ARQUITETURA DO GERENCIADOR DE CONTEÚDO DA AGÊNCIA DE INFORMAÇÃO EMBRAPA

Sergio Aparecido Braga da Cruz
sergio@cnptia.embrapa.br

Maria Fernanda Moura
fernanda@cnptia.embrapa.br

Adriana Delfino dos Santos
adriana@cnptia.embrapa.br

Maria Angélica de Andrade Leite
angelica@cnptia.embrapa.br

Embrapa Informática Agropecuária
Av. André Tosello, 209 - Barão Geraldo - Caixa Postal 6041
13083-886 - Campinas, SP
Tel: (19) 3789-5700 - Fax: (19) 3789-5711

RESUMO

O Gerenciador de Conteúdo da Agência de Informação Embrapa tem o objetivo de organizar e de tornar disponível, via Internet, informação digital relativa às cadeias produtivas do agronegócio, contextualizando-a para os diferentes públicos consumidores. Este gerenciador é materializado na forma de um serviço de Intranet que provê a edição do conteúdo para os diversos centros de pesquisa da empresa de modo colaborativo e distribuído. Usando este serviço os usuários do sistema podem gerar um site que possibilita o acesso a diversas informações e serviços para os consumidores finais da informação. Este trabalho apresenta a experiência na definição da arquitetura de software empregada no desenvolvimento deste gerenciador e discute os resultados obtidos.

Palavras-chave: sistemas de informação, Java, *Model-View-Controller pattern*

ABSTRACT

The goal of the Gerenciador de Conteúdo da Agência de Informação Embrapa (Embrapa Information Agency Content Manager) is the organization and Internet presentation of digital information about the agribusiness productive chains, and also put this information in a context that can reach different profiles of consumers. The Content Manager was built as a site through which the access of different services and information are provided. There is an Intranet service too, that provides the content edition in a collaborative and distributed workflow through several Embrapa research centers. This paper presents the software solutions architecture employed in the content manager development and some discussions about their results.

Key words: information systems, Java, *Model-View-Controller pattern*

1 Introdução

O Gerenciador de Conteúdo da Agência de Informação Embrapa é um sistema de informação desenvolvido pela parceria entre os centros de pesquisa: Embrapa Informação Tecnológica, Embrapa Gado de Corte e Embrapa Informática Agropecuária. A sua criação foi motivada pela necessidade de manutenção eficiente do conteúdo de um portal *web* que organiza informações técnico-científicas resultantes dos projetos de pesquisa da Empresa Brasileira de Pesquisa Agropecuária – Embrapa. Este portal é conhecido como Agência de Informação Embrapa, ou simplesmente Agência.

Na Agência, as informações estão organizadas sob várias Agências de Produtos, onde cada uma delas agrupa informação técnica relevante para o agronegócio a respeito de um produto específico. Esta informação está estruturada sob a ótica da cadeia produtiva do produto e atende a diferentes perfis de consumidores, tais como: extensionistas, pesquisadores, técnicos, produtores rurais, professores, estudantes, etc. Para isso, cada Agência de Produto utiliza uma metodologia de organização da informação desenvolvida especialmente para este fim, a qual estrutura o conhecimento de uma cadeia produtiva numa forma hierárquica, denominada *árvore do conhecimento*. Nos primeiros níveis desta árvore, que são aqueles mais próximos à raiz, estão os conhecimentos mais genéricos e nos níveis mais profundos, estão os conhecimentos específicos. Cada nó desta árvore contém um texto sobre um tema relacionado ao seu produto, onde este texto é resultante da compilação do conhecimento já produzido por pesquisadores, técnicos extensionistas e agricultores, podendo ser complementado através da referenciação de recursos de informação disponíveis em meio eletrônico. Estes recursos, que incluem outros textos, imagens, mapas, vídeos, sons e bases de dados, etc. ..., são descritos usando o padrão de metadados Dublin Core (OCLC, 2003), recomendado pela W3C (SOUZA et al, 2002). A **Figura 1** apresenta um texto de nó de árvore do conhecimento. A árvore do conhecimento é representada graficamente por meio de uma árvore hiperbólica, ilustrada na **Figura 2**. A navegação entre os nós da árvore pode ser realizada através dos *hyperlinks* existentes em cada conteúdo de nó ou através da árvore hiperbólica. Também são fornecidos serviços de busca e *bookmark* (cesta de documentos). O conteúdo da Agência é o resultado da integração do trabalho das diversas equipes de Agência de Produto, que em geral estão geograficamente distribuídas e devem trabalhar de forma colaborativa, seguindo os procedimentos de edição pré-estabelecidos.

Para dar suporte à construção e manutenção do portal Agência foi desenvolvido o Gerenciador de Conteúdo da Agência que visa atender às necessidades das várias equipes, ou seja, armazenar e disponibilizar o seu conteúdo garantindo a sua integridade. Além disto, permite uma fácil integração da informação armazenada, e dos seus processos de transformação, com outros sistemas de informação da Embrapa. O gerenciador foi desenvolvido em plataforma *World Wide Web* – WWW, utilizando tecnologia JAVA (SUN, 2003) e sob o padrão de projeto *Model-View-Controller* – MVC (BUSCHMANN et al, 1996).

O foco deste trabalho é apresentar a arquitetura utilizada no desenvolvimento do gerenciador de conteúdo da agência, descrevendo as tecnologias empregadas e mostrando como estas foram utilizadas na implementação. A seção 2 apresenta as funcionalidades do gerenciador e a motivação para o seu desenvolvimento. A seção 3 descreve a arquitetura adotada e a seção 4 ilustra um exemplo de aplicação. Os resultados e conclusões deste trabalho são mostrados na seção 5.

Figura 1: Nó Adubação da Árvore do Conhecimento da Cultura do Feijoeiro

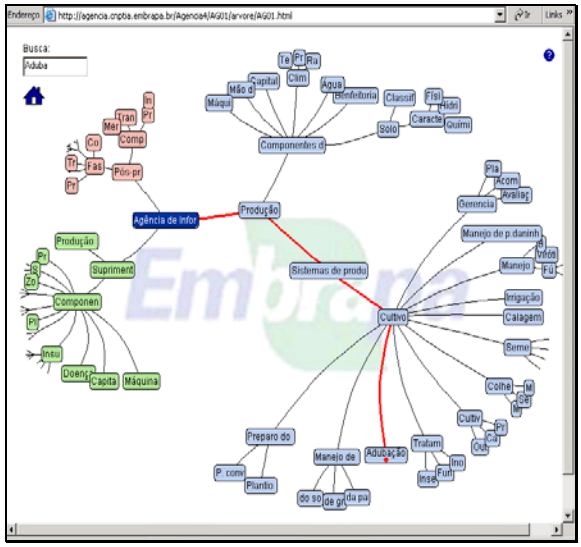

Figura 2: Navegação na Árvore do Conhecimento da Cultura do Feijoeiro

2 Desenvolvimento do Gerenciador de Conteúdo

Existem diversas ferramentas que possibilitam o gerenciamento de conteúdo de portais tais como Zope (ZOPE, 2003) e OpenCMS (ALKACON, 2003). Entretanto, segundo Souza (2002), os resultados de busca em grandes portais que utilizam tais ferramentas, apresentam

baixa revocação (razão entre o número de documentos recuperados relevantes e o número de documentos relevantes existentes) e baixa precisão (razão entre o número de documentos recuperados relevantes e o número de documentos recuperados). Além disso, percebe-se que a informação recuperada é de baixa qualidade, pois os resultados recuperados geralmente diferem daquilo que efetivamente se busca. A principal causa deste comportamento é a falha dos mecanismos de busca em organizar conceitualmente os documentos em uma maneira mais próxima às necessidades do usuário. Buscando atender aos requisitos específicos do projeto e também melhorar o resultado obtido com o uso dessas ferramentas, decidiu-se pelo desenvolvimento de um novo gerenciador para a Agência.

Na Agência, a solução adotada para melhorar a revocação, precisão e qualidade da informação recuperada pode ser dividida em duas partes. A primeira corresponde à adoção de metadados para caracterização das unidades de informação utilizando o padrão Dublin Core e definindo um procedimento criterioso para preenchimento dos seus elementos com a finalidade de caracterizar cada documento da forma mais precisa possível. Com isto espera-se uma melhoria dos resultados de busca na Agência. A segunda corresponde à especificação de conceitos para estabelecimento de linguagem padrão para intercâmbio e reuso de conhecimentos (SOUZA, 2002). Estes conceitos são materializados na forma da árvore de conhecimento, que é utilizada tanto no direcionamento do processo de integração de novos conceitos, quanto na navegação no site pelo usuário final. Cada conceito é representado na árvore por um nó.

A necessidade de prover uma ferramenta que possibilitasse a catalogação de recursos eletrônicos no padrão Dublin Core e oferecesse facilidades para organização dos conceitos das cadeias produtivas em uma árvore do conhecimento reforçou a motivação para o desenvolvimento de um novo gerenciador de conteúdo para portais.

O Gerenciador de Conteúdo da Agência de Informação Embrapa implementa quatro processos básicos de gestão de uma Agência de Produto: Edição de Árvore do Conhecimento, Edição de Nó da Árvore, Catalogação de Recursos de Informação e Publicação de Conteúdo. Estes processos estão representados na **Figura 3**. A execução dos mesmos é realizada por uma equipe editorial multidisciplinar composta por editores (especialistas da cadeia produtiva, de comunicação e de propriedade intelectual) e profissionais da área de informação.

Os especialistas da cadeia produtiva definem a estrutura hierárquica do conhecimento e a manipulam graficamente através do processo de *Edição de Árvore do Conhecimento*. Através do processo de *Edição de Nó da Árvore* estes especialistas, juntamente com os da área de comunicação, elaboram os textos de cada nó em uma linguagem apropriada para os consumidores de informação, selecionam e referenciam recursos de informação que complementam estes textos. Os profissionais da área de Informação descrevem, classificam e atribuem palavras-chave aos recursos de informação selecionados através do processo *Catalogação de Recursos de Informação*. Os especialistas da área de propriedade intelectual são responsáveis pelo controle de autorização de publicação de todo recurso de informação disponível na Agência e definem as condições de autorização a serem utilizadas pelo processo de *Publicação de Conteúdo*. Neste processo, os conteúdos armazenados são transformados em páginas estáticas, formando um *website* de informações sobre determinada cadeia produtiva. Durante esta transformação, as páginas geradas são indexadas para posteriormente serem pesquisadas pelo mecanismo de busca disponível no *website* gerado.

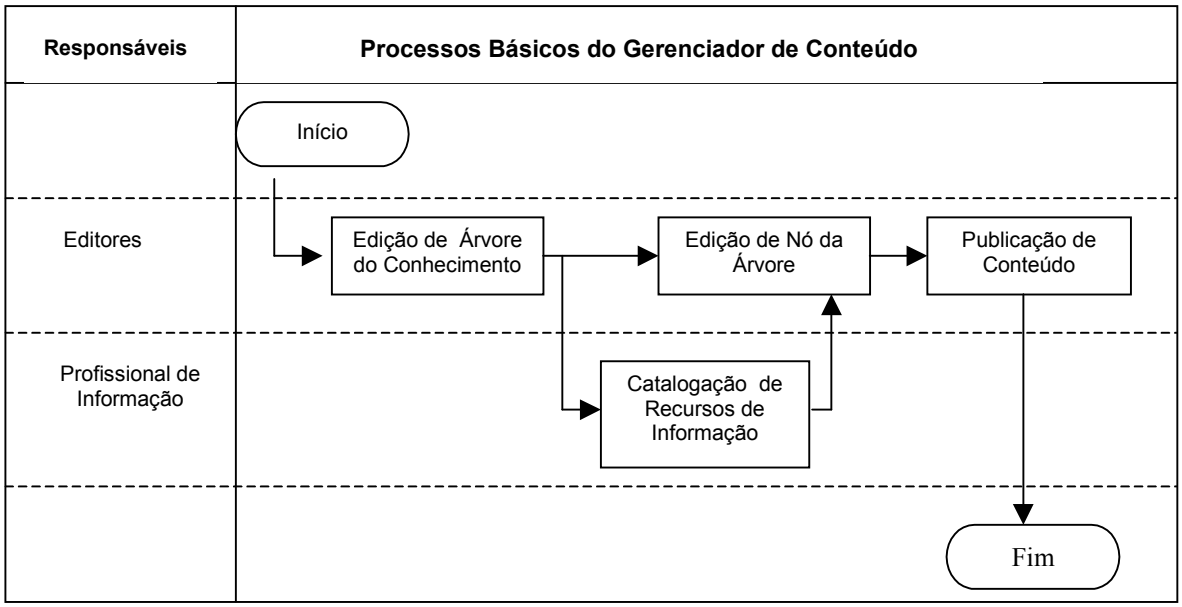

Figura 3: Esquema dos processo básicos do Gerenciador de Conteúdo da Agência de Informação Embrapa

3 Arquitetura do Gerenciador de Conteúdo

O Gerenciador de Conteúdo pode ser dividido em duas partes principais: aplicação e API (*Application Program Interface*). A definição de uma API propicia o encapsulamento de

funcionalidades e a flexibilidade de uso de componentes em diferentes aplicações. A aplicação consiste na implementação de uma solução específica utilizando a API, que neste caso é o próprio Gerenciador. Para desenvolvimento do sistema foi escolhida a tecnologia Java que é uma linguagem orientada a objetos. Neste contexto, foram inicialmente identificados os principais componentes (classes) necessários. Estes componentes definem a API e segundo o padrão de projeto MVC (*Model-View-Controller*) compõem a camada *Model*, a qual determina basicamente as funcionalidades e a lógica de negócio do gerenciador. A camada *View*, correspondente à representação visual dos dados, é implementada com o apoio de recursos da WWW, sendo dependente da aplicação. A camada *Model,* por sua vez, inclui todo o suporte a funcionalidades de persistência, a qual provê uma interface para recursos de armazenamento de longo prazo. Na Figura 4 é apresentado um esquema geral da arquitetura do gerenciador. A camada *Controller*, na aplicação, corresponde à parte relacionada com o tratamento da interação com o usuário.

Através de uma definição de interface clara entre as camadas do padrão MVC podemos obter as seguintes vantagens:

- podem ser criados vários componentes definindo vários *Views* para apresentação diferenciada de um mesmo dado. Estes *Views* podem ser usados simultaneamente, permitindo que um mesmo dado seja apresentado de várias maneiras ao mesmo tempo. Os *Views* que serão utilizados podem ser selecionados em tempo de execução. Isto significa que a representação das informações geradas pelo gerenciador podem ser alteradas sem afetar as demais camadas do esquema MVC;
- a separação em camadas do MVC permite a troca dos componentes na camada *View* e na *Controller* sem afetar a camada *Model*. As camadas *View* e *Controller* definem a interface do gerenciador, que poderá ser então substituída completamente, até mesmo em tempo de execução, sem afetar a camada de processamento de dados principal;
- a baixo acoplamento entre a interface (camadas *View* e *Controller*) e a camada de processamento principal (*Model*) aumenta a portabilidade do sistema interativo, uma vez que a interface pode ser alterada para se adequar aos elementos de interface disponíveis para uma nova plataforma. A camada *Model* representada pela API poderá ser utilizada no desenvolvimento de uma aplicação convencional (fora da WWW) sem necessitar de modificação.

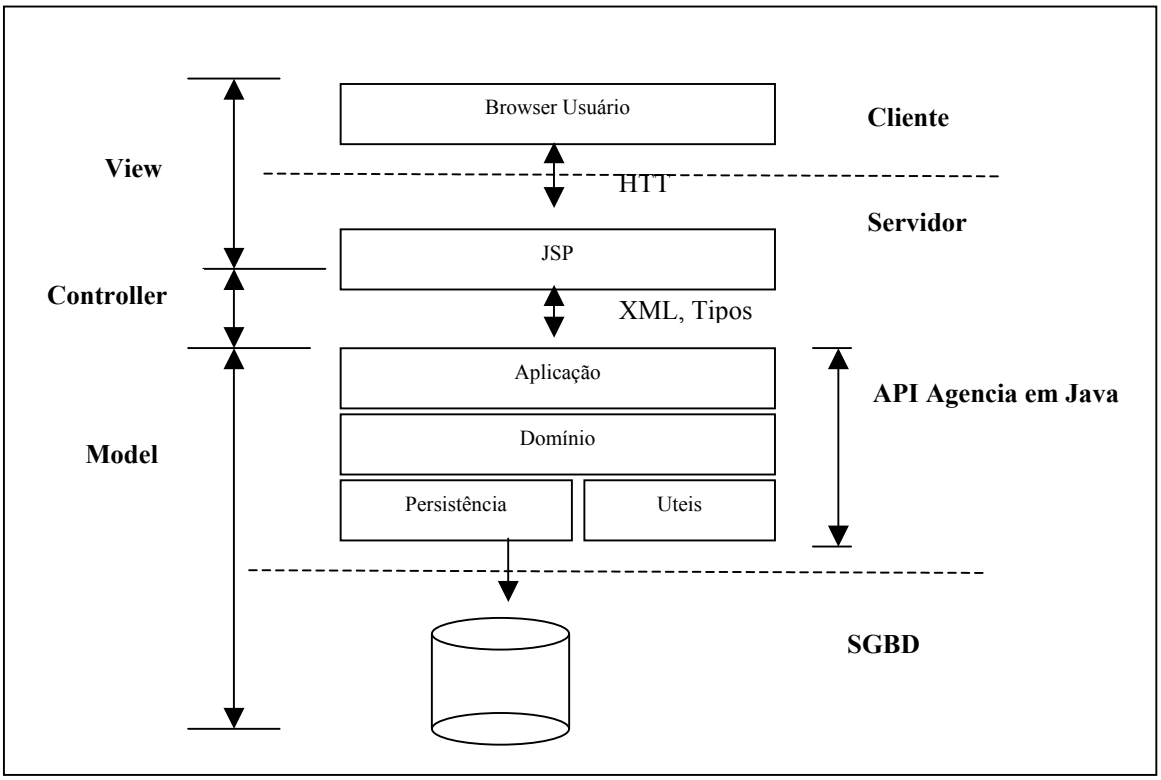

Figura 4: Esquema de camadas no padrão MVC

3.1 Camada de Apresentação (*View*)

A camada de apresentação do Gerenciador de Conteúdo é implementada utilizando a tecnologia JSP e Servlets, que são soluções na plataforma Java para construção da parte servidor das aplicações. A representação visual final, que é aquela vista pelo usuário do sistema, corresponde a páginas na linguagem HTML, as quais podem ser enriquecidas com Applets e scripts em JavaScript, facilitando a navegação entre as diversas páginas geradas pelo gerenciador, que utiliza scripts JSP como *templates*.

O site da Agência de Informação gerado pelo Gerenciador de Conteúdo, também faz parte da camada de apresentação do sistema, uma vez que corresponde a um relatório navegável e *off-line* do conteúdo do gerenciador. Na geração do site, também é utilizada a ferramenta *Velocity* (JAKARTA, 2003a) que permite definir e interpretar *templates.*

3.2 Camada de Controle (*Controller*)

A camada de controle é responsável pela "reação" do sistema às interações realizadas pelo usuário. Na plataforma WWW estas interações correspondem à geração de requisições HTTP que devem ser atendidas pelo sistema. A abordagem adotada na implementação desta solução foi utilizar o esquema Modelo JSP 2 (ALLAMARAJU et al, 2001). Neste esquema a responsabilidade pelo tratamento de requisições e pela apresentação de dados é dividida entre uma servlet ou um script JSP controlador principal e um conjunto de páginas de apresentação de dados. Esta abordagem tem como vantagem reduzir a complexidade que pode ser gerada caso o tratamento de requisição e de apresentação de dados seja realizado por um mesmo script JSP (Modelo JSP 1).

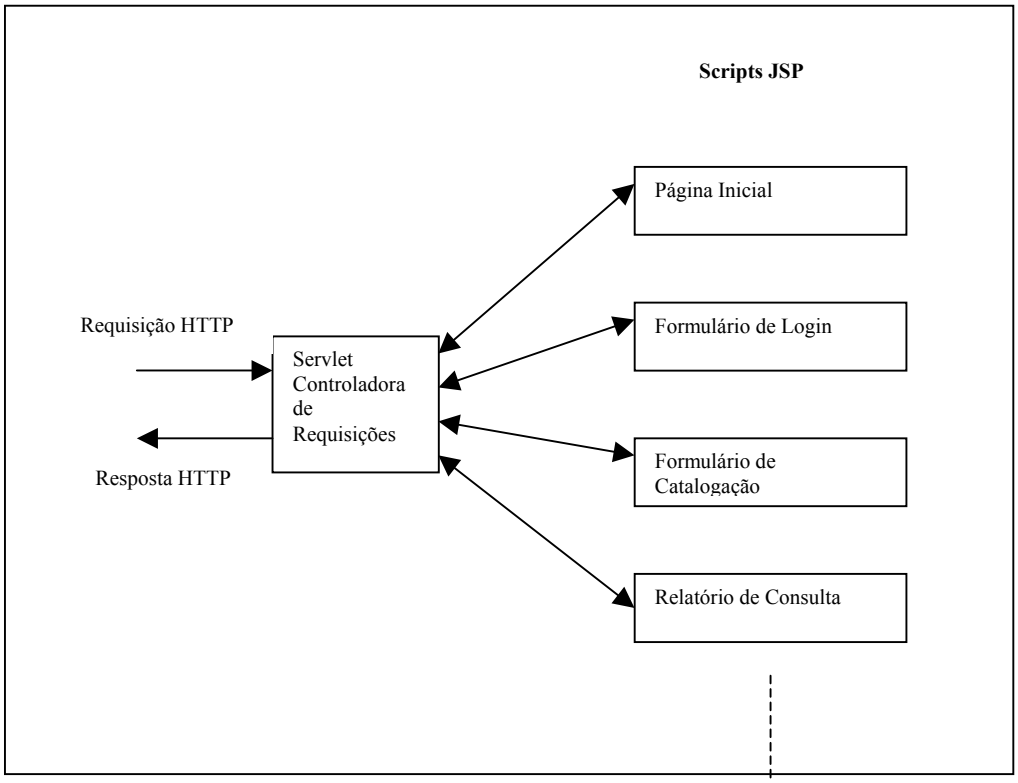

Figura 5: Modelo JSP 2

Neste esquema, ilustrado na Figura 5, a servlet controladora recebe requisições HTTP e executa o processamento adequado. Ao final deste processamento, a servlet controladora dispara a execução de um dos scripts JSP para apresentação de uma resposta ao usuário. Os dados trocados entre a servlet controladora e os scripts JSP estão encapsulados em classes Java.

Para realizar este processamento, a servlet controladora comunica-se com a camada *Model* através de chamadas de métodos da API.

3.3 Camada do Modelo (*Model*)

A Camada *Model* do Gerenciador de Conteúdo é representada pelo conjunto de classes que definem uma API. Esta API foi definida de maneira modular e genérica o suficiente para que partes dela possam ser utilizadas na construção de outras aplicações relacionadas às funcionalidades do Gerenciador de Conteúdo. Em Java, esta modularidade é representada pelos *packages*, que agrupam as classes que implementam a API. A **Figura 6** apresenta, segundo a notação UML (*Unified Modeling Language*), os *packages* atualmente definidos na API do Gerenciador de Conteúdo.

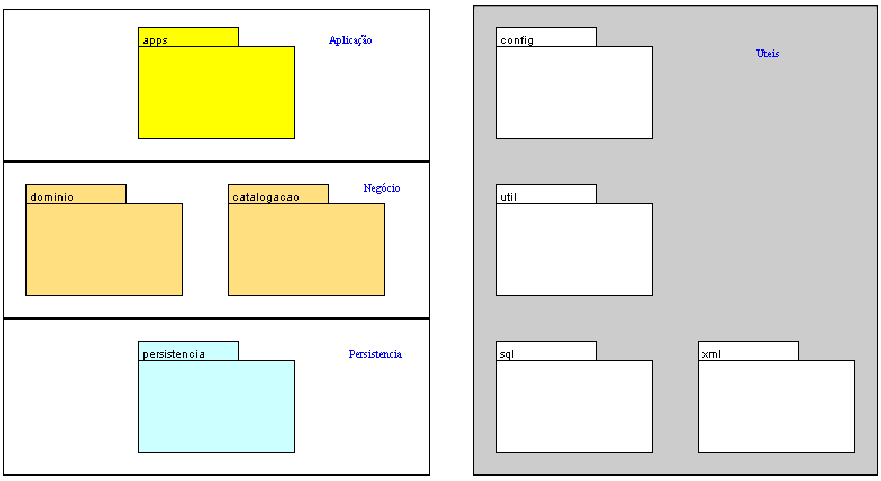

Figura 6: Packages da API do Gerenciador de Conteúdo

O *package domínio* agrupa classes Java relacionadas com o processo de edição de uma árvore do conhecimento e de edição de nós da árvore do conhecimento. O *package catalogacao* agrupa classes relacionadas com o processo de catalogação de recursos de informação e define um *subpackage* específico para catalogação de recursos de informação para o padrão de metadado Dublin Core. O *package apps* agrupa classes que implementam a lógica de négocio do Gerenciador de Conteúdo.

O *package persistencia* implementa a comunicação dos *packages dominio* e *catalogacao*, com serviços de persistência atualmente fornecidos por um SGBD Oracle. Os *packages config*, *util*, *sql* e *xml* foram definidos por conveniência e são usados, quando necessário, pelos outros *packages* da API.

As classes do *package persistencia* implementam a funcionalidade de persistência de maneira transversal ao conjunto de classes da camada de negócio, não existindo, então, relacionamento de hierarquia entre elas. Desta maneira, o serviço é implementado através da delegação desta tarefa para as classes de persistência. No caso do *package persistencia* da API do Gerenciador de Conteúdo foi utilizado o padrão de projeto DAO (Data Access Object) (SUN, 2003b) ilustrado na **Figura** 7 na forma de um diagrama de classes em UML.

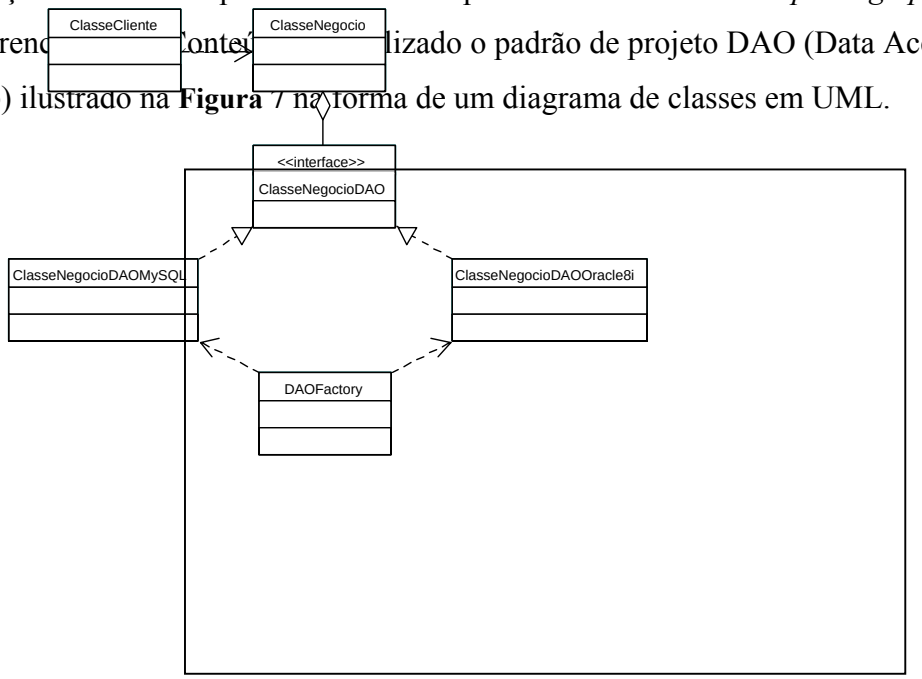

Figura 7: Padrão de projeto DAO

Neste padrão de projeto, uma classe de negócio, representada no diagrama por ClasseNegocio, referencia uma interface que será responsável pela persistência de seus dados. Esta interface pode ser implementada por diversas classes que realizam efetivamente o serviço de persistência e que devem obedecer rigidamente o comportamento determinado por esta interface. Desta forma, existe um baixo acoplamento entre a ClasseNegocio e a classe que realiza o serviço de persistência (ClasseNegocioDAOMySQL e ClasseNegocioDAOOracle). Assim, várias classes de persistência podem ser implementadas para os mais diversos bancos de dados sem que isto afete a implementação da classe de negócio. Porém, toda implementação feita deverá obrigatoriamente respeitar a interface ClasseNegocioDAO.

4 Exemplo de Aplicação

A título de exercício foi construído um protótipo da aplicação utilizando a arquitetura descrita no artigo. Este protótipo segue o esquema de seqüência de páginas apresentado na **Figura 8**, onde os retângulos representam páginas apresentadas ao usuário, as setas indicam

hyperlinks entre as páginas e as setas rotuladas indicam processamento de requisição. Este protótipo segue o esquema Modelo JSP 2 e utiliza como servlet controladora a ferramenta Struts (JAKARTA, 2003b).

Na página inicial encontra-se um *hyperlink* para um formulário de *login*. Ao se submeter os dados preenchidos neste formulário para a aplicação, é executado um tratamento específico para esta requisição. A decisão de qual tratamento será dado à requisição é realizada pela ferramenta Struts a partir do seu arquivo de configuração. Neste caso, os dados de *login* e de senha são pesquisados na base de dados para autenticação do usuário. Após o tratamento da requisição, e sendo um usuário válido, a aplicação apresentará um menu de opções; caso contrário uma mensagem de erro será emitida. No menu de opções, o usuário poderá selecionar a ação desejada: inserção de um novo usuário, alteração dos dados de um usuário cadastrado, listagem de usuários cadastrados ou sair da aplicação (*logout*).

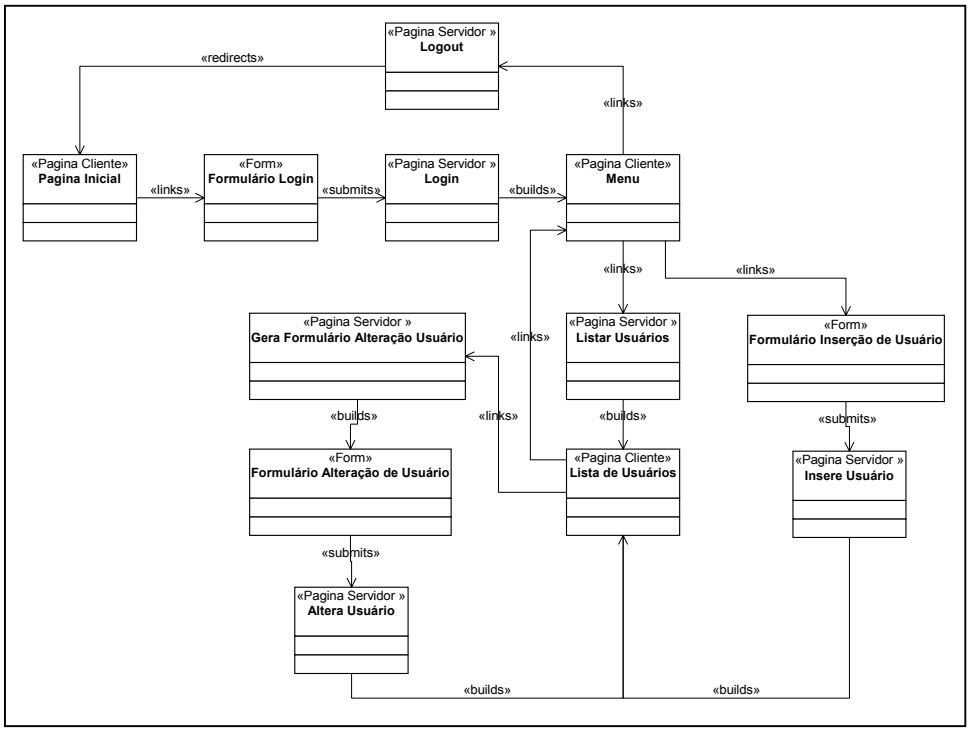

Figura 8: Esquema de páginas de protótipo

Caso o usuário siga o *hyperlink* para inserção de um novo usuário, será apresentado um formulário com um exemplo para o preenchimentos dos dados. Estes dados devem ser preenchidos em formato XML (eXtensible Markup Language) (W3C, 2003), de acordo com o exemplo apresentado. A confirmação da operação de inserção gera uma requisição que será

tratada, utilizando-se de classes de todas as camadas da API do Gerenciador de Conteúdo. Inicialmente, os dados XML são analisados por um *parser* XML, formando um novo objeto em memória. Este objeto é então persistido em banco de dados Oracle com a ajuda das classes de persistência.

Seguindo o *hyperlink* para listagem de usuários, será gerada uma requisição que solicitará a recuperação de todos os dados dos usuários cadastrados na base de dados. Estes dados são enviados para uma página JSP para que sejam apresentados ao usuário. Os usuários listados neste relatório podem ter seus dados alterados caso se siga o *hyperlink* para o formulário de alteração de usuário. Neste caso, os dados do usuário específico são consultados para popular inicialmente um formulário de alteração, onde os dados podem ser alterados e submetidos ao sistema para que sejam novamente armazenados. Ao final do processo de alteração a listagem com os usuários cadastrados é apresentada ao usuário. Para sair do protótipo, o usuário deverá selecionar o botão *logout*.

5 Resultados e Conclusões

Os resultados obtidos após o desenho da arquitetura apresentada foram: a implementação parcial da API; a instalação e a configuração da ferramenta Struts; e a implementação do protótipo que exercitou todas as camadas acima da API.

Nesta implementação do protótipo, observou-se as vantagens de utilizar uma arquitetura modular e encapsulada:

- facilidade na incorporação de novas funcionalidades e na evolução das já existentes, com menor impacto no código já implementado;
- ganho de produtividade no desenvolvimento, pois dada que a API já estava implementada, o esforço de desenvolvimento da aplicação foi reduzido;
- aumento no número de classes que podiam ser desenvolvidas em paralelo pela equipe, também contribuindo para aumentar o ganho de produtividade;
- facilidade na distribuição de trabalho dentro da equipe de desenvolvimento, dado o baixo acoplamento entre as classes.

A utilização do padrão MVC garante a flexibilidade necessária ao gerenciador, dado que suas classes podem ser integradas a outros sistemas de informação da empresa, bem como sua apresentação pode ser facilmente expandida para outros formatos, tais como WAP (Wireless Application Protocol), e interface Palm. A arquitetura desenhada também atende à necessidade imediata da empresa de fornecer a Agência também em CD-ROM, além da plataforma WWW, o que implica em desenvolver uma nova camada de apresentação.

6 Referências Bibliográficas

ALLAMARAJU, S.; BEUST, C.; DAVIES, J.; JEWELL, T.; JOHNSON, R.; LONGSHAW, A.; NAGAPPAN, R.; O´CONNOR, D.; SARANG, P.G.; TOUSSAINT, A.; TYAGI, S.; WATSON. G.; WILCOX, M.; WILLIAMSON, A. **Professional Java Server Programming J2EE 1.3.** *Edition* Wrox Press Ltd, 2001.

BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. **Pattern-oriented software architecture: a system of pattterns.** New York: John Wiley, 1996. 467 p.

ALKACON, Software. **Welcome to the OpenCms Project**. URL:http://www.opencms.org Consultado em 12 jun. 2003

OCLC Research, **Dublin Core Metadata Initiative.** URL: http://dublincore.org Consultado em: 12 jun 2002.

JACARTA. **The Jakarta Project Velocity.** Disponível: site The Jakarta Project URL: http://jakarta.apache.org/velocity/index.html. Consultado em 7 abr. 2003.

JACARTA **The Jakarta Project Struts.** Disponível: site The Jakarta Project URL: http://jakarta.apache.org/struts/index.html. Consultado em 7 abr. 2003.

SOUZA, K. X. S. **Agência Embrapa de produtos e serviços de informação.** In: AGROSOFT: WORKSHOP O AGRONEGÓCIO NA SOCIEDADE DA INFORMAÇÃO, Brasília, 2002. Anais... Disponível em: http:www.agrosoft.com.br/ag2002/workshop/imprimir.php?page=135 7 p. Tipo: SP

SOUZA, M.I.F.; SANTOS, A.D.; HIGA, R.H.; VENDRÚSCULO, L.G. **Use of dublin core and XML for the organization of agricultural information in the web.** In: WORLD CONGRESS OF COMPUTER IN AGRICULTURE AND NATURAL RESOURCES, 2002, Iguaçu Falls. Proceedings... Iguaçu Falls: ASAE, 2001. p. 721-727.

SUN, MicroSystems. **The Source for Java Technology.** URL:http://java.sun.com Consultado em 7 de abr. 2003

SUN, MicroSystems. **Core J2EE(TM) Patterns: DataAccessObject pattern**. URL:http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html Consultado em 7 abr. 2003

ZOPE, Corporation. **Welcome to Zope.** URL:http://www.zope.org Consultado em 12 jun. 2003

W3C, Consortium. **Extensible Markup Language (XML)**. URL: http://www.w3c.org/XML/ Consultado em 7 abr. 2003