UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

INFRA-ESTRUTURA PARA DESENVOLVIMENTO E IMPLEMENTAÇÃO DE APLICAÇÕES WEB EM JAVA UTILIZANDO MÁQUINA DE ESTADOS

JOÃO THIAGO GERALDES
ROBSON PANIAGO DE MIRANDA

ORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JUNIOR

PROJETO FINAL DE GRADUAÇÃO EM ENGENHARIA DE REDES DE COMUNICAÇÃO

PUBLICAÇÃO: UNB.LABREDES.PFG 07/2002

BRASÍLIA / DF: SETEMBRO/2002

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

INFRA-ESTRUTURA PARA DESENVOLVIMENTO E IMPLEMENTAÇÃO DE APLICAÇÕES WEB EM JAVA UTILIZANDO MÁQUINA DE ESTADOS

JOÃO THIAGO GERALDES
ROBSON PANIAGO DE MIRANDA

PROJETO FINAL DE GRADUAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO.

APROVADA POR:

RAFAEL TIMÓTEO DE SOUSA JUNIOR, Doutor, UnB
(ORIENTADOR)

CLÁUDIA JACY BARENCO ABBAS, Doutora, UnB
(EXAMINADOR INTERNO)

DATA: BRASÍLIA/DF, 06 DE SETEMBRO DE 2002.

FICHA CATALOGRÁFICA

GERALDES, JOÃO THIAGO e MIRANDA, ROBSON PANIAGO DE
Infra-estrutura para Desenvolvimento e Implementação de Aplicações Web em Java Utilizando Máquina de Estados [Distrito Federal] 2002.
xix, 126p., 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2002).

Projeto Final de Graduação – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Máquina de Estados Java 2. Arquitetura de Três Camadas
3. Ensino à Distância

I. ENE/FT/UnB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

GERALDES, JOÃO THIAGO e MIRANDA, ROBSON PANIAGO DE (2002). Infra-estrutura para Desenvolvimento de Aplicações Web em Java Utilizando Máquina de Estados. Projeto Final de Graduação, Publicação UnB.Labredes.PFG 07/ANO, Departamento de Engenharia Elétrica, Universidade de Brasília , Brasília , DF, 126p.

CESSÃO DE DIREITOS

NOME DO AUTOR: João Thiago Geraldes e Robson Paniago de Miranda
TÍTULO DA DISSERTAÇÃO: Infra-estrutura para Desenvolvimento de Aplicações Web em Java Utilizando Máquina de Estados
.
GRAU/ANO: Engenheiro/2002.

João Thiago Geraldes
Departamento de Engenharia Elétrica - Campus Universitário Darcy Ribeiro
Caixa Postal 04591
CEP: 70910-900 – Brasília – DF - Brasil

Robson Paniago de Miranda
Departamento de Engenharia Elétrica - Campus Universitário Darcy Ribeiro
Caixa Postal 04591
CEP: 70910-900 – Brasília – DF - Brasil

Dedico esse trabalho aos meus pais que sempre me ajudaram e incentivaram, aos meus irmãos, meus avós e meus amigos que sempre me ajudaram e apoiaram no decorrer do curso e da montagem dessa dissertação.
João Thiago Geraldes

Dedico esse trabalho aos meus pais e minha esposa Claúdia, pela paciência e colaboração durante o decorrer do curso.
Robson Paniago de Miranda

AGRADECIMENTOS

Ao nosso orientador Prof. Dr. Rafael Timóteo de Sousa Júnior, pelo apoio, incentivo, e amizade essenciais para o desenvolvimento deste trabalho e para nosso desenvolvimento como pesquisadores.

Ao Prof. Ricardo Staciarini Puttini, do Curso de Engenharia de Redes de Comunicação - Departamento de Engenharia Elétrica, co-orientador deste trabalho, que, apenas por razões de regulamentação, não pode ser registrado formalmente como tal.

Aos bolsistas do Laboratório de Engenharia de Redes de Comunicação – LabRedes – da Universidade de Brasília, pelas colaborações em diversos aspectos.

A todos os professores que fazem parte do Departamento de Engenharia Elétrica e do Curso de Engenharia de Redes, pelo conhecimento passado e pelas noites de insônia que nos proporcionaram no decorrer do curso de graduação, que foram essenciais para nossa formação como Engenheiros e profissionais, em especial aos professores que ajudaram a consolidar o curso de Engenharia de Redes de Comunicação na UnB.

Além dos professores acima citados, gostaríamos de agradecer especialmente aos professores Profª. Drª. Cláudia Jacy Barenco Abbas, Profª Flavia M. Guerra de Sousa, Prof. Adson Ferreira da Rocha, Ph.D., Prof. Humberto Abdalla Júnior, Dr. Ing.. Prof. Eduardo Wolski, Prof. Sebastião do Nascimento Neto, Prof. Bruno Krost Fagundes e Prof. Maxwell Diógenes. Que nos auxiliaram em vários aspectos, inclusive extracurriculares.

A nossos colegas, que nos acompanharam em inúmeras noites de estudo e também nos proporcionaram uma convivência bastante agradável na universidade, fazendo esse período "suportável". Sabemos que dessa convivência tiraremos amigos para o resto das nossas vidas particulares e profissionais.

A nossos amigos e familiares, que nos "agüentaram" e "compreenderam" durante esse período de graduação que foi bastante cansativo, árduo e estressante, e por isso muito mais saboroso e gratificante. Seria uma injustiça com muitos citar o nome de alguns amigos e colegas, mesmo porque estes realmente sabem sua importância para nós.

A nossos pais que sempre nos apoiaram, antes e durante nossa vida acadêmica e sabemos que continuarão nos apoiando sempre.

A Deus

O presente trabalho foi realizado com suporte do Projeto Infovia de Brasília, rede metropolitana de alta velocidade patrocinada pelo CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico.

A todos, nossos sinceros agradecimentos.

RESUMO

O objeto desta monografia é expor e discutir conceitos relativos a implementação de aplicações utilizando Arquitetura em Três Camadas, tecnologia Java em plataforma J2EE – Java 2 Enterprise Edition e Máquina de Estados. Também é foco desse trabalho discutir e expor conceitos relativos a Ensino à Distância, como a iniciativa ADL e o modelo de conteúdo SCORM. De posse desses conceitos, este trabalho apresenta uma infra-estrutura para o desenvolvimento de aplicações Java utilizando máquina de estados e propõe uma implantação de um *software* Gestor de Cursos pela Internet utilizando essa infra-estrutura e os modelos e padrões de conteúdo expostos.

ABSTRACT

The subject of this work is expose and discuss implementation concepts related to application development using 3-tier Architecture, J2EE – Java 2 Enterprise Edition platform, Java technology and State Machines. This document also focuses on discuss and expose concepts related to Distance Learning like, the ADL initiative and SCORM content model. With those concepts, this work presents an infrastructure for application development in Java platform using a state machine and proposes an implantation of a learning controller software on Internet using the infrastructure and the content models and standards shown here.

ÍNDICE

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

1. INTRODUÇÃO

1.1. DESENVOLVIMENTO DE APLICAÇÕES PARA A INTERNET

O desenvolvimento de aplicações para o ambiente WEB apresenta características diferenciadas de um sistema tradicional, cuja execução ocorre no ambiente do cliente. Tais características são devido ao protocolo HTTP, que não armazena informações sobre o estado da aplicação, ou seja, cada requisição ao servidor é tratada de forma independente das requisições anteriores.

Algumas aplicações em que o controle da navegação passa a ser parte importante do sistema podem se beneficiar de algumas características do protocolo, que permitem ao desenvolvedor obter informações sobre o estado atual do usuário dentro da aplicação.

Tipicamente, aplicações que provêem serviços devem combinar sistemas de informação da empresa – EIS, *enterprise information system* - existentes com novas funções de negócio que disponibiliza serviços a um amplo número de usuários. Estes serviços necessitam ter:

- **Alta Disponibilidade** – para atender as necessidades atuais de ambiente de negócios.
- **Segurança** – para proteger a privacidade dos usuários e a integridade dos dados da empresa.
- **Confiabilidade e Escalabilidade** - assegurar que transações de negócio são processadas exatamente e prontamente.

Por uma variedade de razões, estes serviços são geralmente arquitetados como aplicações distribuídas consistindo de várias camadas – *tiers* –, incluindo clientes no início – *frontend* –, recursos de dados no fim – *backend* –, e uma ou mais camadas de meio entre eles, onde a maioria do desenvolvimento da aplicação é feita. As camadas de meio implementam os novos serviços que integram EISs existentes com as funções de negócio e dados do novo serviço. As camadas de meio protegem a camada de cliente da complexidade da empresa e tira proveito do rápido amadurecimento das tecnologias de Internet para minimizar administração de usuários e treinamento.

O design multicamada simplifica drasticamente o desenvolvimento, instalação e manutenção de aplicações de empresa. Permite a desenvolvedores focalizar especificamente na programação da lógica de negócio, conta com vários serviços de *backend* para proporcionar a infra-estrutura, e aplicações cliente proporcionam a interação com o usuário. Desenvolvida, a lógica de negócios pode ser implementada em servidores apropriados às necessidades existentes em uma organização. Entretanto, apesar destes benefícios claros, o modelo limita a capacidade dos desenvolvedores de construir aplicações de componentes padronizados, implementar uma única aplicação numa ampla variedade de plataformas, ou escalabilizar prontamente aplicações para atender mudanças nas condições de negócio.

Na Sun Microsystems, vários esforços de desenvolvimento levaram ao que tinha se tornado a tecnologia J2EE. Primeiro, a tecnologia de *servlets* Java mostrou que os desenvolvedores estavam ávidos em criar comportamentos semelhantes a CGI que pudessem rodar em qualquer servidor web que suportasse a plataforma de Java. Segundo, a tecnologia JDBC propiciou um modelo para casar o conceito JAVA de "*Write Once, Run Anywhere*" e os sistemas gerenciadores de bases de dados (SGBDs). Finalmente, o êxito da arquitetura de componentes *Enterprise JavaBeans* demonstrou a utilidade de encapsular conjuntos completos de comportamentos em componentes facilmente configuráveis e prontamente reutilizáveis. A convergência destes três conceitos – comportamentos servidor (*server-side behaviors*) escritos em linguagem Java, conectores para capacitar o acesso a sistemas de empresariais existentes, componentes e módulos de fácil implementação – conduziu ao padrão J2EE.

1.1.1.1. Introdução a uma aplicação web de *cliente-enxuto – Thin-client*

Uma aplicação de *cliente enxuto* é aquela que apresenta documentos em HTML puro e recebe requisições HTTP de volta, não existindo código na aplicação cliente – sem *javascript* ou *applets Java*, apenas HTML puro. O *browser* é tratado como um terminal burro, com duas exceções notáveis.

O cliente pode armazenar *dados de estado*, por exemplo, dados podem ser mantidos no cliente através de um dos três métodos: re-escrita de URL, campos escondidos num formulário HTML e *cookies*.

O cliente também pode mudar seu estado atual inesperadamente devido ao uso do botão **voltar** no *browser* e pelo mecanismo de *bookmark*.

Esta arquitetura não determina o uso de código no cliente e aspectos de notificação de um modelo típico MVC – *Model-View-Controler* - podem ser implementados usando *javascripts* mínimos no cliente. Essa tecnologia é denominada *Pushlets*.

1.1.1.2. Eventos de Entrada - Requisições HTTP

Todas as aplicações web são conduzidas por Eventos, como por exemplo, uma requisição HTTP. O servidor recebe requisições HTTP oriundas do cliente, especificamente, eventos g*et* ou *post*.

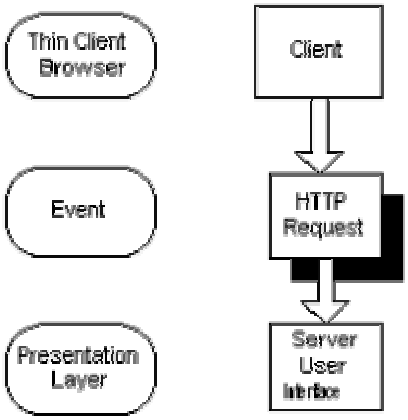

Figura 1 - Requisições HTTP mostradas como um Evento de um Cliente a um Servidor

Como mostrado na Figura 1, o *browser* no cliente envia um Evento para a Camada de Apresentação do Servidor. O Evento é ou um *post* ou um *get* encapsulado em uma requisição HTTP.

1.2. ENSINO À DISTÂNCIA

A utilização da *Internet* para a complementação do ensino de graduação é eficiente, mas sofre principalmente com a falta de uma metodologia bem definida e de padrões e bases teóricas consistentes para essa implementação. Uma grande vantagem da utilização da *Internet* na educação é a complementação do currículo acadêmico.

Para atender às necessidades de complementação e enriquecimento curricular do ensino de graduação foram criados conceitos e padrões que compõem a iniciativa ADL e o modelo SCORM.

Esses conceitos vêm suprir a necessidade da criação der uma base de conteúdo e conhecimento de livre acesso e compartilhada através da Internet.

2. ESTUDO BIBLIOGRÁFICO

2.1. MODELO TRADICIONAL DE APLICAÇÕES

As aplicações tradicionais são desenvolvidas de modo a permitir a "navegação" por suas diversas telas apenas através de ações previamente codificadas. Desta forma, é impossível, por exemplo, a um usuário "atualizar" as informações de um formulário, ou voltar para um formulário anterior, se tais eventos (que são representados por elementos da *interface* com o usuário) não estão presentes.

Tal tipo de desenvolvimento força o usuário a trilhar caminhos perfeitamente definidos, retirando do desenvolvedor a tarefa de codificar testes referentes ao estado atual do usuário.

2.2. APLICAÇÕES WEB

Aplicações para a Internet normalmente são desenvolvidas de modo a serem acessadas via navegador (*browser*). Nestas aplicações, todo ou quase todo o processamento e validação dos dados informados pelo usuário é feito pelo servidor *web* ou um servidor de aplicações.

Nesta tecnologia, é responsabilidade também do servidor *web* gerar dinamicamente ou através de arquivos estáticos uma descrição textual da *interface*, através da codificação HTML ou XML. A transmissão desta definição da *interface* do servidor *web* para o *browser* é feita através do protocolo HTTP.

2.2.1. Protocolo HTTP

O protocolo HTTP (*Hyper Text Transfer Protocol)*, atualmente na sua versão 1.1, é um padrão definido pela IETF *(Internet Engineering Task Force)*, na RFC 2616[1], para "um protocolo de nível de aplicação para sistemas de informação *hipermidia* distribuídos e colaborativos".

Seu desenvolvimento começou em 1990, apenas como um sistema de transmissão de arquivos, com sua versão 0.9. A versão 1.0, introduzida pela RFC 1945[2], já permite a transmissão de vários formatos de arquivos, sendo estes especificados por algumas informações no estilo MIME[3] (*Multimedia Internet Mail Extension*), além de possibilitar a troca de informações que modificam as semânticas da requisição e resposta a um recurso. A versão 1.1 do protocolo HTTP inclui também características para uma melhor operação de

[1] Fielding, R., Irvine, U.C, Gettys, J. et al, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, Junho de 1999

[2] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, Maio 1996.

[3] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, Novembro de 1996.

hierarquias de *proxies* e *caches*, servidores virtuais com o mesmo endereço IP e conexões persistentes.

Conforme especificado na RFC 2616, o HTTP é um protocolo to tipo requisição/resposta, ou seja, o navegador envia uma requisição contendo a URI *(Uniform Resource Identifier*) que será acessada, versão do protocolo, alguns cabeçalhos e modificadores no estilo MIME, e, opcionalmente, um corpo da requisição, usado, por exemplo, para o envio de dados de formulários. Na resposta, o servidor envia uma linha de status, seguida de uma mensagem estilo MIME com informações sobre o conteúdo da resposta e sobre o servidor, e o corpo da resposta.

Há vários cenários de utilização do protocolo HTTP, sendo o mais simples um *user agent* (UA) fazer uma solicitação direta a um servidor, e este responde diretamente para o UA.

Cenários mais complicados envolvem a utilização de *proxies*, *gateways* e túneis como intermediários entre o *user agent* e o servidor.

Os *proxies* são elementos que recebem requisições na forma de URI absoluta, e as repassam para o servidor identificado, podendo ou não, durante o processo, reformatar a requisição antes de repassá-la ao servidor.

Gateways são agentes normalmente em uma camada acima de outro serviço, responsáveis pela tradução das requisições para um protocolo que o outro serviço seja capaz de compreender.

Túneis agem como ponto de repasse entre os dois elementos de uma conexão, passando as requisições sem nenhuma modificação. Um exemplo de túnel é um firewall baseado em *socks5*.

É importante observar que, para cada elemento intermediário, uma nova conexão é estabelecida, e certos parâmetros da conexão HTTP são relevantes apenas ao próximo agente que não seja um túnel, e outros são relevantes apenas aos agentes das pontas da conexão (ou seja, o *user agent* e o servidor).

Outro ponto importante é que qualquer ponto que não seja um túnel (incluindo o servidor e o *user agent*) pode fazer *caches* das respostas, de forma que uma nova solicitação ao mesmo recurso seja processada pelo servidor. No desenvolvimento de uma aplicação é importante ter em mente tais considerações, uma vez que, como grande parte das respostas será gerada sob demanda, estas não devem ser armazenadas em *caches*. Neste capítulo ainda serão vistos cabeçalhos necessários para o controle desta funcionalidade.

O único requisito do HTTP em relação ao protocolo de transporte é que o mesmo seja confiável, ou seja, a entrega dos pacotes na outra ponta da conexão seja garantida, e também entregue na mesma ordem em que foram enviados. Assim, apesar de usualmente as conexões HTTP serem através do protocolo TCP *(Transmisson Control Protocol*), na porta 80, não há nada que impeça sua implementação em outras portas, ou até mesmo em outros protocolos confiáveis.

Algumas partes do protocolo relevantes para o desenvolvimento de uma aplicação estão descritas abaixo. Para a descrição do protocolo, é utilizada a notação BNF estendida, tal como descrita na RFC 2616, seção 2.

2.2.1.1. Parâmetros do Protocolo

2.2.1.1.1. Versão do protocolo

```
HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT
```

Exemplo: HTTP/1.0

A primeira linha que é enviada ao próximo agente (seja ele um *proxy*, um *gateway* ou um servidor) é a versão do protocolo. A versão é utilizada para indicar qual é o formato de passagem de mensagens e as capacidades de entendimento da comunicação.

Esta versão é formada por dois números decimais inteiros separados por um ponto, no formato "*major*.*minor*". O número *major* é utilizado para indicar qual o formato da mensagem, e o *minor* para indicar novas características acrescentadas ao protocolo, sem, no entanto, mudar o algoritmo de interpretação de mensagens.

Durante uma comunicação HTTP (fim-a-fim), a versão do protocolo não é alterada, a não ser que seja necessária uma alteração no formato de passagem de mensagens. Portanto, o acréscimo de novos valores ou cabeçalhos por qualquer um dos agentes intermediários não torna necessária a alteração na versão do protocolo.

Tanto o *major* quanto o *minor* devem ser tratados como inteiros separados, de forma que, por exemplo HTTP/2.45 é uma versão maior que HTTP/2.5, e HTTP/13.2 é uma versão maior que HTTP/2.4.

2.2.1.1.2. Uniform Resource Identifiers

Uniform Resource Idenfiers são os identificadores conhecidos como URL (*Uniform Resource Location)*, URN (*Uniform Resource Name*) ou endereços WWW. Tais identificadores, descritos em detalhes na RFC 2396[4], que, do ponto de vista do HTTP, é uma seqüência de caracteres formatada, possuem a finalidade de identificar um recurso através de um nome, localização ou outra característica qualquer.

No protocolo HTTP, dependendo do contexto, as URLs podem ser representadas na forma absoluta ou relativa a um determinado contexto-base. A forma absoluta é facilmente distinguível da relativa por iniciar por um *esquema* (protocol o), seguido por um ":".

2.2.1.1.3. Data e hora

Os formatos de data e hora no protocolo HTTP são de dois tipos: *full date* e *delta seconds*. O *full date* é utilizado para representar a data / hora completos, e o *delta seconds* é

[4] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax and Semantics", RFC 2396, Agosto de 1998.

empregado ao informar um intervalo de tempo a partir de quando a mensagem é recebida. Um exemplo de utilização do *delta seconds* é para informar ao cliente por quanto tempo um recurso poderá existir em um *cache*.

A descrição do *full date* está definida abaixo:

```
HTTP-date    = rfc1123-date | rfc850-date | asctime-date
rfc1123-date = wkday "," SP date1 SP time SP "GMT"
rfc850-date  = weekday "," SP date2 SP time SP "GMT"
asctime-date = wkday SP date3 SP time SP 4DIGIT
date1        = 2DIGIT SP month SP 4DIGIT
               ; day month year (e.g., 02 Jun 1982)
date2        = 2DIGIT "-" month "-" 2DIGIT
               ; day-month-year (e.g., 02-Jun-82)
date3        = month SP ( 2DIGIT | ( SP 1DIGIT ))
               ; month day (e.g., Jun  2)
time         = 2DIGIT ":" 2DIGIT ":" 2DIGIT
               ; 00:00:00 - 23:59:59
wkday        = "Mon" | "Tue" | "Wed"
             | "Thu" | "Fri" | "Sat" | "Sun"
weekday      = "Monday" | "Tuesday" | "Wednesday"
             | "Thursday" | "Friday" | "Saturday" | "Sunday"
month        = "Jan" | "Feb" | "Mar" | "Apr"
             | "May" | "Jun" | "Jul" | "Aug"
             | "Sep" | "Oct" | "Nov" | "Dec"
```

É importante notar que é obrigatório qualquer agente (*proxy*, *gateway*, *user agent* ou servidor) utilize a data em GMT. No formato *asctime*, que é o resultado padrão da função da biblioteca padrão da linguagem C "*asctime*()", é assumido que tal data está em GMT.

O objeto *delta seconds* está definido abaixo:

```
delta-seconds  = 1*DIGIT
```

2.2.1.1.4. Character Sets

```
charset = token
```

Character Sets (Conjuntos de caracteres) são um método para mapear um ou mais octetos em uma seqüência de caracteres, através de tabelas.

No protocolo HTTP, um *character set* é identificado por um *token* insensível à caixa alta/baixa. Tais *character sets* são definidos pelo *IANA (Internet Authrority for Assigned Numbers)*, e, caso um agente receba um *character set* definido por tal organismo, este deve representar o conjunto de caracteres especificados. Entretanto, é possível utilizar outras codificações não definidas pelo IANA.

Para compatibilidade com o HTTP/1.0, em que alguns clientes utilizam alguma forma de autodetecção, quando não é informado o conjunto de caracteres, os servidores devem sempre incluir um parâmetro de conjunto de caracteres, mesmo quando este é o ISO-8859-1.

2.2.1.1.5. Codificação de conteúdo

```
content-encoding = token
```

Os valores de codificação de conteúdo (content coding) são utilizados para indicar que um determinado tipo de codificação foi aplicado ou pode ser aplicado a um conteúdo. Sua finalidade é permitir que um conteúdo seja compactado ou codificado sem perda de informação, e decodificado apenas no receptor. Estes valores são representados como um token insensível à caixa alta ou baixa.

Os valores permitidos para *content-encoding* são registrados no IANA, como, por exemplo: **gzip** (RFC 1952), **compress** (idêntico ao utilitário *compress* do Unix), **deflate** (RFC 1950) e **identity**.

2.2.1.1.6. Codificação da transferência

```
transfer-coding         = "chunked" | transfer-extension
transfer-extension      = token *( ";" parameter )
parameter               = attribute "=" value
attribute               = token
value                   = token | quoted-string
```

A codificação da transferência é utilizada quando há, ou pode haver, alguma codificação em um corpo da entidade (*entity-body*). Tal codificação é diferente da codificação de conteúdo, em que a codificação de transferência é referente à mensagem, como, por exemplo, para codificar uma mensagem através de um canal compartilhado.

Um tipo específico de codificação de transferência é a codificação em blocos (*chunked*). Esta codificação deve aparecer apenas uma vez em toda a mensagem, e ser a última da lista, é utilizada quando o servidor não possui o tamanho completo da mensagem antes de iniciar a enviar os dados para o destinatário.

A codificação em blocos transfere o corpo de uma mensagem em uma série de blocos, cada um com um indicador de tamanho, seguido de um *trailer* opcional contendo campos *entity-header*. Desta forma, é possível para o destinatário da mensagem saber quando ela acabou.

Cada bloco é definido como especificado abaixo:

```
Chunked-Body   = *chunk
                 last-chunk
                 trailer
                 CRLF
chunk          = chunk-size [ chunk-extension ] CRLF
                 chunk-data CRLF
chunk-size     = 1*HEX
last-chunk     = 1*("0") [ chunk-extension ] CRLF
chunk-extension= *( ";" chunk-ext-name [ "=" chunk-ext-val ] )
chunk-ext-name = token
chunk-ext-val  = token | quoted-string
chunk-data     = chunk-size(OCTET)
trailer        = *(entity-header CRLF)
```

Para indicar o fim da transmissão, é enviando um bloco com tamanho 0 (*last-chunk*), e, em seguida, o *trailer* e uma linha em branco.

2.2.1.1.7. Tipos de mídia

```
media-type     = type "/" subtype *( ";" parameter )
type           = token
subtype        = token
```

Nos campos de cabeçalho *Content-Type* e *Accept* são utilizados os tipos de mídia, de modo a garantir um esquema aberto e extensível para tipificação dos dados e para a negociação do tipo.

Há alguns tipos chamados "*multipart*", utilizados para prover uma ou mais entidades dentro de um único corpo de mensagem. Tais tipos compartilham uma sintaxe comum, definida na RFC 2046[5]

2.2.1.2. Definição do protocolo

As mensagens no protocolo HTTP são divididas em duas:

Request (requisição): mensagem enviada pelo cliente ao servidor; e

Response (resposta): mensagem enviada pelo servidor ao cliente.

As duas mensagens compartilham um formato comum, descrito abaixo através da BNF:

```
generic-message = start-line
                  *(message-header CRLF)
                  CRLF
                  [ message-body ]
start-line      = Request-Line | Status-Line
```

Os cabeçalhos (*message-header)* utilizados nas mensagens utilizam o formato definido na seção 3.1 da RFC 822, sendo compostos por um nome, um dois-pontos (":") e o valor do cabeçalho.

Uma característica relevante para o desenvolvimento de aplicações sobre o protocolo HTTP é a falta do conceito de "estado", ou seja, o protocolo não mantém informação de estado do usuário. Assim, supondo que o usuário esteja visualizando uma página com um formulário para, por exemplo, apagar um dado em um banco de dados, não há, a princípio, uma forma para saber se o usuário passou antes por uma página de autenticação.

Entretanto, o protocolo HTTP, em sua versão 1.0, já introduziu uma forma de contornar este tipo de problema, através do uso de *cookies*. Os *cookies* são uma forma de armazenar

[5] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, Novembro de 1996.

alguma informação no navegador do usuário, e são normalmente associados a uma URL de base.

2.3. DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO A PLATAFORMA JAVA™

2.3.1. XML

2.3.1.1. O que é XML

XML é uma forma padrão de indústria, independente de sistema, de representar dados. Como HTML (*HyperText Markup Language*), o XML confina dados em *tags*, mas existem diferenças significativas entre as duas linguagens de marcação. Primeiro, *tags* XML se relacionam ao significado do texto confinado, ao passo que *tags* HTML especificam como exibir o texto confinado.

O exemplo XML seguinte mostra uma lista de preços com nome e preço de dois cafés.

```
<PriceList>
  <Café>
    <Nome>Java de Mocha</Nome>
    <Preço>11.95</preço>
  </Café>
  <Café>
    <Nome>Sumatra</Nome>
    <Preço>12.50</Preço>
  </Café>
</PriceList>
```

As *tags* `<Café>` e `</Café>` dizem a um *parser*[6] que a informação entre eles é sobre um café. As duas outras *tags* dentro de `<Café>` especificam que a informação confinada equivale ao nome do café e seu preço. Devido ao fato de *tags* XML indicarem o conteúdo e a estrutura dos dados que eles englobam, torna-se possível realizar tarefas como arquivar e procurar.

Uma segunda diferença significativa entre XML e HTML é que o XML é extensível. Com ele, é possível escrever tags personalizadas para descrever o conteúdo em um particular de documento. Com HTML, se está limitado a usar apenas tags que tenham sido predefinidas na sua especificação. Outro aspecto da extensibilidade do XML é que é possível criar um arquivo, chamado *schema*, para descrever a estrutura de um tipo particular de documento XML. Por exemplo, é possível escrever um *schema* para uma lista de preços que especifica *tags* que podem ser usadas e onde elas podem ocorrer. Qualquer documento XML que segue as restrições estabelecidas num *schema* é dito estar em conformidade àquele *schema*.

[6] "(informática) programa de análise, que divide o texto recebido em pequenas partes e faz o processamentos delas;" – Dicionário Babylon (http://www.babylon.com)

Provavelmente o *schema* de linguagem mais utilizado ainda é a *Document Type Definition* (Definição de Tipo de Documento – DTD) porque é parte integral da especificação XML 1.0. Um *schema* escrito nesta linguagem freqüentemente referido como DTD. O DTD abaixo define as *tags* usadas no documento XML acima. Ele especifica quatro *tags* (elementos) e quais tags podem ocorrer (ou são requeridas) em outras *tags*. O DTD também define a estrutura hierárquica de um documento de XML, incluindo a ordem em que devem ocorrer.

```
<! ELEMENT PriceList  (Café) +>
<! ELEMENT Café  (Nome, Preço) >
<! ELEMENT Nome (#PCDATA) >
<! ELEMENT Preço (#PCDATA) >
```

A primeira linha no exemplo fornece o elemento raiz, `PriceList`, que significa que todas as outras *tags* no documento virão entre `<PriceList> e </PriceList>`. A primeira linha também diz que o elemento de `PriceList` deve conter um ou mais elementos `Café` (indicado pelo sinal de mais). A segunda linha especifica que cada elemento `Café` deve conter tanto um elemento `Nome` quanto um elemento `Preço`, nesta ordem. A terceira e quarta linhas especificam que os dados entre as `tags <Nome> e </Nome>` e `<Preço> e </Preço>` são dados que devem ser analisados. O nome e o preço de cada café são o texto real que compõe a lista de preços.

Outra linguagem popular de schema é a *XML Schema,* que está sendo desenvolvida pelo W3C (World Wide Web Consortium) consortium. *XML Schema* é uma linguagem significativamente mais poderosa que DTD, e com sua passagem a uma recomendação W3C, em vigor desde maio de 2001, seu uso e implementações aumentaram. A comunidade de desenvolvedores usando a plataforma Java reconheceu isto, e o grupo de peritos para a API Java™ para Processamento de XML ("JAXP") tem trabalhado em adicionar suporte para *XML Schema* à especificação JAXP 1.2. Esta versão do Java™ Enterprise Edition inclui suporte a *XML Schema.*

2.3.1.2. O Que Torna o XML Portável?

Um *schema* dá portabilidade para dados XML. O DTD de `PriceList` , discutido previamente, é um exemplo simples de um *schema*. Se um documento `PriceList` no formato XML for enviado para uma aplicação e possuir o DTD correspondente, esta pode processar o documento de acordo com as regras especificadas no DTD. Por exemplo, dado o DTD de `PriceList`, um *parser* saberá a estrutura e o tipo de conteúdo para qualquer documento XML baseado neste DTD. Se o *parser* é de validação, ele saberá que o documento não é válido se contiver um elemento não incluído no DTD, tal como o elemento `<Chá>`, ou se os elementos não estiverem na ordem prescrita, como por exemplo o elemento `Preço` precedendo o elemento `Nome`.

Outras características também contribuem para a popularidade do XML como um método para troca de dados. É escrito num formato de texto, que é legível tanto por humanos e quanto por *software* de editor de texto. As aplicações podem analisar e processar documentos XML, e um humano também pode lê-los em caso de erro no processamento. Outra característica é que um documento XML não inclui instruções de formatação, podendo

ser exibido de várias formas. Manter os dados separados de instruções de formatação significa que os mesmos podem ser publicados diferentes meios.

O XML capacita a portabilidade de documento, mas não pode fazer o trabalho isoladamente; isto é, sistemas que usam XML devem concordar com certas condições. Por exemplo, além de concordar em usar XML para comunicar, duas aplicações devem concordar no conjunto de elementos usarão e o que esses elementos querem dizer. Para eles usarem serviços *web*, eles também devem acordar em quais métodos de serviços *web* que eles usarão, o que esses métodos fazem, e a ordem em que eles são invocados quando mais de um método é necessitado.

As empresas têm várias tecnologias disponíveis ajudar satisfazer estes requisitos. Podem usar DTDs e *XML Schemas* para descrever os termos válidos e documentos XML para se comunicarem entre si. Os registros proporcionam um meio para descrever serviços *web* e seus métodos. Para conceitos de nível mais alto, empresas podem usar acordos de sócio e mapas e coreografias de *workflow*.

As APIs Java para XML permitem reescrever aplicações *web* inteiramente em linguagem Java. Elas se encaixam em duas largas categorias: as que lidam diretamente com o processamento dos documentos XML e as que lidam com procedimentos.

Talvez a característica mais importante das APIs Java para XML é o fato de todas elas suportarem padrões de mercado, garantindo interoperabilidade. Muitos grupos de padronização de interoperabilidade de rede, como W3C e OASIS (*Organization for the Advancement of Structured Information Standards*), vêm definindo métodos padrão de realizar tarefas e negócios que seguem esses padrões podem fazer suas aplicações e dados funcionarem em conjunto.

Outra característica das APIs Java para XML é que elas permitem uma alta flexibilidade. Usuários possuem flexibilidade em como utilizam as APIs, Implementadores podem definir requisitos estritos de compatibilidade de componentes para garantir que todas as implementações disponibilizem a funcionalidade padrão, mas também dão a desenvolvedores grande liberdade para prover implementações desenhadas para usuários específicos.

2.3.2. JAVA 2™ Enterprise Edition – J2EE

A plataforma Java 2 define uma arquitetura padrão que é disponibilizada com os seguintes elementos:

- **Plataforma J2EE** - Plataforma padrão para hospedagem de aplicações J2EE, especificada como um conjunto de APIs – *Application Program Interface* – e políticas requeridas.
- **Conjunto de Teste de Compatibilidade J2EE** - Verifica se um produto em feito em plataforma J2EE é compatível com o padrão.
- **Implementação de Referência J2EE** – Demonstração das capacidades do J2EE e para proporcionar uma definição operacional da plataforma J2EE.
- **Diretrizes de Design para J2EE, Sun Blueprints™** – Descrevem um padrão de modelo de programação para desenvolver aplicações de *cliente-enxuto* multicamada.

2.3.2.1. Modelo de Aplicação J2EE

O J2EE é projetado para suportar aplicações que implementam serviços de empresas para clientes, empregados, fornecedores, sócios e outros que fazem exigências ou contribuições para esta. Tais aplicações são inerentemente complexas, potencialmente acessando dados de uma variedade de fontes e distribuindo aplicações a uma variedade de clientes.

Para um melhor controle e administração destas aplicações, as funções de negócio para suportar estes vários usuários são conduzidas na camada de meio, que representa um ambiente controlado pelo departamento de tecnologia de informação da empresa. A camada de meio tipicamente é executada em um servidor dedicado e tem acesso a todos os serviços da empresa.

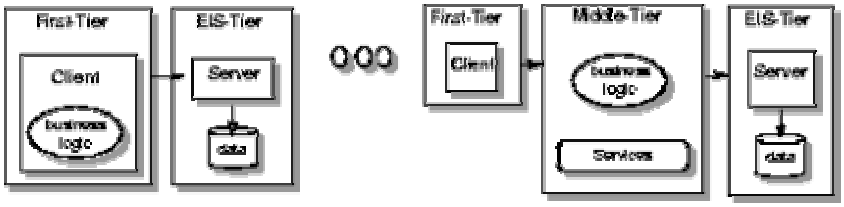

Figura 2 - Modelo de Aplicação de Duas Camadas vs. Multicamadas

Originalmente, o modelo de aplicação cliente-servidor, duas camadas, prometeu melhorar a escalabilidade e a funcionalidade. Infelizmente, a complexidade de entregar serviços EIS diretamente a cada usuário e os problemas administrativos causados pela instalação e manutenção da lógica de negócios em cada estação de usuário provaram ser limitações significativas.

Estas limitações são evitadas pela implementação de serviços de empresa como aplicações multicamadas. As aplicações multicamadas proporcionam uma acessibilidade aumentada que está sendo exigida por todos elementos de uma empresa. Esta mudança tem conduzido investimentos significativos no desenvolvimento de softwares de camada de meio.

O desenvolvimento de serviços multicamada foi complicado pela necessidade desenvolver tanto a função de negócio do serviço quanto um código mais complexo de infra-estrutura é necessário para acessar bases de dados e outros recursos de sistema.

O modelo de aplicação J2EE define uma arquitetura para implementação de serviços como aplicações de multicamada que disponibilizam escalabilidade, acessibilidade, e gerenciabilidade necessários.

Este modelo divide o trabalho necessário para implementar um serviço de multicamada em duas partes: a lógica de negócios e lógica de apresentação.

Além disso, ele proporciona os benefícios de portabilidade e escalabilidade para aplicações de multicamada. Este modelo padrão reduz o custo de treinamento com desenvolvedores e proporciona a empresa uma escolha ampla de servidores e ferramentas de desenvolvimento. Ele começa com a linguagem de programação Java e a máquina virtual Java – *JAVA Virtual Machine*. A portabilidade, segurança, e produtividade de desenvolvimento

que proporcionam formam a base do modelo de aplicação. O modelo de aplicação também inclui o modelo de componentes *JavaBeans*.

Os componentes *JavaBeans* tornam o código Java fácil de ser transformado em componentes para funções comuns, e tais componentes podem ser adequados e combinados visualmente através de ferramentas de desenvolvimento de *JavaBeans*.

Enquanto outros modelos de aplicação requerem medidas de segurança específicas para a plataforma em cada aplicação, o ambiente de segurança da plataforma J2EE possibilita que restrições de segurança sejam definidas durante o desenvolvimento, protegendo aplicações da complexidade de segurança implementada, o que a torna portável a uma grande variedade de implementações de segurança.

A plataforma J2EE define regras de controle de acesso padronizadas a serem definidas pelo desenvolvedor da aplicação e interpretadas quando a aplicação é inserida na plataforma. O mesmo programa trabalha em uma variedade de ambientes diferentes de segurança sem mudança no código fonte.

2.3.2.2. A Camada de Meio

O benefício mais importante do modelo de aplicação J2EE está nas camadas de meio das aplicações de multicamada. Funções de negócios de camada de meio são implementadas como componentes de *Enterprise JavaBean*s. Estes componentes (*beans*) permitem que desenvolvedores de serviço se concentrem na lógica de negócios, deixando o servidor EJB manipular as complexidades do serviço.

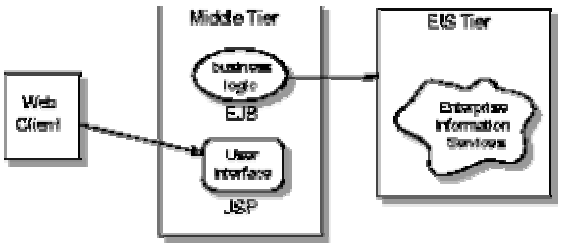

Figura 3 - Componentes de EJB implementando a Lógica de Negócios na camada de Meio

A tecnologia *JavaServer Pages* (JSP) e *servlets* apresentam funções da camada de meio à camada cliente como serviços Internet de simples acesso. Esta tecnologia torna simples para desenvolvedores de *interface* de usuário apresentar páginas geradas dinamicamente a qualquer um com um *browser*. Os *Servlets* provêem a liberdade de implementar apresentações dinâmicas completamente em Java a desenvolvedores de aplicações mais sofisticadas.

2.3.2.3. A Camada de Cliente

A plataforma J2EE suporta vários tipos de clientes. Muitos serviços serão projetados para suportar navegadores web. Estes serviços interagem com seus clientes via páginas e formulários HTML gerados dinamicamente.

Serviços mais sofisticados interagirão diretamente com seus clientes de primeira camada trocando dados de negócio. Aqui, JSPs e Servlets são usados para formatar dados de negócio de maneira que simplifique o trabalho para clientes. Estes clientes podem ser tanto *applets* Java rodando em um *browser* quanto programas baseados em tecnologia Java.

É importante notar que segurança é uma parte chave de todos serviços multicamada.Esta é manipulada quase inteiramente pela plataforma e seus administradores. Na maioria dos casos, nem o serviço ou os clientes requerem lógica de segurança feita pelo desenvolvedor.

2.3.2.4. Clientes Baseados em Páginas HTML

Um serviço pode ser apresentado diretamente a um web browser como páginas HTML geradas dinamicamente. A tecnologia *JavaServer Pages* é um meio simples de compor dinamicamente estas páginas usando um paradigma familiar de *scripting* que combina HTML e código baseado na tecnologia Java. Em alguns casos, um serviço pode necessitar de um código complexo, o que pode ser manipulado colocando o código em um componente *JavaBeans* e o chamando em um JSP. Um serviço também pode ser programado diretamente no código Java usando um servlet.

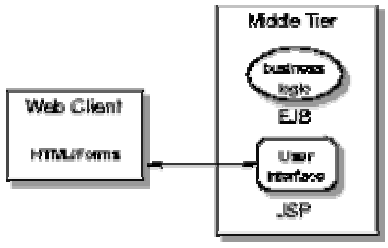

Figura 4 - Apresentando Serviços Diretamente a um Browser

2.3.2.5. Clientes Baseados em Conteúdo HTTP

É freqüentemente útil prover funcionalidade diretamente ao cliente que auxilia um usuário a organizar e reagir com a informação do serviço. Neste caso, o serviço troca o conteúdo bruto (*raw content*) com o cliente em vez de páginas HTML. Este conteúdo está tipicamente na forma de documentos XML que são trocados entre cliente e serviço usando o protocolo HTTP.

Tipicamente este conteúdo XML é manipulado na primeira camada por componentes *JavaBeans* que são providos pelo serviço num *applet* que é automaticamente carregado em um *browser*. Para evitar problemas causados por versões antigas ou não-padronizadas do *Java runtime environment* em um *browser*, o modelo de aplicação J2EE automaticamente baixa e

instala o Java Plug-in. Este conteúdo também pode ser manipulado por um programa Java atuando como um cliente. Este modelo flexível de cliente prove o desenvolvedor com uma ampla gama de escolhas para apresentar uma *interface* de usuário da aplicação distribuída na Internet.

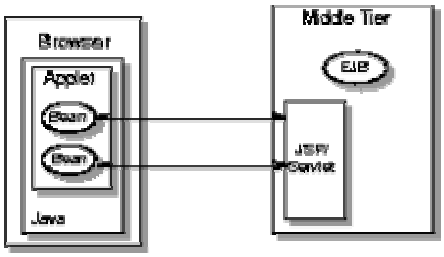

Figura 5 – Provendo componentes JavaBeans em um Browser

2.3.2.6. Clientes de Intranet

Tanto os serviços baseados em página HTML quanto em conteúdo HTTP podem ser eficientemente usados na intranet da empresa assim como na Internet.

Além disso, a intranet proporciona a infra-estrutura extra que permite programas Java acessar diretamente EJBs dentro do domínio de intranet.

2.3.2.7. Outros Tipos de Cliente

Os serviços J2EE apresentados via HTTP normal, HTML e XML são facilmente acessíveis a todos clientes, incluindo clientes da Microsoft tal como Microsoft Visual Basic 2000.

Uma meta da tecnologia de Enterprise Java Beans é definir o padrão CORBA como um mecanismo básico de interoperabilidade. Isto fará qualquer serviço J2EE disponível a qualquer cliente de CORBA, assegurar mais completa integração entre a plataforma J2EE e sistemas existentes de informação de empresa. Enquanto a maioria dos elementos deste padrão estão completas, há alguns itens que estão com desenvolvimento em progresso. Depois que o trabalho final estiver completo, J2EE terá esta requisição de interoperabilidade.

2.3.2.8. Os Sistemas de Informação de Empresa

Um serviço de camada de meio da função de negócios devem acessar e atualizar a informação na camada EIS.

O padrão seguinte de serviços de APIs Java deve prover acesso básico a estes sistemas:

- ✓ **JDBC** – a API padrão para acessar dados relacionais em Java.
- ✓ **Java Naming e Directory *Interface* (JNDI)** – a API padrão para acessar informação nos servidores de nome e diretório da empresa.

- ✓ **Java Message Service (JMS)** – a API padrão para enviar e receber mensagens via sistemas de mensagens de empresa (*enterprise messaging systems*) como IBM MQ Series e TIBCO Rendezvous.
- ✓ **JavaMail** - a API padrão para enviar correio eletrônico.
- ✓ **JavaIDL** - a API padrão para chamar serviços CORBA.

2.3.2.9. Declarações J2EE

Uma meta importante do modelo de aplicação J2EE é minimizar a programação da aplicação. Um dos meios para se obter isso é delegar a realização tarefas comuns à plataforma J2EE. Estas tarefas incluem impor regras de segurança da aplicação, realizar semântica de transações, e ligar seus componentes aos recursos e a outros componentes que sejam requeridos.

O J2EE proporciona um meio simples especificar estes comportamentos. Estas declarações são separadas do código de componentes e armazenadas num *deployment descriptor* que é parte do pacote de aplicação. Estas declarações XML capacitam *application deployers* a modificar o comportamento de uma aplicação sem modificar quaisquer dos seus componentes.

2.3.2.10. Plataforma J2EE

A plataforma J2EE é o ambiente padrão para executar aplicações J2EE. Ela é composta dos seguintes elementos:

- ✓ **J2EE deployment specification** – um padrão que define um meio comum de empacotamento de aplicações para implementação em qualquer plataforma J2EE compatível.
- ✓ **Padrões de tecnologia Java para a plataforma J2EE (*Java technology standards for the J2EE plataform)*** – um conjunto de padrões que todos produtos da plataforma J2EE devem suportar.
- ✓ **Padrões da IETF para a plataforma J2EE (*IETF standards for the J2EE plataform)*** – um conjunto de padrões definidos pela Força de Tarefa de Engenharia da Internet (*Internet Engineering Task Force*) que todos produtos da plataforma J2EE devem suportar.
- ✓ **Padrões CORBA para a plataforma J2EE (*CORBA standards for the J2EE plataform*)** – um conjunto de padrões de CORBA sobre o qual a plataforma J2EE baseia sua interoperabilidade da camada de meio.

2.3.2.11. Montagem e Instalação de Aplicações J2EE

Uma aplicação J2EE é empacotada em uma ou mais unidades padrão para instalação em qualquer sistema de plataforma J2EE compatível. Cada unidade contém um ou mais componentes funcionais (*enterprise beans*, páginas JSP, *servlets*, *applets*, etc.), um descritor padrão de instalação (*deployment descriptor)* que descreve seu conteúdo, e as declarações J2EE que foram especificadas pelo desenvolvedor e montador da aplicação.

Uma vez que uma unidade J2EE foi produzida, está pronta para ser instalado em uma plataforma J2EE, como mostrado abaixo.

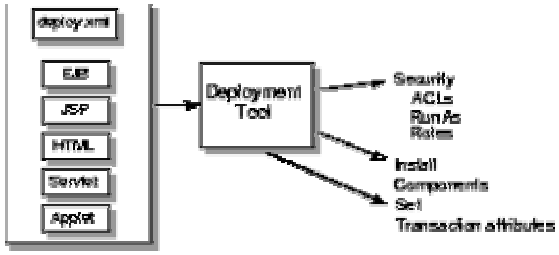

Figura 6 – Instalando (deploying) Aplicações J2EE

A instalação envolve tipicamente o uso de uma ferramenta de instalação (*deployment tool)* da plataforma para especificar informações locais, tais como o nome da base de dados e uma lista de usuários locais que podem acessá-la. Uma vez instalada na plataforma local, a aplicação está pronta para rodar.

2.3.2.12. Padrões de tecnologia Java para a plataforma J2EE

O elemento primário da plataforma J2EE é a lista de padrões de tecnologia Java que todos produtos J2EE devem suportar.

Desde que a plataforma J2EE seja focalizada no desenvolvimento fim-a-fim de soluções de empresa, ela vai além de simplesmente requisitar que cada API Java seja suportada, ele requer que cada API seja plenamente integrada com a plataforma. Isso assegura que a plataforma disponibilize um ambiente fim-a-fim consistente para a instalação de aplicações J2EE.

2.3.2.13. Padrões da IETF para a plataforma J2EE

A emergência da Internet teve um impacto significativo no modo como as aplicações de empresas são desenvolvidas. Esta revolução está baseada no conjunto de padrões estipulados pela Força de Tarefa de Engenharia da Internet (IETF), incluindo HTML, HTTP, e agora XML, o padrão de Internet para comunicação de dados de estruturados.

A linguagem de programação Java, tem crescendo com os padrões da IETF, tornando-se o modo preferido de escrita de aplicações. Em sua iteração atual, a plataforma J2EE suporta clientes HTML e HTTP, e pode suportar clientes XML, e.*deployment descriptors* J2EE utilizam XML para prover informações de aplicação de uma forma independente da plataforma.

2.3.2.14. Padrões da Tecnologia CORBA para a Plataforma J2EE

O OMG (*Object Management Group*) em conjunção com a Sun produziu a especificação de RMI-IIOP. Este padrão define como o protocolo CORBA IIOP é utilizado pelo instrumento de Invocação Remota de Método Java (RMI – *Remote Method Invocation*).

A especificação EJB usa o mapeamento de aplicação para RMI-IIOP como padrão para chamar EJBs. A plataforma J2EE suporta plenamente o uso de RMI-IIOP.

2.3.3. Servidores de aplicação baseados em Servlets/JSP

Os servidores de aplicação baseados em Servlets/JSP são uma solução padronizada pela Sun para o desenvolvimento de páginas com conteúdo dinâmico para a Web.

Este tipo de solução permite ao desenvolvedor construir páginas dinamicamente, obtendo os dados necessários a partir, por exemplo, de um banco de dados.

A API de *servlets* (peças de código que rodam em um servidor de aplicação) foi desenvolvida para ser independente do protocolo de aplicação utilizada. Entretanto, a utilização mais comum de *servlets* é através do protocolo HTTP.

É função dos *servlets* processar as requisições feitas pelo usuário (*interface javax.servlet.ServletRequest*) e gerar as respostas (*interface javax.servlet.ServletResponse*).

A utilização de *Servlets* com o protocolo HTTP está definida no pacote *javax.servlet.http*. Neste pacote, as interfaces relevantes são:

HttpServletRequest é a *interface* utilizada para passar a um *servlet* as informações enviadas ao servidor de aplicação pelo navegador.

Como armazenar informações sobre o acompanhamento do usuário (*user tracking)* em um sistema através de *cookies* não pode ser considerado seguro, já que um usuário mal-intencionado pode alterar, criar ou excluí-los, ou até mesmo forjar uma requisição que inclua os *cookies* necessários, muitas vezes os servidores de aplicação *web* são capazes de acompanhar o estado do usuário através de uma tabela, no servidor, cuja chave é um *cookie* gerado automaticamente pelo servidor e enviado ao navegador do usuário. Com uma chave de tamanho razoável, é pouco provável que haja colisão entre as chaves de usuários diferentes.

Cada linha nesta tabela armazenada no servidor é chamada **sessão**. Como o *cookie* identificando a sessão é enviado em todas as requisições ao servidor, é sempre possível, de acordo com a requisição, saber a que sessão se refere, ou se, em caso da ausência do *cookie*, saber que é uma nova sessão.

Normalmente, o que é armazenado em uma sessão é uma tabela de pares nome/valor. A aplicação utiliza estes dados armazenados no servidor para determinar seu estado atual.

Apesar de já ser possível determinar, a partir dos dados da sessão, qual é o estado em que o usuário deve estar na aplicação, ainda há um outro problema a solucionar: o usuário pode alterar o estado no cliente, através dos botões avançar, voltar e recarregar, ou através de

bookmarks, e a aplicação não vai estar ciente de que a alteração no estado ocorre devido a isso.

2.3.4. Enterprise Java Beans e computação distribuída

Enterprise JavaBeans™ é um componente de arquitetura servidor para plataforma J2EE™. EJB™ possibilita desenvolvimento rápido e simplificado de aplicações Java distribuídas, transacionais, seguras e portáveis.

A arquitetura de *Enterprise JavaBeans* é uma arquitetura de componentes para o desenvolvimento e instalação de aplicações de negócios distribuídas baseadas em componentes. As aplicações feitas nessa arquitetura são escaláveis, transacionais, e multi-usuário. São portáveis, de forma que o código escrito uma única vez pode ser instalado em qualquer plataforma de servidor suporte *Enterprise JavaBeans*.

Desde sua introdução há mais de quatro anos, a tecnologia *Enterprise JavaBeans™* (EJB™) obteve movimento sem precedentes entre provedores e desenvolvedores de empresa. Isso se deu porque o modelo de componentes EJB simplifica o desenvolvimento de aplicações *middleware*[7] por proporcionar suporte automático para serviços tais como transações, segurança, conectividade de base de dados, entre outros.

A especificação EJB 2.1 melhora a arquitetura EJB com suporte para serviços web. Os desenvolvedores empresariais podem implementar e instalar aplicações *web* com o mesmo nível de simplicidade que em outras aplicações servidor.

A especificação EJB 2.1 está sendo conduzido como JSR-153[8] sob o programa JCP℠ (*Java™ Community Program*).

A especificação 2.1 implementa novas características, incluindo suporte a componentes JAXM, aprimoramentos no suporte a EJB QL para agregar outras operações, suporte a ligação de destinos de mensagem, suporte a serviços web usados com EJB e um serviço de *timer* gerenciado por *container*.

[7] **Middleware -** Software que conecta duas aplicações separadas. Isto permite a usuários requisitar dados da base utilizando formulários exibidos no Web browser, e permite que o servidor Web retorne páginas dinâmicas baseadas em requisições e perfis do usuário. O termo middleware é usado para descrever produtos que servem como junção entre duas aplicações. Conecta os dois lados de uma aplicação e troca dados entree les. Em uma arquitetura de três camadas ocupa a camada de meio (ou de negócio) – **Fonte: Internet Glossary**

[8] Em http://jcp.org/jsr/detail/153.jsp

2.4. FERRAMENTAS PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB

2.4.1. Arquitetura em Três Camadas (*3-tier architeture*)

Para separar o código responsável pelas regras de negócio da parte de persistência e apresentação, foi desenvolvido o modelo de arquitetura em 3 camadas:

- Camada de persistência: Responsável por armazenar e recuperar a partir de algum meio persistente as informações solicitadas pelas camadas superiores
- Camada de negócio: Responsável pelo tratamento dos dados e implementação das regras do negócio
- Camada de apresentação: *Interface* com o usuário

Como o desenvolvimento dividido nestas três camadas, é possível facilitar a manutenção da aplicação, uma vez que uma alteração em uma camada não necessariamente implica em uma alteração na camada superior.

Dentre as diversas implementações de servidores de aplicação possíveis, foi escolhido o *JBoss* versão 3.0, que, além de fornecer os serviços de servidor de aplicação, através da tecnologia EJB, também possui um servidor *web* integrado, reduzindo, assim, as chamadas remotas de procedimento, e convertendo todas as chamadas a métodos de objetos em chamadas locais (ou seja, dentro do mesmo processo).

2.4.2. *Statecharts* – Diagramas de Estado

Os *statecharts* (diagramas de estado) definidos pela UML (*Unified Modeling Language*) podem ser utilizados para modelar sistemas reativos, isto é, aqueles que interagem com agentes externos, reagindo a seus estímulos (eventos).

Sua função é descrever "as seqüências possíveis de estados e ações nas quais o elemento [sendo modelado] irá fazer durante o seu tempo de duração dentro do sistema (...)"[9]. Assim, pode ser utilizado para descrever o comportamento dinâmico de uma classe, ou a interação com um usuário.

[9] http://www.npdi.dcc.ufmg.br/membros/rabelo/visao/tbd/introducao.htm

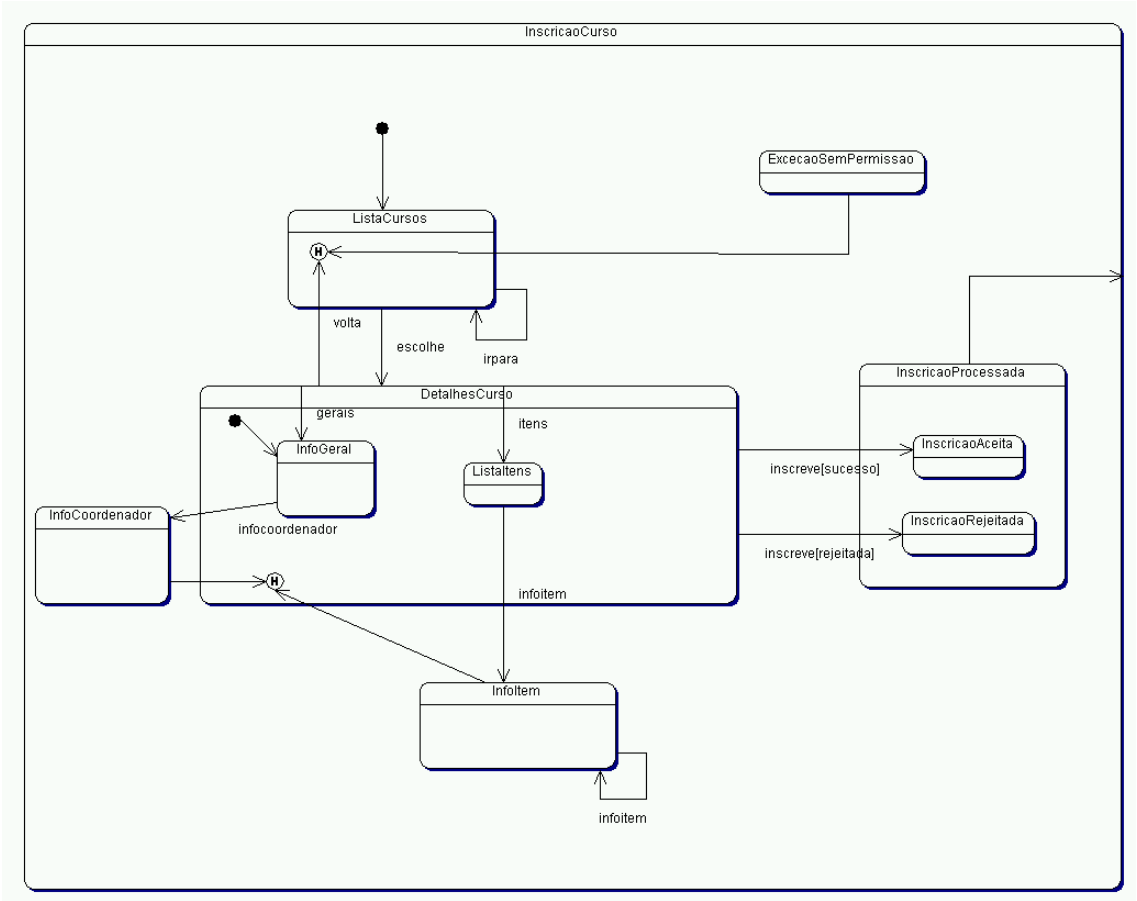

Figura 7 - Exemplo de Statechart

2.4.2.1. Notação

Um mapa de estados é representado por um grafo, em que os arcos dirigidos representam as transições. É importante notar também que os estados podem conter sub-estados, que herdam as transições.

Cada estado, na notação UML, é descrito por:

- Um nome único para cada estado;
- Ações de entrada e saída; e
- Transições internas.

As ações de entrada e saída são executadas sempre que houver uma mudança de estado. Transições internas são aquelas em que não alteração no estado atual. Neste último caso, as ações de entrada e saída não são executadas.

Cada transição possui:

- Um evento associado;
- Uma ação apropriada; e
- Condições de guarda.

O evento é recebido pelo estado atual. A transição correspondente a este evento é efetuada apenas se a condição de guarda associada ao evento puder ser avaliada como verdadeira.

Além disso, existem dois *pseudo-estados*, que não possuem ações nem transições internas associadas, chamado Estado de Início (*Start State*) e Estado de Final (*End State)*. Tais estados existem apenas para possibilitar uma transição para o primeiro estado em que o elemento sendo modelado entrará, e para indicar que o elemento finalizou seu ciclo de vida.

2.4.3. A Arquitetura MVC – Model-View-Controller

Na camada de apresentação, uma nova divisão de funções foi realizada, através da utilização do padrão de projeto MVC.

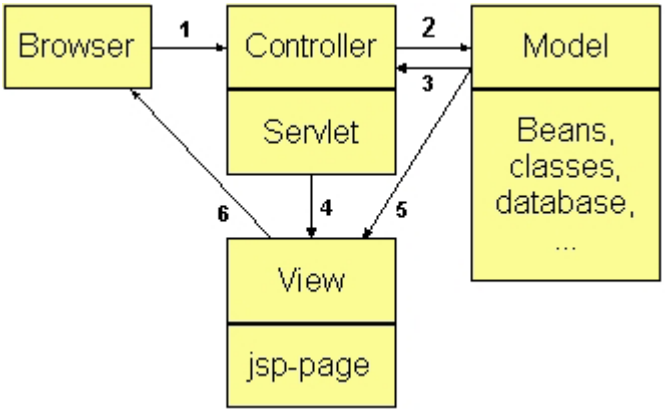

Figura 8 - Arquitetura MVC

Neste modelo, a apresentação é dividida em:

- *Model*: Responsável pela comunicação com a camada de negócio ou persistência
- *View*: Responsável pela criação e apresentação da *interface* com o usuário, obtendo os dados a partir do modelo.
- *Controller*: Responsável por receber os eventos do usuário e efetuar as alterações necessárias do modelo.

Ainda na camada de apresentação, diversas alternativas para a implementação deste modelo foram consideradas, como *Jakarta Struts*, *Oracle MVC Framework*, a implementação direta através de *servlets* e o *JStateMachine*.

Devido à possibilidade de modelagem de uma aplicação através de uma máquina de estados, tanto o *Oracle MVC Framework* e o *JStateMachine* foram considerados. Entretanto, com o desenvolvimento do projeto, algumas adaptações no *framework* foram sendo necessárias, e, por não possuir o código fonte aberto, o *Oracle MVC Framework* foi descartado, sendo a implementação feita no *JStateMachine*.

A camada de apresentação de um servidor web de aplicação raramente é projetada, sendo esta, geralmente, somente codificada. Isto pode funcionar perfeitamente para sites de médio e pequeno porte, mas pode ser problemático em sites maiores.

O aumento da demanda em aplicações web está tornando os sites muito mais complexos, tornando-os tão sofisticados quanto muitas aplicações Cliente/Servidor. Isto vem modificando a idéia do tradicional *cliente enxuto* de uma aplicação web. Aplicações modernas têm de executar transações complexas com o cliente, o que requer que o servidor tenha uma idéia correta do estado de cliente e um bom conhecimento dos laços de transação.

Aplicações web são de certa forma diferentes de aplicações Cliente/Servidor tradicionais, pois o cliente pode ser manipulado utilizando o *browser* e mudar a um estado desconhecido. O servidor deve ser capaz de lidar confortavelmente com Eventos inesperados do cliente que alteram o estado da aplicação como o botão de voltar ou um *bookmark* do *browser*.

2.4.3.1. A Camada de Apresentação Determinística

Em aplicações web, o código do servidor – *server-side code* - que é responsável pela *interface* com o usuário normalmente é referenciado como a Camada de Apresentação, com o intuito de distinguí-lo do *browser* cliente que é denominado de *Interface* de Usuário. Em outras palavras, a camada de apresentação constrói a *interface* de usuário, mas reside no Servidor. O design é freqüentemente mínimo e o código é simplesmente escrito sem muita preocupação com o mesmo. Isto é freqüentemente exacerbado pela natureza rápida e incremental de protótipos de *interface*, que não serve a um design rigoroso e mantém a documentação do design em sincronia com o código desenvolvido. Em muitas aplicações web, um protótipo estático de HTML é construído e então um código servidor é escrito para processar submissões de formulários e construir as páginas html. O protótipo de HTML então é anexado e nenhum ou pouco esforço adicional de design mostra-se necessário.

Isto serve adequadamente para pequenas e médias aplicações, entretanto, é possível tornar-se problemático para aplicações de grande escala, por exemplo, algo em torno de 200 telas e algo entre 6 a 10 programadores para a camada de apresentação. Quando um projeto atinge esta proporção é necessário ter uma documentação de design adequada para o código da camada de apresentação e é necessário ter algum controle determinístico sobre como tudo funciona, em que estado se encontra e o que será feito após. Ter um controle rígido ajuda prevenir falhas e identificá-las quando ocorrerem. Este controle melhora a confiança dos desenvolvedores e o gerenciamento. Um maior gasto de tempo no início do projeto, posteriormente se compensa com um tempo reduzido de depuração e melhores resultados em testes.

Conhecer a informação correta sobre o estado de uma *interface* de usuário torna-se muito útil quando se tem uma aplicação de processamento de transação que deve definir exatamente o âmbito de uma "Unidade de Trabalho" para uma transação lógica. Igualmente para assegurar a integridade e a consistência dos dados estas transações devem ser convenientemente isoladas. Muitas aplicações web até agora evitaram esta complexidade implementando funcionalidade do tipo *stateless* onde transações acontecem simplesmente em requisições http. Entretanto, exigências crescentes por aplicações mais sofisticadas para web

solicitam que aplicações baseadas em *browser* cada vez mais sejam requisitadas a disponibilizar funcionalidades cliente/servidor. Isto pode requerer o que é conhecido como transações conversacionais entre cliente e servidor web, por exemplo transações que distribuem requisições http. Para controlar adequadamente tais transações e proporcionar outra funcionalidade amplamente esperada de uma GUI – *Graphic User Interface* (*Interface* Gráfica de Usuário) - tais como Cancelar (*Cancel*) e Tentar Novamente (*Retry*), é necessário ter uma camada de apresentação determinística.

Uma Camada de Apresentação Determinística pode ser alcançada primeiro modelando a *Interface* de Usuário utilizando *Statecharts* UML – Diagramas de Estado UML - e posteriormente implementando o design diagrama de estados se utilizando de uma *engine* MVC.

2.4.3.2. Avaliação de Eventos

Quando o servidor de aplicação *web* recebe um Evento, deve avaliá-lo e decidir como responder ao cliente. Este processo de avaliação pode envolver interação com a lógica de negócio da aplicação. Sistemas de negócios devem ser avaliados e compreendidos antes de qualquer implementação de sistema de computacional ser viabilizada. Esta análise de negócio, ou análise de domínio do problema como também é conhecida, pode acontecer independentemente do design de tela. A análise de domínio do problema pode ser implementada coma camada de negócio (ou modelo). Esta camada de negócio normalmente deve ser persistente, persistência essa que geralmente é atingida com uma base de dados.

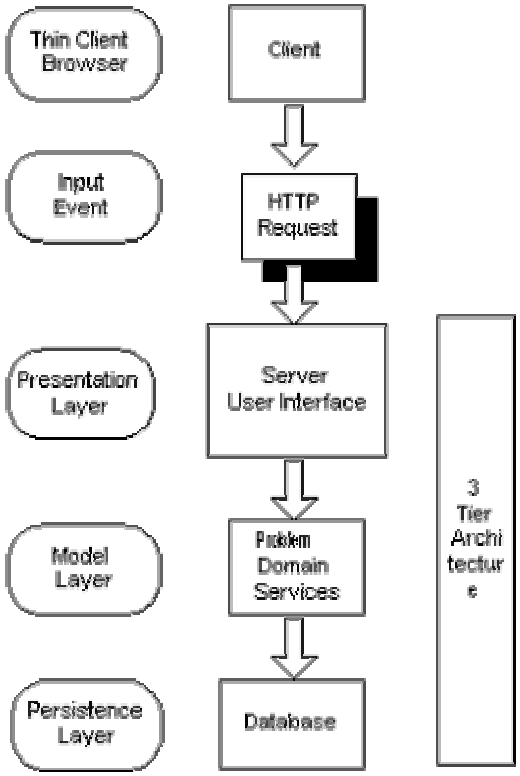

Figura 9 - A camada de apresentação se comunicando com camada de Negócio para acessar storage persistente.

Esta separação da camada de apresentação, camada de negócio e camada de persistência freqüentemente é referida na literatura como modelo de três camadas – *3-tier model*. Isto o separa dos tradicionais formulários de base de dados de cliente/servidor, sendo mais flexíveis. Fisicamente todas as três camadas podem ser implementadas em servidores diferentes, aumentando a escalabilidade e uma capacidade de processar mais requisições de cliente simultaneamente.

2.4.3.3. Estados de Saída - Páginas HTML

A camada de apresentação do servidor de aplicação é também responsável por enviar um *stream* de saída HTML ao cliente. Cada vez que uma nova página (ou frame) HTML é enviada ao cliente, pode-se considerar que o cliente mudou de estado. O *browser* exibe uma nova página.

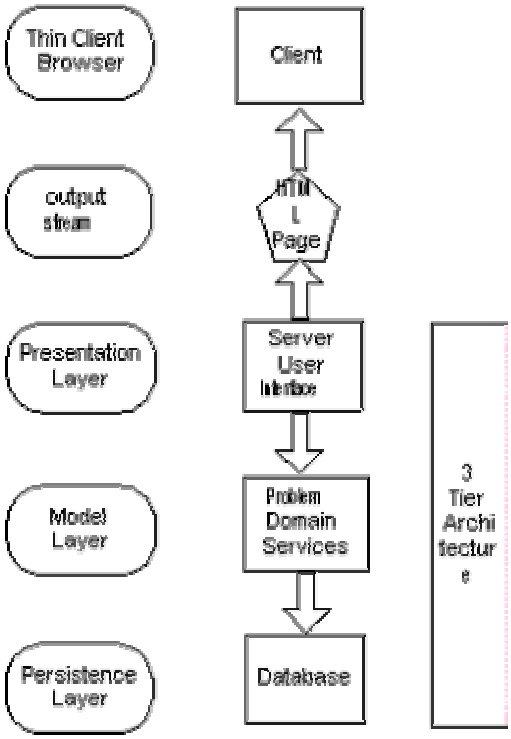

Figura 10 – Fluxo de saída HTML de um Servidor para um Cliente

2.4.3.4. Cliente para Interação da Camada de Apresentação

Se o foco se colocar puramente no cliente para interação da camada de apresentação do servidor de aplicação, podemos ver que o cliente envia a entrada do servidor de aplicação como uma série de requisições HTTP, estas podem ser considerados eventos *get* ou *post* puramente no nível físico. O servidor de aplicação então processa a requisição HTTP e envia de volta um *stream* de dados HTML de saída para o cliente montar a página ou frame requerido.

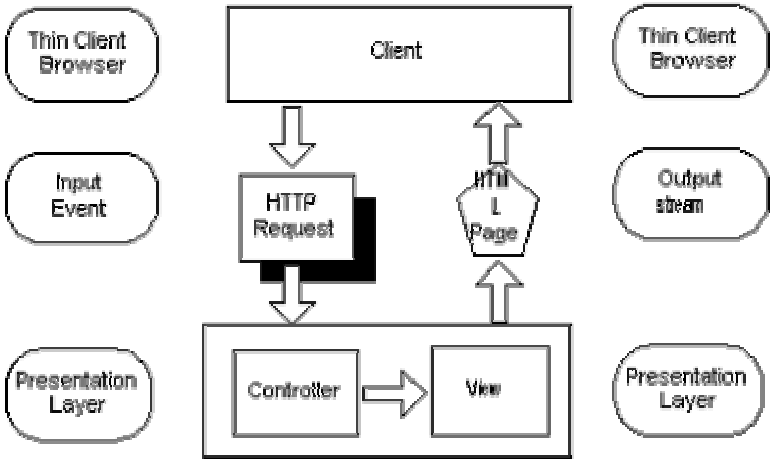

Figura 11 – Interação do Cliente com a Camada de Apresentação do Servidor de Aplicações

2.4.3.5. Modelagem do Cliente para Interação com a Camada de Apresentação

Se o cliente for observado da perspectiva do servidor de aplicação, ele está exibindo uma série de páginas HTML. Eventos podem retornar ao servidor de cada página HTML, que são fisicamente do tipo *get* ou *post*, entretanto, uma lógica semântica pode ser adicionada aos dados que acompanham o evento. Por exemplo, se um "*Submit*" for pressionado, um parâmetro adicional "Evento = *Submit*" pode ser incluído nos dados da requisição HTTP. Isto pode ser usado para transformar o evento físico *post* em um evento *submit* lógico. Este mecanismo amplamente utilizado em aplicações *web* para identificar mensagens específicas do cliente para o servidor, por exemplo, um botão "Adicionar" teria um parâmetro "Evento = Adiciona" da mesma forma que um botão "Remover" teria um “Evento = Remove”.

Dessa forma o cliente se mostra ao servidor exibindo Página 1, então recebe um evento e um processamento no servidor envia Página 2, o cliente exibe Página 2. Outro evento chega e um novo processamento é executado, o cliente então exibe Página 3 e assim por diante.

Dessa forma pode-se montar facilmente o modelo do que acontece no cliente usar Diagramas de Estado. A Página 1 corresponde ao Estado 1, a Página 2 ao Estado 2, a Página 3 ao Estado 3 e assim por diante. O que acontece entre esses estados é denominado Evento.

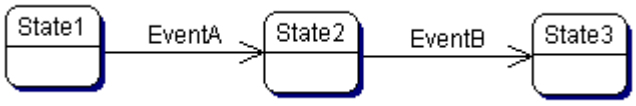

Figura 12 - Diagrama Estados básico para três páginas HTML seqüenciais

O que realmente acontece é geralmente mais complexo. Um cliente em um Estado atual pode produzir um Evento, que pode ser acompanhado por alguns parâmetros. No recebimento deste algumas condições são avaliadas, que podem requerer os parâmetros do Evento. A combinação do Estado atual, do Evento e do resultado da avaliação das condições determinará o próximo Estado e possivelmente algumas ações que devem ser tomadas assim que a

transição for realizada. O diagrama acima mostra um caso básico, onde não existem parâmetros ou condições a avaliar. Pode-se declarar claramente o que é possível com esta definição simples:

[Estado atual + Evento + Condições] *determina* [Ações + próximo Estado]

Há uma arquitetura bem estabelecida para *Interfaces* de Usuário que facilitam sistemas conduzidos por Eventos tal como este, conhecidos como *Model-View-Controller* ou MVC.

Existem dois requisitos para um controlador numa arquitetura MVC. O primeiro, e mais significativo, é que o controlador deve determinar o *controle de fluxo* do sistema. Ele deve processar um Evento de entrada da *interface* de usuário e determinar o próximo estado da mesma. O segundo requisito, que vem causando algumas confusões na bibliografia sobre MVC, que é sinalizar ao Domínio do Problema conduzindo quaisquer ações requisitadas, o que pode modificar o estado do sistema de negócio subjacente. O estado do sistema de negócio subjacente e o estado da *interface* de usuário são coisas diferentes, embora freqüentemente relacionadas.

A aplicação web de *cliente enxuto* está muito próxima de aplicações de terminais burros. Os requerimentos para uma arquitetura MVC num sistema de terminal burro são muito mais simples que algumas implementações modernas de MVC em sistemas GUI. De uma forma simplificada, o funcionamento do MVC se dá da seguinte forma: os Controladores processam a entrada, eles recebem Eventos, consideram o Estado atual do cliente e determinam o próximo Estado do sistema. Baseado no novo Estado, o Controlador invoca uma exibição apropriada para a saída requisitada.

As responsabilidades do Controlador se focam no controle de fluxo e estado da *interface* de usuário em detrimento ao estado subjacente do sistema de negócio.

Como a construção das exibições é feita sob demanda, deve solicitar dados do modelo cada vez que este é instanciado.

As exibições e os controladores juntos podem ser considerados a Camada de Apresentação em uma aplicação *web*. Entretanto, é fácil separar exibições de controladores em uma Aplicação *Web* Servidor.

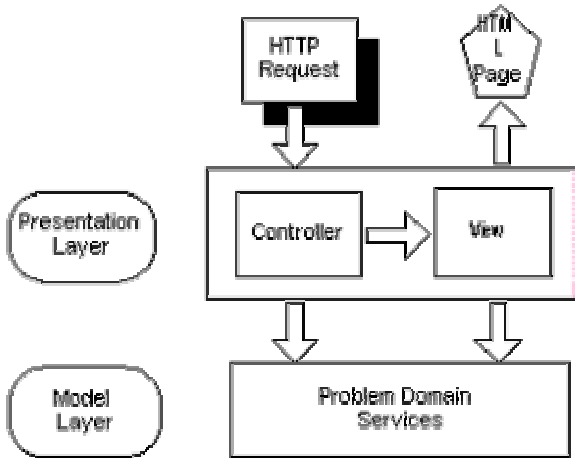

Figura 13 - Cliente para Interação com a Camada de Apresentação mostrando a separação MVC no Servidor

Ao todo, isto é uma implementação simples e enxuta que é facilmente mapeada por uma modelo de implementação de nível da *interface* de usuário utilizando um Diagrama de Estados.

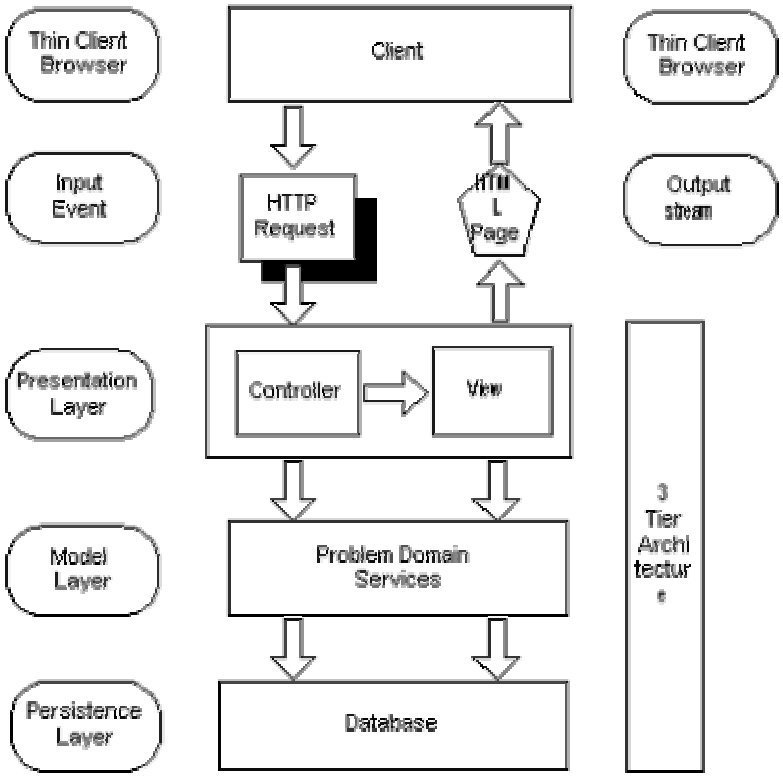

Figura 14 - A Arquitetura MVC completa para um Cliente Enxuto

Finalmente, pode-se considerar este modelo Genérico de MVC como um diagrama de classes UML, mostrando a requisição HTTP empacotada como uma classe. Na sua chegada ao Servidor de Aplicação, invoca um Controlador que avalia algumas condições. Ele decide qual exibição deve ser construída e invoca a classe apropriada para tal através do método *buildPage()*. A exibição, por sua vez, pode requerer alguma informação do modelo como enquanto constrói o HTML para exposição. Por fim, ele manda a saída ao cliente através do método *print*().

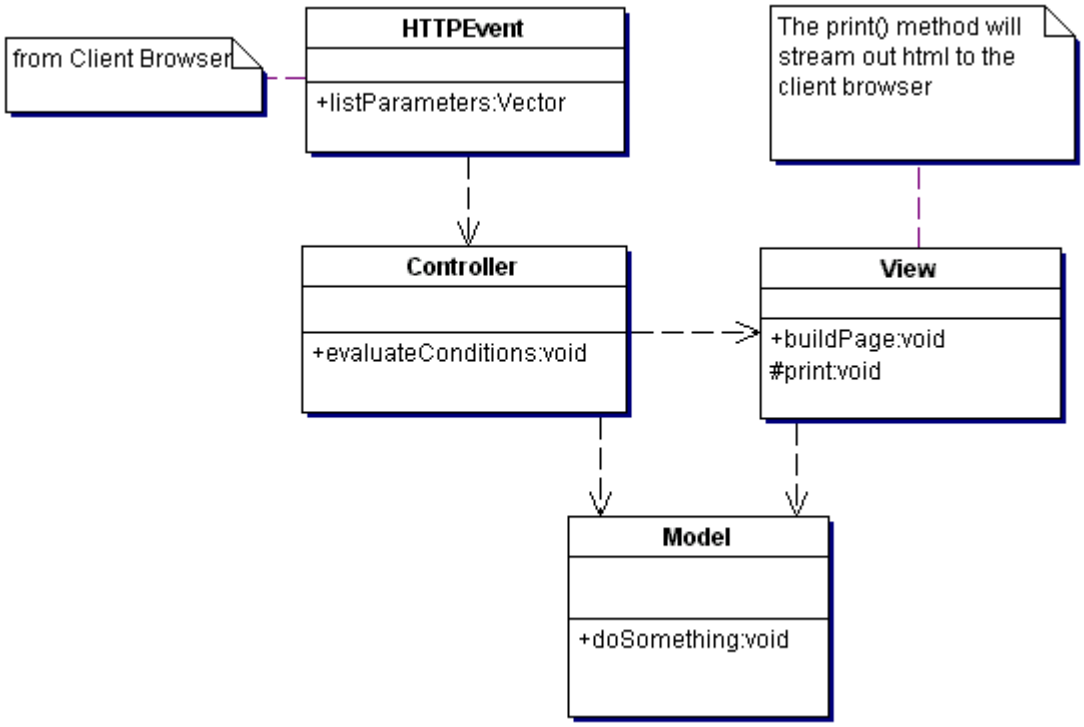

Figura 15 - Diagrama de Classes para uma implementação MVC

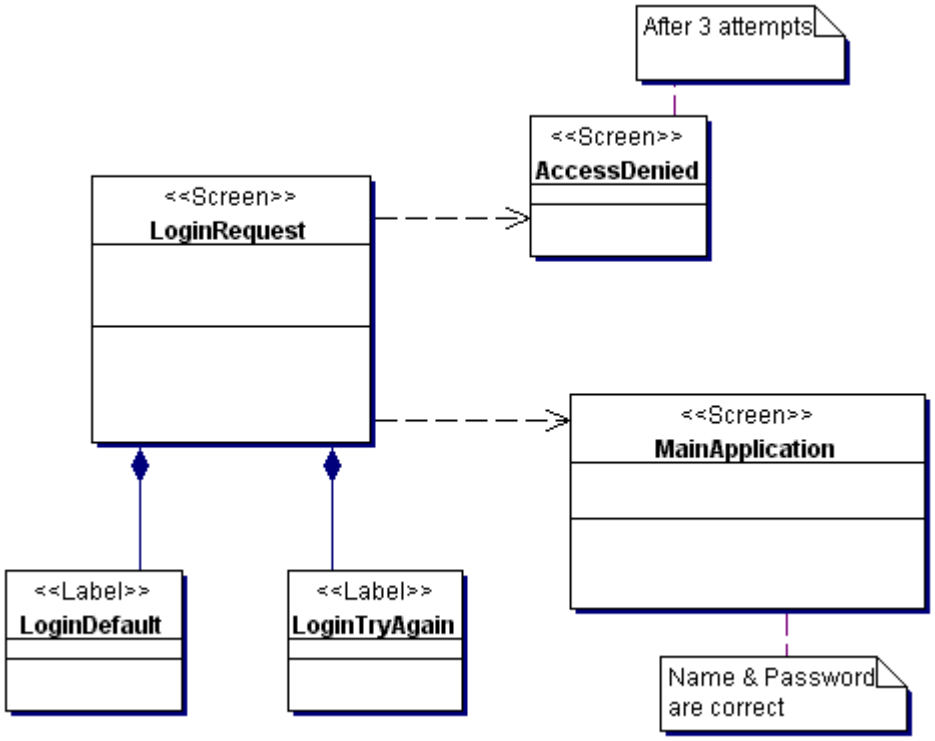

Figura 16 - O Modelo de navegação para uma Requisição de Login e duas telas subseqüentes

Na Figura 16, pode se ver um exemplo de uma requisição de *login* que pode funcionar da seguinte forma: primeiro uma tela de *login* com um de dois rótulos alternativos dizendo "Entra com seu Nome e Senha" ou "Seu Nome ou Senha estão incorretos, Por favor tente

novamente". Isto conduz a uma de duas telas, ou o *Login* é bem sucedido e a aplicação principal é acessada ou mal sucedido e o usuário é rejeitado com a tela de Acesso Negado.

Um modelo de implementação pode ser construído usando um diagrama de estados conforme mostrado na Figura 17.

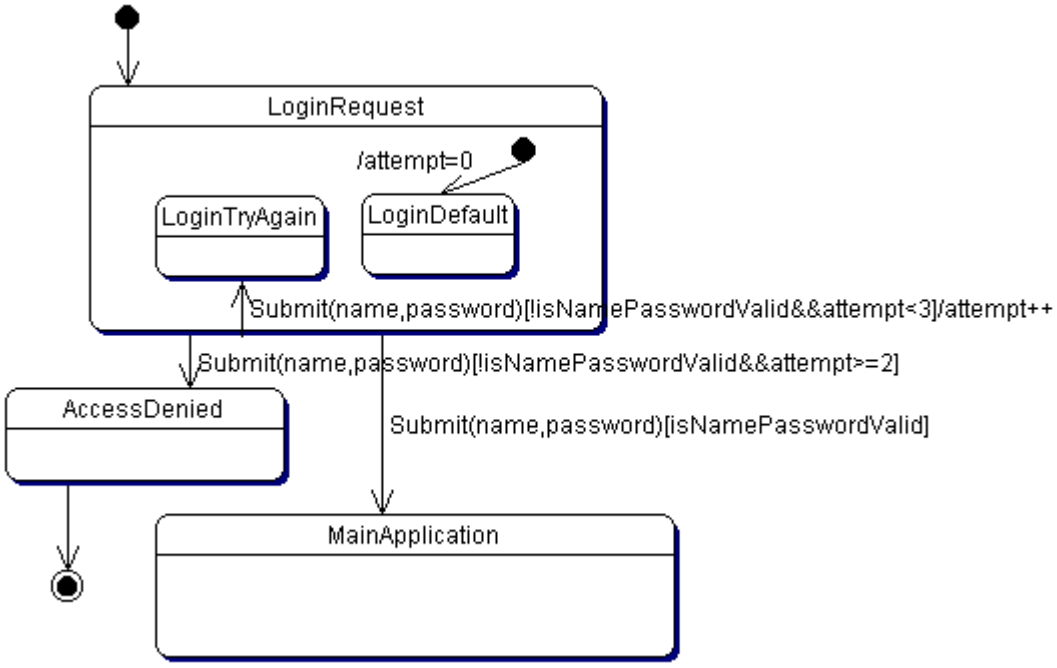

Figura 17 - Modelo para implementação de um Diagrama de Estados para uma requisição de Login e duas telas subseqüentes

Um mapeamento físico deste Modelo de Implementação pode ser providenciado para o controlador de exibição da camada de apresentação, como mostrado na Figura 18.

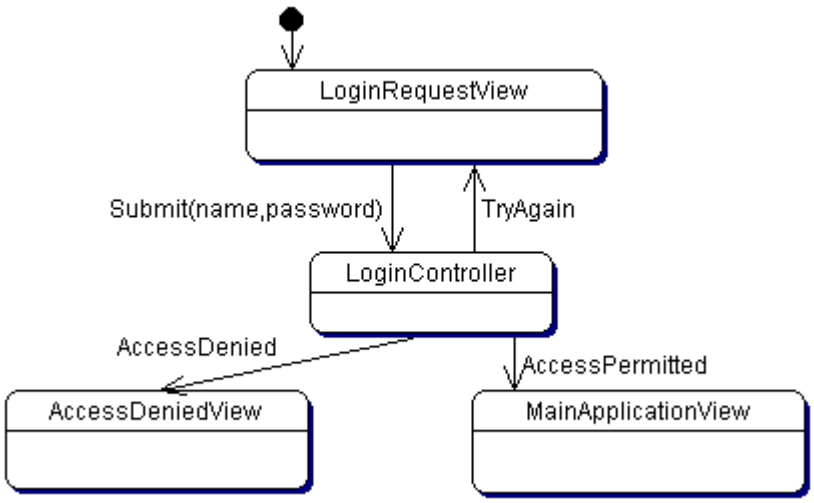

Figura 18 – Mapeamento do Modelo de Implementação.

Com isso é possível implementar este modelo em classes. Cada controlador (ou manipulador de eventos) torna-se uma classe e cada estado que representa uma página ou frame HTML também. O Evento está inserido dentro da classe *HTTPRequest,* como mostrado na Figura 19.

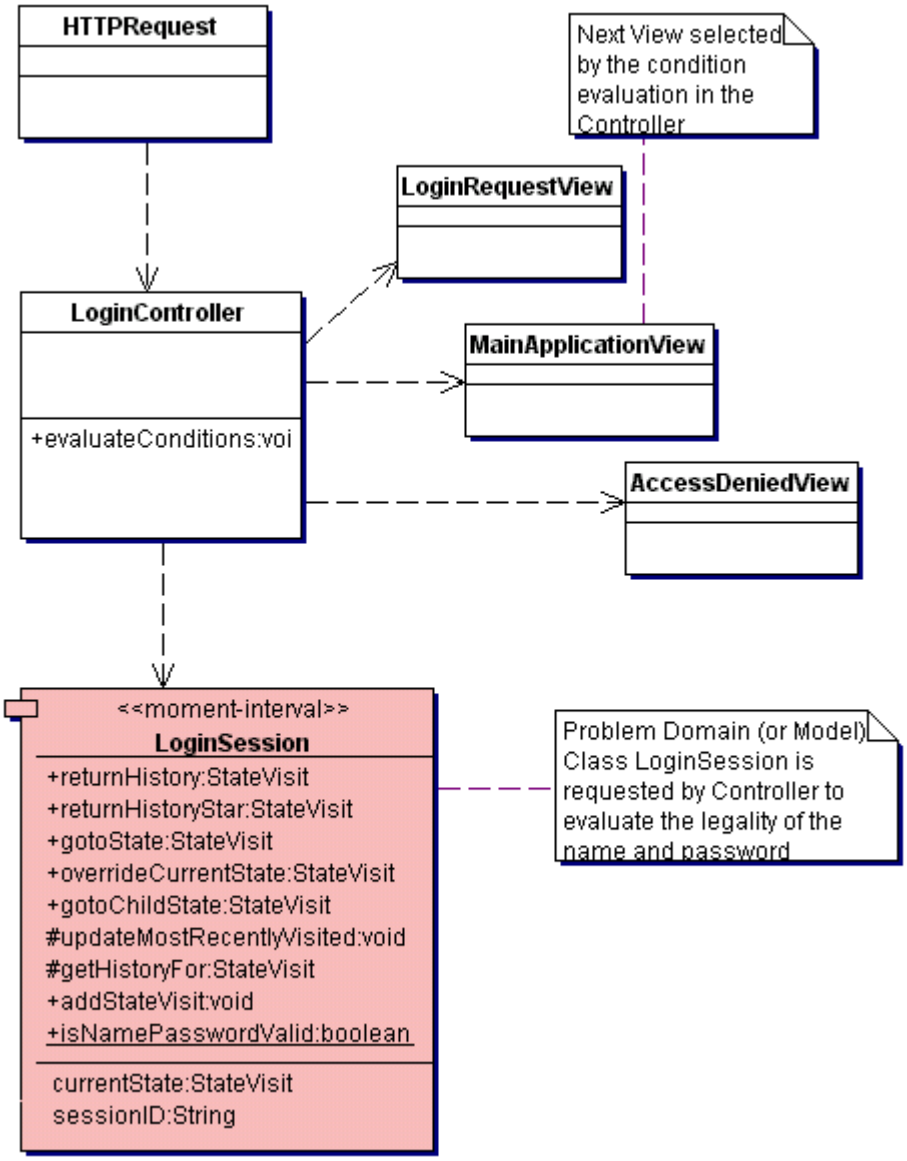

Figura 19 - Diagrama de Classes para o Exemplo de Login

A classe LoginSession encapsula o conceito do período de tempo em que um único cliente visita o web site, o período entre primeiro *Login* e o *Logout*. O *LoginSession* deve também expirar o tempo da sessão no caso do usuário simplesmente ir para outro lugar qualquer na web. O *LoginSession* contem o atributo chave *SessionID* que é usado para identificar um cliente em cada requisição HTTP, este é emitido na primeira requisição HTTP. Uma requisição HTTP de entrada sem um ID de sessão é considerada como oriunda de um novo usuário, ao qual emitido um ID de sessão e solicitado um *Login*.

O *LoginSession* também armazena referências ao *SystemUser* – Usuário do Sistema - a classe posterior ao estabelecimento de um ID e Senha bem sucedidos para um usuário de

sistema. Também armazena uma coleção de Estados visitados para um cliente particular durante esta Sessão. Um *LoginSession* é freqüentemente denominado um Contexto de Usuário ou Cliente.

2.4.3.6. Escalabilidade (Crescimento)

O exemplo mostrado acima é bem simples, com uma única submissão de formulário levando a duas telas subseqüentes. Ela envolve três estados, dois estados subseqüentes e um único controlador. Entretanto, esta arquitetura pode ser expandida para aplicações significativamente maiores e interfaces de usuário mais complexas.

Para implementar tal arquitetura e design num servidor de aplicação web de larga escala, uma *engine* para conduzir a Máquina de Estados e a implementação MVC na camada de apresentação é necessária. Essa *engine* realiza a interpretação de Eventos de entrada oriundos de requisições HTTP e os traduz em Eventos lógicos como "Submeter Senha". Este invocaria a classe de controlador apropriada, baseado no estado atual de cliente, executando as avaliações de condição, conduzindo o cliente ao próximo Estado. A classe correta para este próximo estado seria chamada e iria então gerar a saída HTML apropriada para o cliente. O *browser* cliente terá executado a transição ao novo Estado.

Esta arquitetura pode ser usada para desenvolver uma *interface* de usuário determinística de larga escala para uma aplicação web que se mantém gerenciável e compreensível mesmo quando um grande número de pessoas trabalham no desenvolvimento do código. Este projeto é mais facilmente rastreado para gerar relatórios que podem levar a um controle mais restrito no desenvolvimento da camada de apresentação, melhorando estimativas de desenvolvimento e ajudando a manter os prazos em dia.

2.4.4. JStateMachine

O JStateMachine é uma biblioteca desenvolvida em Java que permite o desenvolvimento de uma aplicação através de sua modelagem em diagramas de estado UML. Outra facilidade provida por esta biblioteca é a criação automática de interfaces com o usuário para protótipo, permitindo ao desenvolvedor iniciar o projeto pelos controladores e modelo.

É composto de cinco partes básicas:

- Os conectores, que são responsáveis por ligar o JStateMachine a diferentes tecnologias de apresentação;
- O gerenciador de sessão do usuário, responsável por armazenar o histórico de acessos, de modo a permitir transições do tipo *HISTORY* e *HISTORY_STAR*;
- A representação abstrata do diagrama de estados;
- As interfaces *IController* (para os controladores) e *IView* (para as visões); e

- A classe *RequestManager* e outras relacionadas, que formam o núcleo da biblioteca, responsáveis por gerenciar todas as requisições, processando os eventos e construindo as visões correspondentes.

Para os conectores, existem duas *interfaces*: *IRequest* e *IResponse*, que são representações abstratas de uma *interface* como usuário orientada a eventos. Os eventos chegam através das requisições, juntamente com outros parâmetros vindos através da UI, e as atualizações da visão são representadas pela resposta. É importante observar que os gerenciadores de sessão possuem uma dependência com o conector utilizado, pois pode ocorrer do gerenciador de sessões utilizar alguma facilidade do sistema de *interface* com o usuário utilizado.

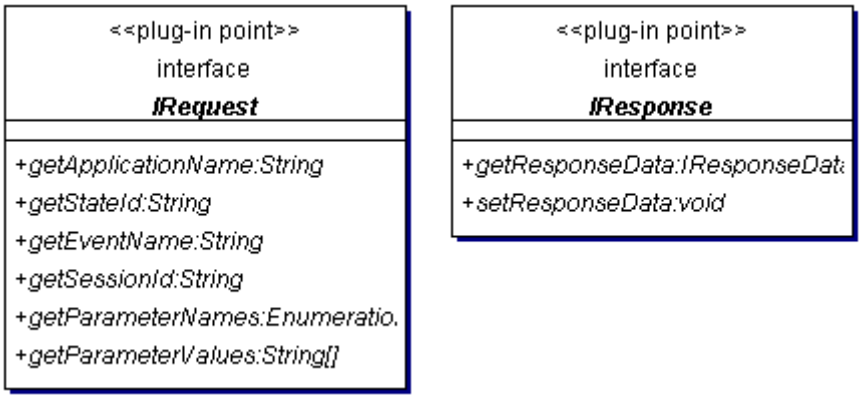

Figura 20 - Interfaces para os conectores

Para a representação abstrata do diagrama de estados, é fornecido um conjunto de interfaces, utilizadas em todo o código, algumas classes que contém os atributos e a hierarquia de estados, e um carregador desenvolvido com a *interface* Jade, responsável por carregar a definição da máquina de estados a partir de um arquivo escrito em XML.

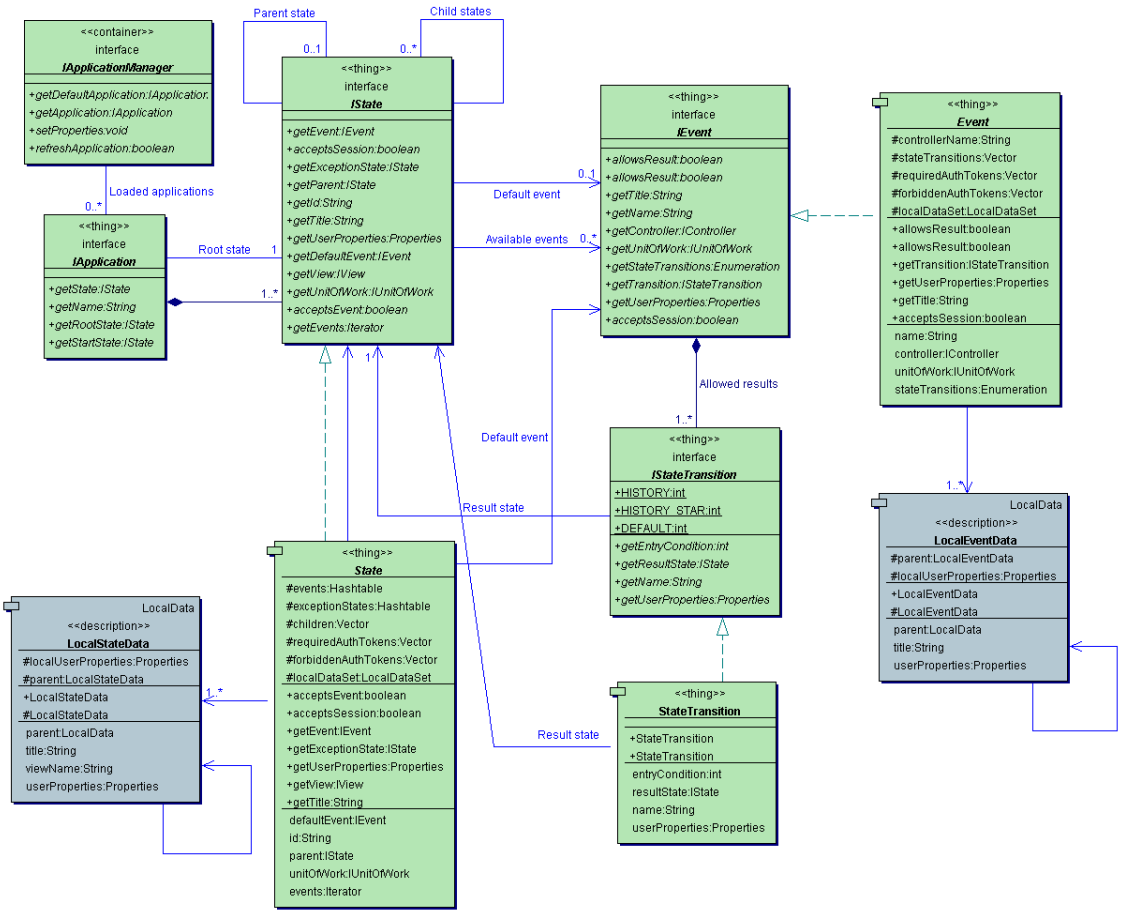

Figura 21 - Diagrama de classes da estrutura abstrata de máquina de estados

A classe *RequestManager* e auxiliares representam o núcleo do *JStateMachine*. Tal classe foi projetada de modo a ser independente da estrutura de *interface* escolhida.

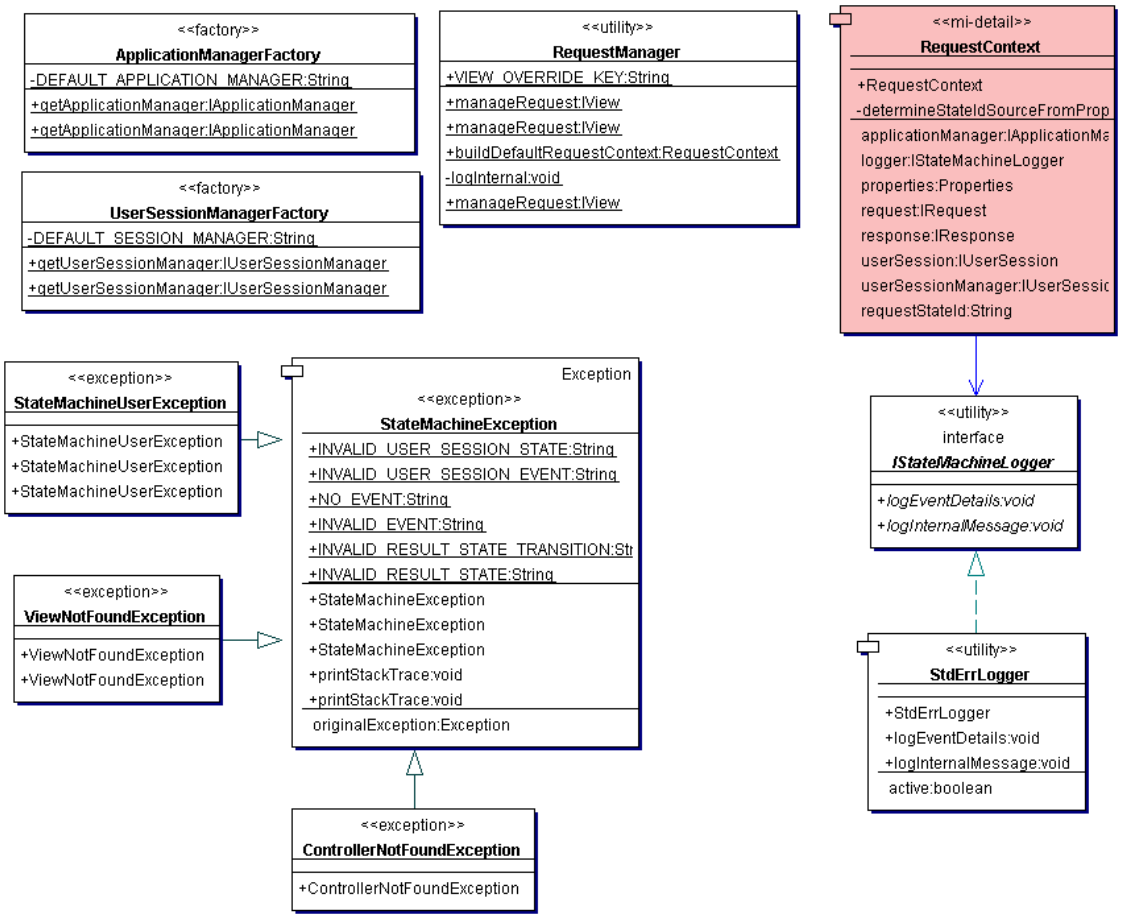

Figura 22 - Classes que formam o núcleo do JStateMachine

A versão utilizada, 0.2, do JStateMachine provê implementações do conector para interfaces baseadas em *web* (via um *servlet*), e também para interfaces baseadas em *Swing*. Também estão presentes dois gerenciadores de sessão, um dependente da utilização do conector para web e outro mais simples, apenas armazenando o estado em memória, destinado à utilização com o conector para interfaces *Swing*.

Sua configuração é feita em um arquivo no formato XML, que será carregado em uma representação abstrata do diagrama de estados. A descrição dos elementos e atributos está definida abaixo:

1. *Application*

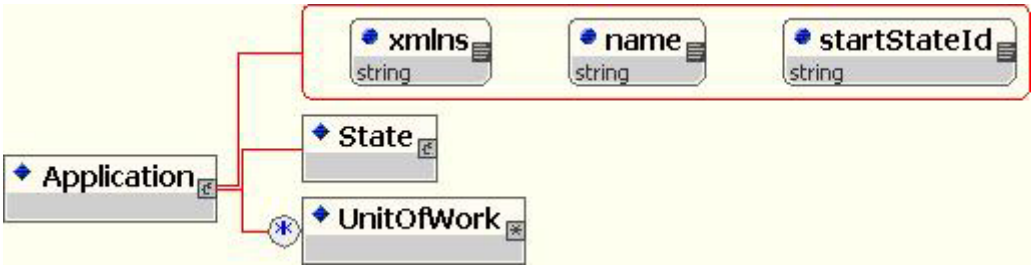

Figura 23 - Elemento Application

Este elemento é a raiz do arquivo de configuração. Contém como atributos: xmlns, que deve ser igual a "java:org.jstatemachine.apploaders.jade", que é o pacote com as classes que definem os elementos da configuração; name, que define o nome da aplicação; e startStateId, para definir qual o estado de início da aplicação. Dentro de Application, deve ser definido apenas um estado, chamado estado raiz, via elemento State.

2. *State*

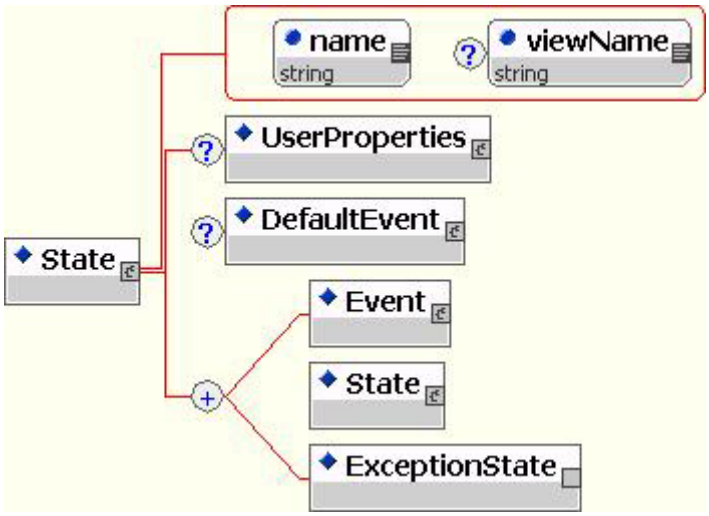

Figura 24 - Elemento State

Define um estado, o qual pode ou não ser uma página a ser mostrada para o usuário. Como atributos, possui: **name**, que define o nome do estado; e **viewName**, que representa o nome da classe responsável por gerar a visão para este estado específico. Caso não seja informado, será pego o nome da classe do estado pai.

No interior de um estado, podem estar definidos outros estados, ou seja, não há limite para a colocação de estados dentro de outros estados.

No interior de *State*, podem estar presentes, além de outras entidades *State*: *UserProperties*, para definir propriedades que estarão disponíveis para a visão; *DefaultEvent*, que define qual é o evento padrão deste estado; *Event*, para definir um evento que pode ser enviado para este estado ou para todos os seus descendentes; e *ExceptionState*, para definir estados especiais relacionados ao tratamento de exceções.

3. *Event*

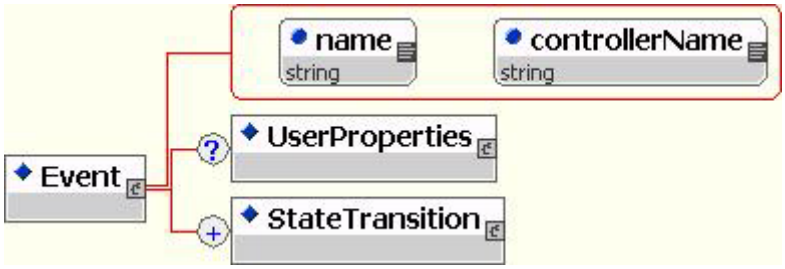

Figura 25 - Elemento Event

Define um evento que pode ser recebido enquanto está no estado atual. Seus atributos são **name**, para definir o nome do evento que a UI deverá gerar; e **controllerName**, para informar o nome da classe do controlador responsável por tratar este evento.

No interior do elemento *Event*, os elementos *UserProperties*, para definir propriedades que o controlador terá acesso; e *StateTransition*, para definir as transições entre os estados.

4. *StateTransition*

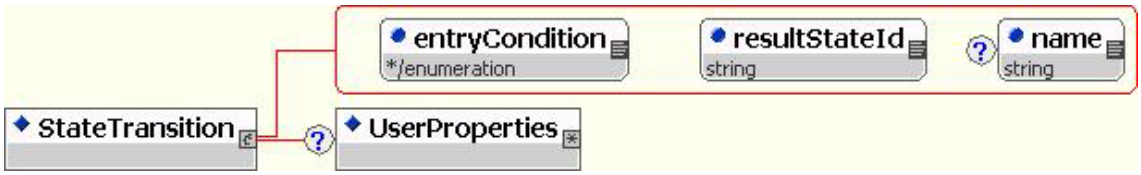

Figura 26 - Elemento StateTransition

Define uma transição. Este elemento faz parte de um *Event*. Como atributos, a **entryCondition** define para qual estado no interior do estado destino a transição irá; **resultStateId** define o estado destino e **name** é um nome para esta transição, que será retornado pelo controlador do evento.

Como condições de entrada (*entry condition*), temos três alternativas:

- DEFAULT: O estado de destino é exatamente o especificado em **resultStateId**.
- HISTORY: O estado de destino é o último estado visitado que seja filho do especificado em **resultStateId**.
- HISTORY_STAR: O estado de destino é o último estado de nível mais profundo que esteja dentro do especificado em **resultStateId**.

Também é possível definir propriedades, via *UserProperties*, que serão acessíveis através da *interface IStateTransition*.

5. *UserProperties*

Figura 27 - Elemento UserProperties

Esta entidade, que pode ser utilizada dentro de diversas outras, permite ao desenvolvedor passar algum parâmetro para as classes utilizadas pelo JStateMachine.

Apenas são permitidos elementos *UserProperty*. Cada elemento *UserProperty* possui dois atributos, *key* e *value*, cuja interpretação depende do contexto e da classe sendo utilizada.

6. *DefaultEvent*

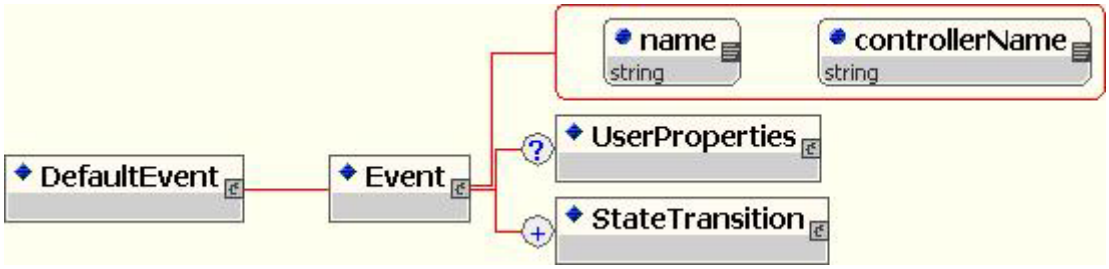

Figura 28 - Elemento DefaultEvent

A entidade *DefaultEvent*, utilizada em *State*, representa um evento que será executado automaticamente (ou seja, sem nenhuma intervenção por parte do usuário) ao ocorrer uma transição para o estado que contém o *DefaultEvent*. Em seu interior, apenas uma entidade *Event* é permitida.

7. *ExceptionState*

Figura 29 - Elemento ExceptionState

A entidade *ExceptionState* é utilizada para definir um estado de exceção. Quando algum controlador lança a exceção *StateMachineUserException*, o texto desta exceção (a mensagem) é utilizado para pesquisar, no estado atual e em seus estados pais, um estado de exceção apropriado.

O atributo **name** define o nome da exceção, conforme passado ao construtor de *StateMachineUserException*, e o **stateId** indica o nome do estado para o qual o fluxo irá se desviar, caso ocorra a exceção especificada.

8. *UnitOfWork*

Figura 30 - Elemento UnitOfWork

A entidade *UnitOfWork* é uma extensão do *JStateMachine* para o diagrama de estados UML, utilizada para estabelecer atividades transacionais, no estilo de *wizards*. Os estados definidos no atributo **containedStates** farão parte desta unidade de trabalho, na qual há apenas um único ponto de entrada, e podem haver vários pontos de saída.

Estes estados se diferenciam dos demais no sentido em que apenas serão aceitas transições entre os estados especificados. Caso haja alguma transição para os estados fora desta unidade, uma exceção será lançada.

3. SISTEMAS DE ENSINO À DISTÂNCIA

3.1. PANORAMA DO ENSINO À DISTÂNCIA NA INTERNET

Atualmente é uma realidade, e até uma necessidade, a utilização da Internet para realizar tarefas corriqueiras e cotidianas como fazer compras, administrar contas bancárias, efetuar pagamentos, entre outras, dessa forma acabou surgindo vários conceitos de utilização da infra-estrutura proporcionada pela Internet para funções mais elaboradas como trabalhar e estudar.

Partindo dessa premissa, torna-se bastante interessante desenvolver aplicações para se adequar a esses conceitos, principalmente no caso da educação a distância, que ainda sofrem com a deficiência na padronização das ferramentas em funcionamento. O projeto *Campus Virtual* se propõe a solucionar a maioria desses problemas, contando com equipamentos e recursos computacionais e conceituais adequados e modernos, de forma a dar credibilidade e funcionalidade real a esse tipo de ensino. A proposta do projeto Campus Virtual não é substituir os métodos de ensino regulares, mas tirar proveito de alguns recursos e conceitos inovadores e as facilidades oferecidas pela Internet para instituir uma forma séria e confiável de ensino complementar, inicialmente voltado à comunidade acadêmica.

Um dos fatores mais significativos para a implementação da educação a distância é a integração da utilização da Internet ao currículo acadêmico do estudante, de modo a complementar o currículo tradicional de ensino de graduação e pós-graduação. Isso também proporciona autonomia aos estudantes, que são responsáveis pelo andamento dos cursos oferecidos.

Outro problema de importância considerável encontrado quando se levanta a questão do ensino à distância é a falta de uma metodologia de ensino padronizada implementada e de uma aplicação adequada para tal. A intenção do projeto Campus Virtual é propor um novo sistema de ensino, suprindo às deficiências identificadas, não para substituir o tradicional, mas para complementar o método regular de ensino em sala de aula, seja na forma de cursos de extensão, seja na forma de estudos complementares de aula, visando agregar conhecimento aos alunos e aumentar o rendimento acadêmico dos mesmos.

O projeto Campus Virtual é decorrente de uma etapa posterior de uma experiência implementada anteriormente em um projeto do PIBIC – Programa de Bolsas de Iniciação Científica, patrocinado pelo CNPq, iniciado em 1998 com duração de um ano. Nesse projeto foi implementado, em caráter experimental, em algumas disciplinas do curso de graduação em Engenharia de Redes de Comunicação da UnB – Universidade de Brasília.

As implementações atuais vem introduzir ao projeto conceitos muito mais avançados de computação e uma metodologia de publicação e disponibilização de conteúdo e base de conhecimento muito mais eficientes.

3.2. A INICIATIVA ADL

Um conceito bastante respeitado no cenário atual de ensino a distância é a iniciativa ADL – *Advanced Distributed Learning* (Aprendizagem Distribuída Avançada) – lançada em novembro de 1997 pelo DoD – *Department of Defense* – e pelo OSTP – *White House Office of Science and Technology Policy* – ambos Norte Americanos.

A missão dessa iniciativa é prover instrução de alta qualidade e auxílio de decisão a qualquer hora, em qualquer lugar e moldada para as necessidades específicas de cada aprendiz. Os desafios para isso não estão baseados na tecnologia em si, mas no entendimento da utilização da infra-estrutura tecnológica para o aprendizado a qualquer hora em qualquer lugar.

O ADL prevê a criação de uma base de conhecimento compartilhada, onde os objetos de aprendizagem são catalogados para distribuição e uso, encontrando-se acessíveis através da Internet.

3.3. A EFICIÊNCIA DA INSTRUÇÃO BASEADA NA TECNOLOGIA

Uma grande vantagem da instrução baseada em tecnologia é a redução de custos quando se pretende alcançar uma grande gama de objetivos. Alguns estudos revelaram que essa redução de custos pode variar entre 30 a 60 por cento[10]. Estes estudos também revelaram uma redução no tempo necessário para se atingir determinados objetivos e uma ampliação no conhecimento dos estudantes para um tempo fixo estipulado.

As tecnologias de aprendizagem, segundo a iniciativa ADL, são classificadas em duas grandes categorias: síncrona e assíncrona.

Os programas tradicionais de ensino a distância enfatizam tecnologias de aprendizagem que provêem treinamento para estudantes que se encontram fisicamente separados dos instrutores. Essas tecnologias podem ser vistas em salas de aula virtuais, por vídeo-conferência ou tele-aulas. O problema é que estas requerem que os estudantes estejam reunidos em um horário específico, e às vezes em um local específico também, no qual serão ministradas as aulas. Essas tecnologias se enquadram na categoria síncrona.

O ADL valoriza as tecnologias assíncronas, que podem prover informações e acompanhamento sem que seja necessário os estudantes estarem todos reunidos em um mesmo horário ou local, estando as informações disponíveis a todo o tempo e em qualquer lugar. Alguns exemplos de tecnologias assíncronas são:

[10] Fonte ADL Network – http://www.adlnet.org/

- Instrução baseada em computador;
- Sistemas Tutoriais Inteligentes;
- Treinamento baseado na Internet.

O projeto Campus Virtual se encaixa nessa tecnologia de aprendizagem distribuída, substituindo o sistema tradicional de aprendizagem baseada em computador, por um sistema de aprendizagem baseada na Internet.

3.4. O SCORM

O SCORM (*Shared Content Object Reference Model*) é um modelo que define a interação entre a LMS (*Learning Management Station*) e o conteúdo apresentado, e também um formato padronizado para a troca de conteúdos, independentes do software utilizado.

Tal modelo permite ao desenvolvedor de conteúdo (professor) reutilizar cursos, aulas ou tópicos desenvolvidos por outros, sem nenhuma necessidade de adaptação. Também permite ao professor a utilização de recursos em outros LMS, de modo manter o conteúdo distribuído pela Internet.

Atualmente, o SCORM é um conjunto de especificações resultantes do desenvolvimento de algumas tecnologias relacionadas ao ensino através da Internet, obtidas por órgãos como o *IMS Global Learning Consortium*, o AICC (*Aviation Industry CBT Comittee*), o ARIADNE (*Alliance of Remote Instructional Authoring & Distribution Networks for Europe*) e o LTSC (*Learning Technology Standards Comittee, do IEEE*).

Tal desenvolvimento resultou em dois tópicos principais: o modelo de agregação de conteúdo e o ambiente de execução.

O modelo de agregação de conteúdo define como a hierarquia do conteúdo é organizada e como será feito o seqüenciamento dos itens através de pré-requisitos.

Além disso, este modelo contém informações sobre como o conteúdo (objetos de ensino) é distribuído, ou seja, contém as meta-informações sobre o conteúdo, como título, descrição, tipo de conteúdo e requisitos para visualização.

Com este modelo, é garantido que o conteúdo pode ser "importado" por qualquer sistema que seja capaz de compreender uma representação padronizada do conteúdo. No caso do SCORM, esta representação é feita em XML.

Para garantir independência à plataforma de ensino, também é padronizada uma *interface* entre o conteúdo e a LMS, de modo que as estruturas de dados sejam visíveis para o conteúdo de uma forma uniforme, independente de onde este esteja sendo exibido ou esteja armazenado.

4. EXTENSÕES FEITAS AO JSTATEMACHINE

Apesar de o JStateMachine resolver alguns dos problemas relativos ao desenvolvimento de aplicações *web*, na versão utilizada ainda haviam alguns problemas:

1. O controle de estado era feito exclusivamente através de um parâmetro passado para o *servlet* de controle (*EntryServlet*);
2. O conceito de efeitos colaterais (*side effects*) não era bem resolvido; e
3. Falta de possibilidade de armazenar variáveis globais (em sessão).

Assim, algumas alterações foram feitas na biblioteca, também disponibilizadas sob licença LPGL, para resolver estes problemas.

Prováveis problemas de escalabilidade também foram resolvidos, uma vez que, na versão original, para cada requisição, era criado um objeto correspondente ao controlador. Nesta versão, o controlador e as ações são instanciados apenas uma vez, e tal instância é reutilizada em todas as solicitações.

Além das alterações, foram adicionadas algumas *taglibs*, de forma a facilitar o desenvolvimento de aplicações para a *web*.

4.1. CONTROLE DE ESTADO

Para poder ter um melhor controle sobre o estado atual, e evitar que um usuário mal-intencionado pudesse alcançar algum estado sem que antes passasse pelos estados anteriores, o controle do último estado visitado passou a fazer parte da *interface IUserSession*.

Outro problema a ser resolvido era a atualização da página pelo cliente (quando o usuário clica sobre o botão *Atualizar*), e o histórico do navegador. Para isto, duas ações foram tomadas: desabilitar o *cache* das páginas geradas, através de cabeçalhos específicos do protocolo HTTP, e o acréscimo de um *token* de 64 bits gerado aleatoriamente, através da classe *Math.SecureRandom* e uma operação ou-exclusivo com o retorno de *System.currentTimeMillis()*.

Com as páginas não sendo armazenadas em *cache*, quando o usuário clica nos botões de voltar ou avançar no navegador, é feita uma nova solicitação ao servidor. Como na solicitação será enviado o *token* antigo, o sistema é capaz de detectar tal tipo de operação e retornar ao usuário os mesmos dados vistos na tela anterior.

Ao utilizar o botão *Atualizar*, também será enviado ao servidor o *token* antigo, e o sistema pode tratar como sendo o mesmo caso acima, e tomar a mesma ação, ou seja, enviar ao usuário a mesma resposta vista anteriormente.

Para poder atingir os objetivos de um melhor controle do usuário, foram acrescentados três métodos em *IUserSession*: *getLastState()* e *getLastStateCookie()* e *isUserInLastState()*. O método *getLastState()* surgiu após a integração com algumas partes da versão 0.3.1 da biblioteca original, juntamente com as constantes definidas também nesta classe.

O método *getLastState()* retorna o último estado visitado pelo usuário, para, a partir daí, definir o estado atual.

getLastStateCookie() é responsável por retornar o *token* a enviar na resposta para o navegador, correspondente ao estado cuja página está sendo retornada.

isUserInLastState() possui como parâmetro o *token* enviado pelo cliente, e o objetivo desta função é verificar se este *token* é realmente o gerado antes do envio da resposta para o cliente. É através desta função que o *RequestManager* é capaz de descobrir se a requisição enviada ao servidor realmente corresponde à esperada (ou seja, ao estado esperado).

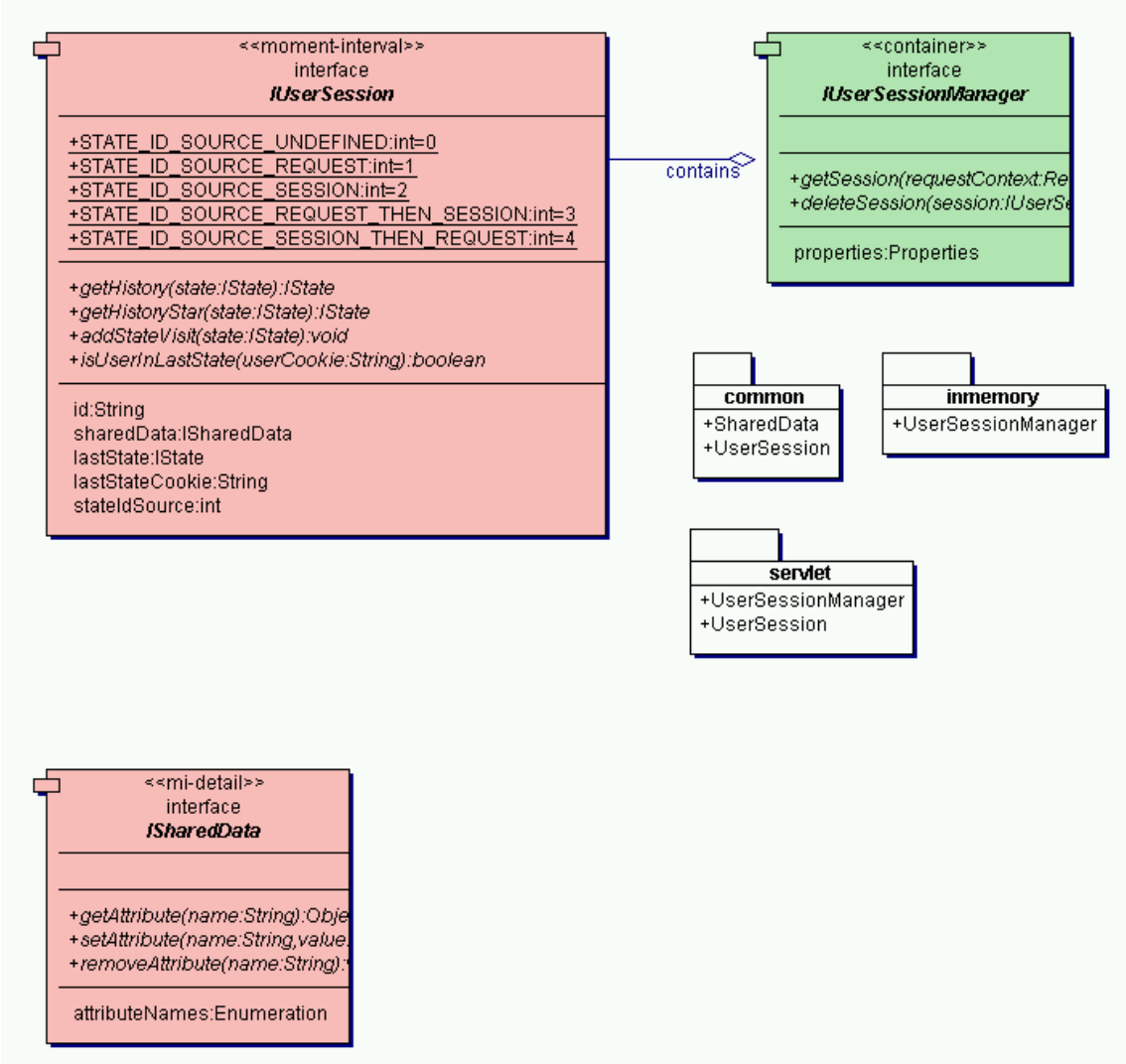

Figura 31- Novo diagrama de classes do pacote org.jstatemachine.sessionmanagers

Com estas alterações, pode-se ver o diagrama de seqüência executado pelo RequestManager na Figura 32logo abaixo.

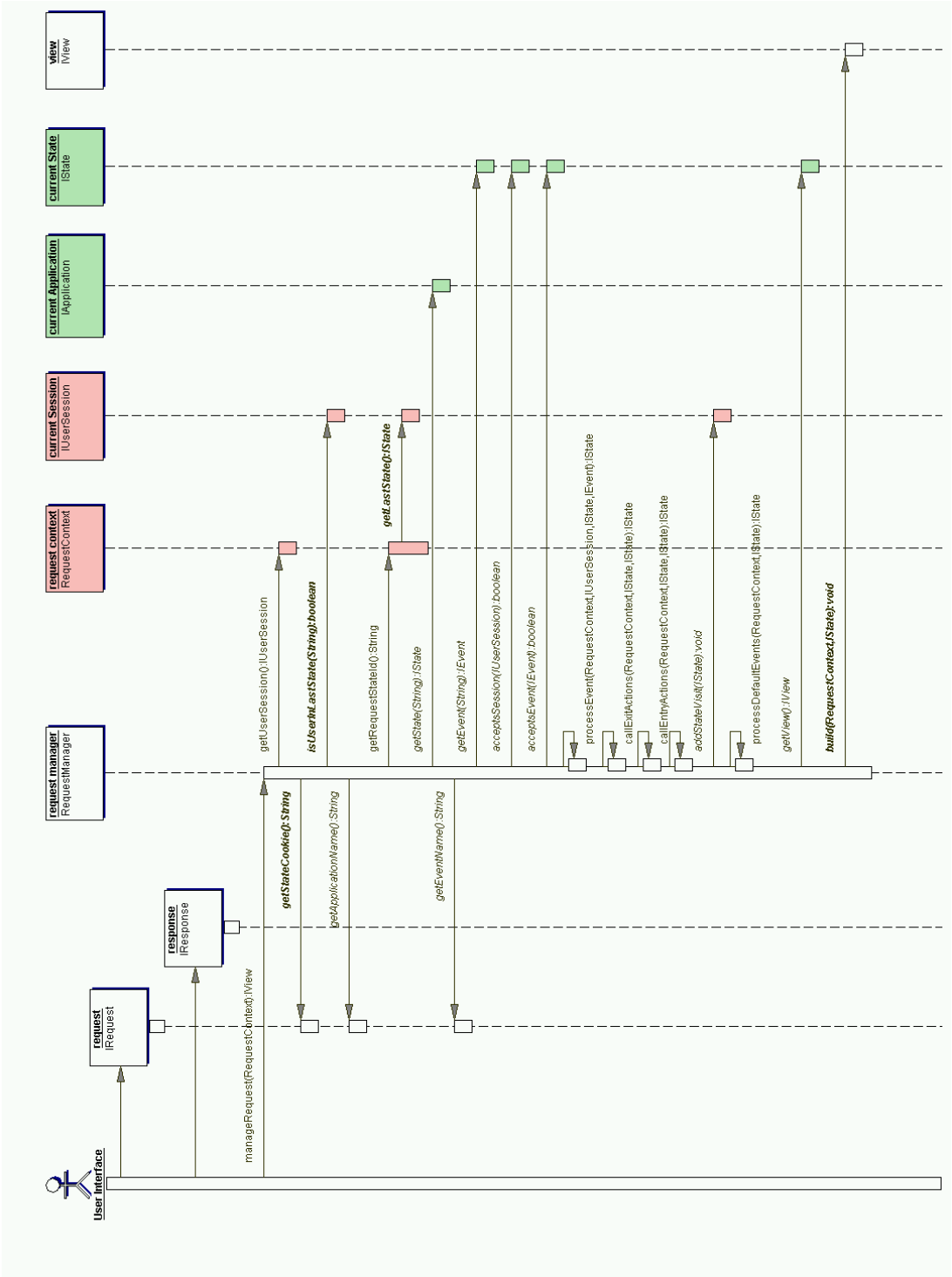

Figura 32 - Diagrama de seqüência para o RequestManager

4.2. EFEITOS COLATERAIS

Como o conceito de Efeitos Colaterais (*side-effects*) não estava bem explícito no JStateMachine, ele foi retirado em favor da colocação de ações de entrada e saída, de acordo com a especificação da UML.

Tais ações são executadas após a transição ser determinada, isto é, após o controlador correspondente ao evento ter seu método *execute()* executado.

Para a execução das ações de entrada, são verificados todos os estados que serão atravessados, a partir do nível mais alto correspondente a esta transição. No exemplo mostrado na Figura 33, abaixo, as ações de entrada para os estados **D** e **E** serão executadas. Tais ações serão executadas a partir do estado mais externo; portanto, a ação correspondente ao estado **D** será executada, e, em seguida, a ação do estado **E**.

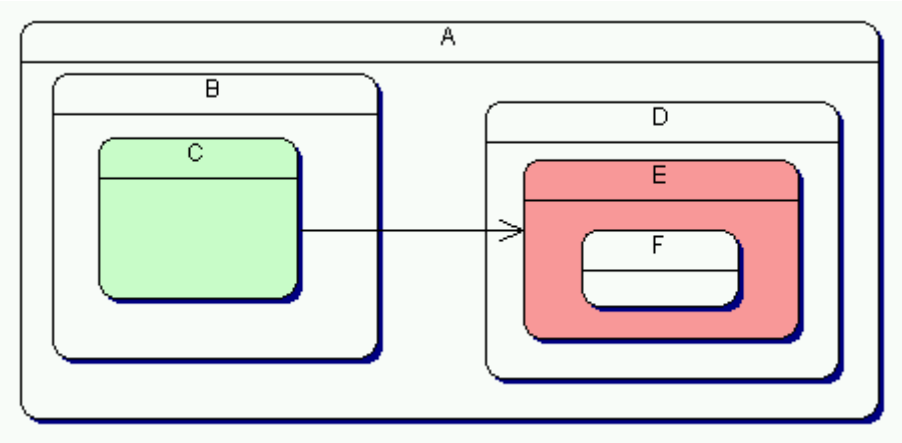

Figura 33 - Exemplo de transição

As ações de saída, executadas antes das ações de entrada, são chamadas em todos os estados atravessados até chegar em um que seja comum tanto ao estado de origem quanto ao de destino. Tais ações são chamadas a partir da mais interna. No exemplo acima, é executada primeira a do estado **C**, e, em seguida, a do estado **B**.

4.3. ATRIBUTOS DE SESSÃO

A versão 0.2 do JStateMachine não suporta a passagem de atributos de um estado para outro. Portanto, o desenvolvimento de aplicações mais complexas não era possível devido a esta deficiência.

Como forma de solucionar este problema, optou-se por remover a *interface* *IResponseData*, responsável por armazenar os dados da resposta associados a cada estado, e acrescentar *ISharedData*, com métodos para listar, alterar e remover atributos que são compartilhados entre todos os estados.

Outro problema encontrado com *IResponseData* é que sua implementação é dependente do gerenciador de sessões utilizado; assim, caso houvesse alguma alteração no gerenciador de sessões, todos os controladores desenvolvidos anteriormente talvez precisassem ser adaptados.

Desta forma, *ISharedData* possui os membros necessários para o acesso aos dados, e também permite o compartilhamento entre todos os estados.

Outra alteração feita quando desta implementação diz respeito às condições de entrada HISTORY e HISTORY_STAR. Na implementação original, quando uma transição desta ocorria, os dados de resposta que são enviados ao usuário são exatamente os mesmos de quando o estado resultante foi visitado pela última vez. Tal forma de operação dificultava a alteração destes dados pelo controlador da transição responsável por esta condição de entrada.

Com a implementação do *ISharedData*, os dados não são alterados quando há uma transição deste tipo, ou seja, não são recuperados os atributos e valores de quando o estado foi visitado pela última vez. Isto permite aos sistemas alterar alguma informação global em um estado cuja transição de saída possua uma condição HISTORY ou HISTORY_STAR.

O diagrama de classes referente a *ISharedData* foi mostrado na Figura 31 Nele podemos observar os métodos *getAttribute()*, cuja finalidade é obter um valor correspondente ao atributo especificado; *setAttribute()*, para ajustar um valor para o atributo especificado; e *removeAttribute()*, para remover um atributo dos dados compartilhados.

Este *design* foi inspirado no Oracle MVC Framework, que possui uma estrutura de compartilhamento de dados semelhante. Entretanto, em tal *framework*, caso tenha ocorrido uma transição não esperada (por exemplo, o usuário pressionou o botão de recarregar, avançar ou voltar no navegador), será pego o conjunto de valores correspondente ao estado especificado nesta transição. Portanto, para cada estado que corresponde a uma apresentação, há uma cópia dos valores compartilhados.

4.4. TAGLIB

Para facilitar o desenvolvimento de páginas *JSP*, de forma a não ser necessário ao *web designer* ter conhecimentos sobre as classes internas do JStateMachine, foram feitas algumas *tags* específicas para esta biblioteca, definidas na *URI http://www.jstatemachine.org/jsm-taglib*. A especificação destas *tags* está descrita na tabela abaixo:

Tag	Descrição
form	Cria um formulário já com os atributos necessários para um evento específico. Atributos: *event* (opcional): nome do evento que este formulário enviará *name* (opcional): nome do formulário

	beforeError (opcional): texto a ser colocado antes de alguma mnsagem de erro que os itens deste formulário tenham associado *afterError* (opcional): texto a ser colocado após alguma mensagem de erro que os itens deste formulário tenham associado *errorPosition* (opcional): define se as mensagens de erro devem ser colocadas antes (*before*) ou depois (*after*) o campo de formulário que contenha erro
currentState	Coloca no atributo de contexto de página especificado pelo parâmetro *id* o nome do estado atual. Esta tag não possui corpo, ou seja, nenhum elemento em seu interior. Além disso, cria a variável com o nome especificado por *id*.
sharedData	Coloca o valor do atributo compartilhado entre os estados, especificado pelo parâmetro *name*, com a classe especificada por *className*, na variável cujo nome é definido por *id*.
error	Coloca na saída o texto de erro correspondente ao campo especificado pelo atributo *name*. Caso *name* não tenha sido especificado, serão colocados os erros globais.
value	Coloca na saída o valor de um atributo compartilhado, especificado pelo atributo *name*, ou um valor, especificado pelo atributo *value*. Esta tag toma o cuidado de substituir os caracteres reservados do HTML (menor que, maior que, aspas simples e duplas, além de caracteres *UNICODE)* pelas entidades correspondentes. É utilizado para evitar que tais caracteres possam ser interpretados pelo navegador como marcações HTML ou finalizadores de *string* em JavaScript.

Tabela 1. Tags acrescentadas ao JStateMachine

Para o suporte a listas divididas em páginas e também a páginas que contém “abas”, ou seja, nem todos os detalhes de uma informação estão disponíveis na mesma página, foi feita uma alteração também no tratamento de eventos, de tal forma que, caso não seja passado um evento, a mesma visão será gerada. Entretanto, é possível passar parâmetros para a visão a partir de mecanismos próprios da tecnologia de apresentação utilizada. No caso do conector para *servlets* é possível passar dados através da requisição, e obtê-los utilizando *getParameter()*, na *interface ServletRequest*.

Utilizando este fato, pode-se construir listas paginadas deixando todos os dados disponíveis em *ISharedData*, e mostrando apenas um subconjunto destes, de acordo, por exemplo, com a página passada como parâmetro.

Para facilitar este desenvolvimento, duas *tags* foram criadas: *pager* e *itemIterator*, cuja documentação está nas tabelas 2 e 3, respectivamente.

Atributo	Obrigatório?	Descrição
attribute	Sim	Nome do atributo em *ISharedData* que contém os dados a serem utilizados.
pageParameter	Sim	Nome do parâmetro com o número da página
pageSize	Sim	Número de entradas por página
previousPageForm	Não	Nome do formulário que será gerado com as ações para ir para a página anterior
nextPageForm	Não	Nome do formulário que será gerado com as ações para ir para a próxima página
currentPageVar	Não	Nome da variável do tipo *Integer* que conterá o número da página atual
isFirstVar	Não	Nome da variável do tipo *Boolean* que conterá um indicador se é a primeira página.
isLastVar	Não	Nome da variável do tipo *Boolean* que conterá um indicador se é a última página
numberOfPagesVar	Não	Nome da variável do tipo Integer que conterá o número total de páginas

Tabela 2. Documentação da tag Pager

Com a tag *pager*, toda a estrutura necessária para o desenvolvimento de uma lista paginada fica facilitada. A tag *itemIterator* é utilizada para pegar os itens e colocá-los na página.

Atributo	Obrigatório?	Descrição
Id	Sim	Nome da variável que conterá o item
className	Não	Nome completo da classe da variável que conterá o item. Caso não especificado, será assumido *java.lang.Object*.
attribute	Não	Caso especificado, a fonte de dados será um atributo compartilhado, cujo nome é especificado aqui. Do contrário, a fonte de dados será a *tag pager* logo acima na hierarquia de tags.

Tabela 3. Atributos de itemIterator

4.5. CONFIGURAÇÃO

Com a inclusão das ações de entrada e saída, algumas alterações no formato do arquivo de configuração foram necessárias.

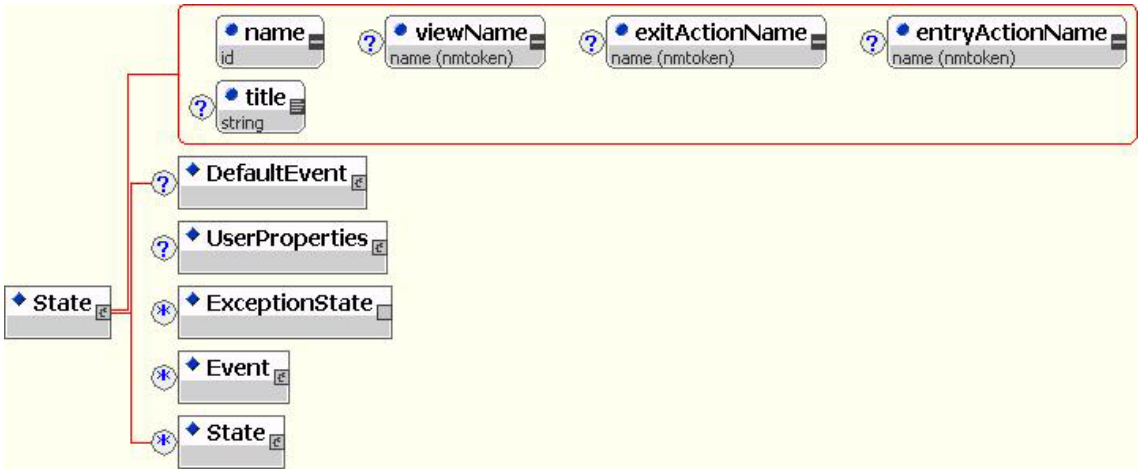

Figura 34 - Elemento State após as alterações no JStateMachine

Como se pode observar na figura acima, foram adicionados dois atributos, *exitActionName* e *entryActionName*. O atributo *title* é resultante da junção da versão desenvolvida com a versão 0.3.1 da biblioteca.

Os atributos *entryActionName* e *exitActionName* definem os nomes das classes que implementam a *interface IAction*, correspondendo, respectivamente, às ações executadas ao entrar em um estado, e a sair de um estado.

5. DESENVOLVIMENTO DE UM SISTEMA GESTOR DE CURSOS PELA INTERNET - O PROJETO CAMPUS VIRTUAL

O Projeto Campus Virtual foi concebido para atender às necessidades de complementação e enriquecimento curricular do ensino de graduação, tomando por base o curso de Engenharia de Redes de Comunicação da Universidade de Brasília.

Tendo em vista o aspecto funcional de ensino do Campus Virtual, o sistema consiste em uma plataforma de ensino a distância baseado na Internet, seguindo os conceitos dispostos pela iniciativa ADL e pelo modelo SCORM.

O site do Campus Virtual possui uma parte de conteúdo aberto com informações sobre o projeto e o funcionamento do sistema, de acesso irrestrito. Alguns cursos, se desejado, podem ser disponibilizados nessa área do site. Existe também uma parte restrita onde se encontram basicamente três grandes áreas: a área dos alunos, a área dos professores e a área dos coordenadores de curso.

Na área destinada aos alunos se encontram as aulas propriamente ditas, informações sobre sua posição no curso, informações sobre os itens do curso, como pré-requisitos, ementa, duração e etc.

Na área destinada aos professores existem ferramentas apropriadas para o acompanhamento e avaliação dos alunos, além como a possibilidade de alteração de alguns itens, sobre os quais foi delegada responsabilidade pelo coordenador.

Na área destinada aos coordenadores existem ferramentas que permitem a alteração dos itens do curso, a delegação de atividades para os professores, aprovação do cadastro de alunos e controle do curso, além de estatísticas detalhadas sobre informações pertinentes ao curso.

Essas áreas serão mais bem desmembradas e discorridas no decorrer deste capítulo, cabendo a essa parte apenas uma visão geral de cada uma e do sistema como um todo.

A seqüência das disciplinas, os pré-requisitos e outras características pertinentes ao curso são definidas pelo autor responsável pela agregação do conteúdo.

O sistema do Campus Virtual permite interação direta com os professores e coordenadores de forma rápida e precisa, visando melhorar o processo de aprendizagem e interagir os alunos com os professores, propiciando um ambiente o mais agradável e o menos impessoal possível.

Do ponto de vista técnico, o Projeto Campus Virtual foi concebido dentro da filosofia do Software Livre, sobre os padrões de licença LGPL, tendo toda a sua infra-estrutura baseada em softwares que se encaixam na mesma filosofia.

Do ponto de vista computacional, o sistema do Campus Virtual foi implementado sobre uma plataforma J2EE, utilizando recursos EJB, JSP e servlets. O software foi implementado em uma arquitetura de três camadas, de cliente-enxuto, e utilizando modelagem de diagramas de estados (*statecharts*).

A infra-estrutura do sistema do Campus Virtual está montada em um ambiente RedHat Linux. Todos os componentes do sistema utilizam o Servidor de Aplicação JBoss, e o Sistema de Banco de Dados utilizado é o MySQL.

Toda a confecção e implementação do sistema foi realizada utilizando softwares livres e gratuitos, de modo a manter a coerência com a filosofia adotada na concepção do mesmo.

Todas as características mencionadas nesse capítulo estão discorridas com maiores detalhes nos capítulos subseqüentes.

5.1. MODELAGEM DO SISTEMA

Para o desenvolvimento do Campus Virtual, utilizou-se de uma abordagem de desenvolvimento em três camadas, ou seja, uma camada de persistência, responsável pelo armazenamento e recuperação de informações; uma camada de negócio, cuja finalidade é o processamento das informações e tomada de decisões; e uma camada de apresentação, responsável por interagir com o usuário.

O projeto foi feito com o foco nos casos de uso, ou seja, em uma visão que os agentes externos (usuários) teriam do sistema. A partir dos requisitos que o sistema possui e também destas visões, foram modelados os diagramas de caso de uso:

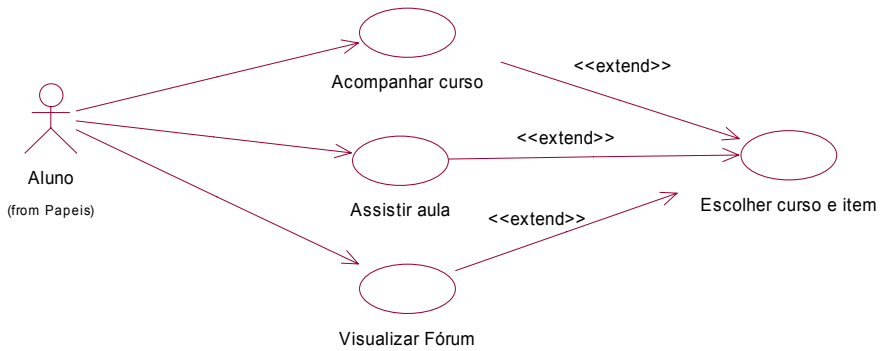

Figura 35 - Tarefas do aluno

O diagrama representado acima na Figura 35 representa as tarefas que um aluno pode executar no sistema. A notação <<*extend*>> quer dizer que o determinado caso de uso herda características de outro. Dessa forma, tanto **Acompanhar curso**, **Assistir aula** e **Visualizar Fórum** herdam características de "**Escolher curso e item**" no exemplo acima.

Na tabela abaixo está a documentação de cada item:

Item	**Descrição**
Aluno	Usuário cadastrado no sistema exercendo o papel de aluno
Acompanhar Curso	O aluno vê quais são os itens de um curso, qual é seu "status" em cada uma dos itens. Pode enviar arquivos para o coordenador, caso seja necessário no curso.
Assistir Aula	O usuário assiste um item. O sistema possui um modelo de execução compatível com o SCORM, assim, qualquer dado associado ao acompanhamento das aulas estará disponível via interfaces padronizadas.
Escolher curso e item	Permite ao aluno navegar pelos cursos em que ele se inscreveu e itens disponíveis.
Visualizar Fórum	O aluno pode enviar e receber mensagens de outros alunos e do coordenador através do fórum de discussão do curso.

Tabela 4. Documentação dos itens das tarefas do aluno

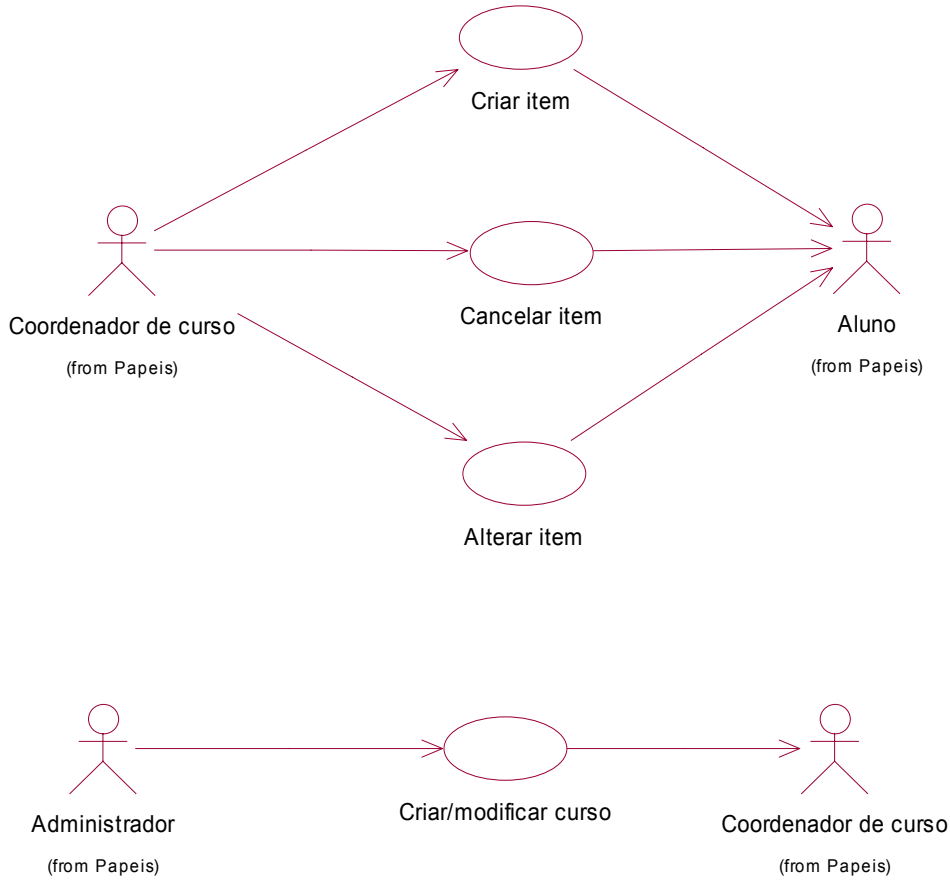

Figura 36 - Atividades de coordenação e administração de curso

As atividades de coordenação de curso, mostradas na Figura 36, consistem basicamente na alteração e criação de itens e curso. O administrador não tem acesso aos itens de um curso, exceto na fase de criação, quando poderá importar um arquivo em conformidade com o Modelo de Agregação de Conteúdo do SCORM (*SCORM Content Aggregation Model)*.

Item	**Descrição**
Coordenador de curso	Usuário cadastrado no sistema com o papel de coordenador de curso
Administrador	Usuário cadastrado no sistema com o papel de administrador
Criar/modificar curso	Cria ou modifica um curso on-line. Este curso pode ser ou não disponível para todos os usuários. Caso não esteja disponível para todos, o coordenador do curso deverá aprovar as solicitações de inscrição. Na criação do curso, são escolhidos alguns professores para serem os coordenadores do curso.

Alterar item	Altera o conteúdo de um item de um curso. Ao confirmar a alteração, os alunos matriculados nos cursos correspondentes serão avisados.
Cancelar item	É cancelado um item. O sistema avisa automaticamente tanto os alunos como os outros coordenadores deste cancelamento, enviando o motivo do mesmo.
Criar item	É criado um novo item. Os coordenadores do curso são avisados, assim como os alunos.

Tabela 5. Documentação das atividades de coordenação e administração

Outro conjunto de atividades pertinentes a esse grupo é o gerenciamento de inscrições, realizado pelo coordenador do curso. Esse conjunto se encontra esquematizado na Figura 37, mostrada abaixo.

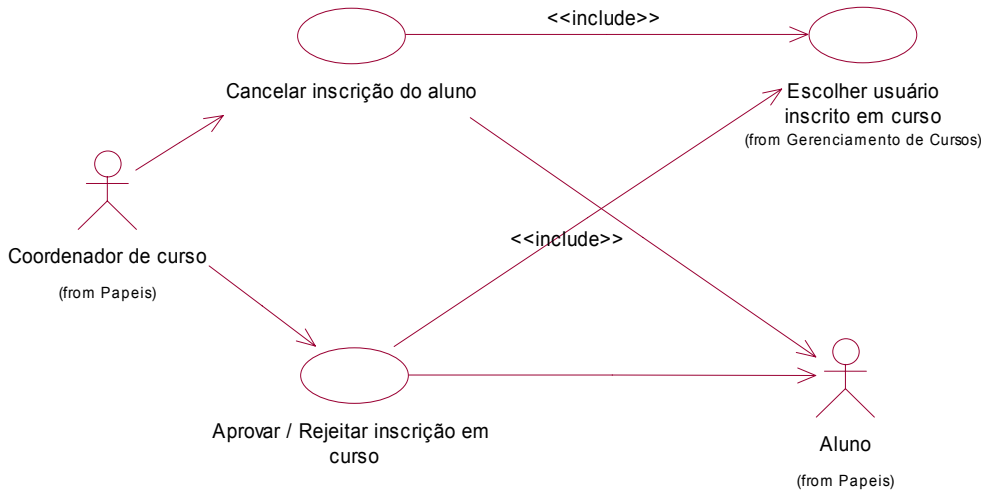

Figura 37 - Atividades de gerenciamento de inscrições

O último conjunto de atividades a merecer atenção é o gerenciamento de usuários, que inclui atividades relacionadas ao gerenciamento e manutenção das contas de usuário, sendo estas operações responsabilidades tanto do administrador quanto do usuário em si.

Tais operações estão descritas na Tabela 6 abaixo:

Item	**Descrição**
Solicitar cadastro	Um novo usuário solicita o cadastro no sistema. Após sua solicitação, é enviado um e-mail para ele com informações de como confirmar seu cadastro, além de seu nome de usuário e data limite para confirmação de seu cadastro.

Inscrever em curso	Um usuário com o cadastro confirmado pode escolher um dos cursos disponíveis para inscrição
Confirmar cadastro	Primeiro acesso do usuário, em que ele confirma informações sobre seu perfil. Nesta etapa é tida a garantia de que o usuário é alcançável de alguma forma pelo sistema, pois ele chegará aqui pois foi avisado pelo sistema que foi cadastrado com sucesso. É instruído a colocar uma contra-senha que foi enviada para ele no e-mail de cadastro.
Atualizar cadastro	O usuário quer modificar alguma informação sua, como nome e e-mail.
Alterar senha	O usuário deseja alterar a senha no sistema.
Editar usuário	Dá ao administrador a possibilidade de editar / apagar um usuário. Na edição, ele pode dar ao usuário o acesso de administrador e/ou professor, e também retirar este acesso. Ao final da operação, as alterações de papel do usuário (professor / administrador) são comunicadas via sistema ao usuário.
Escolher usuário	É escolhido um usuário já existente ou feito um novo cadastro. Lista os usuários e oferecemos uma opção para cadastrar o usuário.
Cadastrar usuário	Cadastra um usuário no sistema, podendo já deixá-lo confirmado.
Escolher usuário existente	Escolhe um usuário já existente para a alteração de seus dados ou seu cadastro em alguma outra operação.
Autentica	Cria uma nova sessão para o usuário, através do nome de usuário e senha. Em seguida, mostra as informações de abertura do sistema para este usuário.

Tabela 6 - Operações relacionadas a contas de usuários

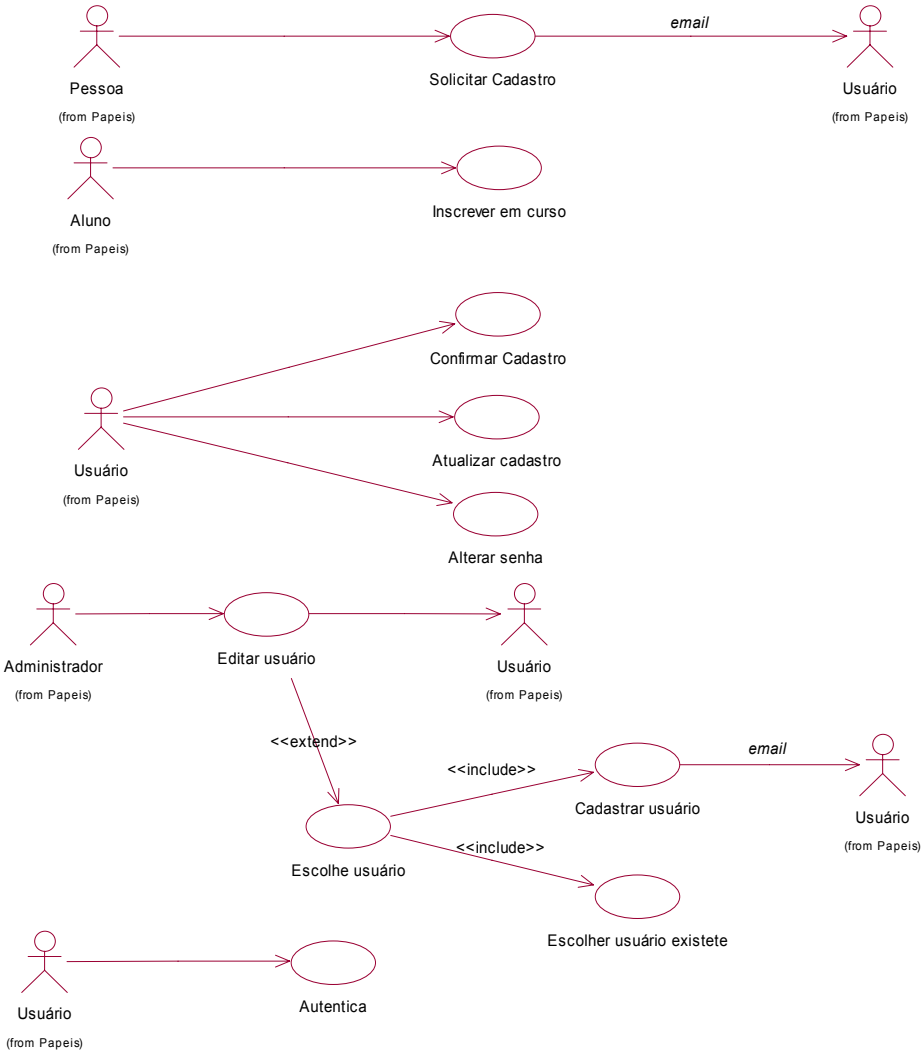

Figura 38 - Operações de gerência e manutenção de usuários.

5.2. MODELAGEM DAS CAMADAS DE PERSISTÊNCIA E NEGÓCIO

A partir dos casos de uso levantados e de um requisito do sistema – a implementação do SCORM – pode ser feito um levantamento dos dados que precisam ser armazenados de que forma tal armazenamento seja mais eficiente. Então foi iniciado o desenvolvimento de um software que pudesse oferecer pelo menos a funcionalidade básica desejada para um sistema de ensino à distância: o cadastramento de alunos e seu acesso às aulas.

Daí foi criado o modelo de classes para a camada de negócio, que engloba, além dos dados, uma estrutura sobre como eles são acessados. Como tecnologia de implementação, foi escolhido o EJB (*Enterprise Java Beans*). Desta forma, cada objeto é construído como um componente, e, neste caso, composto de duas interfaces e uma classe: a *interface home*, que contém o contrato necessário para a criação ou obtenção de uma instância do componente; a

interface remota, com o contrato para a utilização do componente, ou seja, disponibiliza os métodos que constituem as regras de negócio; e a própria implementação do componente.

Alguns componentes, principalmente os que fazem acesso ao banco de dados, foram implementados apenas com interfaces locais, uma vez que estes não serão expostos à camada de apresentação.

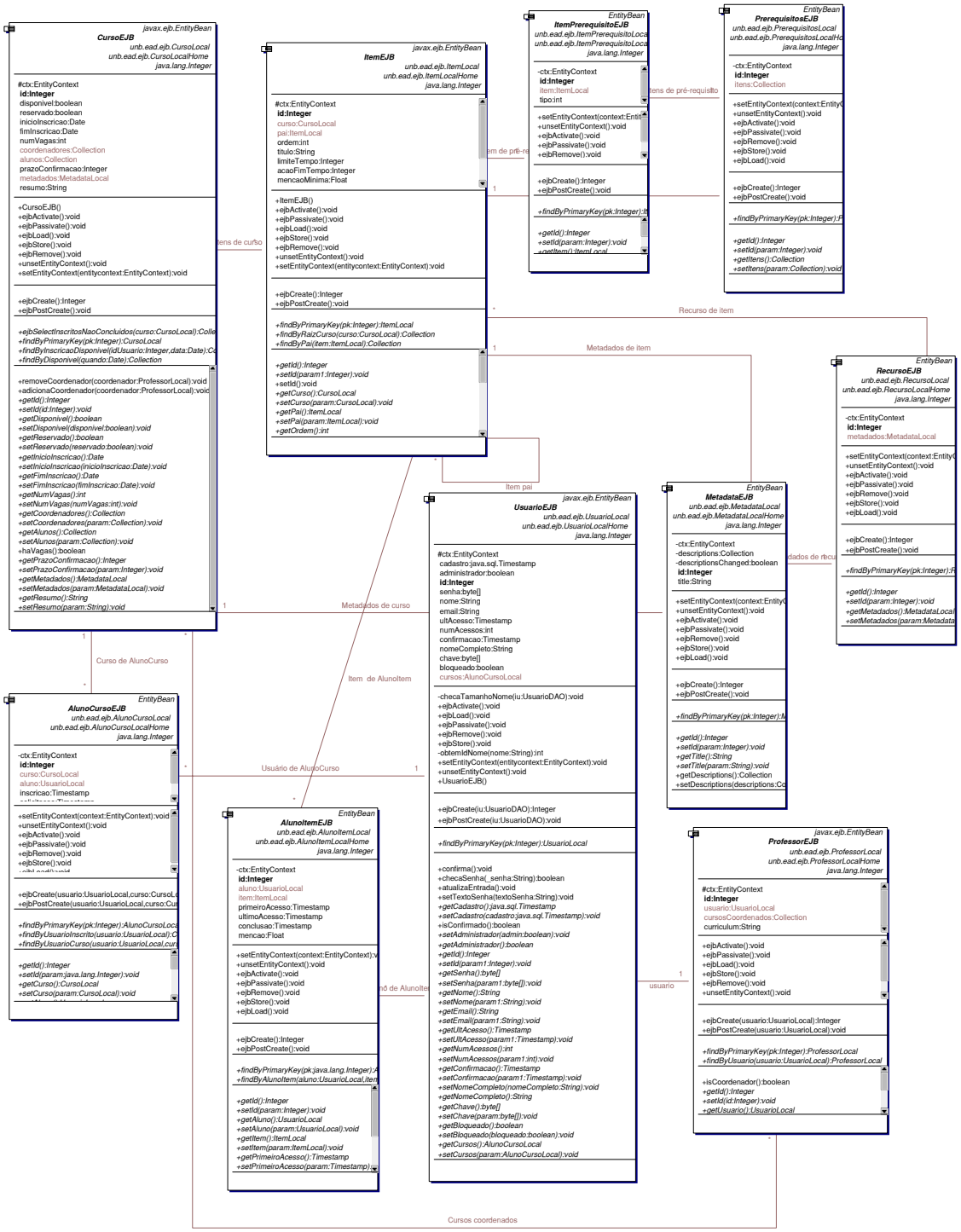

Figura 39 - Diagrama de classes para os elementos persistentes

Na figura 39, vê-se os elementos persistentes e suas relações. A partir de tal diagrama foi possível modelar o banco de dados.

Devido à falta de uma ferramenta disponível para a modelagem gráfica de um banco de dados MySQL, não há um diagrama de banco de dados. Entretanto, o *script* em SQL gerado foi suficientemente comentado, facilitando a visualização do diagrama de entidades e relacionamentos (ERD) correspondentes. Tal *script* se encontra no Anexo I.

Entretanto, não são necessários apenas os componentes relacionados à persistência para o desenvolvimento completo de uma aplicação em três camadas. Assim, utilizando o *design pattern Data Access Object*, foram criadas fábricas de objetos e também os componentes relacionados à tomada de decisões e processamento dos dados. O diagrama de classes correspondente a estes componentes está mostrado abaixo:

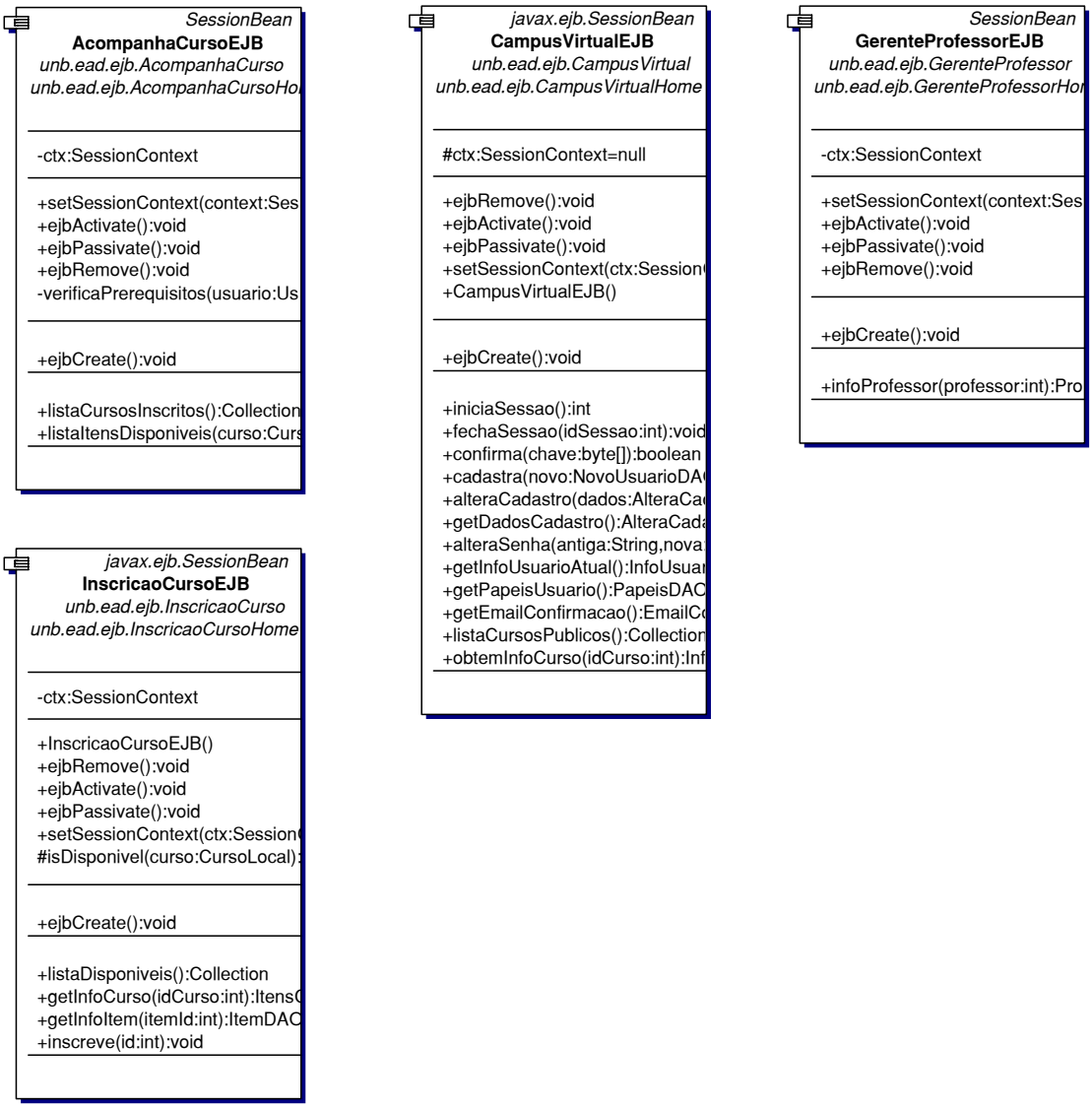

Figura 40 - Diagrama de classes de negócio

Estes componentes são os quais a camada de apresentação possui contato, definindo, assim, um contrato bem definido.

Na forma com a qual tais componentes foram definidos, cada um é responsável por um ou mais grupos de casos de uso, facilitando, assim, seu desenvolvimento e manutenção.

Para cada grupo de classes de acesso de dados (*data access object*) foi criado um pacote, contendo, cada um, uma fábrica de objetos (*factory*).

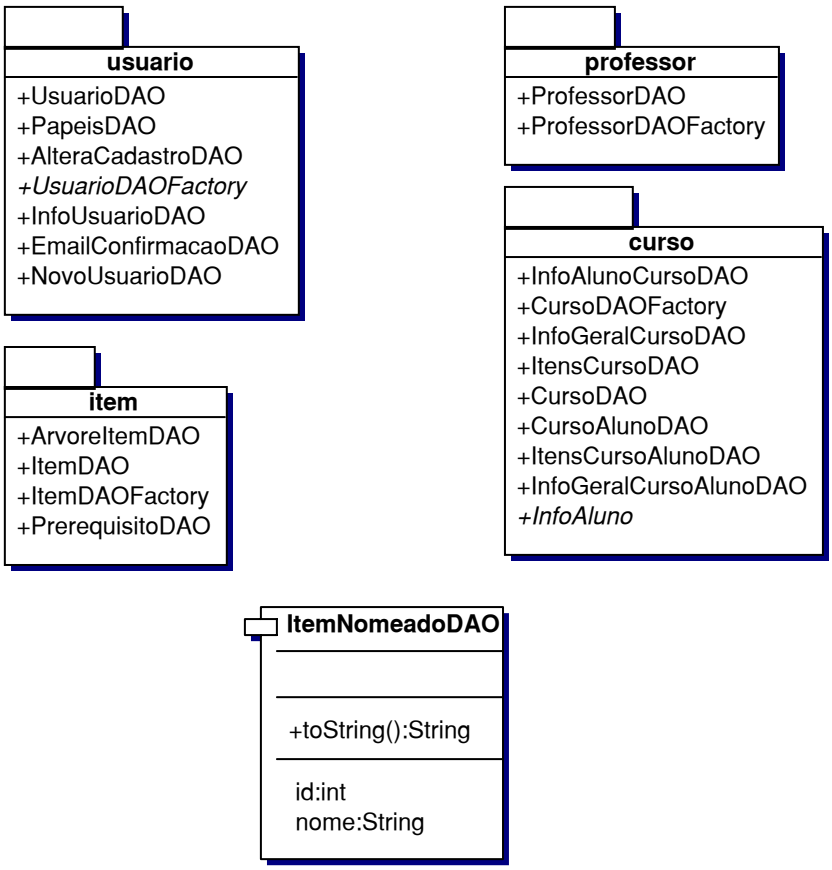

Figura 41 - Pacotes e classes sobre unb.ead.dao

Pela ilustração acima, é possível ver todos os pacotes presentes em *unb.ead.dao*, além da classe *ItemNomeadoDAO*, utilizada para representar itens que se caracterizam por um nome e um valor.

5.3. CAMADA DE APRESENTAÇÃO

Para o desenvolvimento da camada de apresentação, foi utilizado o *JStateMachine* com as alterações já descritas anteriormente.

Assim, a primeira etapa do desenvolvimento foi a utilização do modelo de casos de uso para desenhar os mapas de estado.

Como um único diagrama de estados para toda a aplicação se tornaria ilegível, este diagrama foi dividido em várias etapas, sendo construído a partir dos de mais alto nível.

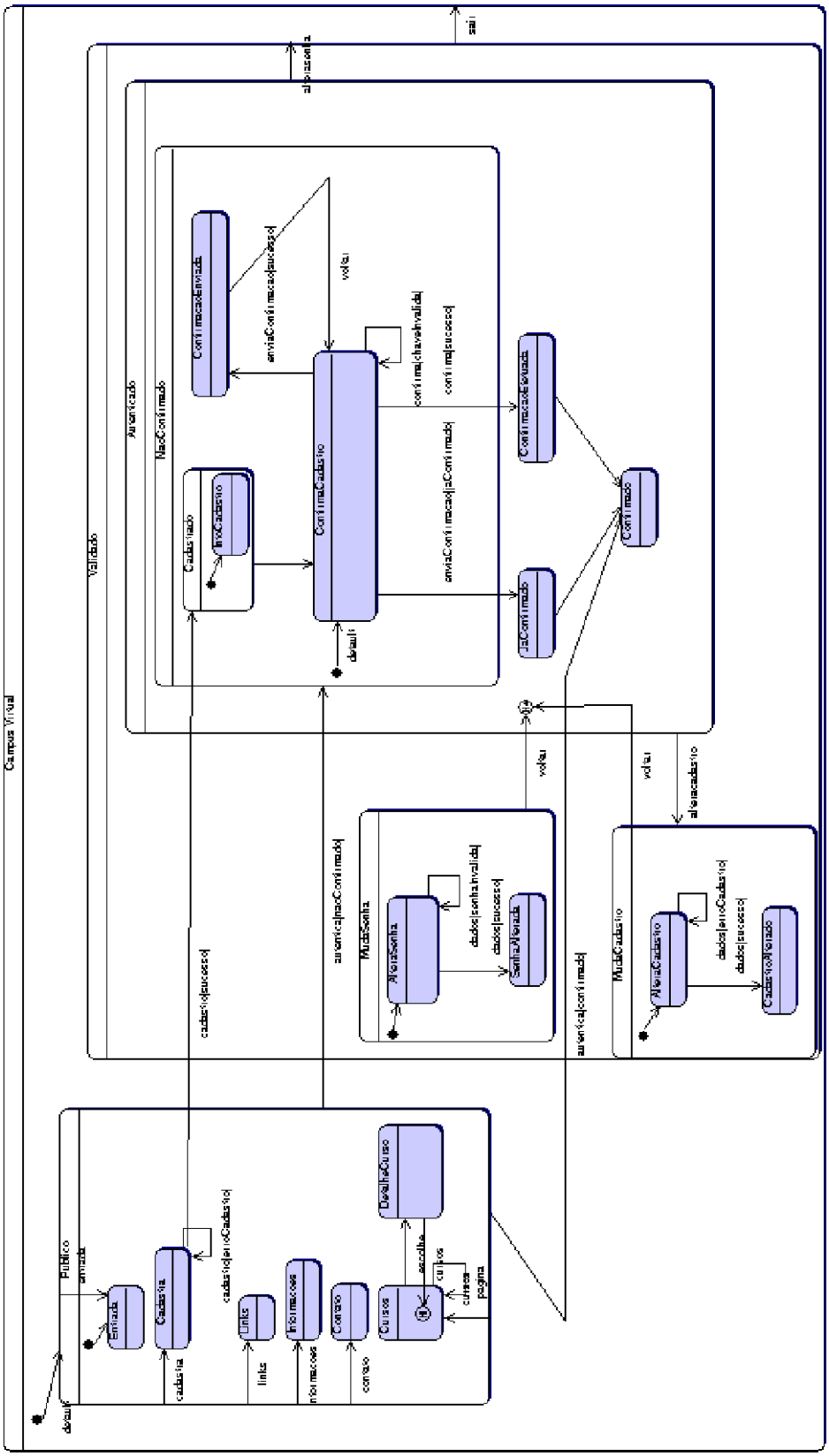

Figura 42 - Diagrama de alto nível e autenticação do usuário.

Na Figura 42 acima, pode-se visualizar o diagrama de mais alto nível da aplicação, e a parte que contém os estados referentes ao cadastramento e autenticação de usuário, além da parte pública do *site*. Como se pode observar, nem todos os estados correspondem a uma página a ser apresentada. Nos estados em que há um estado inicial, e uma transição *default*, o estado que será apresentado corresponde ao destino da transição.

Também nesta etapa é possível fazer um teste da navegabilidade, utilizando o controlador padrão *SimpleController*, que efetua uma transição para o primeiro estado destino especificado, através da geração automática de interfaces presente no *JStateMachine*.

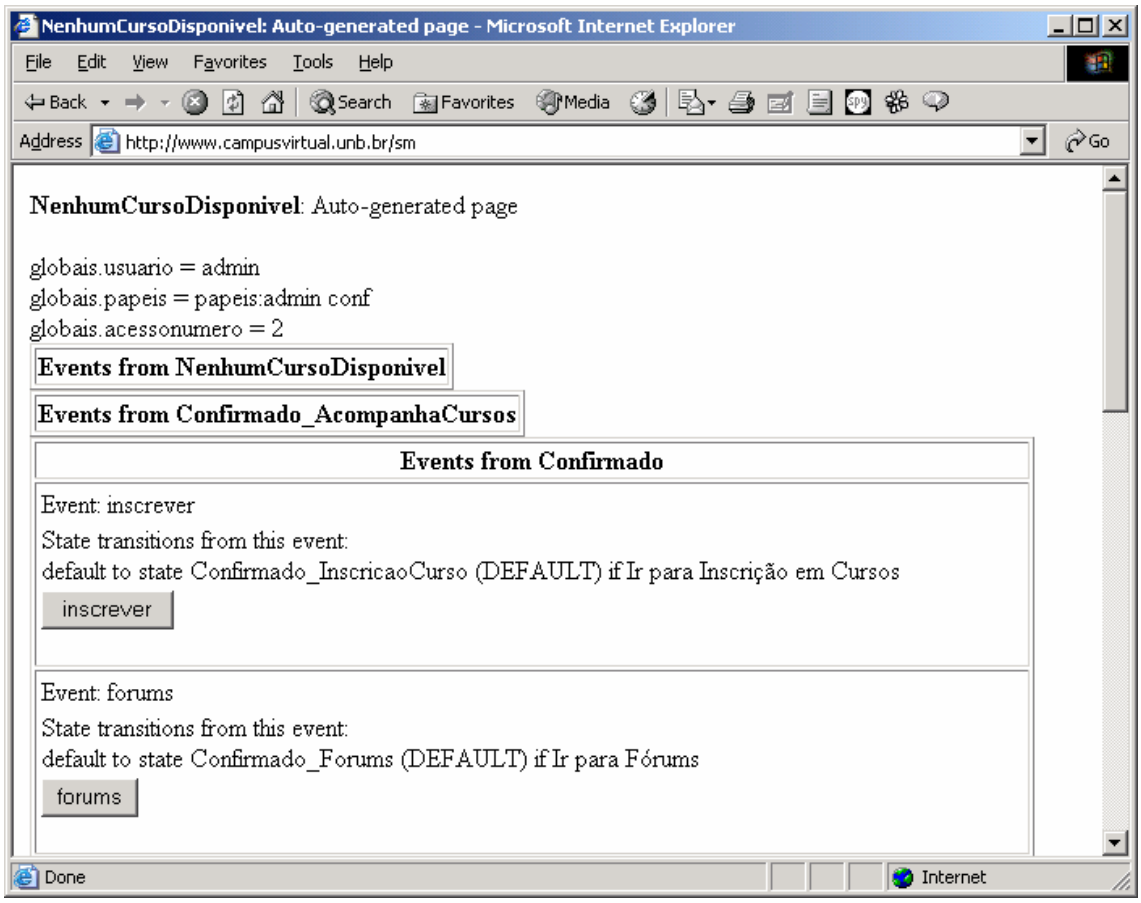

Figura 43 - Exemplo de interface gerada automaticamente (AutoView)

Também durante o desenvolvimento destes estados mais externos foi feito o projeto do *layout* da página. Todas as páginas foram montadas em tecnologia JSP.

O conteúdo das páginas pode ou não ser gerado dinamicamente, e para se adequar a essa situação o *layout* principal do documento foi segmentado em três partes: Topo (*top*), Meio (*body*) e Rodapé (*bottom*). Essas camadas são reunidas por meio de *tags* `<@include>` que compõem o documento final para exibição.

O topo contém quase todas as informações necessárias para a formatação do layout do documento, sobrando um a pequena porção, basicamente o fechamento das tags, ao rodapé da página, isto simplifica e agiliza quaisquer mudanças de layout a serem efetuadas no

documento. O topo carrega o menu de navegação apropriado também por intermédio de uma *tag* `<@include>` e disponibiliza algumas estatísticas e dados do usuário, além de mostrar imagens e conteúdo estático.

O corpo do documento é onde se encontram as informações propriamente ditas, referentes ao estado atual do sistema, nesse ponto é que o conteúdo pode ser estático ou dinâmico, sendo que a formatação de exibição permanece inalterada, devido a utilização de *Cascade Style Sheets* (CSS) para tal. Todo o processamento do conteúdo e transições é executado no servidor, cabendo ao cliente apenas a exibição da *interface* e a interação do usuário com o sistema. Todo o conteúdo é carregado em uma camada, do tipo , no documento de modo a preservar o layout e a formatação da página.

O rodapé do documento, que também é incluído pelo corpo do documento, pode ser usado para exibir informações sobre o projeto bem como exibir estatísticas, mensagens ou sugestões.

Em uma primeira etapa composição da *interface* foi feita de maneira simples e enxuta e um modelo de formulário pode ser visualizado na Figura 44 abaixo.

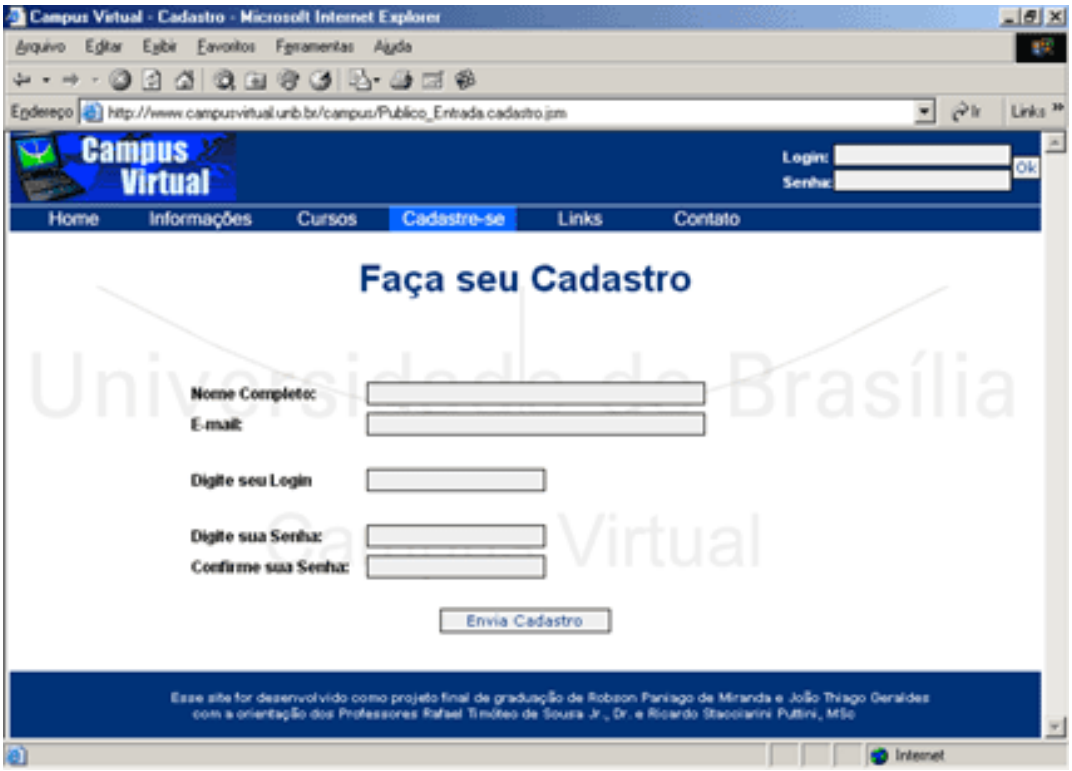

Figura 44 - Formulário de Cadastro do Campus Virtual

Já em uma etapa posterior, o layout e a diagramação do conteúdo foram alterados, mantendo o conteúdo já existente, implementação que foi feita rapidamente, despendendo se tempo apenas para a produção do layout. Nas figuras abaixo estão alguns exemplos de telas do Campus Virtual

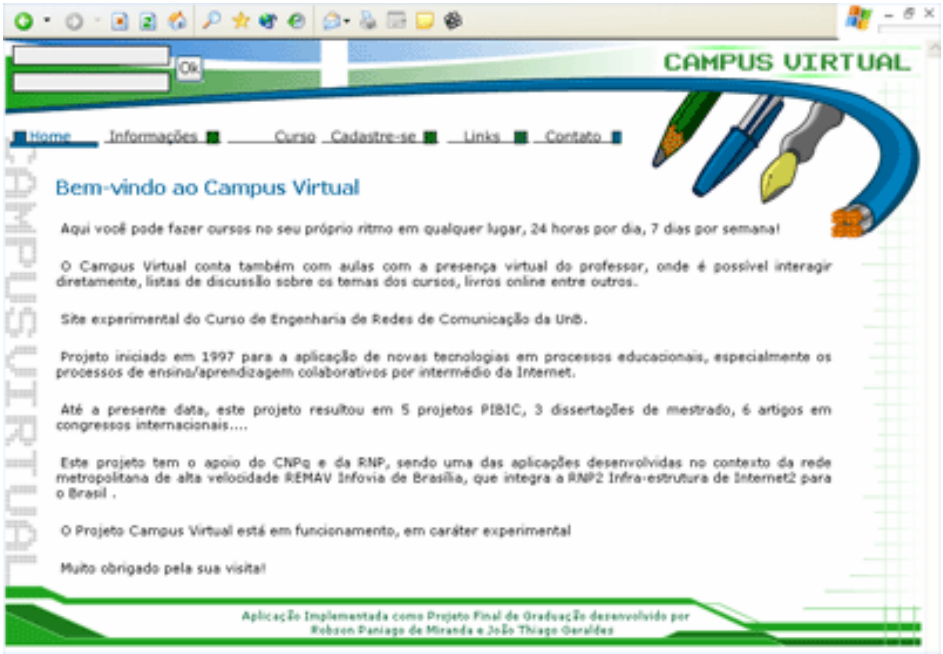

Figura 45 - Tela Inicial do Campus Virtual

O Site do Campus foi concebido para uma resolução de 800x600 e não carrega as barras de navegação, com o intuito de evitar cliques nos botões avançar, voltar e atualizar, que não funcionam na aplicação, sendo toda a navegação pelos estados amarrada na página.

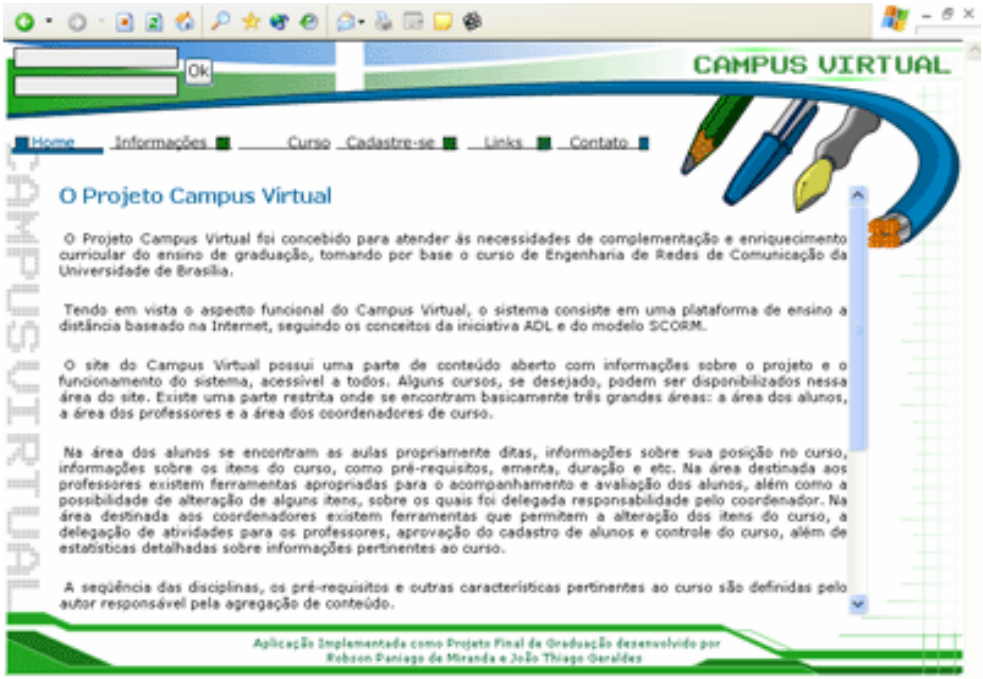

Figura 46 - Página de Informações

Aqui pode se observar a barra de rolagem de conteúdo dentro do documento, preservando o layout

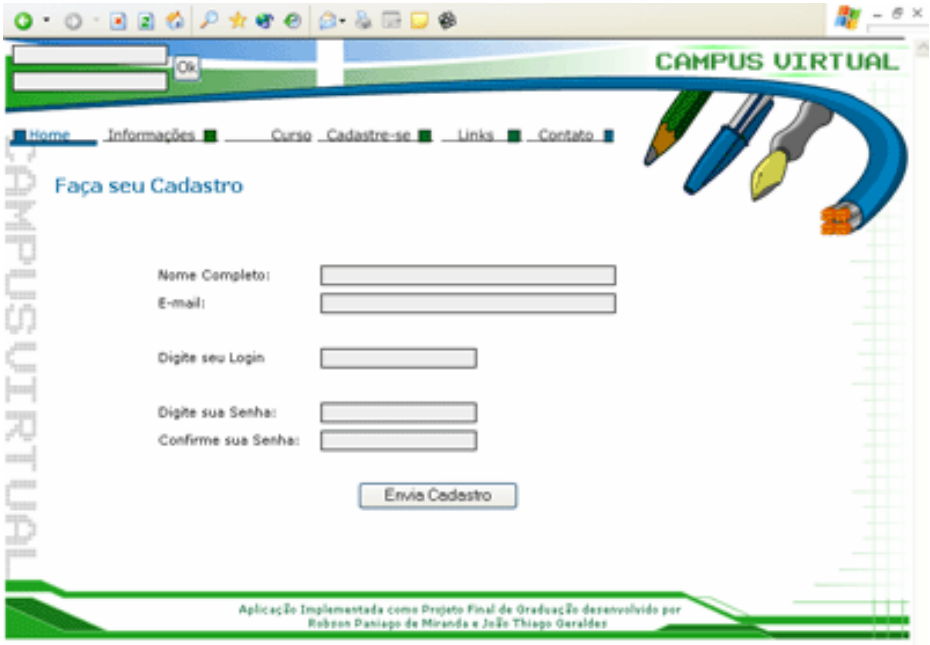

Figura 47 - Novo Formulário de Cadastro

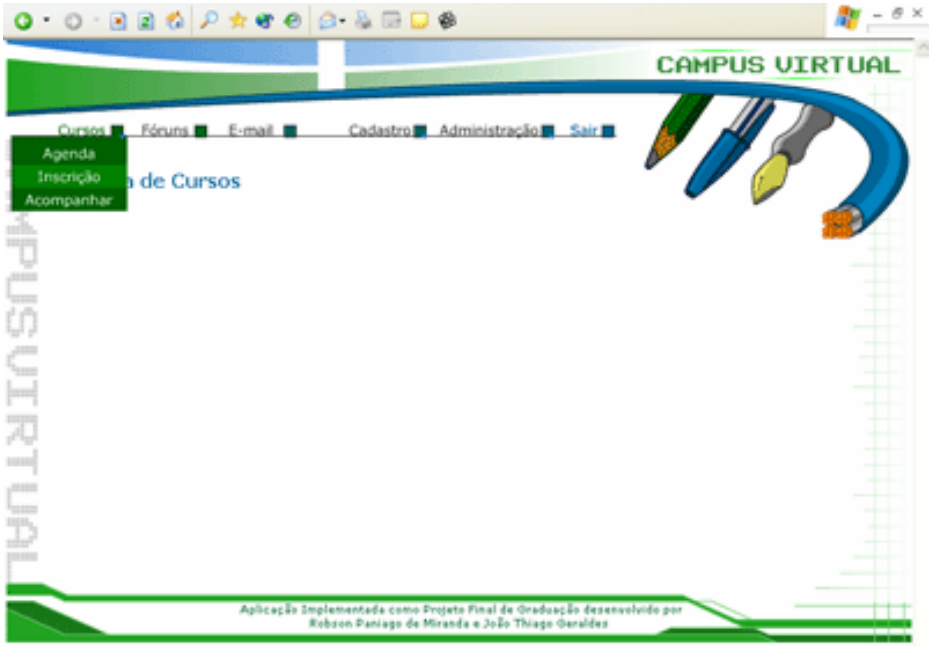

Figura 48 - Página Inicial da Área Restrita

Na Área restrita pode se observar que os menus mudam e que o formulário de login some, ou seja, um novo topo de documento é carregado para a estrutura em questão.

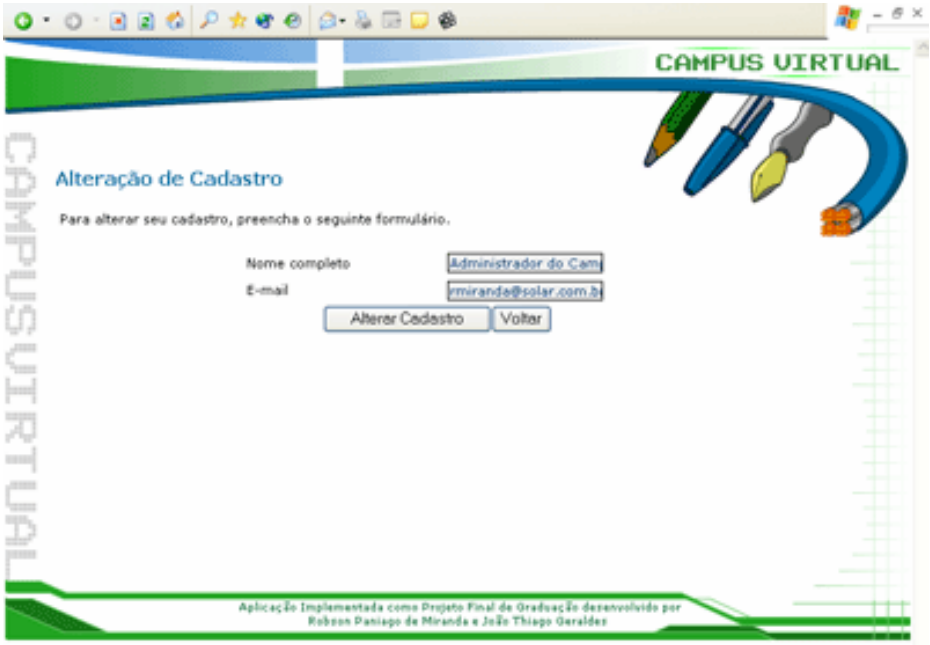

Figura 49 - Formulário de Alteração de Cadastro

No formulário acima pode se notar a inexistência do menu de navegação, sendo este desnecessário ao atual estado da aplicação.

A composição das partes do documento é feita de forma adequada ao atual estado do sistema, existindo vários tipos de topos e rodapés diferentes para tal, se adequando a função e estado do sistema, além do fato de existirem também componentes diferenciados para as áreas restrita e aberta.

O menu de navegação foi desenvolvido utilizando um *JavaScript,* por questões puramente estéticas, de forma que este pudesse dispor maiores recursos e opções de forma simples e bonita, como itens *drop down,* alteração de imagens e destaque de texto.

A utilização das folhas de estilo (CSS) foi bastante providencial no que diz respeito a composição do documento, limitando os dados introduzidos no documento a *tags* HTML simples, como <h1> ou <p> por exemplo, sem informações adicionais no dentro destas, o que facilita a formatação estética de conteúdos gerados dinamicamente.

Foram utilizados nas páginas tags JSP personalizadas, com o intuito de proteger a integridade do sistema e dos dados que trafegam por este. Estas tags foram usadas para substituir as tags padrão de formulário, por exemplo a tag <input type = “text” name = “nome”> foi substituída por <jsm:text name="nome">.

Em seguida, os controladores e ações de entrada e saída foram definidos. Algumas transições mais básicas continuaram utilizando o controlador básico, e as ações de saída foram definidas basicamente para remover valores de *ISharedData*.

Os passos seguintes foram os mapas de estados relativos aos estados de níveis mais baixos.

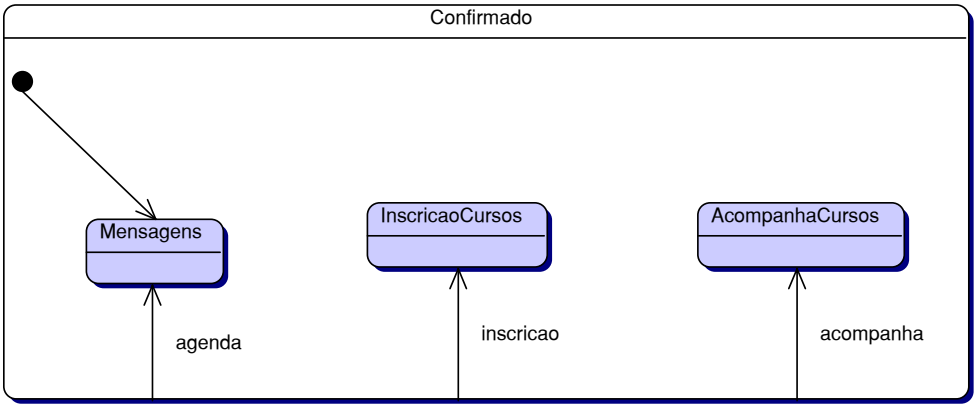

Figura 50 - Mapa de estados para o estado Confirmado

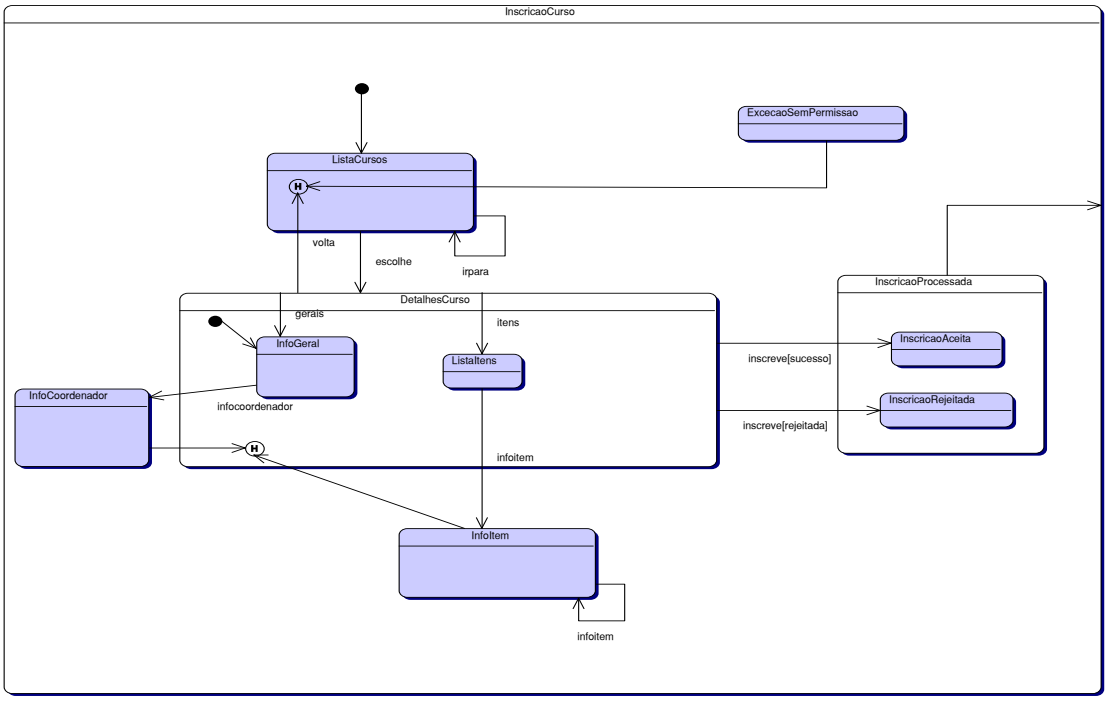

Figura 51 - Estados para Inscrição em Curso

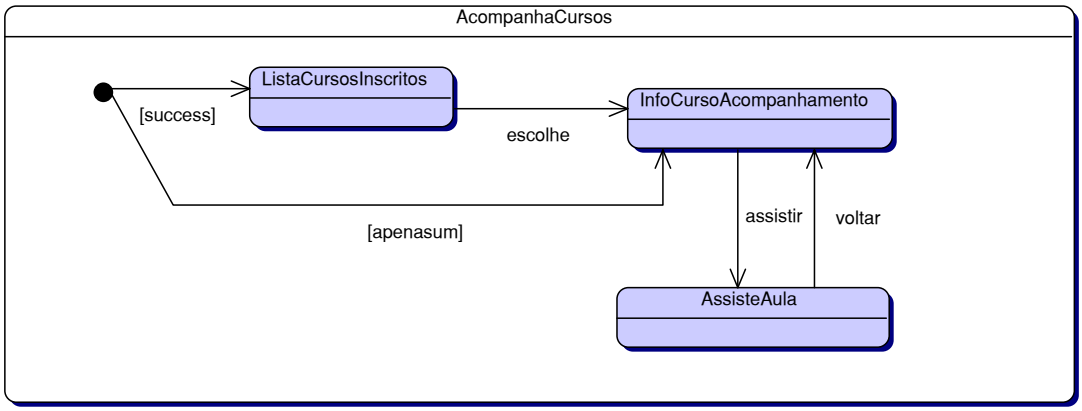

Figura 52 - Estados para Acompanhamento de Curso

Após o desenvolvimento destas estruturas, também foram feitos os controladores e o arquivo de configuração, conforme mostrado no Anexo I.

Para a importação de cursos, foi desenvolvido um aplicativo *stand-alone*, que possui como parâmetro um arquivo no formato *SCORM Content Agregation Model*. O aplicativo procura pelo arquivo que descreve a estrutura, após obter uma representação abstrata da mesma, a envia ao componente existente na camada de negócio, que é responsável pela conversão em entidades correspondentes e seu respectivo armazenamento no banco de dados. O diagrama de classes está mostrado abaixo:

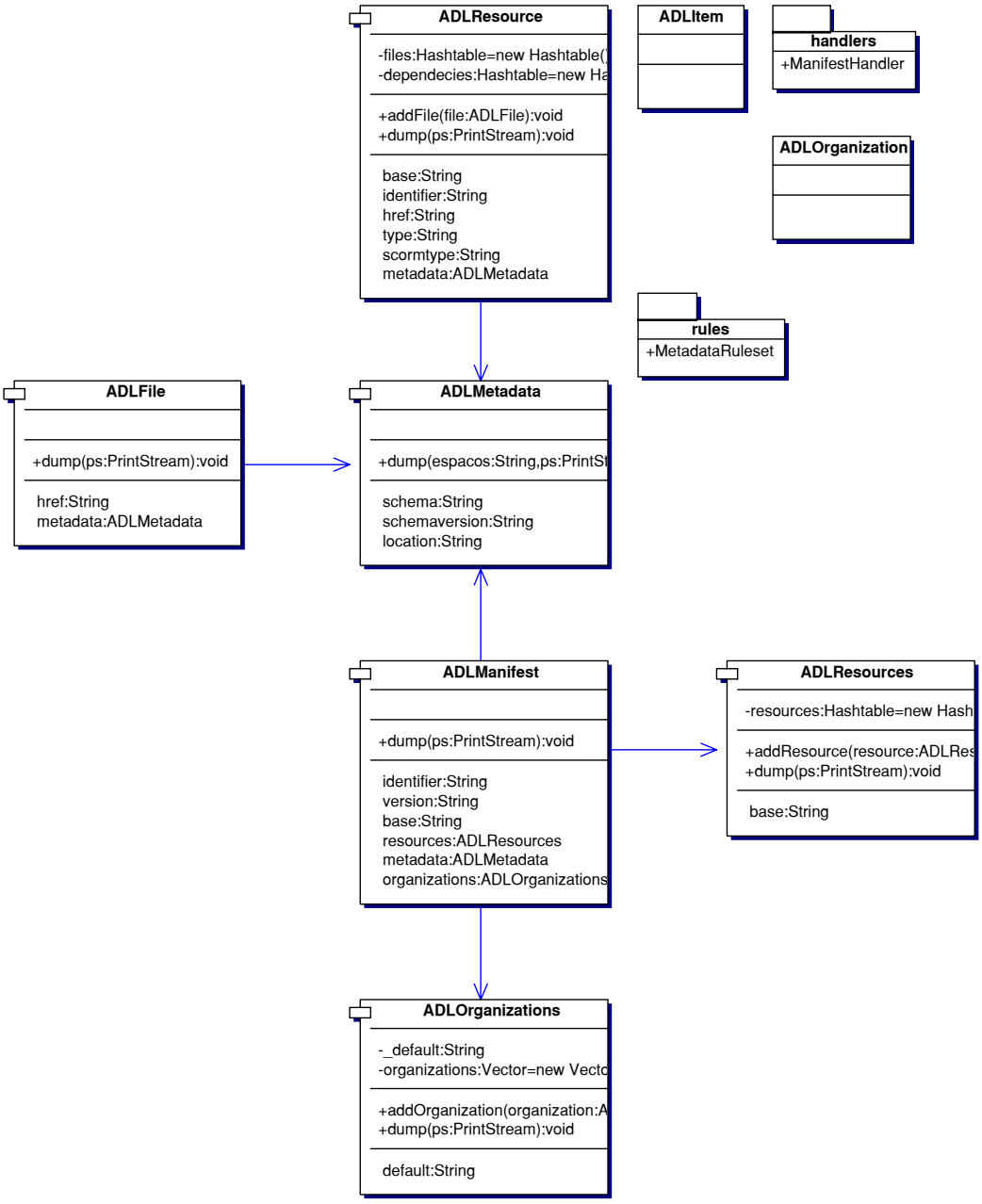

Figura 53 - Classes para representação abstrata de uma estrutura SCORM

Em todas as classes acima foi acrescentado um método *dump* para facilitar o diagnóstico, cuja função é mostrar em forma *string* o conteúdo do objeto.

Além deste pacote, *unb.ead.scorm*, há também os pacotes *unb.ead.scorm.rules*, que contém classes de regras utilizadas pelo analisador do arquivo, o *Jakarta Digester*; e o pacote *unb.ead.scorm.handlers*, contendo as classes responsável pela manipulação da estrutura hierárquica.

Após a análise do arquivo de descrição, os recursos nele descritos são armazenados no banco de dados, e, caso toda a importação ocorra com sucesso, o processo é considerado concluído e o curso já estará disponível para acesso pelos alunos.

6. CONCLUSÃO

As aplicações em arquitetura de três camadas podem ser enquadradas como o próximo passo na evolução das aplicações cliente-servidor ao redor do mundo. Principalmente devido ao fato destas serem muito mais eficientes e seguras que as aplicações tradicionais, pois cabe ao cliente apenas o processamento da *interface* e entrada de requisições, o que faz com que a aplicação seja facilmente distribuída e gerenciada.

A utilização de padrões J2EE imprime uma portabilidade impressionante ao código das aplicações, que podem rodar em qualquer plataforma compatível com os padrões de J2EE. A integração com componentes EJB torna as aplicações ainda mais complexas e poderosas. Essa tecnologia prepara os sistemas para a disponibilização em plataformas web, necessitando aos clientes apenas um navegador, reduzindo assim custos instalação, treinamento e desenvolvimento.

O objetivo deste projeto foi propor um infra-estrutura adequada para a implementação de qualquer tipo de aplicação utilizando os conceitos de máquina de estados e J2EE aqui expostos e explorados.

Para testes e validação foi implementado um sistema gestor de conteúdo para ensino pela Internet que se mostrou extremamente satisfatório e eficiente nos resultados aos quais se propôs. Cabe ressaltar a adequação da arquitetura aqui proposta para sistemas de grande porte devido a tecnologia e equipamentos utilizados. Infelizmente, não puderam ser executados testes com alto volume de processamento e acessos para comprovar essa adequação, devido ao estado no qual se encontra a aplicação.

Com isso, o projeto em questão entrou também no âmbito do ensino a distância, implementando características e conceitos inovadores visando solucionar problemas a muito conhecidos, que tornavam esse tipo de aplicação muito pouco difundida.

Como o conhecimento é algo bastante evolutivo, necessita de constantes atualizações. Quanto mais dinâmicas essas renovações forem, mais eficientes se tornam os métodos de ensino. Nesse quadro o conceito de ensino a distância, aplicado no Projeto Campus Virtual, se encaixa perfeitamente, vindo complementar os métodos tradicionais de ensino.

A intenção do sistema aqui construído é se incorporar ao currículo do curso de Engenharia de Redes de Comunicação da UnB, como forma de disponibilizar cursos de extensão e cursos complementares aos ministrados em sala de aula.

O projeto Campus Virtual vem dar continuidade a um projeto de iniciação científica, utilizado no Curso de Engenharia de Redes de Comunicação da Universidade de Brasília.

A filosofia do Software Livre vem inserir o projeto em uma posição de vanguarda dentro das novas tendências de desenvolvimento de sistemas, adotada amplamente em todo o mundo. A tecnologia empregada no projeto é considerada de ponta, estando mais do que adequada para a operacionalização do mesmo.

7. BIBLIOGRAFIA

[1] FIELDING, R., IRVINE, U.C, GETTYS, J. *Hypertext Transfer Protocol -- HTTP/1.1,* RFC 2616, Junho de 1999.

[2] BERNERS-LEE, T., FIELDING, R. AND H. FRYSTYK, *Hypertext Transfer Protocol -- HTTP/1.0*, RFC 1945, Maio 1996.

[3] FREED, N. AND N. BORENSTEIN, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC 2045, Novembro de 1996.

[4] FREED, N. AND N. BORENSTEIN, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, RFC 2046, Novembro de 1996.

[5] BERNERS-LEE, T., FIELDING, R. AND L. MASINTER, *Uniform Resource Identifiers (URI): Generic Syntax and Semantics*, RFC 2396, Agosto de 1998.

[6] ALUR, DEEPAK et al. *Core J2EE (TM) patterns: As melhores práticas e estratégias de design*; Tradução de Altair Dias et al; Rio de Janeiro, Campus, 2002.

[7] CAY S. HORSTMANN E GARY CORNELL; Core Java 2 - Volume 1 - Fundamentos; 1a. Edição, Makron Books, 2000

[8] HALL, MARTY*, Core Servlets And Javaserver Pages*; 1st Edition, Prentice-Hall, 2000

[9] GRIFFITH, ARTHUR, *JAVA, XML, and JAXP*, John Wiley Computer, 1 Edição, 2002

[10] KOCHMER, CASEY, FRANDSEN, ERICA, *JSP and XML integrating XML and web services in your JSP application*, SAMS, 1st Edition, 2002

[11] ROMAN, ED , AMBLER, SCOTT W., JEWELL, TYLER, MARINESCU, FLOYD, *Mastering Enterprise JavaBeans (2nd Edition),* John Wiley & Sons; 2nd edition, 2001

[12] *MySQL.org,* http://www.mysql.org.

[13] *JBoss LLC Inc.,* http://www.jboss.org

[14] JAVA COMMUNITY PROCCESS, *JSR-153 – Java Spcifications Request,* http://jcp.org/jsr/detail/153.jsp,

[15] *Diagrama Mapas de Estados (Statechart Diagram),* http://www.npdi.dcc.ufmg.br/membros/rabelo/visao/tbd/introducao.htm

[16] O`BYRNE, BRIAN, *JStateMachine*, http://www.jstatemachine.org

[17] *Server-Side MVC Architeture - Part 1*, http://www.uidesign.net/1999/papers/webmvc_part1.html,

[18] SUN MICROSYSTEMS, *Java Core Documentation*, http://java.sun.com

[19] GERALDES, JOÃO THIAGO E MIRANDA, ROBSON PANIAGO DE, *Campus Virtual: Plataforma de Ensino à Distância na Internet,* 1º Congresso de Novas Tecnologias de Ensino de Graduação, UnB, 2002.

[20] GERALDES, JOÃO THIAGO E MIRANDA, ROBSON PANIAGO DE E MARTINS, ALEXANDRE DE ARAÚJO E MOURA, RICARDO NOGUEIRA E DANTAS, ANDREA DE QUADROS, CAMPUS VIRTUAL. *5º Congresso de Iniciação Científica da Universidade de Brasília*; Anais. páginas 228 a 232. Brasília-DF: UnB. 1999

[21] DE SOUSA JR., RAFAEL T. E PUTTINI, RICARDO S., *Ambiente de ensino multimídia via rede para as disciplinas fundamentais do Curso de Graduação em Engenharia de Redes de Comunicação da UnB.* DEG, UnB, 1999.

[22] ADVANCED DISTRIBUTED LEARNING, *Sharable Content Object Reference Model, The SCORM Overview*. ADL Net. 2001.

[23] *ADL Net* – http://www.adlnet.org/

[24] RIBEIRO, G.S.N.; SOUSA JR. R.T.: *Webquest: Protótipo de Um Ambiente de Aprendizagem Colaborativa a Distância Empregado a Internet*. Acessado em 10/04/2002 em http://www.abed.org.br/congresso2001/index.html

8. ANEXO I

8.1. ARQUIVOS DE INSTALAÇÃO DO APLICATIVO NO JBOSS VERSÃO 3.0.

8.1.1. ejb-jar.xml

Este arquivo é utilizado para indicar ao *JBoss* quais são as classes que formam os componentes, os tipos de componentes, os campos que o *container* deve gerenciar para persistência, declarações de segurança e transações.

```
<?xml version="1.0" encoding="ISO-8859-1"?>


<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans
2.0//EN' 'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>
<ejb-jar>
    <enterprise-beans>

        <!-- Alunos inscritos em curso -->

        <entity>
            <display-name></display-name>
            <ejb-name>AlunoCurso</ejb-name>
            <local-home>unb.ead.ejb.AlunoCursoLocalHome</local-home>
            <local>unb.ead.ejb.AlunoCursoLocal</local>
            <ejb-class>unb.ead.ejb.AlunoCursoEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>AlunoCurso</abstract-schema-name>
            <cmp-field>
                <field-name>id</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>inscricao</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>solicitacao</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>conclusao</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>expSolicitacao</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>

            <!-- Esta consulta retorna todos os cursos em que o usuário está
inscrito -->
            <query>
                <query-method>
                    <method-name>findByUsuarioInscrito</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.UsuarioLocal</method-param>
                    </method-params>
```

```
                </query-method>
                <ejb-ql>
                    <![CDATA[SELECT OBJECT(o) FROM AlunoCurso AS o WHERE o.aluno =
?1 AND o.inscricao IS NOT NULL]]>
                </ejb-ql>
            </query>

            <!-- Retorna um objeto que é a relação entre usuário e curso -->
            <query>
                <query-method>
                    <method-name>findByUsuarioCurso</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.UsuarioLocal</method-param>
                        <method-param>unb.ead.ejb.CursoLocal</method-param>
                    </method-params>
                </query-method>
                <ejb-ql>
                    <![CDATA[SELECT OBJECT(o) FROM AlunoCurso AS o WHERE o.aluno =
?1 AND o.curso = ?2]]>
                </ejb-ql>
            </query>
        </entity>

        <!-- Item de um curso -->
        <entity>
            <display-name></display-name>
            <ejb-name>Item</ejb-name>
            <local-home>unb.ead.ejb.ItemLocalHome</local-home>
            <local>unb.ead.ejb.ItemLocal</local>
            <ejb-class>unb.ead.ejb.ItemEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Item</abstract-schema-name>
            <cmp-field>
                <description>Chave primária</description>
                <field-name>id</field-name>
            </cmp-field>
            <cmp-field>
                <description>Ordem dos itens neste nível da
hierarquia</description>
                <field-name>ordem</field-name>
            </cmp-field>
            <cmp-field>
                <description>Título do item, que vai aparecer para o usuário na
lista de cursos.</description>
                <field-name>titulo</field-name>
            </cmp-field>
            <cmp-field>
                <description>Limite de tempo do item</description>
                <field-name>limiteTempo</field-name>
            </cmp-field>
            <cmp-field>
                <description>Ação a tomar ao fim do tempo</description>
                <field-name>acaoFimTempo</field-name>
            </cmp-field>
            <cmp-field>
                <description>Menção mínima que o usuário precisa ter para ser
considerado aprovado neste ítem</description>
                <field-name>mencaoMinima</field-name>
            </cmp-field>
            <cmp-field>
                <description>Indica se um determinado item está ou nao visível para
o usuário.</description>
```

```
                <field-name>visivel</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>
            <query>
                <query-method>
                    <method-name>findByPai</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.ItemLocal</method-param>
                    </method-params>
                </query-method>
                <ejb-ql><![CDATA[SELECT OBJECT(o) FROM Item o WHERE o.pai =
?1]]></ejb-ql>
            </query>
            <query>
                <query-method>
                    <method-name>findByRaizCurso</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.CursoLocal</method-param>
                    </method-params>
                </query-method>
                <ejb-ql><![CDATA[SELECT OBJECT(o) FROM Item o WHERE o.curso = ?1
AND o.pai IS NULL]]></ejb-ql>
            </query>
        </entity>

        <!-- Curso -->
        <entity>
            <display-name></display-name>
            <ejb-name>Curso</ejb-name>
            <local-home>unb.ead.ejb.CursoLocalHome</local-home>
            <local>unb.ead.ejb.CursoLocal</local>
            <ejb-class>unb.ead.ejb.CursoEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Curso</abstract-schema-name>
            <cmp-field>
                <field-name>id</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>disponivel</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>reservado</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>inicioInscricao</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>fimInscricao</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>numVagas</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>prazoConfirmacao</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>resumo</field-name>
            </cmp-field>

            <primkey-field>id</primkey-field>
```

```
            <!-- Esta consulta retorna todos os alunos que ainda não concluiram o
curso, mas estão matriculados -->
            <query>
                <query-method>
                    <method-name>ejbSelectInscritosNaoConcluidos</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.CursoLocal</method-param>
                    </method-params>
                </query-method>
                <ejb-ql>
                    <![CDATA[
                    SELECT OBJECT(o) FROM AlunoCurso AS o WHERE
                    o.curso = ?1 AND o.inscricao IS NOT NULL AND o.conclusao IS NOT
NULL
                    ]]>
                </ejb-ql>
            </query>


            <!-- Consulta para listar os cursos disponíveis para inscrição -->
            <query>
                <query-method>
                    <method-name>findByInscricaoDisponivel</method-name>
                    <method-params>
                        <method-param>java.lang.Integer</method-param>
                        <method-param>java.sql.Date</method-param>
                    </method-params>
                </query-method>
                <ejb-ql>
                    <![CDATA[SELECT OBJECT(o) FROM Curso AS o
                    WHERE o.disponivel AND
                     (o.inicioInscricao IS NULL OR o.inicioInscricao <= ?2) AND
                     (o.fimInscricao    IS NULL OR o.fimInscricao >= ?2)
                    ]]>
                </ejb-ql>
            </query>
            <query>
                <description>Lista os cursos públicos disponíveis para
inscrição.</description>
                <query-method>
                    <method-name>findByDisponivel</method-name>
                    <method-params>
                        <method-param>java.sql.Date</method-param>
                    </method-params>
                </query-method>
                <ejb-ql>
                    <![CDATA[SELECT OBJECT(o) FROM Curso AS o
                      WHERE o.disponivel AND (NOT o.reservado) AND
                       ((o.inicioInscricao IS NULL OR o.inicioInscricao <= ?1) AND
                        (o.fimInscricao    IS NULL OR o.fimInscricao >= ?1))
                    ]]>
                </ejb-ql>
            </query>
        </entity>

        <!-- Recursos disponíveis para os itens de um curso -->
        <entity>
            <display-name></display-name>
            <ejb-name>Recurso</ejb-name>
            <local-home>unb.ead.ejb.RecursoLocalHome</local-home>
            <local>unb.ead.ejb.RecursoLocal</local>
            <ejb-class>unb.ead.ejb.RecursoEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
```

```
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Recurso</abstract-schema-name>
            <cmp-field>
                <description>Chave primária</description>
                <field-name>id</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>
        </entity>

        <!-- Professor -->
        <entity>
            <display-name></display-name>
            <ejb-name>Professor</ejb-name>
            <local-home>unb.ead.ejb.ProfessorLocalHome</local-home>
            <local>unb.ead.ejb.ProfessorLocal</local>
            <ejb-class>unb.ead.ejb.ProfessorEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Professor</abstract-schema-name>
            <cmp-field>
                <field-name>id</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>curriculum</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>
            <query>
                <query-method>
                    <method-name>findByUsuario</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.UsuarioLocal</method-param>
                    </method-params>

                </query-method>
                <ejb-ql>
                    <![CDATA[SELECT OBJECT(p) FROM Professor AS p WHERE p.usuario =
?1 ]]>
                </ejb-ql>
            </query>
        </entity>

        <!-- Usuário -->
        <entity>
            <display-name></display-name>
            <ejb-name>Usuario</ejb-name>
            <local-home>unb.ead.ejb.UsuarioLocalHome</local-home>
            <local>unb.ead.ejb.UsuarioLocal</local>
            <ejb-class>unb.ead.ejb.UsuarioEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Usuario</abstract-schema-name>
            <cmp-field>
                <description>Data de cadastro do usuário</description>
                <field-name>cadastro</field-name>
            </cmp-field>
            <cmp-field>
                <description>Indica se o usuário está bloqueado</description>
                <field-name>bloqueado</field-name>
            </cmp-field>
            <cmp-field>
                <description>Verifica se o usuário é administrador</description>
```

```
            <field-name>administrador</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>id</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>senha</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>nome</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>email</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>ultAcesso</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>numAcessos</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>confirmacao</field-name>
        </cmp-field>
        <cmp-field>
            <description>Obtém o nome completo</description>
            <field-name>nomeCompleto</field-name>
        </cmp-field>
        <cmp-field>
            <field-name>chave</field-name>
        </cmp-field>
        <primkey-field>id</primkey-field>
    </entity>

    <!-- Meta-dados -->
    <entity>
        <description>Representação dos meta-dados de um objeto</description>
        <display-name></display-name>
        <ejb-name>Metadata</ejb-name>
        <local-home>unb.ead.ejb.MetadataLocalHome</local-home>
        <local>unb.ead.ejb.MetadataLocal</local>
        <ejb-class>unb.ead.ejb.MetadataEJB</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Integer</prim-key-class>
        <reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
        <abstract-schema-name>Metadata</abstract-schema-name>
        <cmp-field>
            <description>Identificador dos meta-dados.</description>
            <field-name>id</field-name>
        </cmp-field>
        <cmp-field>
            <description>Título do objeto</description>
            <field-name>title</field-name>
        </cmp-field>
        <primkey-field>id</primkey-field>
    </entity>

    <!-- Pré-requisitos -->
    <entity>
        <display-name>Um conjunto de pré-requisitos de um item</display-name>
        <ejb-name>Prerequisitos</ejb-name>
        <local-home>unb.ead.ejb.PrerequisitosLocalHome</local-home>
        <local>unb.ead.ejb.PrerequisitosLocal</local>
        <ejb-class>unb.ead.ejb.PrerequisitosEJB</ejb-class>
        <persistence-type>Container</persistence-type>
```

```
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Prerequisitos</abstract-schema-name>
            <cmp-field>
                <description>Identificador do pré-requisito.</description>
                <field-name>id</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>
        </entity>
        <entity>
            <description>Componente parte do relacionamento n-n entre pré-requisito
e item.</description>
            <display-name></display-name>
            <ejb-name>ItemPrerequisito</ejb-name>
            <local-home>unb.ead.ejb.ItemPrerequisitoLocalHome</local-home>
            <local>unb.ead.ejb.ItemPrerequisitoLocal</local>
            <ejb-class>unb.ead.ejb.ItemPrerequisitoEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>ItemPrerequisito</abstract-schema-name>
            <cmp-field>
                <field-name>id</field-name>
            </cmp-field>
            <cmp-field>
                <description>Tipo de pré-requisito:
1 = Passed
2 = Completed
3 = Browsed
4 = Failed
5 = Not attempted
6 = Incomplete</description>
                <field-name>tipo</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>
        </entity>

        <!-- Situação de um item em relação a um aluno -->
        <entity>
            <display-name></display-name>
            <ejb-name>AlunoItem</ejb-name>
            <local-home>unb.ead.ejb.AlunoItemLocalHome</local-home>
            <local>unb.ead.ejb.AlunoItemLocal</local>
            <ejb-class>unb.ead.ejb.AlunoItemEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>AlunoItem</abstract-schema-name>
            <cmp-field>
                <field-name>id</field-name>
            </cmp-field>
            <cmp-field>
                <description>Data/hora do primeiro acesso a este item</description>
                <field-name>primeiroAcesso</field-name>
            </cmp-field>
            <cmp-field>
                <description>Data/hora do último acesso a este item.</description>
                <field-name>ultimoAcesso</field-name>
            </cmp-field>
            <cmp-field>
                <description>Data/hora da conclusão deste item.</description>
                <field-name>conclusao</field-name>
```

```
            </cmp-field>
            <cmp-field>
                <description>Menção dada ao aluno (normalizada de 0 a 100), ou null
se não foi dada nenhuma menção.</description>
                <field-name>mencao</field-name>
            </cmp-field>
            <primkey-field>id</primkey-field>
            <query>
                <description>Encontra o objeto correspondente a um relacionamento
entre um aluno e um item</description>
                <query-method>
                    <method-name>findByAlunoItem</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.UsuarioLocal</method-param>
                        <method-param>unb.ead.ejb.ItemLocal</method-param>
                    </method-params>
                </query-method>
                <ejb-ql>SELECT OBJECT(o) FROM AlunoItem AS o WHERE o.aluno = ?1 AND
o.item = ?2</ejb-ql>
            </query>
        </entity>

        <!-- Número serial -->
        <entity>
            <display-name></display-name>
            <ejb-name>Serial</ejb-name>
            <local-home>unb.ead.ejb.SerialLocalHome</local-home>
            <local>unb.ead.ejb.SerialLocal</local>
            <ejb-class>unb.ead.ejb.SerialEJB</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.String</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-version>2.x</cmp-version>
            <abstract-schema-name>Serial</abstract-schema-name>
            <cmp-field>
                <field-name>tabela</field-name>
            </cmp-field>
            <cmp-field>
                <field-name>serial</field-name>
            </cmp-field>
            <primkey-field>tabela</primkey-field>
        </entity>

        <session>
            <display-name></display-name>
            <ejb-name>SessaoSerial</ejb-name>
            <local-home>unb.ead.ejb.SessaoSerialLocalHome</local-home>
            <local>unb.ead.ejb.SessaoSerialLocal</local>
            <ejb-class>unb.ead.ejb.SessaoSerialEJB</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
            <env-entry>
                <env-entry-name>tentativas</env-entry-name>
                <env-entry-type>java.lang.Integer</env-entry-type>
                <env-entry-value>5</env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>tamBloco</env-entry-name>
                <env-entry-type>java.lang.Integer</env-entry-type>
                <env-entry-value>5</env-entry-value>
              </env-entry>
        </session>
        <session>
            <display-name></display-name>
            <ejb-name>CampusVirtual</ejb-name>
```

```
            <home>unb.ead.ejb.CampusVirtualHome</home>
            <remote>unb.ead.ejb.CampusVirtual</remote>
            <ejb-class>unb.ead.ejb.CampusVirtualEJB</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
        <session>
            <display-name></display-name>
            <ejb-name>InscricaoCurso</ejb-name>
            <home>unb.ead.ejb.InscricaoCursoHome</home>
            <remote>unb.ead.ejb.InscricaoCurso</remote>
            <ejb-class>unb.ead.ejb.InscricaoCursoEJB</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
        <session>
            <display-name></display-name>
            <ejb-name>GerenteProfessor</ejb-name>
            <home>unb.ead.ejb.GerenteProfessorHome</home>
            <remote>unb.ead.ejb.GerenteProfessor</remote>
            <ejb-class>unb.ead.ejb.GerenteProfessorEJB</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
        <session>
            <display-name></display-name>
            <ejb-name>AcompanhaCurso</ejb-name>
            <home>unb.ead.ejb.AcompanhaCursoHome</home>
            <remote>unb.ead.ejb.AcompanhaCurso</remote>
            <ejb-class>unb.ead.ejb.AcompanhaCursoEJB</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
    </enterprise-beans>


    <relationships>
        <!-- Relação entre alunos e cursos -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Curso de AlunoCurso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoCurso to Curso</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>AlunoCurso</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>curso</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Curso to AlunoCurso</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Curso</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>alunos</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
```

```
            <description></description>
            <ejb-relation-name>Usuário de AlunoCurso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Usuario to AlunoCurso</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Usuario</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>cursos</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoCurso to Usuario</ejb-
relationship-role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>AlunoCurso</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>aluno</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Coordenadores de curso -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Cursos coordenados</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Curso to Professor</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>Curso</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>coordenadores</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Professor to Curso</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>Professor</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>cursosCoordenados</cmr-field-name>

                </cmr-field>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Relacionamentos entre os cursos e seus itens -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Itens de curso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
```

```
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>curso</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Curso</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Auto-relacionamento entre os itens -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Item pai</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>pai</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Relacionamento entre recursos e itens -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Recurso de item</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>recurso</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Recurso</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
```

```
        <!-- Usuário correspondente a um professor -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>usuario</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Professor UNIDIRECTIONAL</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Professor</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>usuario</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Professor</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Usuario</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Pré-requisitos -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Pré-requisito de item</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>preRequisitos</cmr-field-name>
                    <cmr-field-type>java.util.Collection</cmr-field-type>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>Prerequisitos</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Itens de pré-requisito</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Prerequisitos UNIDIRECTIONAL</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Prerequisitos</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>itens</cmr-field-name>
                    <cmr-field-type>java.util.Collection</cmr-field-type>
                </cmr-field>
            </ejb-relationship-role>
```

```
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Prerequisitos</ejb-
relationship-role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>ItemPrerequisito</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Item de item de pré-requisito.</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>ItemPrerequisito UNIDIRECTIONAL</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>ItemPrerequisito</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>item</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL ItemPrerequisito</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Relacionamentos entre Aluno e Item -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Aluno de AlunoItem</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoItem UNIDIRECTIONAL</ejb-
relationship-role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
                    <ejb-name>AlunoItem</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>aluno</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL AlunoItem</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Usuario</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Item  de AlunoItem</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoItem UNIDIRECTIONAL</ejb-
relationship-role-name>
                <multiplicity>Many</multiplicity>
                <relationship-role-source>
```

```
                    <ejb-name>AlunoItem</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>item</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL AlunoItem</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Metadados -->
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Metadados de item</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Item</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>metadados</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Metadata</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <description></description>
            <ejb-relation-name>Metadados de recurso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Recurso UNIDIRECTIONAL</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Recurso</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>metadados</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Recurso</ejb-
relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Metadata</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <description></description>
```

```
            <ejb-relation-name>Metadados de curso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Curso UNIDIRECTIONAL</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Curso</ejb-name>
                </relationship-role-source>
                <cmr-field>
                    <cmr-field-name>metadados</cmr-field-name>
                </cmr-field>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Curso</ejb-relationship-
role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>Metadata</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
    </relationships>

    <assembly-descriptor>
        <security-role>
            <role-name>autenticado</role-name>
        </security-role>
        <method-permission>
            <role-name>autenticado</role-name>
            <method>
                <ejb-name>CampusVirtual</ejb-name>
                <method-name>*</method-name>
            </method>
            <method>
                <ejb-name>InscricaoCurso</ejb-name>
                <method-name>*</method-name>
            </method>
            <method>
                <ejb-name>GerenteProfessor</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <unchecked/>
            <method>
                <ejb-name>CampusVirtual</ejb-name>
                <method-name>create</method-name>
            </method>
            <method>
                <ejb-name>CampusVirtual</ejb-name>
                <method-name>cadastra</method-name>
                <method-params>
                    <method-param>unb.ead.dao.usuario.NovoUsuarioDAO</method-param>
                </method-params>
            </method>
            <method>
                <ejb-name>CampusVirtual</ejb-name>
                <method-name>fechaSessao</method-name>
                <method-params>
                    <method-param>int</method-param>
                </method-params>
            </method>
            <method>
                <ejb-name>CampusVirtual</ejb-name>
                <method-name>listaCursosPublicos</method-name>
```

```
            <method-params>
            </method-params>
        </method>
        <method>
            <ejb-name>CampusVirtual</ejb-name>
            <method-name>obtemInfoCurso</method-name>
            <method-params>
                <method-param>int</method-param>
            </method-params>
        </method>
        <method>

            <ejb-name>AlunoCurso</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Curso</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Recurso</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Item</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Professor</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Usuario</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Serial</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Metadata</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>Prerequisitos</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>ItemPrerequisito</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>AlunoItem</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>SessaoSerial</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>AcompanhaCurso</ejb-name>
            <method-name>*</method-name>
        </method>
    </method-permission>
    <container-transaction>
```

```
    <method>
        <ejb-name>Serial</ejb-name>
        <method-name>getSerialeIncrementa</method-name>

        <method-params>
            <method-param>int</method-param>
        </method-params>
    </method>
    <method>
        <ejb-name>Serial</ejb-name>
        <method-name>create</method-name>
        <method-intf>LocalHome</method-intf>
        <method-params>
            <method-param>java.lang.String</method-param>
        </method-params>
    </method>
    <trans-attribute>RequiresNew</trans-attribute>
</container-transaction>
<container-transaction>
    <method>
        <ejb-name>CampusVirtual</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Serial</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>AlunoCurso</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Usuario</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Curso</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Professor</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Recurso</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Item</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Metadata</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>Prerequisitos</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>ItemPrerequisito</ejb-name>
        <method-name>*</method-name>
    </method>
    <method>
        <ejb-name>AlunoItem</ejb-name>
```

```
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>SessaoSerial</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>InscricaoCurso</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>GerenteProfessor</ejb-name>
            <method-name>*</method-name>
        </method>
        <method>
            <ejb-name>AcompanhaCurso</ejb-name>
            <method-name>*</method-name>
        </method>
        <trans-attribute>Required</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

8.1.2. jbosscmp-jdbc.sql

Este arquivo é utilizado para o mapeamento de objetos para banco de dados relacional, utilizado pelo *plugin* de comunicação entre o *JBoss* e o banco de dados.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE jbosscmp-jdbc>

<jbosscmp-jdbc>

  <defaults>
    <datasource>java:/CampusDS</datasource>
    <datasource-mapping>mySQL</datasource-mapping>
    <create-table>false</create-table>
    <remove-table>false</remove-table>
    <row-locking>false</row-locking>
    <read-only>false</read-only>
    <read-time-out>300</read-time-out>
  </defaults>

  <type-mappings>
    <mapping>
      <java-type>java.sql.Time</java-type>
      <jdbc-type>TIME</jdbc-type>
      <sql-type>DATETIME</sql-type>
    </mapping>
  </type-mappings>

    <enterprise-beans>
        <!-- Número sequencial -->
        <entity>
            <ejb-name>Serial</ejb-name>
            <table-name>t_sequencia</table-name>

            <cmp-field><field-name>tabela</field-name><column-
name>SEQ_NOME</column-name></cmp-field>
            <cmp-field><field-name>serial</field-name><column-
name>SEQ_VALOR</column-name></cmp-field>
        </entity>
```

```
        <!-- Usuário -->
        <entity>
            <ejb-name>Usuario</ejb-name>
            <table-name>t_usuario</table-name>
            <cmp-field><field-name>id</field-name><column-name>USU_ID</column-
name></cmp-field>
            <cmp-field><field-name>nomeCompleto</field-name><column-
name>USU_NOME</column-name></cmp-field>
            <cmp-field><field-name>nome</field-name><column-name>USU_LOGIN</column-
name></cmp-field>
            <cmp-field><field-name>email</field-name><column-
name>USU_EMAIL</column-name></cmp-field>
            <cmp-field><field-name>administrador</field-name><column-
name>USU_ADMINISTRADOR</column-name></cmp-field>
            <cmp-field><field-name>ultAcesso</field-name><column-
name>USU_ULTACESSO</column-name></cmp-field>
            <cmp-field><field-name>numAcessos</field-name><column-
name>USU_NUMACESSOS</column-name></cmp-field>
            <cmp-field><field-name>bloqueado</field-name><column-
name>USU_BLOQUEADO</column-name></cmp-field>
            <cmp-field><field-name>cadastro</field-name><column-
name>USU_CADASTRO</column-name></cmp-field>
            <cmp-field><field-name>confirmacao</field-name><column-
name>USU_CONFIRMACAO</column-name></cmp-field>
            <cmp-field>
                <field-name>senha</field-name>
                <column-name>USU_SENHA</column-name>
                <jdbc-type>BLOB</jdbc-type>
                <sql-type>CHAR(16) BINARY</sql-type>
            </cmp-field>
            <cmp-field>
                <field-name>chave</field-name>
                <column-name>USU_CHAVE</column-name>
                <jdbc-type>BLOB</jdbc-type>
                <sql-type>CHAR(20) BINARY</sql-type>
            </cmp-field>
        </entity>


        <!-- Professor -->
        <entity>
            <ejb-name>Professor</ejb-name>
            <table-name>t_professor</table-name>
            <cmp-field><field-name>id</field-name><column-name>PROF_ID</column-
name></cmp-field>
            <cmp-field><field-name>curriculum</field-name><column-
name>PROF_CURRICULUM</column-name></cmp-field>
        </entity>

        <!-- Curso -->
        <entity>
            <ejb-name>Curso</ejb-name>
            <table-name>t_curso</table-name>
            <cmp-field><field-name>id</field-name><column-name>CUR_ID</column-
name></cmp-field>
            <cmp-field><field-name>disponivel</field-name><column-
name>CUR_DISPONIVEL</column-name></cmp-field>
            <cmp-field><field-name>reservado</field-name><column-
name>CUR_RESERVADO</column-name></cmp-field>
            <cmp-field><field-name>inicioInscricao</field-name><column-
name>CUR_INSCR_INICIO</column-name></cmp-field>
            <cmp-field><field-name>fimInscricao</field-name><column-
name>CUR_INSCR_FIM</column-name></cmp-field>
```

```
            <cmp-field><field-name>numVagas</field-name><column-
name>CUR_NUMVAGAS</column-name></cmp-field>
            <cmp-field><field-name>prazoConfirmacao</field-name><column-
name>CUR_PRAZOCONFIRM</column-name></cmp-field>
            <cmp-field><field-name>resumo</field-name><column-
name>CUR_RESUMO</column-name></cmp-field>

            <!-- Consulta para listar os cursos disponíveis para inscrição -->
            <query>
                <query-method>
                    <method-name>findByInscricaoDisponivel</method-name>
                    <method-params>
                        <method-param>java.lang.Integer</method-param>
                        <method-param>java.sql.Date</method-param>
                    </method-params>
                </query-method>
                <declared-sql>
                    <select>
                        <distinct/>
                        <alias>c</alias>
                    </select>
                    <from><![CDATA[
INNER JOIN t_metadados md ON c.md_id = md.md_id
LEFT JOIN t_usuario_curso uc ON c.cur_id = uc.cur_id
LEFT JOIN t_coord_curso cc ON c.cur_id = cc.cur_id
LEFT JOIN t_professor p ON p.prof_id = cc.prof_id
LEFT JOIN t_usuario u1 ON uc.usu_id = u1.usu_id
LEFT JOIN t_usuario u2 ON p.usu_id  = u2.usu_id
]]>
                    </from>
                    <where><![CDATA[
c.cur_disponivel AND
((c.cur_inscr_inicio IS NULL or c.cur_inscr_inicio <= {1}) AND
 (c.cur_inscr_fim IS NULL or c.cur_inscr_fim >= {1})) AND
(u2.usu_id <> {0} OR u2.usu_id IS NULL) AND
((u1.usu_id <> {0} OR u1.usu_id IS NULL) OR
(u1.usu_id = {0} AND uc.uc_inscricao IS NULL))
]]>
                    </where>
                    <order>md.md_titulo</order>
                </declared-sql>
            </query>

            <!-- Lista os cursos públicos disponíveis para inscrição. -->
            <query>
                <query-method>
                    <method-name>findByDisponivel</method-name>
                    <method-params>
                        <method-param>java.sql.Date</method-param>
                    </method-params>
                </query-method>
                <jboss-ql>
                    <![CDATA[SELECT OBJECT(o) FROM Curso AS o
                      WHERE o.disponivel AND (NOT o.reservado) AND
                       ((o.inicioInscricao IS NULL OR o.inicioInscricao < ?1 OR
o.inicioInscricao = ?1) AND
                        (o.fimInscricao IS NULL OR o.fimInscricao > ?1 OR
o.fimInscricao = ?1))
                      ORDER BY o.metadados.title
                    ]]>
                </jboss-ql>
            </query>

        </entity>
```

```
        <!-- AlunoCurso -->
        <entity>
            <ejb-name>AlunoCurso</ejb-name>
            <table-name>t_usuario_curso</table-name>
            <cmp-field><field-name>id</field-name><column-name>USU_CUR_ID</column-
name></cmp-field>
            <cmp-field><field-name>inscricao</field-name><column-
name>UC_INSCRICAO</column-name></cmp-field>
            <cmp-field><field-name>solicitacao</field-name><column-
name>UC_SOLICITACAO</column-name></cmp-field>
            <cmp-field><field-name>conclusao</field-name><column-
name>UC_CONCLUSAO</column-name></cmp-field>
            <cmp-field><field-name>expSolicitacao</field-name><column-
name>UC_EXPSOLICIT</column-name></cmp-field>
        </entity>

        <!-- Itens -->
        <entity>
            <ejb-name>Item</ejb-name>
            <table-name>t_item</table-name>
            <cmp-field><field-name>id</field-name><column-name>ITEM_ID</column-
name></cmp-field>
            <cmp-field><field-name>titulo</field-name><column-
name>ITEM_TITULO</column-name></cmp-field>
            <cmp-field><field-name>ordem</field-name><column-
name>ITEM_ORDEM</column-name></cmp-field>
            <cmp-field><field-name>limiteTempo</field-name><column-
name>ITEM_LIMITETEMPO</column-name></cmp-field>
            <cmp-field><field-name>acaoFimTempo</field-name><column-
name>ITEM_ACAOTEMPO</column-name></cmp-field>
            <cmp-field><field-name>mencaoMinima</field-name><column-
name>ITEM_MENCAO</column-name></cmp-field>
            <cmp-field><field-name>visivel</field-name><column-
name>ITEM_VISIVEL</column-name></cmp-field>
            <query>
                <query-method>
                    <method-name>findByPai</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.ItemLocal</method-param>
                    </method-params>
                </query-method>
                <jboss-ql><![CDATA[SELECT OBJECT(o) FROM Item o WHERE o.pai = ?1
ORDER BY o.ordem]]></jboss-ql>
            </query>
            <query>
                <query-method>
                    <method-name>findByRaizCurso</method-name>
                    <method-params>
                        <method-param>unb.ead.ejb.CursoLocal</method-param>
                    </method-params>
                </query-method>
                <jboss-ql><![CDATA[SELECT OBJECT(o) FROM Item o WHERE o.curso = ?1
AND o.pai IS NULL ORDER BY o.ordem]]></jboss-ql>

            </query>
        </entity>

        <!-- Recursos -->
        <entity>
            <ejb-name>Recurso</ejb-name>
            <table-name>t_recurso</table-name>
            <cmp-field><field-name>id</field-name><column-name>REC_ID</column-
name></cmp-field>
        </entity>
```

```
        <!-- Situação de um item em relação a um aluno -->
        <entity>
            <ejb-name>AlunoItem</ejb-name>
            <table-name>t_usuario_item</table-name>
            <cmp-field><field-name>id</field-name><column-name>UI_ID</column-
name></cmp-field>
            <cmp-field><field-name>primeiroAcesso</field-name><column-
name>UI_PRIMACESSO</column-name></cmp-field>
            <cmp-field><field-name>ultimoAcesso</field-name><column-
name>UI_ULTACESSO</column-name></cmp-field>
            <cmp-field><field-name>conclusao</field-name><column-
name>UI_CONCLUSAO</column-name></cmp-field>
            <cmp-field><field-name>mencao</field-name><column-
name>UI_MENCAO</column-name></cmp-field>
        </entity>

        <!-- Metdadados -->
        <entity>
            <ejb-name>Metadata</ejb-name>
            <table-name>t_metadados</table-name>
            <cmp-field><field-name>id</field-name><column-name>MD_ID</column-
name></cmp-field>
            <cmp-field><field-name>title</field-name><column-
name>MD_TITULO</column-name></cmp-field>
        </entity>

        <!-- Pré-requisitos -->
        <entity>
            <ejb-name>Prerequisitos</ejb-name>
            <table-name>t_prerequisitos</table-name>
            <cmp-field><field-name>id</field-name><column-name>PREREQ_ID</column-
name></cmp-field>
        </entity>
        <entity>
            <ejb-name>ItemPrerequisito</ejb-name>
            <table-name>t_itemprereq</table-name>
            <cmp-field><field-name>id</field-name><column-name>IP_ID</column-
name></cmp-field>
            <cmp-field><field-name>tipo</field-name><column-name>IP_TIPO</column-
name></cmp-field>
        </entity>
    </enterprise-beans>

    <relationships>
        <!-- Associação de Professor para Usuário -->
        <ejb-relation>
            <ejb-relation-name>usuario</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Professor UNIDIRECTIONAL</ejb-
relationship-role-name>
                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Professor</ejb-
relationship-role-name>
                <key-fields>
                  <key-field>
                    <field-name>id</field-name>
                    <column-name>USU_ID</column-name>
                  </key-field>
                </key-fields>
            </ejb-relationship-role>
        </ejb-relation>
```

```
        <!-- Associação entre Professores e Cursos (Coordenadores) -->
        <ejb-relation>
            <ejb-relation-name>Cursos coordenados</ejb-relation-name>
            <relation-table-mapping>
              <table-name>t_coord_curso</table-name>
            </relation-table-mapping>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Professor to Curso</ejb-relationship-
role-name>
                <key-fields>
                  <key-field>
                    <field-name>id</field-name>
                    <column-name>PROF_ID</column-name>
                  </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Curso to Professor</ejb-relationship-
role-name>
                <key-fields>
                  <key-field>
                    <field-name>id</field-name>
                    <column-name>CUR_ID</column-name>
                  </key-field>
                </key-fields>
            </ejb-relationship-role>
        </ejb-relation>


        <!-- Associação entre usuários e AlunoCurso -->
        <ejb-relation>
            <ejb-relation-name>Usuário de AlunoCurso</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Usuario to AlunoCurso</ejb-
relationship-role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>USU_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoCurso to Usuario</ejb-
relationship-role-name>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Associação entre cursos e AlunoCurso -->
        <ejb-relation>
            <ejb-relation-name>Curso de AlunoCurso</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoCurso to Curso</ejb-relationship-
role-name>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Curso to AlunoCurso</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>CUR_ID</column-name>
                    </key-field>
```

```
                </key-fields>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Relacionamentos entre os cursos e seus itens -->
        <ejb-relation>
            <ejb-relation-name>Itens de curso</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>

                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>CUR_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Auto-relacionamento entre os itens -->
        <ejb-relation>
            <ejb-relation-name>Item pai</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>ITEM_ITEM_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Relacionamento entre recursos e itens -->
        <ejb-relation>
            <ejb-relation-name>Recurso de item</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>REC_ID</column-name>
                    </key-field>
                </key-fields>
```

```
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Pré-requisitos -->
        <ejb-relation>
            <ejb-relation-name>Pré-requisito de item</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>ITEM_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <key-fields/>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <ejb-relation-name>Itens de pré-requisito</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Prerequisitos UNIDIRECTIONAL</ejb-
relationship-role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>PREREQ_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Prerequisitos</ejb-
relationship-role-name>
                <key-fields/>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <ejb-relation-name>Item de item de pré-requisito.</ejb-relation-name>

            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>ItemPrerequisito UNIDIRECTIONAL</ejb-
relationship-role-name>
                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL ItemPrerequisito</ejb-
relationship-role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>ITEM_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Relacionamentos entre Aluno e Item -->
        <ejb-relation>
```

```
            <ejb-relation-name>Aluno de AlunoItem</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoItem UNIDIRECTIONAL</ejb-
relationship-role-name>
                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL AlunoItem</ejb-
relationship-role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>USU_ID</column-name>
                    </key-field>
                 </key-fields>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <ejb-relation-name>Item  de AlunoItem</ejb-relation-name>
            <foreign-key-mapping/>
            <ejb-relationship-role>
                <ejb-relationship-role-name>AlunoItem UNIDIRECTIONAL</ejb-
relationship-role-name>
                <key-fields/>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL AlunoItem</ejb-
relationship-role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>ITEM_ID</column-name>
                    </key-field>
                 </key-fields>
            </ejb-relationship-role>
        </ejb-relation>

        <!-- Metadados -->
        <ejb-relation>
            <ejb-relation-name>Metadados de item</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Item UNIDIRECTIONAL</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>MD_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Item</ejb-relationship-
role-name>
                <key-fields/>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <ejb-relation-name>Metadados de recurso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Recurso UNIDIRECTIONAL</ejb-
relationship-role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
```

```
                        <column-name>MD_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Recurso</ejb-
relationship-role-name>
                <key-fields/>
            </ejb-relationship-role>
        </ejb-relation>
        <ejb-relation>
            <ejb-relation-name>Metadados de curso</ejb-relation-name>
            <ejb-relationship-role>
                <ejb-relationship-role-name>Curso UNIDIRECTIONAL</ejb-relationship-
role-name>
                <key-fields>
                    <key-field>
                        <field-name>id</field-name>
                        <column-name>MD_ID</column-name>
                    </key-field>
                </key-fields>
            </ejb-relationship-role>
            <ejb-relationship-role>
                <ejb-relationship-role-name>UNIDIRECTIONAL Curso</ejb-relationship-
role-name>
                <key-fields/>
            </ejb-relationship-role>
        </ejb-relation>
    </relationships>
</jbosscmp-jdbc>
```

8.1.3. jboss.xml

Neste arquivo é definido o domínio de autenticação e também como cada componente será disponibilizado no diretório.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 3.0//EN"
"http://www.jboss.org/j2ee/dtd/jboss_3_0.dtd">
<!-- Autenticação -->

<jboss>
    <security-domain>java:jaas/CampusVirtual</security-domain>

    <enterprise-beans>
        <session>
            <ejb-name>SessaoSerial</ejb-name>
            <local-jndi-name>ejb/ead/SessaoSerial</local-jndi-name>
        </session>
        <entity>
            <ejb-name>Serial</ejb-name>
            <local-jndi-name>ejb/ead/Serial</local-jndi-name>
        </entity>

        <entity>
            <ejb-name>Usuario</ejb-name>
            <local-jndi-name>ejb/ead/Usuario</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>Professor</ejb-name>
            <local-jndi-name>ejb/ead/Professor</local-jndi-name>
```

```
        </entity>
        <entity>
            <ejb-name>Curso</ejb-name>
            <local-jndi-name>ejb/ead/Curso</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>AlunoCurso</ejb-name>
            <local-jndi-name>ejb/ead/AlunoCurso</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>Usuario</ejb-name>
            <local-jndi-name>ejb/ead/Usuario</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>Item</ejb-name>
            <local-jndi-name>ejb/ead/Item</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>Recurso</ejb-name>
            <local-jndi-name>ejb/ead/Recurso</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>Metadata</ejb-name>
            <local-jndi-name>ejb/ead/Metadata</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>Prerequisitos</ejb-name>
            <local-jndi-name>ejb/ead/Prerequisitos</local-jndi-name>
        </entity>
        <entity>
            <ejb-name>ItemPrerequisito</ejb-name>
            <local-jndi-name>ejb/ead/ItemPrerequisito</local-jndi-name>
        </entity>
    </enterprise-beans>
</jboss>
```

8.2. CONFIGURAÇÃO DO JSTATEMACHINE

No arquivo CampusVirtual.xml está representado o mapa de estados, da forma como foram mostrados anteriormente.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Created by rmiranda on 15 de Maio de 2002, 09:45 -->
<Application xmlns="java:org.jstatemachine.apploaders.jade" name="CampusVirtual"
startStateId="Publico">
       <State name="CampusVirtual"
viewName="org.jstatemachine.connectors.servlet.AutoView">
             <!-- Evento padrao quando caímos no estado da aplicação. -->
             <DefaultEvent>
                    <Event name="Default"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico" name="Default"/>
                    </Event>
             </DefaultEvent>
             <!-- Parte pública da aplicação -->
             <State name="Publico"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                    <UserProperties>
                           <UserProperty key="atributos" value="autentica.nome"/>
                    </UserProperties>
```

```
                    <DefaultEvent>
                           <Event name="Default"
controllerName="org.jstatemachine.apps.SimpleController">
                                  <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Entrada" name="Default"/>
                           </Event>
                    </DefaultEvent>
                    <!-- Exibir página inicial -->
                    <Event name="entrada"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Entrada" name="Default"/>
                    </Event>
                    <!-- Usuário informou nome e senha -->
                    <Event name="autentica"
controllerName="unb.ead.web.controlador.autenticacao.Autentica">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado" name="sucesso"/>
                           <StateTransition entryCondition="DEFAULT"
resultStateId="NaoConfirmado" name="naoConfirmado"/>
                           <StateTransition entryCondition="HISTORY_STAR"
resultStateId="Publico" name="usuarioInvalido"/>
                    </Event>
                    <!-- Novo usuário -->
                    <Event name="cadastro"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Cadastra" name="cadastro"/>
                    </Event>
                    <!-- Informações -->
                    <Event name="informacoes"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Informacoes" name="informacoes"/>
                    </Event>
                    <!-- Cursos disponíveis -->
                    <Event name="cursos"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Cursos" name="cursos"/>
                    </Event>
                    <!-- Links -->
                    <Event name="links"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Links" name="links"/>
                    </Event>
                    <!-- Contato -->
                    <Event name="contato"
controllerName="org.jstatemachine.apps.SimpleController">
                           <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Contato" name="contato"/>
                    </Event>
                    <!-- Estado de entrada -->
                    <State name="Publico_Entrada"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                           <UserProperties>
                                  <UserProperty key="JSPName"
value="/pages/aberto/index.jsp"/>
                           </UserProperties>
                    </State>
                    <!-- Informações -->
                    <State name="Publico_Informacoes"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                           <UserProperties>
```

```
                                <UserProperty key="JSPName"
value="/pages/aberto/info.jsp"/>
                        </UserProperties>
                    </State>
                    <!-- Links -->
                    <State name="Publico_Links"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                        <UserProperties>
                                <UserProperty key="JSPName"
value="/pages/aberto/links.jsp"/>
                        </UserProperties>
                    </State>
                    <!-- Contato -->
                    <State name="Publico_Contato"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                        <UserProperties>
                                <UserProperty key="JSPName"
value="/pages/aberto/contato.jsp"/>
                        </UserProperties>
                    </State>
                    <!-- Cadastro de usuário -->
                    <State name="Publico_Cadastra"
viewName="org.jstatemachine.connectors.servlet.JSPView"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                        <UserProperties>
                                <UserProperty key="atributos"
value="cadastro.nome,cadastro.nomecompleto,cadastro.email"/>
                                <UserProperty key="JSPName"
value="/pages/aberto/cadastro.jsp"/>
                        </UserProperties>
                        <Event name="cadastro"
controllerName="unb.ead.web.controlador.autenticacao.NovoUsuario">
                                <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_Cadastra" name="erroCadastro"/>
                                <StateTransition entryCondition="DEFAULT"
resultStateId="Cadastrado" name="sucesso"/>
                        </Event>
                    </State>
                    <!-- Cursos -->
                    <State name="Publico_Cursos"
entryActionName="unb.ead.web.actions.publico.CursosEntry"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                        <UserProperties>
                                <UserProperty key="atributos"
value="publico.cursos.lista"/>
                        </UserProperties>
                        <DefaultEvent>
                                <Event name="Default"
controllerName="org.jstatemachine.apps.SimpleController">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_ListaCursos" name="Default"/>
                                </Event>
                        </DefaultEvent>
                        <State name="Publico_ListaCursos"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                <UserProperties>
                                        <UserProperty key="JSPName"
value="/pages/aberto/cursos.jsp"/>
                                </UserProperties>
                                <Event name="escolhe"
controllerName="org.jstatemachine.apps.SimpleController">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_DetalheCurso" name="escolhe"/>
                                </Event>
                        </State>
```

```
                        <State name="Publico_DetalheCurso"
exitActionName="unb.ead.web.actions.RemoveAtributosExit"
entryActionName="unb.ead.web.actions.publico.CursoEntry">
                                <UserProperties>
                                        <UserProperty key="atributos"
value="publico.cursos.infocurso"/>
                                </UserProperties>
                                <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="Publico_ListaCursos" name="cursos"/>
                                </Event>
                        </State>
                </State>
            </State>
            <!-- Daqui para baixo o usuário está autenticado -->
            <State exitActionName="unb.ead.web.actions.ValidadoExit"
entryActionName="unb.ead.web.actions.ValidadoEntry" name="Validado">
                <Event name="sair"
controllerName="org.jstatemachine.apps.SimpleController">
                        <!-- Sair do sistema -->
                        <StateTransition entryCondition="DEFAULT"
resultStateId="CampusVirtual" name="sair"/>
                </Event>
                <State name="MudaCadastro"
entryActionName="unb.ead.web.actions.AlteraCadastroEntry"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                        <UserProperties>
                                <UserProperty key="atributos"
value="altcadastro.nomecompleto,altcadastro.email"/>
                        </UserProperties>
                        <DefaultEvent>
                                <Event name="Default"
controllerName="org.jstatemachine.apps.SimpleController">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="AlteraCadastro" name="Default"/>
                                </Event>
                        </DefaultEvent>
                        <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                <StateTransition entryCondition="HISTORY_STAR"
resultStateId="Autenticado" name="cancela"/>
                        </Event>
                        <State name="AlteraCadastro"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                <UserProperties>
                                        <UserProperty key="JSPName"
value="/pages/restrito/mudacadastro.jsp"/>
                                </UserProperties>
                                <Event name="alteraCadastro"
controllerName="unb.ead.web.controlador.autenticacao.MudaCadastro">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="AlteraCadastro" name="erroCadastro"/>
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="CadastroAlterado" name="sucesso"/>
                                </Event>
                        </State>
                        <State name="CadastroAlterado"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                <UserProperties>
                                        <UserProperty key="JSPName"
value="/pages/restrito/cadastroalterado.jsp"/>
                                </UserProperties>
                        </State>
                </State>
```

```
                    <State name="Autenticado">
                          <!-- Alterar senha -->
                          <Event name="alterasenha"
controllerName="org.jstatemachine.apps.SimpleController">
                                <StateTransition entryCondition="DEFAULT"
resultStateId="MudaSenha" name="alterasenha"/>
                          </Event>
                          <!-- Alterar cadastro -->
                          <Event name="alteracadastro"
controllerName="org.jstatemachine.apps.SimpleController">
                                <StateTransition entryCondition="DEFAULT"
resultStateId="MudaCadastro" name="alteracadastro"/>
                          </Event>
                          <!-- Usuário já confirmado -->
                          <State name="JaConfirmado">
                                <Event name="principal"
controllerName="org.jstatemachine.apps.SimpleController">
                                      <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado" name="principal"/>
                                </Event>
                          </State>
                          <!-- Confirmaçao de cadastro efetuada -->
                          <State name="ConfirmacaoEfetuada"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                <UserProperties>
                                      <UserProperty key="JSPName"
value="/pages/restrito/benvindo.jsp"/>
                                </UserProperties>
                                <Event name="principal"
controllerName="org.jstatemachine.apps.SimpleController">
                                      <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado" name="principal">
                                            <UserProperties>
                                                  <UserProperty key="Condition"
value="Ir para página principal"/>
                                            </UserProperties>
                                      </StateTransition>
                                </Event>
                          </State>
                          <State name="NaoConfirmado">
                                <!-- Evento padrao para um usuário nao confirmado
(inicial) -->
                                <DefaultEvent>
                                      <Event name="Default"
controllerName="org.jstatemachine.apps.SimpleController">
                                            <StateTransition
entryCondition="DEFAULT" resultStateId="ConfirmaCadastro" name="Default"/>
                                      </Event>
                                </DefaultEvent>
                                <!-- Informaçoes do cadastro -->
                                <State name="Cadastrado">
                                      <DefaultEvent>
                                            <Event name="Default"
controllerName="unb.ead.web.controlador.autenticacao.EnviaEmail">
                                                  <UserProperties>
                                                        <UserProperty key="smtp-
host" value="@smtp.host@"/>
                                                  </UserProperties>
                                                  <StateTransition
entryCondition="DEFAULT" resultStateId="InfoCadastro" name="sucesso"/>
                                                  <StateTransition
entryCondition="DEFAULT" resultStateId="InfoCadastro" name="jaConfirmado"/>
                                            </Event>
                                      </DefaultEvent>
```

```
                                        <Event name="confirma"
controllerName="org.jstatemachine.apps.SimpleController">
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="ConfirmaCadastro" name="confirma">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Ir para confirmaçao de cadastro"/>
                                                        </UserProperties>
                                                </StateTransition>
                                        </Event>
                                        <State name="InfoCadastro"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                                <UserProperties>
                                                        <UserProperty key="JSPName"
value="/pages/restrito/cadastroaceito.jsp"/>
                                                </UserProperties>
                                        </State>
                                </State>
                                <!-- Confirmaçao do cadastro -->
                                <State name="ConfirmaCadastro"
viewName="org.jstatemachine.connectors.servlet.JSPView"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                                        <UserProperties>
                                                <UserProperty key="atributos"
value="confirma.chave"/>
                                                <UserProperty key="JSPName"
value="/pages/restrito/confirma.jsp"/>
                                        </UserProperties>
                                        <Event name="enviaConfirmacao"
controllerName="unb.ead.web.controlador.autenticacao.EnviaEmail">
                                                <UserProperties>
                                                        <UserProperty key="smtp-host"
value="@smtp.host@"/>
                                                </UserProperties>
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="JaConfirmado" name="jaConfirmado"/>
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="ConfirmacaoEnviada" name="sucesso"/>
                                        </Event>
                                        <Event name="confirma"
controllerName="unb.ead.web.controlador.autenticacao.ConfirmaCadastro">
                                                <UserProperties>
                                                        <UserProperty
key="parameter.chave" value="chave"/>
                                                </UserProperties>
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="ConfirmacaoEfetuada" name="sucesso">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Cadastro confirmado"/>
                                                        </UserProperties>
                                                </StateTransition>
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="ConfirmaCadastro" name="chaveInvalida">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Chave inválida"/>
                                                        </UserProperties>
                                                </StateTransition>
                                        </Event>
                                </State>
                                <!-- E-Mail de confirmação -->
                                <State name="ConfirmacaoEnviada"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                        <UserProperties>
```

```
                                                <UserProperty key="JSPName"
value="/pages/restrito/confenviada.jsp"/>
                                          </UserProperties>
                                          <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="ConfirmaCadastro" name="confirma">
                                                      <UserProperties>
                                                            <UserProperty
key="Condition" value="Ir para confirmaçao de cadastro"/>
                                                      </UserProperties>
                                                </StateTransition>
                                          </Event>
                                    </State>
                              </State>
                              <State name="Confirmado">
                                    <!-- Área para usuários confirmados -->
                                    <DefaultEvent>
                                          <Event name="default"
controllerName="org.jstatemachine.apps.SimpleController">
                                                <StateTransition
entryCondition="DEFAULT" resultStateId="Confirmado_Agenda"/>
                                          </Event>
                                    </DefaultEvent>
                                    <Event name="agenda"
controllerName="org.jstatemachine.apps.SimpleController">
                                          <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado_Agenda" name="agenda">
                                                <UserProperties>
                                                      <UserProperty key="Condition"
value="Ir para Agenda"/>
                                                </UserProperties>
                                          </StateTransition>
                                    </Event>
                                    <Event name="inscrever"
controllerName="org.jstatemachine.apps.SimpleController">
                                          <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado_InscricaoCurso" name="default">
                                                <UserProperties>
                                                      <UserProperty key="Condition"
value="Ir para Inscrição em Cursos"/>
                                                </UserProperties>
                                          </StateTransition>
                                    </Event>
                                    <Event name="acompanhar"
controllerName="org.jstatemachine.apps.SimpleController">
                                          <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado_AcompanhaCursos" name="default">
                                                <UserProperties>
                                                      <UserProperty key="Condition"
value="Ir para Acompanhamento de Cursos"/>
                                                </UserProperties>
                                          </StateTransition>
                                    </Event>
                                    <Event name="forums"
controllerName="org.jstatemachine.apps.SimpleController">
                                          <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado_Forums" name="default">
                                                <UserProperties>
                                                      <UserProperty key="Condition"
value="Ir para Fórums"/>
                                                </UserProperties>
                                          </StateTransition>
                                    </Event>
```

```
                                <Event name="email"
controllerName="org.jstatemachine.apps.SimpleController">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="Confirmado_Email" name="default">
                                                <UserProperties>
                                                        <UserProperty key="Condition"
value="Ir para e-mail"/>
                                                </UserProperties>
                                        </StateTransition>
                                </Event>

                                <!-- Inscriçao em cursos -->
                                <State name="Confirmado_InscricaoCurso"
entryActionName="unb.ead.web.actions.inscricao.InscricaoCursoEntry"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                                        <UserProperties>
                                                <UserProperty key="atributos"
value="unb.ead.inscricao.cursos"/>
                                        </UserProperties>
                                        <DefaultEvent>
                                                <Event name="Default"
controllerName="org.jstatemachine.apps.SimpleController">
                                                        <StateTransition
entryCondition="DEFAULT" resultStateId="InscricaoCurso_ListaCursos"/>
                                                </Event>
                                        </DefaultEvent>
                                        <State name="InscricaoCurso_ListaCursos">
                                                <Event name="escolhe"
controllerName="org.jstatemachine.apps.SimpleController">
                                                        <UserProperties>
                                                                <UserProperty
key="parameter.curso" value="curso"/>
                                                        </UserProperties>
                                                        <StateTransition
entryCondition="DEFAULT" name="sucesso" resultStateId="InscricaoCurso_Detalhes">
                                                                <UserProperties>
                                                                        <UserProperty
key="Condition" value="Ver detalhes de curso"/>
                                                                </UserProperties>
                                                        </StateTransition>
                                                </Event>
                                        </State>
                                        <State name="InscricaoCurso_Detalhes"
entryActionName="unb.ead.web.actions.inscricao.InfoCursoEntry"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                                                <UserProperties>
                                                        <UserProperty key="atributos"
value="unb.ead.inscricao.infocurso"/>
                                                </UserProperties>
                                                <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                                        <StateTransition
entryCondition="DEFAULT" resultStateId="InscricaoCurso_ListaCursos"/>
                                                </Event>
                                                <Event name="inscreve"
controllerName="unb.ead.web.controlador.insccurso.Inscreve">
                                                        <UserProperties>
                                                                <UserProperty
key="parameter.curso" value="curso"/>
                                                        </UserProperties>
                                                        <StateTransition
entryCondition="DEFAULT" name="sucesso"
resultStateId="InscricaoCurso_InscricaoAceita">
                                                                <UserProperties>
```

```
                                                                <UserProperty
key="Condition" value="Inscrição aceita sem problemas"/>
                                                        </UserProperties>
                                                </StateTransition>
                                                <StateTransition
entryCondition="DEFAULT" name="rejeitada"
resultStateId="InscricaoCurso_InscricaoRejeitada">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Inscrição não aceita"/>
                                                        </UserProperties>
                                                </StateTransition>
                                                <StateTransition
entryCondition="DEFAULT" name="semvagas" resultStateId="InscricaoCurso_SemVagas">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Sem vagas disponíveis"/>
                                                        </UserProperties>
                                                </StateTransition>
                                        </Event>
                                        <Event name="infocoordenador"
controllerName="unb.ead.web.controlador.ObtemInfoProfessor">
                                                <UserProperties>
                                                        <UserProperty
key="parameter.professor" value="professor"/>
                                                </UserProperties>
                                                <StateTransition
entryCondition="DEFAULT" name="default"
resultStateId="InscricaoCurso_InfoCoordenador">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Informações sobre um coordenador"/>
                                                        </UserProperties>
                                                </StateTransition>
                                        </Event>
                                        <Event name="infoitem"
controllerName="unb.ead.web.controlador.insccurso.ObtemDetalheItem">
                                                <UserProperties>
                                                        <UserProperty
key="parameter.item" value="item"/>
                                                </UserProperties>
                                                <StateTransition
entryCondition="DEFAULT" name="default" resultStateId="InscricaoCurso_InfoItem"/>
                                        </Event>
                                </State>

                                <State
name="InscricaoCurso_InfoCoordenador">
                                        <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                                <StateTransition
entryCondition="HISTORY" name="default" resultStateId="InscricaoCurso_Detalhes">
                                                        <UserProperties>
                                                                <UserProperty
key="Condition" value="Voltar"/>
                                                        </UserProperties>
                                                </StateTransition>
                                        </Event>
                                </State>
                                <State name="InscricaoCurso_InfoItem">
                                        <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                                <UserProperties>
                                                        <UserProperty
key="parameter.item" value="itemAnterior"/>
```

```
                                                    </UserProperties>
                                                    <StateTransition
entryCondition="DEFAULT" name="semItems" resultStateId="InscricaoCurso_Detalhes">
                                                          <UserProperties>
                                                                <UserProperty
key="Condition" value="Voltar para Curso"/>
                                                          </UserProperties>
                                                    </StateTransition>
                                                    <StateTransition
entryCondition="DEFAULT" name="anterior" resultStateId="InscricaoCurso_InfoItem">
                                                          <UserProperties>
                                                                <UserProperty
key="Condition" value="Voltar para item anterior"/>
                                                          </UserProperties>
                                                    </StateTransition>

                                              </Event>
                                              <Event name="infoitem"
controllerName="unb.ead.web.controlador.insccurso.ObtemDetalheItem">
                                                    <UserProperties>
                                                          <UserProperty
key="parameter.item" value="item"/>
                                                    </UserProperties>
                                                    <StateTransition
entryCondition="HISTORY" name="default" resultStateId="InscricaoCurso_InfoItem">
                                                          <UserProperties>
                                                                <UserProperty
key="Condition" value="Ver pré-requisito"/>
                                                          </UserProperties>
                                                    </StateTransition>
                                              </Event>
                                        </State>

                                        <State
name="InscricaoCurso_InscricaoProcessada">
                                              <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                                    <StateTransition
entryCondition="DEFAULT" name="default" resultStateId="Confirmado_InscricaoCurso">
                                                          <UserProperties>
                                                                <UserProperty
key="Condition" value="Voltar para lista de cursos"/>
                                                          </UserProperties>
                                                    </StateTransition>
                                              </Event>
                                              <State
name="InscricaoCurso_InscricaoAceita"/>
                                              <State
name="InscricaoCurso_InscricaoRejeitada"/>
                                              <State
name="InscricaoCurso_SemVagas"/>
                                        </State>
                                        <!-- Estado de exceção -->
                                        <ExceptionState
name="InscricaoCurso_SemPermissao" stateId="InscricaoCurso_ExcecaoSemPermissao"/>
                                        <State
name="InscricaoCurso_ExcecaoSemPermissao">
                                              <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                                    <StateTransition
entryCondition="HISTORY" name="default" resultStateId="InscricaoCurso_ListaCursos">
                                                          <UserProperties>
                                                                <UserProperty
key="Condition" value="Voltar para lista de cursos"/>
                                                          </UserProperties>
```

```
                                                      </StateTransition>
                                                </Event>
                                          </State>
                                    </State>
                                    <!-- Fim da inscrição em cursos -->

                                    <!-- Acompanhamento de cursos -->
                                    <State name="Confirmado_AcompanhaCursos">
                                          <DefaultEvent>
                                                <Event name="default"
controllerName="unb.ead.web.controlador.acompanhamento.ObtemCursosInscritos">
                                                      <StateTransition
entryCondition="DEFAULT" name="sucesso" resultStateId="ListaCursosInscritos"/>
                                                      <StateTransition
entryCondition="DEFAULT" name="apenasum" resultStateId="AcompanhaCursoEscolhido"/>
                                                      <StateTransition
entryCondition="DEFAULT" name="nenhum" resultStateId="NenhumCursoDisponivel"/>
                                                </Event>
                                          </DefaultEvent>
                                          <State name="ListaCursosInscritos"/>
                                          <State name="AcompanhaCursoEscolhido"/>
                                          <State name="NenhumCursoDisponivel"/>
                                    </State>
                                    <!-- Fim do acompanhamento de cursos -->
                                    <State name="Confirmado_Agenda"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                          <UserProperties>
                                                <UserProperty key="JSPName"
value="/pages/restrito/agenda.jsp"/>
                                          </UserProperties>
                                    </State>
                                    <State name="Confirmado_Forums"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                          <UserProperties>
                                                <UserProperty key="JSPName"
value="/pages/restrito/forum.jsp"/>
                                          </UserProperties>
                                    </State>
                                    <State name="Confirmado_Email"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                          <UserProperties>
                                                <UserProperty key="JSPName"
value="/pages/restrito/envia_email.jsp"/>
                                          </UserProperties>
                                    </State>
                              </State>
                        </State>
                        <State name="MudaSenha"
exitActionName="unb.ead.web.actions.RemoveAtributosExit">
                              <DefaultEvent>
                                    <Event name="default"
controllerName="org.jstatemachine.apps.SimpleController">
                                          <StateTransition entryCondition="DEFAULT"
name="default" resultStateId="Senha_AlteraSenha"/>
                                    </Event>
                              </DefaultEvent>
                              <Event name="voltar"
controllerName="org.jstatemachine.apps.SimpleController">
                                    <StateTransition entryCondition="HISTORY_STAR"
resultStateId="Autenticado" name="voltar"/>
                              </Event>
                              <State name="Senha_AlteraSenha"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                    <UserProperties>
```

```
                                        <UserProperty key="JSPName"
value="/pages/restrito/mudasenha.jsp"/>
                                    </UserProperties>
                                    <Event name="alteraSenha"
controllerName="unb.ead.web.controlador.autenticacao.MudaSenha">
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="Senha_AlteraSenha" name="senhaInvalida"/>
                                        <StateTransition entryCondition="DEFAULT"
resultStateId="Senha_SenhaAlterada" name="sucesso"/>
                                    </Event>
                                </State>
                                <State name="Senha_SenhaAlterada"
viewName="org.jstatemachine.connectors.servlet.JSPView">
                                    <UserProperties>
                                        <UserProperty key="JSPName"
value="/pages/restrito/senhaalterada.jsp"/>
                                    </UserProperties>
                                </State>
                            </State>
                        </State>
                    </State>
                </Application>
```

8.3. CONFIGURAÇÃO DO BANCO DE DADOS MYSQL

O script abaixo acresenta no MySQL as tabelas necessárias para o suporte ao Campus Virtual.

```
drop table if exists t_sequencia;
create table t_sequencia (seq_nome varchar(32) primary key, seq_valor integer
default 1) type=InnoDB;
insert into t_sequencia  (seq_nome) values ('lista_acessos');
insert into t_sequencia  (seq_nome) values ('usuario');

drop table if exists t_recurso;
drop table if exists t_usuario cascade;
drop table if exists t_curso cascade;
drop table if exists t_item cascade;
drop table if exists t_forum cascade;
drop table if exists t_forum_curso cascade;
drop table if exists t_forum_mensagem cascade;
drop table if exists t_forum_msg_dados;
drop table if exists t_professor cascade;
drop table if exists t_usuario_curso cascade;
drop table if exists t_usuario_item cascade;
drop table if exists t_acesso cascade;
drop table if exists t_coord_curso cascade;
drop table if exists t_prerequisitos;
drop table if exists t_itemprereq;
drop table if exists t_metadados;
drop table if exists t_metadados_descricao;


--
-- Metadados
create table t_metadados
(
    MD_ID               INTEGER              not null,   -- Identificador do meta-
dado
    MD_TITULO            TINYTEXT                not null,    -- Título do objeto
associado
    primary key (MD_ID)
) type=InnoDB;
```

```
create table t_metadados_descricao
(
    MD_ID               INTEGER               not null,    -- A que metadados esta
descrição se refere
    MDD_ORDEM            INTEGER                 not null,    -- Número sequencial
referenciando esta descrição.
    MDD_TEXTO          TEXT                  not null,    -- Dados da descrição
    primary key (MD_ID, MDD_ORDEM),
    index idx_metadados_descricao (MD_ID),
    foreign  key  fk_metadados_descricao_metadados  (MD_ID)  references  t_metadados
(MD_ID)
) type=InnoDB;


--
-- Usuários
create table t_usuario
(
    USU_ID             INTEGER             not null,    -- Identificador do usuário
    USU_LOGIN          VARCHAR(16)         not null,    -- Nome de usuário
    USU_SENHA           CHAR(16) BINARY      null,       -- Senha digerida por MD5,
NULL se o usuário foi desabilitado.
    USU_BLOQUEADO      BOOL                not null,    -- Usuário está bloqueado?
    USU_CADASTRO       DATETIME            not null,    -- Data/hora de cadastro
    USU_ULTACESSO       DATETIME              null     ,    -- Data/hora do último
acesso
    USU_NUMACESSOS     INTEGER             not null,    -- Total de acessos
    USU_ADMINISTRADOR  BOOL                not null,    -- É administrador?
    USU_NOME            VARCHAR(64)           not null,    -- Nome completo. O nome
completo está na tabela t_usuario_nome....
    USU_CHAVE           CHAR(20) BINARY      null,       -- Chave de confirmação do
cadastro
    USU_EMAIL          VARCHAR(128)         not null,    -- E-Mail do usuário
    USU_CONFIRMACAO    DATETIME             null    ,    -- Data/hora da confirmação
do email
    primary key (USU_ID)
) type=InnoDB;
create index idx_usuario_login_senha on t_usuario (usu_login, usu_senha);
create index idx_usuario_administrador on t_usuario (usu_administrador);
create index idx_usuario_confirmacao on t_usuario (usu_confirmacao);

--
-- Cursos disponíveis
create table t_curso
(
    CUR_ID             INTEGER             not null,    -- Identificador do curso
    CUR_DISPONIVEL     BOOL                not null,    -- Está disponível?
    CUR_INSCR_INICIO    DATE                   null     ,    -- Período de início de
inscrições (NULL para qualquer período)
    CUR_INSCR_FIM       DATE                     null     ,    -- Período de fim de
inscrições (NULL para qualquer período)
    CUR_RESERVADO       BOOL                  not null,    -- Reservado? (precisa de
autorização do administrador)
    CUR_NUMVAGAS        INTEGER              not null,    -- Número de vagas ou 0 se
não houver este tipo de restrição
    CUR_PRAZOCONFIRM    INTEGER                 null     ,    -- Número de dias para
confirmar o cadastro, caso seja curso
                                                         -- restrito
    CUR_RESUMO          TEXT                   null     ,    -- Resumo do curso (até
65536 caracteres)
    MD_ID              INTEGER             null    ,    -- Metadados do curso
    primary key (CUR_ID)
) type=InnoDB;

--
```

```
-- Recursos que compõem um curso
create table t_recurso
(
    CUR_ID               INTEGER             not null,   -- Curso ao qual o recurso
está associado
    REC_ID              INTEGER             not null,   -- Identificador do recurso
    REC_ENTRADA          INTEGER              null,        -- Arquivo de entrada do
recurso (para um bem, pode ser NULL)
    REC_TIPOSCORM        INTEGER              not null,   -- Tipo SCORM do recurso
(Objeto ou Bem)
    REC_TIPO             INTEGER             not null,   -- Tipo do recurso (apenas
1 = WebContent está definido agora)
    MD_ID                INTEGER             null,       -- Metadados do recurso
    primary key (REC_ID),
    index idx_recurso_curso (CUR_ID),
    constraint fk_recurso_curso foreign key (CUR_ID) references t_curso (CUR_ID)
) type=InnoDB;


--
-- Ítens de um curso
create table t_item
(
    CUR_ID              INTEGER              not null,   -- Curso ao qual este item
está associado
    REC_ID              INTEGER              null,       -- Recurso associado a este
item, ou null se não houver recurso
    ITEM_ID             INTEGER             not null,   -- Identificador do item
    ITEM_ITEM_ID        INTEGER             null,       -- Item pai
    ITEM_VISIVEL        BOOL                not null,   -- O item é ou não visível?
    ITEM_ORDEM           INTEGER               not null,   -- Ordem dos itens neste
nivel hierarquico
    ITEM_TITULO         VARCHAR(64)         not null,   -- Nome do item
    ITEM_LIMITETEMPO    INTEGER             null,       -- Limite de tempo do item
    ITEM_ACAOTEMPO       INTEGER               null,        -- Ação a tomar caso o
tempo tenha sido extrapolado
    ITEM_MENCAO         FLOAT                null,       -- Nota mínima que deve ser
obtida
    MD_ID               INTEGER             null,       -- Metadados do item

    primary key (ITEM_ID),
    index idx_item_curso (CUR_ID),
    index idx_item_recurso (REC_ID),
    constraint fk_item_curso foreign key (CUR_ID) references t_curso (CUR_ID)
--  constraint fk_item_recurso foreign key (REC_ID) references t_recurso (REC_ID)
--    constraint fk_item_item_pai foreign key (ITEM_ITEM_ID) references t_curso
(ITEM_ID)
) type=InnoDB;

--
-- Fórums
create table t_forum
(
    FOR_ID              INTEGER             not null,   -- Identificador do fórum
    FOR_NOME            VARCHAR(128)        not null,   -- Nome descritivo do fórum
    FOR_RESTRITO         BOOL                  not null,   -- Fórum restrito, apenas
quem tiver cadastro no curso vinculado
                                                        -- tem acesso
    primary key (FOR_ID)
) type=InnoDB;

--
-- Discussões nos fórums
create table t_forum_mensagem
(
```

```
    FOR_ID              INTEGER             not null,   -- Identificador do fórum a
que esta mensagem pertence
    USU_ID              INTEGER             not null,   -- Quem postou a mensagem??
    FOR_MSG_ID             INTEGER                 not null,    -- Identificador da
mensagem
    FOR_MSG_ID_PAI      INTEGER              not null,   -- Identificador do pai da
mensagem (está respondendo a quem??)
    FOR_MSG_DATA        INTEGER             not null,   -- Data/hora da postagem da
mensagem no fórum
    primary key (FOR_MSG_ID),
    index idx_forum_mensagem_forum   (FOR_ID),
    index idx_forum_mensagem_usuario (USU_ID),
    constraint fk_forum_mensagem_forum    foreign key (FOR_ID) references t_forum
(FOR_ID),
    constraint fk_forum_mensagem_usuario foreign key (USU_ID) references t_usuario
(USU_ID)
) type=InnoDB;

--
-- Dados dos fórums
create table t_forum_msg_dados
(
    FOR_MSG_ID             INTEGER                 not null,    -- Identificador da
mensagem
    FOR_MSG_DADOS       MEDIUMTEXT          null,       -- Mensagem
    primary key (FOR_MSG_ID),
    constraint    fk_forum_msg_dados    foreign    key    (FOR_MSG_ID)    references
t_forum_mensagem (FOR_MSG_ID)
) type=MyISAM;
create fulltext index ft_forum_mensagem_dados on t_forum_msg_dados (FOR_MSG_DADOS);

--
-- Fórums de curso
create table t_forum_curso
(
    FOR_ID              INTEGER             not null,
    CUR_ID              INTEGER             not null,
    primary key (FOR_ID, CUR_ID),
    index idx_forum_disciplina_forum (FOR_ID),
    index idx_forum_disciplina_curso (CUR_ID),
    constraint   fk_forum_curso_forum   foreign   key   (FOR_ID)   references   t_forum
(FOR_ID),
    constraint   fk_forum_cursp_disc    foreign   key   (CUR_ID)   references   t_curso
(CUR_ID)
) type=InnoDB;


--
-- Professores cadastrados
create table t_professor
(
    PROF_ID               INTEGER                 not null,    -- Identificador do
professor.
    USU_ID               INTEGER                not null,    -- Qual usuário é este
professor?
    PROF_CURRICULUM    MEDIUMTEXT           null    ,   -- Curriculum em XML
    primary key (PROF_ID),
    index idx_professor_usuario (USU_ID),
    constraint   fk_professor_usuario   foreign   key   (USU_ID)   references   t_usuario
(USU_ID)
) type=InnoDB;

--
-- Alunos de um curso
create table t_usuario_curso
```

```
(
    USU_CUR_ID          INTEGER              not null,   -- Identificador da relação
curso/usuário
    USU_ID              INTEGER             null,       -- Identificador do usuário
    CUR_ID              INTEGER             null,       -- Identificador do curso
    UC_SOLICITACAO      DATETIME              null    ,   -- Data da solicitação do
cadastro
    UC_EXPSOLICIT       DATETIME              null    ,   -- Data de expiração da
solicitação
    UC_INSCRICAO        DATETIME              null    ,   -- Data da inscrição do
aluno no curso
                                                      --    (quando    feita    a
solicitação/curso não restrito ou
                                                      --    quando  o  coordenador
confirma o cadstro, para curso restrito)
    UC_CONCLUSAO        DATETIME              null    ,   -- Data da conclusão do
curso pelo aluno
    primary key (USU_CUR_ID),
    index idx_usuario_curso_curso (CUR_ID),
    index idx_usuario_curso_aluno (USU_ID),
    unique index idx_usuario_curso_usr_curso (CUR_ID, USU_ID)
--     constraint fk_usuario_curso_curso    foreign key (CUR_ID) references t_curso
(CUR_ID),
--         constraint  fk_usuario_curso_usuario  foreign  key  (USU_ID)  references
t_usuario(USU_ID)
) type=InnoDB;

--
-- Coordenadores de um curso
create table t_coord_curso
(
    PROF_ID               INTEGER                 not null,    -- Identificador do
professor
    CUR_ID              INTEGER             not null,   -- Identificador do curso
    primary key (PROF_ID, CUR_ID),
    index idx_coord_curso_curso (CUR_ID),
    index idx_coord_curso_prof  (PROF_ID),
    constraint   fk_coord_curso_professor   foreign   key   (PROF_ID)   references
t_professor (PROF_ID),
    constraint fk_coord_curso_curso      foreign key (CUR_ID)   references t_curso
(CUR_ID)
) type=InnoDB;

--
-- Pré-requisitos
create table t_prerequisitos
(
    PREREQ_ID             INTEGER                 not null,    -- Identificador do
conjunto de pré-requisitos
    ITEM_ID             INTEGER             not null,   -- Identificador do item
    primary key (PREREQ_ID),
    index idx_prerequisitos_curso (ITEM_ID),
    constraint  fk_prerequisitos_item  foreign  key  (ITEM_ID)  references  t_item
(ITEM_ID)
) type=InnoDB;
create table t_itemprereq
(
    IP_ID               INTEGER               not null,   -- Identificador do pré-
requisito
    PREREQ_ID             INTEGER                 not null,    -- Identificador do
conjunto de pré-requisitos
    ITEM_ID             INTEGER             not null,   -- Item deste pré-requisito
    IP_TIPO             INTEGER             not null,   -- Tipo de pré-requisito
    primary key (IP_ID),
    index idx_itemprereq_item (ITEM_ID),
```

```
    index idx_itemprereq_prerequisitos (PREREQ_ID),
    constraint fk_itemprereq_item foreign key (ITEM_ID) references t_item
(ITEM_ID),
    constraint fk_itemprereq_prerequisitos foreign key (PREREQ_ID) references
t_prerequisitos (PREREQ_ID)
) type=InnoDB;


--
-- Relatório de acessos
create table t_acesso
(
    ACES_ID             INTEGER             not null,
    USU_ID              INTEGER             not null,
    ACES_ENTRADA        DATETIME            not null,
    ACES_SAIDA          DATETIME            null    ,
    primary key (ACES_ID),
    index idx_acesso_usuario   (USU_ID),
    constraint fk_acesso_usuario foreign key (USU_ID) references t_usuario (USU_ID)
) type=InnoDB;

--
-- Acompanhamento dos alunos em relação aos ítens de um curso
create table t_usuario_item
(
    UI_ID               INTEGER             not null,   -- Chave primária
    USU_ID              INTEGER             not null,   -- Identificador do usuário
    ITEM_ID             INTEGER             not null,   -- Identificador do item
    UI_PRIMACESSO       DATETIME               not null,    -- Data/hora do primeiro
acesso
    UI_ULTACESSO         DATETIME                not null,    -- Data/hora do último
acesso
    UI_CONCLUSAO        DATETIME            null,       -- Data/hora de conclusão
    UI_MENCAO           FLOAT               null,       -- Menção, caso haja.
    primary key (UI_ID),
    index idx_usuario_item_item    (ITEM_ID),
    index idx_usuario_disc_usuario (USU_ID),
    constraint fk_usuario_disc_item      foreign key (ITEM_ID) references t_item
(ITEM_ID),
    constraint fk_usuario_disc_usuario foreign key (USU_ID)   references t_usuario
(USU_ID)
) type=InnoDB;
```