Arquitetura JavaEE do Sistema de Informações para o Ensino (SIE)

Marcos Vinícius B. De Souza[1], Giuliano Lopes Ferreira[1], Marcius da Silva Fonseca[1], Márcio Frick[1]
{marcos, giuliano, sf.marcius, mfrick}@cpd.ufsm.br
Centro de Processamento de Dados da Universidade Federal de Santa Maria
Av. Roraima, 1000. Prédio 43.
CEP 97105-900
Santa Maria, RS

1. Introdução

Devido a necessidade de disponibilização das funcionalidades do SIE para a internet, tornou-se necessário um estudo para a formação de um arquitetura. Durante tais estudos, levou-se em consideração as tecnologias mais usadas e adequadas para as demandas das aplicações desenvolvidas. Muitas das propriedades presentes no SIE foram estudadas a fundo para que fosse possível encontrar tecnologias correspondentes na plataforma JavaEE (*Java Enterprise Edition*), como, por exemplo, controle de transações (JTA), envio assíncrono de mensagens (JMS)[1], etc.

2. Componentes da arquitetura

Como primeira premissa, definiu-se construir a arquitetura em quatro camadas: Visão, Controle, Negócio e Modelo. A utilização dessas camadas é uma adaptação do padrão de projeto MVC (*Model-View-Controller*) que permite o isolamento de implementação e reduz o impacto de possíveis mudanças em cada uma [2].

A camada Visão é responsável por construir a interface e interagir com o usuário. A camada Controle agrupa a lógica necessária para receber a interação do usuário e interagir com a camada Negócio, que possui os métodos implementados para a realização da lógica de negócios em si. A camada Modelo fornece os métodos para a realização das operações CRUD (*Create, Retrieve, Update, Delete* – Criar, Buscar, Atualizar e Remover) nas entidades referenciadas no sistema em questão. A separação das camadas visa isolar possíveis mudanças na arquitetura através de interfaces bem definidas entre elas. Dessa forma, a correção de uma busca de dados na camada Modelo, por exemplo, não afeta as demais, aumentando a estabilidade do sistema como um todo.

3. Tecnologias

Como interface com o BD (Banco de Dados), optou-se por utilizar o *framework* Hibernate [3] como ferramenta de mapeamento objeto-relacional (ORM). Dessa forma, obtém-se uma independência da implementação específica do Sistema Gerenciador de Banco de Dados (SGBD). Além disso, torna-se possível a mudança do SGBD sem que as funcionalidades do sistema sejam afetadas, bastando a modificação em um arquivo de configuração do sistema.

A camada Negócio das aplicações é implementada com o uso da tecnologia *Enterprise Java Beans* (EJB) [4]. Através dessa tecnologia, é possível distribuir os objetos responsáveis pela lógica de negócio em diferentes servidores, permitindo uma solução escalável e distribuída. Além desses benefícios, conforme a quantidade de aplicações se expande, é possível formar uma Arquitetura Orientada a Serviços (SOA), facilitando a interação com ferramentas atuais construídas com esse foco.

A camada da interface gráfica, em contato direto com o usuário final, é desenvolvida através da tecnologia JSP (*Java Server Pages*), em conjunto com tecnologias como: HTML (*HyperText Markup Language*), Javascript, CSS (*Cascading StyleSheet*), Ajax, etc. De forma a não violar a separação entre as camadas do sistema, a interface gráfica não possui nenhuma lógica de negócio ou acesso direto ao BD, mas sim, deve possuir componentes e regras para a renderização correta dos dados.

A interface gráfica dos sistemas *web* integrantes do SIE, levam em conta, ainda, as premissas de usabilidade, facilitando o seu uso e tornando-a o mais natural possível. Para a realização desses objetivos, leva-se em consideração o ambiente no qual o usuário está imerso, para que não hajam quebras de comunicabilidade no sistema.

O *framework* Spring[5] é utilizado para fazer o elo entre a interface gráfica do usuário e a lógica de negócios, presente nos EJBs. São desenvolvidos controladores (*controllers*) que permitem a invocação dos métodos de negócio para a apresentação de dados na interface, bem como, permite o uso de formulários para cadastros/alterações necessários na maioria das aplicações[5]. Graças a infra-estrutura presente no Spring, a implementação dessas funcionalidades é facilitada e aumenta a produtividade da equipe de desenvolvedores.

Para suportar a constante mudança da lógica de negócios de certas áreas do SIE, foi criado o mecanismo de planilhas. Esse mecanismo permite que sejam escritos códigos-fonte na linguagem LUA que são armazenados em BD e carregados em tempo de execução. De acordo com as novas necessidades, esse código dinâmico pode ser re-escrito sem que o código-fonte das aplicações seja recompilado, facilitando a manutenção do do sistema [6].

Para arquitetura *web*, criou-se um mecanismo semelhante, porém com a linguagem de programação Groovy, por ser mais próxima da linguagem de programação Java [7]. Foram criados componentes que permitem ao código estático fazer o carregamento de classes do banco sempre que necessário, construindo, ainda uma cache dos objetos mais recentemente utilizados no sistema. Dentre os sistemas *web* presentes, o processo de geração do número de matrícula é realizado através de código dinâmico construído em Groovy, pois seguidamente requer mudanças na maneira da sua construção.

4. Conclusões

A construção da arquitetura para aplicações *web* do SIE foi realizada a partir de vários estudos, visando a utilização de padrões da industria e padrões de fato, usados em projetos mundialmente conhecidos. Conforme a demanda por aplicações *web* expandiu-se, foram pesquisadas tecnologias que atendessem aos requisitos e que tivessem funcionalidades equivalentes às constantes na arquitetura SIE.

Como forma de validação e utilização da arquitetura construída, forma desenvolvidos os sistemas: Portal do Aluno, Portal do Professor, Portal do DERCA, Sistema de Controle de Restaurantes Universitários, Sistema de Solicitações de Bolsas, etc. Tais sistemas são utilizados em ambiente de produção na UFSM atualmente e fornecem meios para a melhoria constante da própria arquitetura.

Referências

[1] Sun Developer Network. “*JavaEE 6 Technologies*”. Disponível em: <http://java.sun.com/javaee/technologies/>. Acesso em 14 de abril de 2010.

[2] Gamma E.; et al. (1995). “*Design Patterns: Elements of Reusable Object-Oriented Software*”. Addison Wesley, USA.

[3] JBoss Community. “*Hibernate. Relational Persistence for Java & .Net*”. Disponível em: <http://hibernate.org/>. Acesso em 14 de abril de 2010.

[4] Sun Developer Network. “Enterprise JavaBeans Techonology”. Disponível em: <http://java.sun.com/products/ejb/>. Acesso em 14 de abril de 2010.

[5] Johnson, R.; et al . “Spring Documentarion”. Disponível em: <http://www.springsource.org/documentation>. Acesso em 14 de abril de 2010.

[6] Lerusalimschy, R.; Figueiredo, L. H.; Celes, W. “*Lua 5.1 Reference Manual*”. Disponível em: <http://www.lua.org/>. Acesso em 14 de abril de 2010.

[7] CodeHaus Foundation. “*Groovy – Tutotial 1 - Getting Started*”. Disponível em: <http://groovy.codehaus.org/Tutorial+1+-+Getting+started>. Acesso em 14 de abril de 2010.