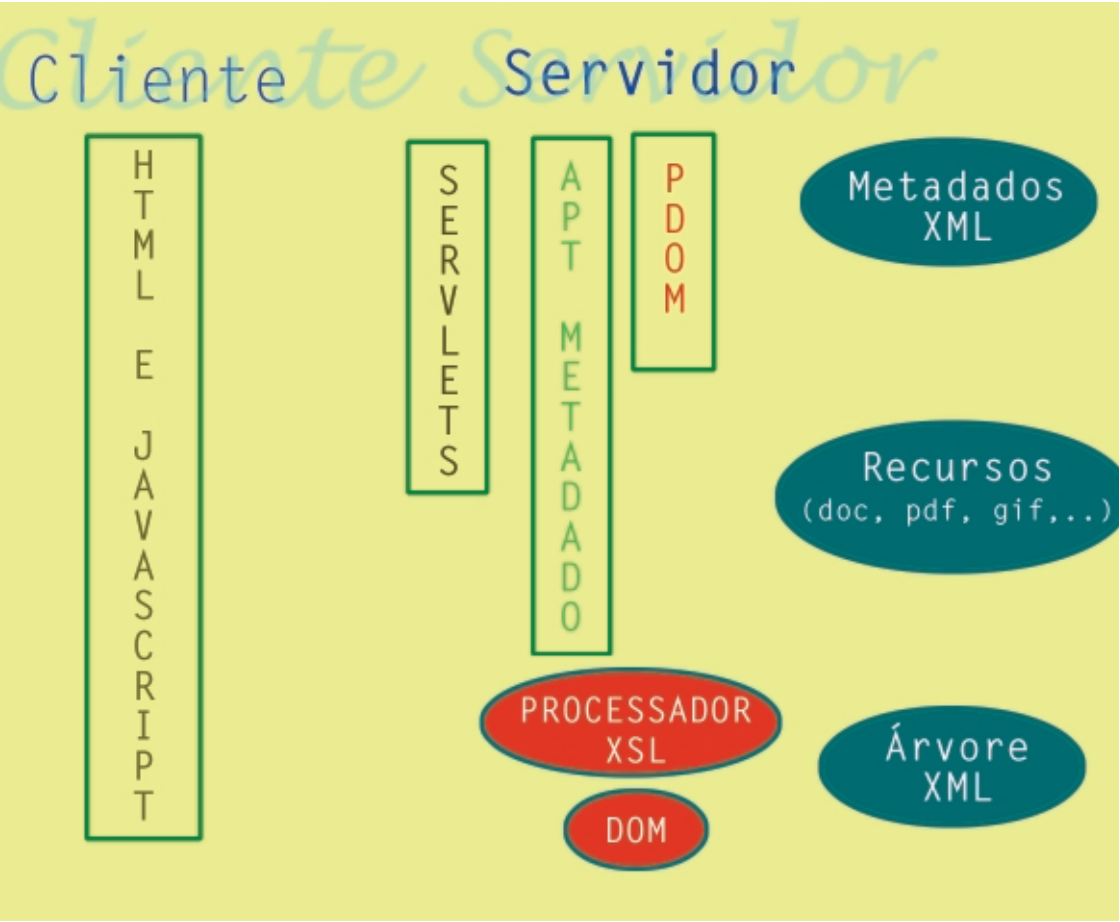
Ministério da Agricultura, Pecuária e Abastecimento

Boletim de Pesquisa e Desenvolvimento 6

Novembro, 2002

ISSN 1677-9266

Arquitetura de Software para a Manutenção de Dados da Agência de Informação Embrapa



Embrapa

República Federativa do Brasil

Fernando Henrique Cardoso
Presidente

Ministério da Agricultura, Pecuária e Abastecimento

Marcus Vinicius Pratini de Moraes
Ministro

Empresa Brasileira de Pesquisa Agropecuária - Embrapa

Conselho de Administração

Márcio Fortes de Almeida
Presidente

Alberto Duque Portugal
Vice-Presidente

Dietrich Gerhard Quast
José Honório Accarini
Sérgio Fausto
Urbano Campos Ribeiral
Membros

Diretoria Executiva da Embrapa

Alberto Duque Portugal
Diretor-Presidente

Bonifácio Hideyuki Nakasu
Dante Daniel Giacomelli Scolari
José Roberto Rodrigues Peres
Diretores-Executivos

Embrapa Informática Agropecuária

José Gilberto Jardine
Chefe-Geral

Tércia Zavaglia Torres
Chefe-Adjunto de Administração

Kleber Xavier Sampaio de Souza
Chefe-Adjunto de Pesquisa e Desenvolvimento

Álvaro Seixas Neto
Supervisor da Área de Comunicação e Negócios

ISSN 1677-9266
Novembro, 2002

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Boletim de Pesquisa e Desenvolvimento 6

Arquitetura de Software para a Manutenção de Dados da Agência de Informação Embrapa

Maria Fernanda Moura
Maria Angélica Andrade Leite
Silvio Roberto Medeiros Evangelista
Kleber Xavier Sampaio
Adriana Delfino dos Santos

Campinas, SP
2002

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. André Tosello, 209
Cidade Universitária "Zeferino Vaz" – Barão Geraldo
Caixa Postal 6041
13083-970 – Campinas, SP
Telefone (19) 3789-5743 - Fax (19) 3289-9594
URL: http://www.cnptia.embrapa.br
e-mail: sac@cnptia.embrapa.br

Supervisor editorial**:** *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel Fugisawa Souza*
Capa: *Intermídia Produções Gráficas*
Editoração eletrônica: *Intermídia Produções Gráficas*

1ª. edição
on-line - 2002

Arquitetura de software para a manutenção de dados da Agência de Informação Embrapa / Maria Fernanda Moura ... [et al.] – Campinas: Embrapa Informática Agropecuária, 2002.

25 p. : il. – (Boletim de Pesquisa e Desenvolvimento / Embrapa Informática Agropecuária ; 6)

ISSN 1677-9266

1. Ferramenta Agência. 2. Agência de Informação Embrapa. 3. Aplicativo Internet. I. Moura, Maria Fernanda. II. Título. III. Série.

CDD – 21st ed.
004.678

Sumário

Arquitetura de Software para a Manutenção de Dados da Agência de Informação Embrapa

Maria Fernanda Moura[1]
Maria Angélica Andrade Leite[2]
Sílvio Roberto Medeiros Evangelista[3]
Kleber Xavier Sampaio[4]
Adriana Delfino dos Santos[5]

Resumo

A Agencia de Informação Embrapa disponibiliza na internet informação qualificada e organizada e, muitas vezes, também aquelas geradas pela própria Embrapa. As soluções de software expostas neste trabalho são dirigidas ao gerenciamento dessas informações, que são armazenadas em base de dados centralizada e atualizadas via Internet por aplicativos deste sistema. O objetivo de apresentar essas soluções é contribuir para o desenvolvimento de sistemas com orientação metodológica similar. Este sistema teve como principal identificação de requisitos as falhas existentes na primeira versão do mesmo, que foi orientada exclusivamente para

[1] M.Sc. em Engenharia Elétrica, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

[2] M.Sc. em Ciência da Computação, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

[3] Ph.D. em Engenharia Elétrica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

[4] Ph.D. em Engenharia Elétrica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

[5] M.Sc. em Engenharia Elétrica, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

manipulação de dados formatados em XML. A nova versão traz uma arquitetura baseada nas orientações Java 2 Enterprise Edition (J2EE): modelo em camadas (orientação *Model View Controler* – MVC), uso de *containers* e sistema gerenciador de banco de dados. O resultado é um sistema mais robusto em seu todo, além das melhorias de manutenabilidade.

Termos para indexação: J2EE, XML, PDOM, Model view controller – MVC, Oracle.

A Software Architecture for Embrapa Information Agency Data Maintenance

Abstract

Agência de Informação Embrapa (Embrapa Information Agency) makes available through internet qualified and organized information, and sometimes those ones produced by Embrapa itself. This paper presents software solutions to manage those information, which are storaged in a central data base and updated by using Internet applications – these applications are developed in the context of this system. The goal of this paper is to contribute for the solutions of similar software systems. The requisite analysis for this system was based on the failures of the first version. The first version was completely embased in the exchange of XML data. The new version has the architeture based on the Java 2 Enterprise Edition (J2EE) recommendations: n-Tier model employing the Model-View-Controller (MVC) pattern; web container and a database. The result is a more realiable system, besides the manutenability improvements.

Index terms: J2EE, XML, PDOM, model view controller – MVC, Oracle.

Introdução

A Agência de Informação da Embrapa, em linhas gerais, corresponde a um conjunto de informações relevantes para o agronegócio, especializado por produto, organizado e armazenado em meio eletrônico, cujo principal objetivo é fornecer informação qualificada ao público em geral (produtores rurais, extensionistas, pesquisadores, técnicos, professores, estudantes, etc). Para isso, montou-se um repositório de conteúdos de informações, com acesso por produto, categorizados segundo os diferentes ambientes de consumo da informação (Embrapa Informática Agropecuária, 2002a). Para atingir esses objetivos foi construído um site onde são disponibilizadas as informações das diversas agências de produtos.

Para manter os dados desse site íntegros e constantemente atualizados, fez-se necessário construir um sistema automatizado para inserção, alteração e exclusão de dados das diversas agências, permitindo a troca e o reuso desses dados. Adicionalmente, o sistema precisava ter uma arquitetura cliente servidor a fim de ser utilizado nas diversas unidades descentralizadas da Embrapa mantendo os dados em uma base única acessível por todas.

O objetivo deste trabalho é apresentar as diversas soluções e métodos utilizados em cada etapa do desenvolvimento deste sistema e o porquê de suas escolhas, de modo a contribuir com o desenvolvimento de sistemas que utilizem orientação metodológica similar. Desta forma, neste trabalho são encontradas as descrições dos requisitos que motivaram o desenvolvimento deste sistema, seus objetivos, seus elementos, idéias gerais do modelos de dados e de classes desenvolvidos, os principais métodos e ferramentas utilizados, e, finalmente, os prós e contras identificados nas escolhas efetuadas.

Metodologia

A primeira versão do software responsável pelo tratamento de dados das diversas agências foi finalizada em abril de 2001. Essa versão fora desenhada para os seguintes elementos de dados:

- **Árvore do Conhecimento:** é uma organização hierárquica das informações sobre um determinado produto, desde seus requisitos de produção, processo de produção e processo de comercialização

e/ou industrialização. Ela tem seu embasamento na cadeia produtiva, sendo que cada nó possui um conteúdo textual que explica seu papel na cadeia e, ainda, referencia recursos de informação sobre o seu tema.

- **Recursos de informação:** são documentos publicados, ou em alguns casos não publicados, sobre os temas de interesse da agência. Nessa primeira versão, eram considerados apenas os recursos eletrônicos de informação, isto é, documentos em meio eletrônico, preferencialmente arquivos resultantes de editores de textos ou imagens.
- **Metadados de recursos:** são os dados sobre os recursos de informação, isto é, suas "fichas de catalogação". Para esses metadados havia sido escolhido o padrão Dublin Core (Dublin Core Metadata Initiative, 2002), recomendado pelo consórcio W3C (World Wide Web Consortium, 2002b), que é responsável por vários padrões de comunicação e representação de dados na Internet. Nesse padrão são guardados até 15 elementos de dados, passando por títulos da obra, autores, publicadores, endereços de onde encontrá-lo (URLs - ***U****niform* ***R****esource* ***L****ocator*s ou URIs - *Uniform Resource Identifiers*[6]), etc.[7].

Na versão abril/2001 do software, apenas os metadados de recursos possuíam um aplicativo completamente automatizado para a sua manutenção. A árvore era criada manualmente, digitada em um editor de textos e transportada para um editor XML - *Extensible Markup Language* (World Wide Web Consortium, 2002a). XML é um formato de texto simples, porém muito flexível, derivado do SGML (ISO, 2002) – de onde se deriva também o HTML - Hipertext Markup Language. O XML foi originalmente projetado para estender as soluções aplicadas às publicações eletrônicas em larga escala e vem desempenhando um crescente e importante papel na troca de uma grande variedade de dados na *web*.

Assim, a árvore era guardada no sistema como um arquivo XML, que para ser apresentado ao usuário final passava por um processador XSL -

---

[6] URI é o termo genérico para todos os nomes e endereços que se referem a objetos na *World Wide Web*. Uma URL é um tipo de URI.

[7] Maiores detalhes podem ser obtidos em Embrapa Informática Agropecuária (2002b).

Extensible Stylesheet Language. XSL é uma linguagem para expressar folhas de estilo, que especificam como uma classe de documentos XML deve ser apresentada, descrevendo as transformações para apresentação de cada instância da classe (World Wide Web Consortium, 2002c). As folhas de estilo contêm as regras de transformação, implicando na necessidade de um processador para essas regras. Os metadados dos recursos eram armazenados e recuperados em XML, o que implicava em também passá-los por um processador XSL para apresentá-los no navegador do cliente. A arquitetura desse conjunto de aplicativos é esquematizada na Fig.1.

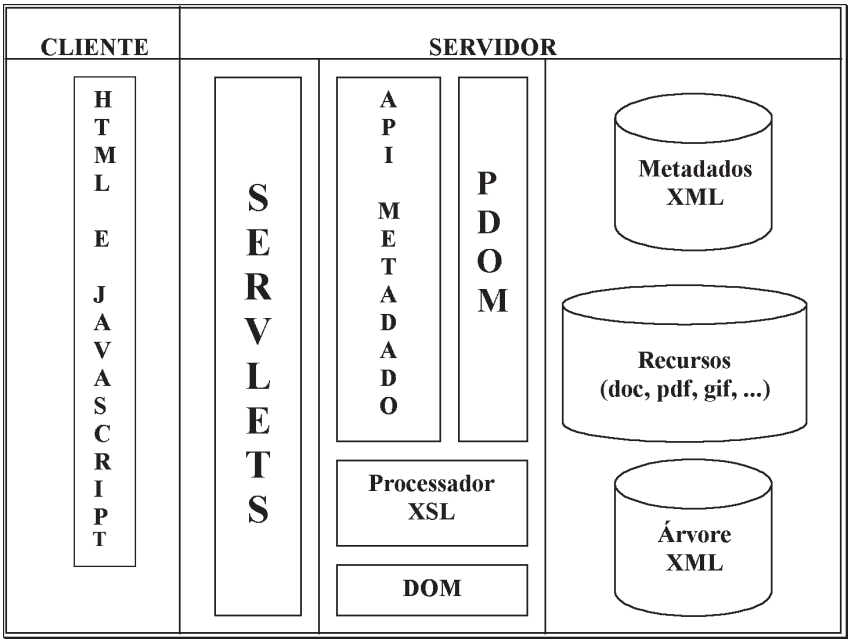


Fig. 1. Esquema da arquitetura da versão abril/2001.

Nota-se na Fig. 1 que, do lado cliente as partes executadas no navegador usuário são apenas HMTL (*Hipertext Markup Language*) e *javascript* (*scripts* JAVA™ que são executados no navegador); não são explorados outros recursos para processamento locais. Do lado servidor a arquitetura não era muito robusta, dado que se vinha trabalhando com arquivos XML

e não com algum tipo de gerenciamento de dados que permitisse manter integridade de acesso e consistências entre os dados. Um sério problema de consistência dessa versão era que os recursos de informação eram referenciados na árvore, mas não haviam checagens automáticas quanto aos recursos não terem sido removidos; dado que não havia um aplicativo de entrada de dados para a árvore. Também, alterar a árvore era uma tarefa árdua, dado a inexistência de aplicativo especificamente desenvolvido para essa finalidade. No entanto, deve-se notar que a arquitetura já previa evoluções do seu modelo tendo sido projetada em camadas razoavelmente independentes.

Os *servlets* (Zeiger, 2002) são módulos de código JAVA™ que rodam sob um servidor de aplicações para responder a requisições do cliente. Eles não são amarrados a um protocolo de cliente-servidor, porém são mais usados com o HTTP - ***H****yper****T****ext* ***T****ransfer* **P**rotocol. O protocolo HTTP é usado na *web* para definir como as mensagens são formatadas e transmitidas e quais ações os servidores e navegadores devem tomar em resposta a alguns comandos (Jupitermedia Corporation, 2002). Esses *servlets*, ilustrados na Fig.1, especificamente, tratavam as *http-queries* transformando-as em objetos do sistema e vice-versa, assim como transformavam objetos do sistema em formulários HTML. A API Metadados era responsável por trocar dados de uma *http-query* já tratada (dados convertidos para um objeto do sistema) com o modelo PDOM – *Persistent Document Object Model*.

Algumas definições são importantes, como DOM, PDOM e XQL. DOM – *Document Object Model* é uma interface de aplicação para documentos em XML e HTML, que define a estrutura lógica do documento e o modo como ele deva ser acessado e manipulado. O termo documento é genericamente empregado, dado que XML não é utilizado apenas para representar documentos textuais e, ainda, XML representa os dados como documentos enquanto o DOM é usado para gerenciá-los (Robie, 2002). O PDOM implementa toda a API W3C-DOM (API – *application program interface*) em arquivos binários indexados. Os documentos são analisados sintaticamente e então armazenados em uma forma binária, acessível para operações DOM que não exigirão outros processos de *parser* (análise sintática); e, ainda, há uma arquitetura com memória cache que melhora o desempenho. Um processador XQL pode ser usado para processar consultas em arquivos DOM. XQL é uma linguagem para consultas em arquivos XML, logo usa XML como modelo de dados e possui expressões facilmente analisáveis (GMD, 2002).

A partir dessa versão foram sentidas algumas dificuldades em relação às tecnologias utilizadas e, também, começaram a surgir novos requisitos que precisavam ser incorporados ao sistema, assim como a necessidade de construção de uma ferramenta para edição da árvore do conhecimento. Das dificuldades encontradas com a tecnologia, a mais importante foi a não adequação do modelo de armazenamento/acesso de dados. Promover um controle sincronizado dos acessos apenas com o uso de PDOM, e, conseqüentemente, manter a integridade dos dados era uma tarefa bastante complexa. O modelo PDOM ainda não era tão padronizado e não existiam ferramentas para cobrir essas necessidades, bem validadas; pensava-se em trocar o modelo por um gerenciador de banco de dados relacional ou orientado a objetos. Algumas outras dificuldades eram relativas à manutenção dos *servlets*, especialmente os que geravam código HTML. Como o código HTML era misturado ao código JAVA™ [8] que o gerava, estando todo o HTML dentro de chamadas a procedimentos de escrita JAVA™, tornava-se particularmente difícil mudar os leiautes desses formulários. Com essas dificuldades e mais o surgimento de novos requisitos, optou-se por desenvolver um novo sistema, utilizando-se as especificações válidas desse já desenvolvido.

Na versão abril do software, um esquema geral da arquitetura do sistema atual é apresentado na Fig. 2. O modelo adotado é o de camadas (ou *layers*), que ajuda a estruturar aplicações que podem ser decompostas em grupos de subtarefas. Cada grupo está em um particular nível de abstração (Bushman et al., 1996) – de modo que a comunicação entre os objetos seja sempre de uma camada para outra. Por exemplo, no caso a seguir, apenas as classes da camada de nome Básicas podem interagir com as classes da **API-BD** e vice-versa.

Em geral esse tipo de arquitetura é encontrado em sistemas de informação, que costumam ser divididos nas seguintes camadas: apresentação (no caso, os processos do navegador e os **JSPs**); aplicação (camada de **Aplicação** na figura); domínio (classes **Básicas**); e banco de dados (**API-BD**, **JDBC** e o SGBD Oracle 8i (Oracle Corporation, 2002)[9]).

---

[8] Para mais detalhes sobre a linguagem de programação JAVA consulte http://www.javasun.com/ .

[9] A melhor referência para o Oracle, de acordo com a visão dos autores, é o site do mesmo. Todos os artigos da OTN (*Oracle Technology Network*) relativos ao Oracle 8i foram consultados e considerados na análise do sistema de software; ou consulte Loney & Koch (2000).

Esse modelo facilita o reuso dos componentes criados, especialmente aqueles da camada do domínio e permite trocar camadas inteiras sem prejuízos do restante do sistema, o que facilita a adoção de padrões (trocar camadas inteiras por novos padrões) e mantém as dependências entre camadas sob controle (dependências locais) (Bushman et al., 1996).

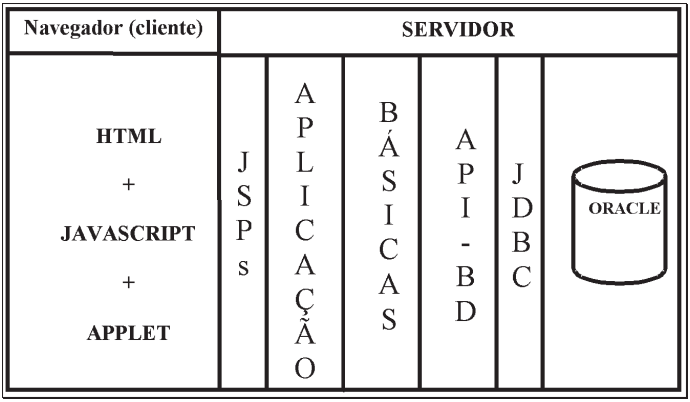


Fig. 2. Esquema da arquitetura atual.

As tecnologias e padrões adotadas no modelo seguem as recomendações de desenvolvimento da plataforma J2EE - JAVA 2 Enterprise Edition (Subrahmanyam et al., 2001). Usou-se no modelo da Fig. 2 a orientação de divisão em camadas mais as idéias do padrão de desenvolvimento MVC – Model-View-Controller (Subrahmanyam et al., 2001; Bushman et al., 1996)), no qual a idéia central é separar os dados (modelo, *model*) da forma de apresentação dos mesmos (interface com o usuário, *view*), colocando-se o controle entre eles (*control*). Na arquitetura atual tem-se basicamente o seguinte:

- **Dados**: usou-se o sistema gerenciador de bancos de dados Oracle (especificamente, o 8i, versão 8.1.7) com o driver **JDBC** (JAVA Database Connectivity) e uma API para encapsular as operações do banco de dados específico versus o padrão de dados do sistema – a **API-BD**;
- **Controle**: formado pelas classes da camada de **Aplicação** e as **Básicas**. Na camada de **Aplicação** tem-se todas as operações que o sistema precisa realizar com os dados (catalogação de recursos de informação, controle de edição da árvore do conhecimento, etc.)

e na camada **Básicas** tem-se as classes de manipulação de cada elemento do sistema (árvore do conhecimento, conteúdos dos nós da árvore, recursos de informação, etc.);

- **Forma de apresentação**: (*view*) corresponde à camada de **JSPs** e a parte do sistema que é executada (ou interpretada) no navegador do cliente (HTMLs e *Javascript*s). JSP – JAVA Server Pages, é uma extensão da tecnologia de *servlets*. Enquanto os *servlets* são programas JAVA™ os JSPs são textos, divididos em duas partes: conteúdo estático (HTML ou XML) e conteúdo dinâmico (*tags* JSP e/ou *scripts* JAVA™). O objetivo dos JSPs é facilitar a geração de hipertextos com conteúdos dinâmicos, podendo-se produzir múltiplas visões a partir do mesmo JSP. Os JSPs funcionam como templates: uma página de marcações com preenchimentos especiais (gerados pelas *tags* ou *scriplets*[10]). Na apresentação executada no navegador foram exploradas várias funcionalidades em *javascript* – para encapsular operações de consistência de dados dos formulários e até editores de dados, assim como o uso de um *applet* para navegação gráfica na árvore do conhecimento. *Applets* são programas escritos em JAVA™ que podem ser incluídos em uma página HTML, de forma similar à inclusão de uma imagem; o *applet* é executado no navegador (Sun Microsystems, 2002a).

O modelo de classes da camada **Básicas** teve sua construção baseada no modelo de entidades e relacionamentos (MER) dos dados. Primeiro foi projetado o MER e em seguida projetadas todas as classes **Básicas** – um trabalho inverso ao apresentado por Bourret (Bourret, 2002). No trabalho de Bourret cada classe (abstração dos objetos do sistema) dá origem a uma ou mais tabelas de dados, no caso deste sistema relacionou-se um a um cada tabela de dados com uma única classe.

Por exemplo, os recursos de informação são formados por catorze tabelas de dados: recurso (informações únicas, sem repetições), títulos (um ou mais títulos), criadores (zero ou mais criadores), publicadores (zero ou mais publicadores), identificadores (um ou mais identificadores), palavra-chave (uma ou mais palavras-chaves), direito (zero ou mais donos de direitos autorais), relações (zero ou mais relações com outras obras),

[10] *Scriplets* são *pequenos scripts*. A grande recomendação do uso de JSP é não sobrecarregar os *scriplets* com código JAVA.

cobertura (zero ou mais coberturas), categoria (uma ou mais categorias), colaboradores (zero ou mais colaboradores), usuário (um ou mais usuários responsáveis pela catalogação desses dados), perfil (um ou mais perfis de consumidores do recurso) e referência ao nó da árvore (zero ou mais referências). Para abstrair os recursos de informação foi criada uma classe Recurso, que possui entre seus atributos treze listas compostas por objetos de treze diferentes classes; pois, cada nova tabela de zero ou mais elementos mapea uma nova classe. E, completando o mapeamento do MER para o modelo de classes, os atributos de cada classe são nomeados exatamente da mesma forma que cada coluna das tabelas de dados (exemplo na Fig. 3).

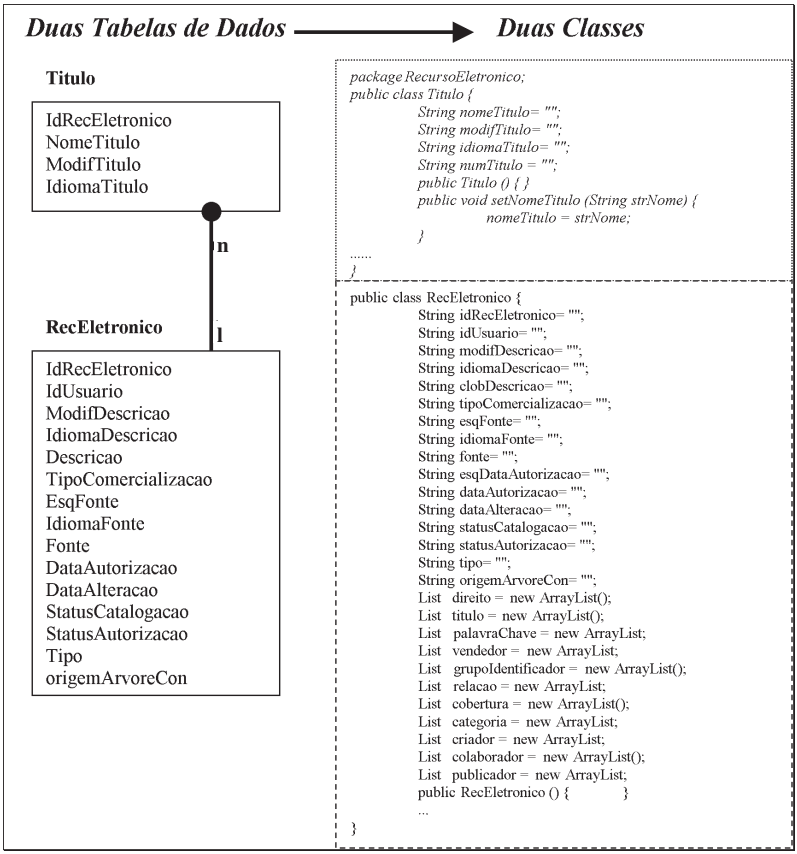


Fig. 3. Mapeamento do MER para classes.

Cada classe da camada **Básicas** possui métodos que permitem recuperar ou gravar uma instância delas na base de dados, passando por chamadas a métodos da camada **API-BD**. A principal função da **API-BD** é reconhecer os objetos do sistema e mapeá-los para as tabelas de dados específicas, encapsulando as características do sistema gerenciador de banco de dados utilizado.

A única classe da camada **Básicas** que possui um formato ligeiramente diferente do esperado é **ÁrvoreCon** (que corresponde ao elemento do sistema denominado árvore do conhecimento). O mapeamento esperado seria manter o modelo hierárquico da estrutura de uma árvore nas tabelas de dados e, então, manter métodos de recuperação e gravação dessa árvore como tal. Porém, manter tabelas de dados que reflitam a estrutura hierárquica de uma árvore é bastante complexo, dado que as *sql queries*[11] ficam complexas para serem projetadas e executadas. Adicionalmente, havia a necessidade de construir um editor para a árvore do conhecimento, isto é, uma ferramenta de entrada de dados para a estrutura da árvore e seus conteúdos com uma interface gráfica que permitisse visualizar a hierarquia entre os nós.

Para atender a necessidade da edição da árvore, buscou-se algum componente que oferecesse uma forma de visualização gráfica da hierarquia da árvore e pudesse ser integrado às demais ferramentas. Dos componentes analisados, escolheu-se a ferramenta Star Tree Studio™ (Inxight Software Incorporated, 2002). Esse componente permite editar uma árvore, criar toda a hierarquia de nós e gerar um arquivo XML com a especificação dessa árvore e/ou gerar um *applet* que permite navegá-la graficamente. O *applet* foi integrado à ferramenta de edição da árvore e, adicionalmente, o arquivo XML que a Star Tree Studio™ gera foi utilizado como a especificação da árvore que é armazenada no banco de dados. Dessa forma a classe **ÁrvoreCon** possui um atributo que corresponde a um objeto grande e não formatado (para o Oracle é o tipo CLOB – Character Large Object, que suporta até 4GB) equivalente ao conteúdo de um arquivo XML. A hierarquia entre os nós e a ligação com cada conteúdo textual do nó (os conteúdos pertencem a outras tabelas) é estabelecida somente quando o arquivo XML entra ou sai do sistema, através de uma análise sintática da especificação da árvore.

---

11 *Sql queries*: expressões SQL - *Standard Query Language* (Date & Darwen, 1993), que permitem a obtenção das tabelas de interesse com base nas tabelas existentes no banco de dados.

A camada de **Aplicação** contém tantas classes quantas são as funcionalidades identificadas no sistema, sendo as principais:

- **Catalogação de recursos de informação**: as classes que cobrem essa funcionalidade contêm métodos de inserção, alteração, exclusão e consulta a recursos;
- **Manutenção da árvore do conhecimento**: as classes envolvidas nessas funcionalidades possuem métodos para tratar *download* e *upload* de arquivos (que é como as especificações da árvore em arquivos XML entram ou saem do sistema), analisador sintático de XML e, métodos de checagem de consistências da árvore em cada passo executado. Como a estrutura da árvore é efetivamente editada na ferramenta Star Tree Studio™ que é para ambiente Windows, sempre que se deseje alterá-la será necessário copiar o XML guardado na base de dados (procedimento de *download*), editá-lo em ambiente Windows e entrar com a nova especificação no sistema (procedimento de *upload*);
- **Edição de conteúdos textuais dos nós de uma árvore**: engloba todas as classes utilizadas pela ferramenta de edição dos conteúdos e controla as relações entre as classes que compõem o conteúdo de um nó.

A camada de apresentação é dividida em **JSPs** do lado servidor e HTMLs e *javascript*s do lado cliente. Os **JSPs** foram projetados um para cada método das classes de **Aplicação** que manipulam dados, de modo que correspondem a cada formulário HTML de inclusão, alteração, exclusão ou consulta a dados, complementados pelas *tags* JSPs e *scriptlets* JAVA™ que geram os conteúdos dinâmicos desses formulários. Assim, também a camada **JSPs** é subdividida em classes correspondentes às da camada **Aplicação**.

Do lado cliente há formulários HTML de conteúdos fixos e bibliotecas *javascript* divididos em classes. As bibliotecas são utilizadas em checagens de valores de dados, tais como validação de datas, tamanhos de campos, intervalos de valores aceitos em determinados campos, e outros, formando uma classe **Miscelânea** (miscelânea de funcionalidades de tratamento de campos). Outra classe distinta, do lado cliente, é formada por funcionalidades de um editor de textos HTML, que foi integrado à ferramenta[12] de edição do

[12] Deve-se notar que o termo ferramenta, neste caso específico, está sendo usado para designar a integração de várias classes entre as várias camadas completando uma determinada funcionalidade do sistema.

conteúdo textual da árvore. Para construir e controlar o uso dos formulários de manipulação dos recursos de informação foi utilizada uma biblioteca, contendo várias classes distintas, capaz de trabalhar com formulários cujos campos podem ter uma ou mais ocorrências, apresentando-as à medida que são requisitadas. Todo esse processamento ocorre na máquina cliente, liberando o servidor desse controle e, principalmente, evitando idas e vindas de *queries http* do cliente para o servidor. Essa biblioteca tem sido utilizada junto a *templates* de entrada de dados com metadados no padrão Dublin Core e foi adaptada ao ambiente de software deste projeto.

Algumas tecnologias foram utilizadas na implementação desse sistema desde seu projeto, de acordo com as recomendações da plataforma J2EE – JAVA 2 Enterprise Edition:

- **Tomcat** – Jakarta-Tomcat-3.3: foi o *web container* utilizado. Um *container*, para a arquitetura J2EE, gerencia, em tempo de execução, os componentes da aplicação desenvolvidos de acordo com as especificações dessa arquitetura e, conseqüentemente, também provê o acesso entre as diversas APIs (Subrahmanyam et al., 2001). Especificamente um *web container* hospeda e gerencia JAVA serviste e páginas JSP. O Tomcat é desenvolvido sob o projeto JAKARTA (Apache Software Foundation, 2002);
- Linguagens de programação: foram utilizados JAVA™ em toda a parte servidor e *javascript* na parte cliente. Principalmente foram deixados na parte cliente as checagens de consistência de dados quanto a tamanhos, formatos, intervalos válidos, campos obrigatórios e um editor de HTML (construído em *javascript*);
- Banco de dados: foi usado o ORACLE 8i (Oracle Corporation, 2002), apenas como gerenciador do banco de dados.

Recursos adicionais: usou-se WinCVS como controlador de versões do software (DevGuy's..., 2002); jEdit como editor de código fonte (Pestov, 2002); o pacote com.oreilly.servlet (Hunter, 2002) para *uploads* e *downloads* de arquivos; o analisador sintático Aelfred XML Parser (O´Reilly & Associates, 2002) para XML e a ferramenta StarTree Studio™ (Inxight Software Incorporated, 2002) para edição da estrutura da árvore do conhecimento.

Resultados e Discussão

A arquitetura ficou mais leve e fácil de manter que a anterior, devido às recomendações seguidas e à maior robustez das tecnologias utilizadas. No entanto, não apenas a arquitetura atual utiliza um modelo em camadas, permitindo trocas de camadas sem efeitos colaterais para as demais, a anterior também utilizou esse modelo. A adoção de JSP e de uma camada de aplicação separando apresentação de controle é que permitiu à versão atual ter uma melhoria em seu modelo comparando-os aos *servlets* da versão anterior.

A troca do PDOM por um sistema gerenciador de banco de dados (SGBD) foi importante, dado que este era o principal problema de manipulação de dados na versão anterior. Pensava-se em manter o uso de XML nas camadas de importação e exportação de dados, por isso o primeiro sistema gerenciador de bancos de dados a ser avaliado foi o Oracle 8i. Essa versão do Oracle englobava um ambiente de ferramentas que era adequado para desenvolvimento web, além de ser capaz de armazenar dados em um modelo relacional e recuperá-los em *sql queries*[13] que resultavam em uma especificação XML dos dados – tratável por um processador XSL.

Esse ambiente foi estudado porque deveria permitir que se desenvolvesse aplicativos mais rapidamente e que se conservasse boa parte das tecnologias utilizadas na versão anterior; principalmente que a migração dos dados de uma para outra versão ocorresse de forma mais simples e sua manipulação fosse mais ou menos conservada. No entanto, por estabelecer uma fortíssima dependência do software em relação ao ambiente de desenvolvimento da Oracle e pelo fato das *sql queries* para recuperar XML não serem triviais optou-se por estudar mais soluções, ficando-se com a arquitetura J2EE e utilizando-se o Oracle apenas como SGBD, sob o JDBC e uma API do sistema. Desta forma, qualquer outro SGBD relacional poderia ter sido usado.

Com a escolha de uso da arquitetura J2EE e de JSPs na camada de apresentação, a primeira solução de implementação experimentada foi usar JAVA *beans* nas camadas de **Aplicação** e **Básicas**. JAVA *beans* são

---

[13] Uma *sql query* é uma expressão escrita em SQL (*Standard Query Language*), que é um padrão de especificação dessas *queries* para bancos de dados (Date & Darwen, 1993).

componentes (classes) reusáveis de código JAVA, que respeitam a um determinado padrão de interface, de modo a serem reconhecidos (por métodos de introspeção) e manipulados por outras aplicações, sem que seja necessário um pré-conhecimento dos mesmos. O modelo adotado para os *beans* neste projeto foi o JAVA Beans™ (Sun Microsystems, 2002b). O uso de *beans* teve como objetivo utilizar as *tags* JSP para importar e exportar dados entre formulários e *beans*, o que facilitaria muito a implementação e promoveria uma melhor separação entre apresentação e controle – dado que seriam mais usadas *tags* que *scriplets* nas páginas JSPs.

Aparentemente a escolha teria sido uma excelente solução, mas perdeu-se a idéia do uso de XML para exportar e importar dados, que a versão anterior preconizava. Este procedimento é bastante recomendável e vem se tornando um padrão para desenvolvimento *web* (Grønbæk et al., 2002). O XML facilita trocas de dados com outros sistemas de informação, especialmente no caso dos recursos de informação, as marcações do XML seguiam o padrão Dublin Core – exportável para bibliotecas virtuais que o vem utilizando ou utilizam o MARC (o formato MARC é o padrão para a representação e comunicação de informação bibliográfica em uma forma que possa ser lida por alguma máquina, para maiores detalhes de MARC, Dublin Core e conversões consulte Library of Congress (2002)). Poder-se-ia ter utilizado uma camada a mais entre o controle e a de apresentação para converter e reverter as especificações dos *beans* em XML, de modo a manter esse tipo de padrão. Há várias bibliotecas que podem ser utilizadas para esse fim, entre elas a Digester do Jakarta Project (Apache Software Foundation, 2002).

Outra solução que trouxe um problema colateral mais difícil de ser resolvido foi o uso da Star Tree Studio™ para gerenciar a estrutura da árvore do conhecimento. Ela trouxe a navegação sobre uma árvore hiperbólica para o editor de conteúdo dos nós (e também utilizada no site final), que promoveu um efeito gráfico excelente para o software, facilitando a localização de nós. Outro ganho valioso foi evitar o gerenciamento de uma árvore guardada em tabelas do banco de dados – o que implicaria em *queries* complicadíssimas e demoradas. No entanto, o efeito colateral é a dependência de uma ferramenta que precisa ser executada em um PC, sob ambiente Windows, implicando em *downloads* e *uploads* do arquivo XML de especificação da árvore. Quem controla esse vaivém da árvore do banco para o Windows é o editor responsável pela árvore, ou seja, um especialista do domínio da cadeia produtiva.

Caso esse editor não mantenha um trabalho estritamente organizado, ele pode acabar provocando perdas ou inconsistências de dados. Um problema dessa nova versão do software é conseguir uma integração mais direta dessa ferramenta com o restante do sistema desenvolvido.

Sobre o modelo de dados, houve uma inversão do processo natural, pois na programação orientada a objetos é o modelo de classes que leva ao modelo de dados. Neste caso, os dados foram analisados e modelados antes das classes, devido ao fato de a primeira preocupação ter sido migrar das DTDs originais para o modelo relacional. As DTDs são especificações dos modelos de documentos XML (World Wide Web Consortium, 2002a), no caso deste projeto era a especificação dos dados na versão abril de 2001. Não foi possível migrar diretamente de DTDs para o modelo relacional, pois durante o processo surgiram novos requisitos e novos elementos do sistema que foram modelados em novas tabelas. Com isso, o MER que ficou pronto antes do modelo de classes acabou por ditar o modelo da camada **Básicas**. A vantagem obtida foi que qualquer engenheiro de software da equipe do projeto que conhecesse o modelo de dados também conheceria o modelo de classes da camada **Básicas** e vice-versa.

Quanto ao item desempenho não se pode comparar as duas versões, tanto por serem arquiteturas que usaram soluções tecnológicas bastante diferentes, quanto pelo fato de estarem funcionalmente desiguais. Deve-se lembrar que na versão anterior não havia uma ferramenta específica para a edição da árvore e seus conteúdos e que o principal objetivo da nova versão era colocar os dados sob um SGBD robusto.

Deve-se notar que o site final, onde as informações de cada Agência de produtos serão disponibilizadas, não é coberto neste trabalho, pois o foco deste sistema é a manutenção dos dados. O site pode ser gerado a partir dos dados armazenados como se queira apresentá-los, seguindo-se o padrão MVC, bastando obter mais uma visão dos dados.

Conclusões

1. Os dados da versão atual encontram-se no banco de dados Oracle, permitindo que eles sejam melhor gerenciados.

2. Uma ferramenta para gerenciar os conteúdos da árvore do conhecimento com interface amigável, de acordo com os requisitos dos usuários, está presente na nova versão.
3. O modelo de desenvolvimento de software utilizado na versão atual é adequado à incorporação de evoluções e novas tecnologias, além de utilizar as técnicas mais recentes preconizadas para desenvolvimento *web*.
4. Uma melhor integração entre o gerenciamento da estrutura da árvore e o restante do sistema deve ser incorporada à versão atual.

Agradecimentos

À Laurimar Gonçalves Vendrúsculo, membro da equipe que iniciou os primeiros trabalhos de infra-estrutura para o projeto Agência. À Roberto Hiroshi Higa que construiu a API-PDOM da versão anterior e auxiliou a montagem da arquitetura anterior. À Marcia Izabel Fugisawa Souza e Maria das Dores Rosa Alves pelo trabalho de padronização da catalogação dos recursos de informação. À Ricardo Cassefo e Cármine Marrone que trabalharam na implementação na versão atual e a Maria Carolina Ribeiro da Silva pelos testes da versão atual. À Marcelo Narciso pelos auxílios de suporte computacional ao Oracle e ao Wagner Correa Ramos pelas consultas e dúvidas no uso do Oracle.

Referências Bibliográficas

APACHE SOFTWARE FOUNDATION. **The Apache Jakarta Project**: the Jakarta site: Apache Tomcat. Disponível em: <http://jakarta.apache.org/tomcat/>. Acesso em: 02 nov. 2002.

BOURRET, R. Mapping DTDs to databases. In: O´REILLY & ASSOCIATES. **XML.com**: XML from the inside out - XML development, XML resources, XML specifications. Disponível em: <http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html>. Acesso em: 30 out. 2002.

BUSHMAN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; SATL, M. **Pattern-oriented software architeture**. West Sussex: John Wiley, 1996. 468 p.

APACHE SOFTWARE FOUNDATION. **The Apache Jakarta Project**: Jakarta commons. Disponível em: <http://jakarta.apache.org/commons/digester.html>. Acesso em: 02 nov. 2002.

DATE, C.J.; DARWEN, H. A. **Guide to the SQL standard**. 3rd ed. Boston: Addison-Wesley, 1993. 414 p.

DEVGUY´S official CVS software list. Disponível em: <http://www.devguy.com/fp/cfgmgmt/cvs/software.htm>. Acesso em: 02 nov. 2002.

DUBLIN CORE METADATA INITIATIVE. **Dublin Core Metadata Initiative [home page]**. Disponível em: <http://www.dublincore.org/>. Acesso em: 23 out. 2002.

EMBRAPA INFORMÁTICA AGROPECUÁRIA. **Ferramentas automatizadas para a manutenção e acesso a dados das Agências de Informacão da Embrapa.** [Campinas]: Embrapa Informática Agropecuária: Embrapa Gado de Corte: Embrapa Informação Tecnológica, [2002]. Disponível em: <http://agenciahome.cnptia.embrapa.br/materialtreinamento/materialtreinamento.html>. Acesso em: Acesso em: 02 nov. 2002a.

EMBRAPA INFORMÁTICA AGROPECUÁRIA. **Manual de catalogação de recursos eletrônicos, versão 1.0**. Campinas, 2002. 48 p. Disponível em: <http://agenciahome.cnptia.embrapa.br/materialtreinamento/materialtreinamento.html>. Acesso em: Acesso em: 02 nov. 2002b.

GMD. **GMD-IPSI XQL engine**: version 1.0.2. Disponível em: <http://xml.darmstadt.gmd.de/xql/>. Acesso em:30 out. 2002.

GRØNBÆK, K.; SLOTH, L.; BOUVIN, N. O. Open hypermedia as user controlled meta data for the web. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE – THE WEB – THE NEXT GENERATION, 9., 2000, Amsterdam. **WWW9 proceedings**: table of contents. [S.l.]: Foretec Seminars, 2000. Disponível em: <http://www9.org/w9cdrom/index.html>. Acesso em: 02 nov. 2002.

HUNTER, J. What is the com.oreilly.servlet package? In: HUNTER, J. **Servlets.com - com.oreilly.servlet FAQ**. Disponível em: <http://www.servlets.com/cos/faq.html>. Acesso em: 02 nov. 2002.

INXIGHT SOFTWARE INCORPORATED. **Inxight [home page]**: download now. Disponível em: <http://www.inxight.com/products/stsv.php>. Acesso em: 01 nov. 2002.

ISO. **ISO 8879:**1986 information processing - text and office systems - Standard Generalized Markup Language (SGML). <http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=1638>. Acesso em: 01 nov. 2002.

JUPITERMEDIA CORPORATION. HTTP. In: JUPITERMEDIA CORPORATION. **Webopedia**: online dictionary for computer and Internet terms. Disponível em: <http://www.pcwebopedia.com/TERM/H/HTTP.html>. Acesso em: 30 out. 2002.

LIBRARY OF CONGRESS (ESTADOS UNIDOS). Network Development and MARC Standards Office. **Dublin core/MARC/GILS crosswalk**. Disponível em: <http://www.loc.gov/marc/dccross.html>. Acesso em: 02 nov. 2002.

LONEY, K.; KOCH, G. **Oracle 8i**: the complete. Berkely: Osborne/McGraw-Hill, 2000. 1314 p.

ORACLE CORPORATION. **Oracle8i –extending leadership to the Internet**. Disponível em: <http://technet.oracle.com/products/oracle8i/content.html>. Acesso em: 31 out. 2002.

O´REILLY & ASSOCIATES. **XML.com**: Ælfred XML parser for palm. In: O´REILLY & ASSOCIATES. Disponível em: <http://www.xml.com/pub/r/216>. Acesso em: 02 nov. 2002.

PESTOV, S.—**jEdit - open source programmer's text editor**. Disponível em: <http://www.jedit.org/>. Acesso em: 02 nov. 2002.

ROBIE, J. **What is the Document Object Model?**. Disponível em: <http://www.w3.org/TR/REC-DOM-Level-1/introduction.html>. Acesso em: 30 out. 2002.

SUBRAHMANYAM, A.; BUEST, C.; DAVIES, J.; JEWELL, T.; JOHNSON, R.; LONGSHAW, A.; NAGAPPAN, R.; SARANG, P. G.; TOUSSAINT, A.;TYAGI, S.; WATSON, G.; WILCOX, M.; WILLIAMSON, A.; O´CONNOR, D. **Professional Java server programming J2EE 1.3 edition**. Birmingham: Wrox Press, 2001. 1250 p.

SUN MICROSYSTEMS. Applets. In: SUN MICROSYSTEMS. **The source for Java technology**. Disponível em: <http://java.sun.com/applets/>. Acesso em: 01 nov. 2002a.

SUN MICROSYSTEMS. **Java 2 platform SE v1.3.1**: interface – java.beans DesignMode. Disponível em: <http://java.sun.com/j2se/1.3/docs/api/java/beans/DesignMode.html>. Acesso em: 02 nov. 2002b.

WORLD WIDE WEB CONSORTIUM. **eXtensible Markup Language (XML)**. Disponível em: <http://www.w3.org/XML/>. Acesso em: 28 out. 2002a.

WORLD WIDE WEB CONSORTIUM. **Metadata activity**. Disponível em: <http://www.w3.org/Metadata/Activity.html>. Acesso em: 28 out. 2002b.

WORLD WIDE WEB CONSORTIUM. **The eXtensible Stylesheet Language (XSL)**. Disponível em: <http://www.w3.org/Style/XSL/>. Acesso em: 28 out. 2002c.

WORLD WIDE WEB CONSORTIUM. **World Wide Web Consortium – Extensible Markup Language (XML) 1.0 (second edition)**: W3C recommendation 6 October 2000. Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: 30 out. 2002d.

ZEIGER, S. **Servlets essentials**: version 1.3.5. Disponível em: <http://novocode.de/doc/servlet-essentials/chapter1.html#1_1>. Acesso em: 28 out. 2002.


Embrapa
Informática Agropecuária