Um Framework de Componentes para o Desenvolvimento de Aplicações Web Robustas de Apoio à Educação

Elaine Quintino da Silva, Dilvan de Abreu Moreira
Instituto de Ciências Matemáticas e de Computação-ICMC, Universidade de São Paulo – USP
{elaine,dilvan}@icmc.usp.br

Resumo: *Apesar do grande número de ambientes de apoio à educação existentes atualmente, nota-se que, em grande parte desses ambientes, a extensão ou modificação de alguma funcionalidade, quando possível, é uma tarefa extremamente complexa principalmente em função da arquitetura de implementação utilizada. No intuito de flexibilizar a utilização desse tipo de aplicação, este trabalho apresenta um framework para o desenvolvimento de aplicações educacionais baseadas na Web. Por estar baseado nos componentes da arquitetura Java 2 Enterprise Edition, aplicações desenvolvidas sobre esse framework podem ser consideradas robustas em função da presença de características como confiabilidade, segurança e escalabilidade extraídas dos servidores de aplicação no qual os componentes são mantidos. Para garantir a flexibilidade de extensão e modificação de funcionalidades de uma forma simplificada, esse framework foi implementado com base no padrão de projeto conhecido como MVC (Model View Controller) e nas características de modularização dos sitemas de gerenciamento de conteúdos comerciais, conhecidos como CMSs (Content Management Systems).*

1. Introdução

Atualmente existe uma variedade de ambientes virtuais e ferramentas de software que apóiam o processo de ensino-aprendizagem. Como exemplos pode-se citar os ambientes comerciais WebCT [1] e Blackboard [2], e os acadêmicos AulaNet [3] e TelEduc [4], que oferecem suporte à educação a distância via Internet; o projeto eClass [5], que oferece apoio à sala de aula por meio da captura de experiências e posterior disponibilização na *web*, entre outros.

Problemas comuns na maioria dos ambientes educacionais atuais são a falta de interoperabilidade entre ambientes diferentes e a dificuldade de estender, modificar e reusar suas funcionalidades [6,7]. Isso ocorre, principalmente, porque a maioria destes ambientes é construída sobre uma arquitetura integrada de difícil modificação com interesses educacionais mais gerais ou interesses muito específicos.

Para minimizar os problemas de interoperabilidade, o reuso, aspecto importante no desenvolvimento de software [8], vem sendo explorado na área de Informática na Educação por meio da padronização dos recursos educacionais. Motivados pela necessidade de portar conteúdos educacionais entre ambientes e reutilizar conteúdos já desenvolvidos, órgãos como o IMS [9], AICC [10], ADL [11], IEEE-LTSC [12], entre outros, iniciaram o desenvolvimento de padrões (baseados na definição de metadados) para a criação de objetos de aprendizagem (*Learning Objects*) que podem ser interpretados em diferentes ambientes.

Atualmente, ambientes educacionais como WebCT [1] e Blackboard [2] já implementam o reuso de objetos de aprendizagem, porém os problemas relacionados com a extensão, modificação e reuso de funcionalidades (ferramentas) ainda continua sendo pouco explorado. O Aulanet é um exemplo raro de ambiente educacional que está sendo re-estruturado sobre uma arquitetura de componentes para permitir maior grau de customização [13].

Neste contexto, este artigo apresenta um *framework* - WebMoDE (*Framework for **Mo**dular **D**evelopment of **Web**-based **E**ducational Applications*) - para desenvolvimento de aplicações *web* customizáveis e robustas de apoio à educação. Aplicações educacionais desenvolvidas sobre o WebMoDE podem ser consideradas robustas porque seus componentes passam a fazer parte dos componentes de um servidor de aplicação J2EE (para isso tiveram que seguir regras especificadas no padrão J2EE) responsável por gerenciá-los e garantir-lhes escalabilidade, confiabilidade, tolerância a falhas e segurança [23]. O desenvolvimento modular, baseado nos sistemas de gerenciamento de conteúdos comerciais – CMS (*Content Management Systems*) - e

a separação entre lógica e apresentação de dados, baseada no modelo arquitetural MVC (*Model View Controller*), também são características importantes no WebMoDE (cujo uso em software educacional é uma das contribuições deste trabalho) que dão flexibilidade para a extensão e modificação das funcionalidades da aplicação.

Este artigo está organizado da seguinte forma: na seção 2 apresenta-se o tipo de modularização utilizado no WebMoDE; na seção 3 descreve-se o uso de *frameworks* no desenvolvimento *web* em geral e na área educacional; na seção 4 são apresentados aspectos relacionados a arquitetura e a implementação do WebMoDE; na seção 5 são citados seus módulos educacionais, e, por fim, na seção 6 são apresentadas as considerações finais.

2. Desenvolvimento Baseado em Módulos

A possibilidade de estender, modificar e reusar as funcionalidades de aplicações desenvolvidas sobre o WebMoDE se dá em função do desenvolvimento modular que o *framework* propõe. Por meio de uma arquitetura de componentes, são disponibilizados, além de um conjunto de especificações e classes abstratas (padrão de todo *framework*), vários módulos previamente implementados que podem ser diretamente utilizados em aplicações ou modificados. Com as especificações e classes abstratas, novos módulos também podem ser implementados. Os módulos são agrupados para formar uma aplicação de apoio à educação baseada na web.

A arquitetura modular do WebMoDE foi baseada nos princípios de desenvolvimento dos Sistemas de Gerenciamento de Conteúdos (CMS) utilizados na construção de *web sites* comerciais. Os CMSs (mais especificamente, os *open-source*) não são concebidos como pacotes de software, mas como *toolboxes* para a criação customizada de *web sites* [14]. Geralmente, esses sistemas oferecem um conjunto básico de módulos específicos para algumas tarefas relacionadas com o gerenciamento de conteúdos de *sites* comerciais, mas permitem, além de mudanças no *layout*, o desenvolvimento de outros módulos "plugáveis" que são construídos sobre uma arquitetura base para formar aplicações diferenciadas. Alguns CMSs incentivam a disponibilização de módulos publicamente para que possam ser reutilizados por outros interessados [15,16], enquanto outros incentivam o *feedback* de seus usuários para a agregação de novos módulos ao próprio *toolbox* [17]. Em ambos os casos, o interesse é o de se reaproveitar os esforços de desenvolvimento já realizados.

O WebMoDE segue essa mesma premissa de reaproveitamento de esforços (reuso), porém, com o objetivo de permitir o desenvolvimento modular de aplicações de apoio à educação baseada na web. Esse desenvolvimento modular, no qual cada módulo agrega um conjunto de funcionalidades específicas, enfatiza os seguintes aspectos:

- facilidade para instanciação de aplicações de gerenciamento de atividades didáticas: o WebMoDE oferece um conjunto básico de funcionalidades que pode ser diretamente utilizado na criação de uma aplicação educacional;
- facilidade para adição de módulos ao *framework*: módulos novos podem ser "plugados" sem prejuízo das aplicações já instanciadas;
- facilidade para agregar módulos às aplicações: aplicações são instanciadas a partir da seleção de módulos a serem utilizados.

Os módulos também podem ser alterados sendo que essa alteração pode ser feita em dois níveis: mudando-se sua configuração em termos de suas funcionalidades ou alterando-se a lógica (código) dos componentes utilizados pelo módulo.

3. *Frameworks* no Desenvolvimento Web

Do ponto de vista do propósito, um *framework* pode ser definido como o esqueleto de uma aplicação que pode ser customizado por um desenvolvedor de aplicações, ou seja, é uma

aplicação semi-completa e reutilizável que, quando especializada, produz aplicações personalizadas [18]. Nos últimos anos, muitas aplicações *web* vêm sendo desenvolvidas a partir de *frameworks* em virtude, principalmente, das facilidades que esse tipo de implementação oferece, tais como: aproveitamento de código implementado, redução de custos de implementação, estrutura bem definida e comum a várias aplicações, etc.

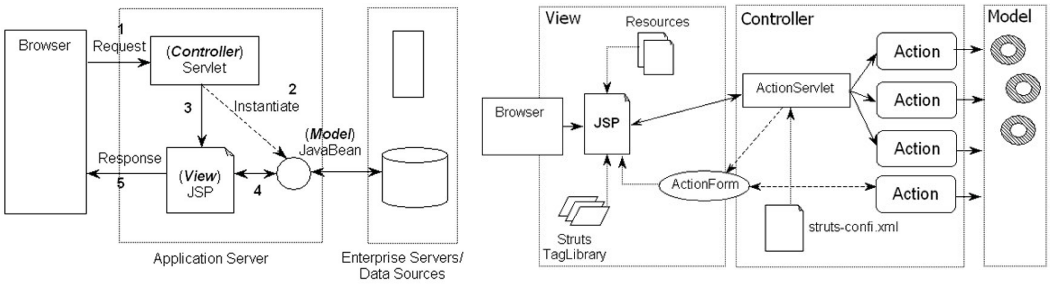
Um dos aspectos mais explorados na construção dos *frameworks* atuais tem sido a utilização do modelo arquitetural MVC [19] introduzido pelos desenvolvedores da linguagem Smalltalk na década de 80. O MVC divide a aplicação em três camadas:

- modelo (*model*): representa os dados e as regras específicas da aplicação (*business data e business logic*) que gerenciam o acesso e a atualização desses dados;
- visão (*view*): renderiza o conteúdo de um modelo acessando seus dados e especificando como eles são apresentados;
- controlador (*controller*): define o comportamento da aplicação enviando as requisições do usuário para o modelo e selecionando as visões para apresentar os resultados destas requisições.

Essa divisão em camadas do MVC permite organizar melhor a aplicação e facilita sua manutenção uma vez que cada camada pode ser modificada sem afetar diretamente as outras.

Na arquitetura J2EE, o MVC pode ser implementado de duas formas: *model* 1 e *model* 2. No MVC *model* 1 (MVC-1), uma JSP (*Java Server Page*) é responsável por receber a requisição, solicitar a execução da ação a um componente JavaBean e responder ao cliente [20]. No MVC *model* 2 (MVC-2) há uma divisão mais clara entre as camadas da aplicação pela utilização do Java Servlet como controlador, conforme se observa na **Figura 1a.**

Atualmente, o MVC-2, com algumas pequenas variações, constitui a base da maioria das aplicações e *frameworks* para desenvolvimento *web* utilizando tecnologia Java, como é o caso do Struts (descrito na próxima seção) e Turbine do projeto Apache Jakarta (http://jakarta.apache.org); do Webwork (http://www.opensymphony.com/webwork), dentre outros.



(a) **(b)**

Figura 1- (a) Arquitetura original do MVC-2 [20];
(b) Arquitetura do Struts

3.1. O *Struts framework*

De modo especial, o Struts [21] tem sido um dos *frameworks* mais utilizados para a construção de aplicações *web* no padrão MVC-2. Uma aplicação *web* desenvolvida com o Struts (cuja arquitetura pode ser observada na **Figura 1b**) tem os seguintes componentes:

- um Java Servlet implementando o controlador da aplicação (`ActionServlet`);
- páginas JSP implementando as visões (usam Struts Tag Library para apresentar dados);

- várias classes Java implementando as ações da aplicação (`Actions`);
- vários componentes JavaBeans implementando formulários de ação (`ActionForm`);
- um arquivo de configuração (`struts-config.xml`) definindo aspectos de relacionamento entre as ações, formulários de ações e visões da aplicação.

O fluxo de execução de uma aplicação baseada no Struts é o seguinte: o controlador recebe a requisição, armazena as informações do formulário de entrada no formulário de ação específico da ação requisitada e solicita a execução da ação à classe de ação específica; a classe de ação executa a ação, armazena os dados de resposta em um formulário de ação (pode ser o mesmo ou outro mais específico) e disponibiliza esse formulário para acesso das páginas JSP; o controlador faz um redirecionamento para a página JSP que irá apresentar os dados; então, essa página utiliza as bibliotecas de *tags* do Struts e arquivos de recursos da aplicação para apresentar os dados do formulário de ação.

O Struts *framework* não constitui uma aplicação que pode ser diretamente utilizada, mas ele oferece um conjunto de APIs (*Application Programmin Interfaces*) e especificações que permitem criar uma aplicação sobre sua arquitetura.

De modo geral, pode-se dizer que o Struts segue adequadamente o modelo MVC e permite o desenvolvimento de aplicações *web* bem estruturadas. Entretanto, a complexidade para implementar uma aplicação, mesmo que simples, é uma de suas maiores desvantagens [22]. Parte dessa complexidade vem do próprio modelo MVC que divide a aplicação em camadas, mas características como a definição de relacionamentos de ações, formulários e páginas de visualização em um único arquivo de configuração (`struts-config.xml`), além da complexidade no desenvolvimento das classes de ação, tornam o processo de desenvolvimento distante do modelo usual (baseado no uso Java Server Pages e Servlets) e mais difícil a medida em que a aplicação fica mais completa.

Originalmente, houve o interesse em se reusar as vantagens oferecidas por *frameworks* bem estabelecidos como o Struts para implementar as funcionalidades de um *framework* educacional. Entretanto, por causa da complexidade de desenvolvimento e da necessidade de se incorporar características dos CMSs comerciais, foi implementada uma nova arquitetura, inspirada tanto nos CMSs quanto nos *frameworks* para desenvolvimento *web* que usam MVC, para suportar o *framework* proposto neste artigo. Nesta nova arquitetura (apresentada em detalhes na Seção 4), a aplicação desenvolvida sobre o WebMoDE é formada por vários módulos que são implementados como aplicações MVC-2.

3.2. *Frameworks* em Aplicações Educacionais

Embora o uso de *frameworks* tenha importância potencial para o desenvolvimento de aplicações *web* comerciais, essa forma de desenvolvimento de aplicações vem ganhando atenção também na área educacional.

Lindquist et al. [6] apresentam um *framework* de componentes que pode ser usado para estender uma aplicação *web* existente. Seus componentes são implementados na forma de Java Servlets específicos para cada tipo de funcionalidade. Esses componentes são invocados via Java Applets inseridos em documentos HTML de aplicações *web* já existentes. As funcionalidades oferecidas pelos componentes de Lindquist et al. incluem serviços para gerenciamento de estudantes, notas, calendários de projetos, entre outros. Aspectos sobre a criação de uma aplicação mais completa não são contemplados no *framework* de Lindquist et al., sendo que seus componentes oferecem apenas serviços adicionais que podem ser acoplados a uma aplicação existente.

Anido et al. [7] apresentam um *framework* distribuído em camadas para suportar o desenvolvimento de aplicações *web* interativas e colaborativas. Os componentes são

implementados por meio de Java Applets agrupados para criar a aplicação. Gerenciamento de usuários, ferramenta de auditoria, quadro de notas, *White-board*, agenda e gerenciamento de projetos são exemplos de componentes oferecidos no *framework* de Anido et al.

Há várias características que diferem o WebMoDE dos demais *frameworks* com mesmo propósito - apoio à educação baseada na *web* - encontrados na literatura, tais como:

- modularidade inspirada nas características dos CMSs comerciais: permite a construção de uma aplicação *web* modular a partir de uma aplicação básica que já implementa recursos comuns (por exemplo, *login*, *logout*, alteração de dados cadastrais, etc);
- arquitetura inspirada nos *frameworks* mais comuns para desenvolvimento *web* (como o Struts): permite maior facilidade de desenvolvimento e manutenção da aplicação, uma vez que cada módulo é desenvolvido como sendo uma aplicação MVC-2;
- robustez da aplicação adquirida pelo uso de componentes da arquitetura J2EE como o EJB (*Enterprise JavaBeans*): os componentes J2EE rodam sobre servidores de aplicação que possuem serviços, tais como controle de transação e acesso a bases de dados, replicação para alta disponibilidade com o uso de *clusters*, *caching*, entre outros, para gerenciar os seus componentes e garantir-lhes confiabilidade, tolerância a falhas, segurança e escalabilidade.
- reusabilidade proporcionada por meio do uso de componentes e módulos.

Conforme se observa, esses aspectos são essenciais para uma aplicação de apoio à educação que seja robusta e de fácil customização.

4. Arquitetura e Implementação do WebMoDE

Mais do que um conjunto de APIs e especificações, o WebMoDE é uma aplicação *web* enterprise (básica) que pode ser disponibilizada em um servidor de aplicações J2EE. Essa aplicação é formada por um conjunto de módulos (e componentes) específicos para o gerenciamento de atividades educacionais.

Para obter as características citadas na seção anterior, o WebMoDE foi desenvolvido utilizando-se tecnologias da arquitetura de componentes J2EE, tais como [23]:

- **Java Servlets:** componentes *web* que rodam no servidor para estender suas funcionalidades.
- **Java Server Pages:** componentes *web* com função similar a dos servlets, porém mais centrados na parte de apresentação; uma JSP é uma página HTML que inclui código para efetuar processamento no servidor.
- **Enterprise JavaBeans (EJB):** componentes de negócio que executam de forma distribuída sobre um servidor de aplicação e são gerenciados por um *container* EJB.

Os componentes JavaBeans, componentes mais simples com métodos *get* e *set* para suas propriedades, também foram utilizados para a troca de informação entre os componentes da arquitetura J2EE [23]. Além disso, o Hibernate, um mecanismo que oferece persistência a objetos Java de maneira transparente (por meio do mapeamento desses objetos para base de dados relacionais) [24] foi utilizado para o armazenamento persistente de dados no *framework*.

Cada módulo do WebMoDE é desenvolvido como uma aplicação baseada no MVC-2. Os componentes EJB, que representam o **modelo**, implementam a lógica específica da aplicação e utilizam objetos persistentes (Hibernate) para manipular os dados. Um servlet, que é uma subclasse de um servlet controlador genérico (`ModuleSupport`) oferecido pelo *framework*, faz o papel de **controlador** do módulo. Ele recebe requisições de um controlador central (do núcleo do *framework*), valida as entradas de dados (método `validade`), identifica as ações (método `getAction`) e faz chamadas aos EJBs para executá-las (método `runAction`). As

visões são implementadas por páginas JSP que utilizam a biblioteca padrão de *tags* JSTL (*JSP Standard Tag Library)* [25] para recuperar dados de um objeto JavaBean padrão. Esse objeto, que contém os dados a serem apresentados ao usuário, é fornecido pelo *framework* e pode armazenar qualquer tipo de objeto serializável.

A opção pelo uso de EJBs como representantes do modelo se deve às suas três principais características:

- é um padrão bem aceito pela indústria e baseado no conceito "*train once, code anywhere*", pois uma vez que se conhece a especificação fica fácil entender outros componentes;
- é portável, ou seja, um EJB pode executar em qualquer servidor de aplicações J2EE;
- o desenvolvimento torna-se rápido já que pode-se aproveitar os serviços oferecidos pelos servidores de aplicação (transação, *caching*, *clustering*, etc), ou seja, não é necessário implementar certos serviços nos próprios componentes.

O WebMoDE foi desenvolvido sobre um núcleo básico, capaz de unir e gerenciar os módulos da aplicação. Esse núcleo pode ser reutilizado para a implementação de vários outros tipos de aplicações baseadas na web com as mesmas características de modularidade. Ele é composto por (a representação dos componentes pode ser vista na Figura 2):

- Um controlador principal (`Main Controller`) que recebe cada requisição e a redireciona para o módulo referenciado na própria requisição. O módulo executa seu processamento e devolve ao controlador um objeto JavaBean (`FormBean`) com os dados a serem apresentados para o cliente. O controlador aciona um componente interno que adiciona informações necessárias para a construção da interface da aplicação ao `FormBean` e o redireciona para o JSP *template*.
- Um JSP *template* (`Main Page`) formado por arquivos JSP que representam diferentes porções da apresentação de um documento *web* (esses arquivos são descritos a seguir).
- Componentes EJB e módulos básicos que implementam a lógica mínima para o funcionamento do *framework*.

Desse núcleo básico, também faz parte o mecanismo de autenticação de usuários. As opções de *login* e *logout* são implementadas com base no serviço JAAS (*Java Authentication and Authorization Service*) que permite o uso de diferentes fontes de autenticação, tais como: autenticação em sistema operacional, em servidores LDAP (*Lightweight Directory Access Protocol*), em bases de dados específicas, entre outros. Esse mecanismo de autenticação, dá à aplicação desenvolvida sobre o WebMoDE a capacidade de reusar uma base de dados de usuários já existente. O acesso às opções dos módulos é controlado em função dos papéis que um usuário possui na aplicação. Tais papéis são cadastrados para cada aplicação.

A arquitetura do WebMoDE é apresentada na **Figura 2**. Os números na figura indicam a ordem em que a comunicação acontece entre os componentes para que uma requisição seja atendida. De acordo com a figura, o *template* JSP (`Main Page`) é formado pelos seguintes arquivos: `Header Page`, `SideBar Page`, `Content Page` e `Footer Page`. Essa construção está baseada no padrão de projeto da arquitetura J2EE conhecido como *Composite Views*. Esse padrão sugere a utilização de páginas compostas (ou *templates*) para facilitar a manutenção da aparência da aplicação, mantendo-a consistente. Assim, para alterar a aparência de uma das áreas de apresentação do *template*, basta alterar um dos arquivos que o compõe, e para mudar o *layout* que organiza as partes da apresentação, basta alterar o próprio *template*.

No WebMoDE, os arquivos JSP que definem o cabeçalho e rodapé (`Header Page` e `Footer Page`) são responsáveis por incluir os arquivos de apresentação dessas áreas que são específicos para cada nova aplicação. O arquivo que define a lateral da página (`SideBar Page`) faz parte do núcleo do *framework* e renderiza os itens de navegação dinamicamente a

partir dos módulos disponíveis. O arquivo JSP que representa o conteúdo central da página (`Content Page`) é responsável por incluir o arquivo (página) resultante de uma resposta a uma requisição processada. O *template*, bem como os arquivos que o compõe, está descrito em um arquivo XML (*Extensible Markup Language*) e é processado por um documento XSL (*Extensible Stylesheet Language*) [26] que define seus aspectos de apresentação.

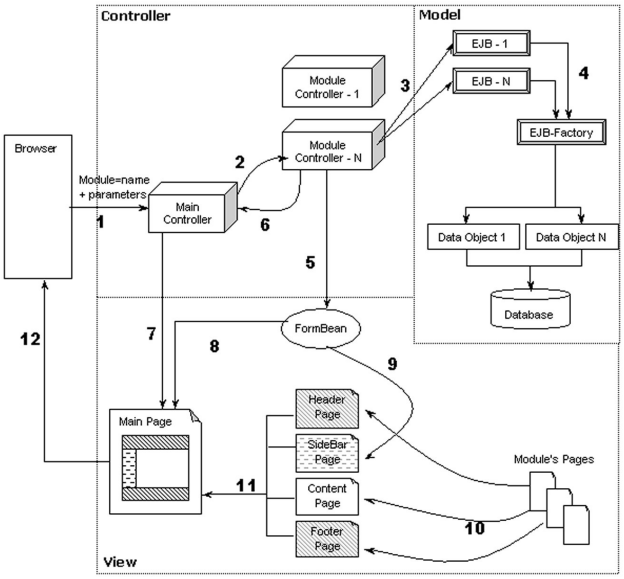


Figura 2 - Arquitetura geral do WebMoDE

Um módulo do *framework* pode ser disponibilizado como uma aplicação *web* (`.war`), quando não requer a inclusão de novos componentes EJBs, ou como uma aplicação enterprise (`.ear`), quando novos componentes são necessários. O novo módulo pode ser agregado ao pacote principal do *framework* ou incluído separadamente no servidor de aplicações (mais indicado). Um arquivo XML que descreve as ações do módulo é processado quando o mesmo é disponibilizado no servidor (dentro ou fora do pacote principal). Isso permite que o núcleo do *framework* tenha conhecimento do módulo assim que ele é disponibilizado, não sendo necessário parar a execução das aplicações para adicionar novos módulos. Depois de disponibilizado, o módulo pode ser adicionado a uma aplicação e customizado. Essa característica é obtida por meio do serviço de *hot-deploy* presente nos servidores de aplicação J2EE.

O WebMoDE está sendo desenvolvido com base no uso de software livre para que possa ser facilmente instanciado e modificado sem dependência de plataforma proprietária. O servidor de aplicações JBoss [27], o único servidor de aplicações disponível gratuitamente (software livre), está sendo utilizado no seu desenvolvimento como ambiente de teste.

5. Módulos para Gerenciamento de Atividades Educacionais

Conforme citado, o WebMoDE é disponibilizado como uma aplicação composta por um conjunto de módulos voltados para o gerenciamento de atividades educacionais. Esses módulos podem ser alterados ou substituídos de acordo com interesses individuais. Embora ainda não estejam totalmente desenvolvidos, alguns módulos essenciais para esse tipo de aplicação são:

a) *Administration*: (parte do núcleo) oferece recursos para a administração das aplicações. Uma aplicação é instanciada por meio da escolha de módulos, a partir dos disponíveis, e da configuração de aspectos como páginas de cabeçalho, rodapé, base de dados, etc.

b) User: oferece recursos para o gerenciamento de usuários. Entre as funcionalidades desse módulo, há a possibilidade de se associar papéis aos usuários, além de incluir ou excluir usuários dependendo do tipo de autenticação utilizado. Os papéis são importantes no WebMoDE porque o acesso às funcionalidades das aplicações é controlado por meio deles. Por *default,* o WebMoDE define os papéis *nobody* e *user* que representam respectivamente, os usuários visitantes da aplicação *web* e os usuários que efetuaram *login*. A interface da aplicação é customizada para o tipo de papel que o usuário possui.

c) Course: permite gerenciar os cursos da aplicação, por exemplo: cadastrar, apagar ou editar informações de cursos.

d) Team, Teacher, Monitor, Candidate e *Student*: oferecem recursos para o gerenciamento de turmas, professores, monitores, candidatos a cursos e estudantes.

e) Grade: oferece recursos para o gerenciamento e visualização de notas dos estudantes.

f) Activity: oferece opções para gerenciamento das atividades dos cursos.

g) Calendar: permite gerenciar o calendário dos cursos disponíveis.

Para oferecer suporte ao método educacional baseado no uso de Peer Review para avaliação de estudantes, apresentado por Moreira e Silva [28], o módulo *Activity* incorpora recursos para o gerenciamento de atividades que usam esse método, mas outros tipos de atividades também podem ser suportados.

Embora os módulos básicos estejam centrados no gerenciamento das atividades didáticas, módulos mais específicos, por exemplo para colaboração e cooperação, como os chats, foruns, entre outros, podem ser desenvolvidos de acordo com as necessidades de cada aplicação.

Um módulo pode ou não ter componentes EJB agregados. O conjunto de EJBs disponibilizado com o WebMoDE pode ser utilizado por vários módulos e deve ser suficiente para criar uma boa aplicação de gerenciamento de atividades, mas a inclusão de novos EJBs também pode ser feita.

Esse *framework* ainda está em fase de implementação e, portanto, nem todos os módulos estão completamente desenvolvidos.

6. Considerações Finais

O desenvolvimento desse trabalho teve como agente motivador a necessidade de se criar um software de apoio à educação que pudesse ser modificado para atender interesses específicos de conjuntos de usuários. Embora os ambientes educacionais existentes já ofereçam um grande conjunto de funcionalidades, ainda há muitas entidades que têm dificuldades em adaptar esses ambientes às suas necessidades. Nesses ambientes, mudanças simples, quando possíveis, são difíceis de realizar.

No intuito de dar flexibilidade para a construção desse tipo de ambiente e permitir extender, modificar e reusar funcionalidades de aplicações educacionais, foi proposto um *framework* baseado nas premissas de desenvolvimento dos CMSs comerciais (que pregam a construção da aplicação de forma modular) e nas características dos *frameworks* para desenvolvimento *web*, como o Struts.

A arquitetura apresentada na **Figura 2** já foi totalmente implementada e um protótipo de aplicação já foi desenvolvido e testado com sucesso. Esse protótipo (**Figura 3**) permitiu validar a capacidade de integração entre as várias tecnologias utilizadas e as premissas de desenvolvimento do *framework*. Tal protótipo foi criado a partir do WebMoDE e roda em um *cluster* de computadores (6 nós - 12 CPUs) sobre um servidor de aplicações J2EE (JBoss Application Server). Esse servidor gerencia todos os processos de transação, segurança,

balanceamento de carga, entre outros, que garantem robustez e eficiência, sem a necessidade de agregar código ao *framework* ou ao protótipo.

Pelo fato de nem todos os módulos educacionais estarem totalmente implementados, ainda não é possível apresentar resultados numéricos sobre reusabilidade e desempenho. Porém, o bom funcionamento do protótipo, mesmo rodando num *cluster,* demonstra que não será uma tarefa complexa alcançar um bom nível de reusabilidade ou desempenho do *framework*.

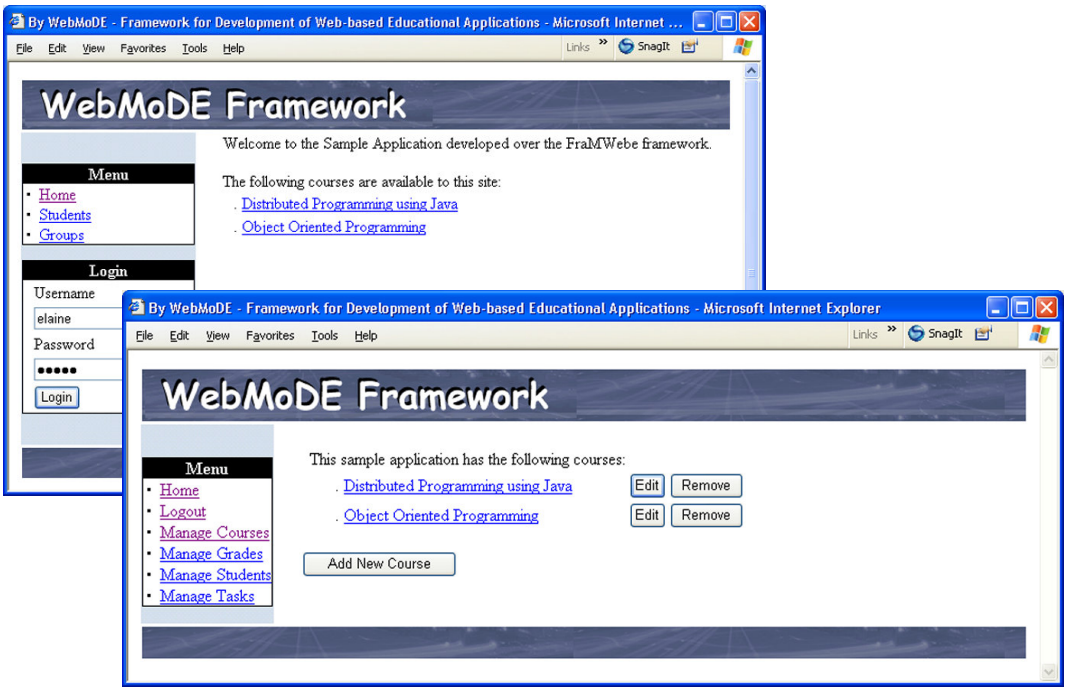


Figura 3 – Interfaces do protótipo de aplicação desenvolvido sobre o WebMoDE

Como continuação deste trabalho, após a implementação dos módulos citados na Seção 5, será feita uma avaliação, com a implementação e teste de uma aplicação educacional, para verificar aspectos de reusabilidade e performance. Resultados numéricos desta avaliação devem ser reportados em artigos posteriores.

7. Agradecimentos

Agradecimentos à CAPES, ao ICMC-USP e à FINEP (no contexto do projeto SAFE – Software Engineering Available For Everyone, MCT/FINEP/CT-INFO - 01/2003, convênio 0210/04) pelo apoio financeiro a este trabalho.

8. Referências Bibliográficas

[1] WebCT homepage. http://www.webct.com.

[2] Blackboard Learning System. http://www.blackboard.com/.

[3] H. Fuks, "Aprendizagem e Trabalho Cooperativo no Ambiente AulaNet", *Revista Brasileira de Informática na Educação*, SBC, n.6, Abril 2000, pp 53-73.

[4] H. V. Rocha, "O ambiente TelEduc para educação a distância baseada na web: Princípios, funcionalidades e perspectivas de desenvolvimento", em M. C. Moraes, *Educação a distância: Fundamentos e práticas,* UNICAMP/NIED, Campinas-SP, 2002, cap. 11, p. 197-212.

[5] G. D. Abowd, "Classroom 2000: an experiment with the instrumentation of a living educational environment", *IBM Systems Journal*, 1999, pp. 508-530.

[6] T. E. Lindquist, K. A. Gary, H. E. Koehnemann and H. Naccache, "Component Framework for Web-Based Learning", *Proceedings of the Frontiers in Education Conference FIE'1999 29th Annual*, San Juan, Puerto Rico, November 1999. IEEE Education/Computer Society, S12C3, pp. 23-28.

[7] L. Anido-Rifón, M. Llamas-Nistal, M. J. Fernández-Iglesias, "A Component Model for Stardardized Web-based Education", *Proc.Tenth international conference on World Wide Web*, Hong Kong, Hong Kong, 2001, pp. 86 – 95.

[8] V. Basili, L. Briand and W. Melo, "How reuse influences productivity in object-oriented systems", *Communications of the ACM*, n.39, v.10, 1996, 104-116.

[9] IMS Global Learning Consortium. http://www.imsproject.org/

[10] AICC – Aviation Industry CBT Committee. http://www.aicc.org/

[11] ADL – Advanced Distributed Learning. http://www.adlnet.org/

[12] IEEE-LTSC – Learning Technology Standards Committee. http://ltsc.ieee.org/.

[13] H. Fuks, M. A. Gerosa, M.G. Pimentel, A. B. Raposo, L. H. R. G. Mitchell and C.J.P Lucena, "Evoluindo para uma Arquitetura de Groupware Baseada em Componentes: o Estudo de Caso do Learningware AulaNet". *Anais Eletrônicos do III Workshop de Desenvolvimento Baseado em Componentes*, São Carlos-SP, Setembro 2003.

[14] J. Robertson, "A content management project presents unique challenges", *Step Two Designs*. http://www.steptwo.com.au/papers/cmb_unique/index.html.

[15] PHPNukes CMS. http://www.warez.fr/nuked.html.

[16] PostNuke CMS. http://www.postnuke.com.

[17] Nukes on JBoss. http://www.jboss.org.

[18] R. E. Johnson, "Frameworks, Components, Patterns", *Communications of the ACM*, v.40, n.10, pp. 39-42, 1997.

[19] G. Krasner, S. T. Pope, "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80", *Journal of Object-Oriented Programming*, v.1 n.3, Aug./Sept. 1988, pp.26-49.

[20] G. Seshadri, "Understanding JavaServer Pages Model 2 architecture - Exploring the MVC design pattern", *Java World*, December 1999. http://www.javaworld.com.

[21] S. Spielman, *The Struts Framework*", Elsevier Science, USA, 2003.

[22] Core Servlets homepage, "Demystifying Jakarta Struts-An Introductory Apache Struts Tutorial". http://www.coreservlets.com.

[23] E. Roman, S. W. Ambler, T. Jewell, *Mastering Enterprise JavaBeans*, 2º ed., John Wiley & Sons, Canadá, 2002.

[24] Hibernate Project, JBoss Group, http://www.hibernate.org.

[25] D. M. Geary, *Core JSTL-Mastering the JSP Standard Tag Library*", Sun Microsystems, Prentice-Hall, USA, 2003.

[26] W3C Specifications. http://www.w3.org/.

[27] JBoss Application Server. http://www.jboss.org.

[28] D.A. Moreira, E.Q. Silva, "Increasing Student Interaction using Student Groups and Peer Review over the Internet", *IFIP Journal of Education and Information Technologies,* v.8, n.1, March 2003, Kluwer Academic Publishers, pp. 47-54.