home Page visitada em Junho de 1999.

[OMG2003] **Object Management Group – Event Service**. Disponível em: http://www.omg.org/technology/documents/formal/event_service.htm.

Acesso em: fevereiro de 2003.

[Riccioni2000] RICCIONI, P. R. **Introdução à Objetos Distribuídos com Corba. –** 1 ed. – Visual Books, 2000.

[RMI99] **JavaÔ Remote Method Invocation**.

Disponível em:

http://java.sun.com/products/jdk/rmi/index.html

Acesso em: junho de 1999.

[Sanches2002] SANCHES, I. C.; Prado, A. F. **Framework para Ensino a Distância via Web**. UFSCAR.

Disponível em:

http://www.tci.ufal.br/sbie2000/ht-docs/sub-prop/artigos-

aceitos.htm

Acesso em: outubro de 2002.

[SOAP2003] **Simple Object Acess Protocol**.

Disponível em: http://www.w3.org/TR/SOAP/

Acesso em: fevereiro de 2003.

[Sun2002] **Sun Microsystems**. The Java Tutorial. Sun Microsystems.

Disponível em: http://java.sun.com/docs/books/tutorial

Acessado em: outubro de 2002.

[Trinta2000] TRINTA, F. A. M.; FERRAZ, C. A. G.**Co-autoria Distribuída de Cursos na Internet Utilizando CORBA e Java.** XX Congresso da Sociedade Brasileira de Computação – XXVII Seminário Integrado de Software e Hardware, Curitiba-PR,. 2000.

[Virtus2003] Projeto Virtus – Educação e Ciberespaço.

Disponível em: http://www.virtus.ufpe.br/

Acessado em: fevereiro de 2003.