

# Uso de Design Patterns e J2EE: um estudo de caso

Rogério Sorroche (FURB)

rs@furb.br

Maurício Capobianco Lopes (FURB)

mclopes@furb.br

**Resumo.** Este trabalho apresenta um estudo de caso sobre o desenvolvimento e a implementação de um software utilizando *design patterns* e a tecnologia *Java 2 Enterprise Edition*, destacando a modelagem de um sistema que agrega estas duas tecnologias.

**Palavras-chave:** Design Patterns, J2EE.

## 1 Introdução

Segundo Sun (2002) a plataforma *Java 2 Enterprise Edition* (J2EE) define um padrão para o desenvolvimento de aplicações multicamadas. Nesta arquitetura, a camada que contém as regras de negócio, normalmente implementada utilizando *Enterprise JavaBeans*, pode ficar concentrada no servidor de aplicações, sendo compartilhada com diversas aplicações clientes. As aplicações clientes não contêm a lógica do negócio, atendo-se somente à camada de apresentação. Na camada de apresentação normalmente são utilizadas as tecnologias de *Servlets* e *JavaServer Pages*.

Segundo Marinescu (2002), sem um conjunto de boas práticas de modelagem, o desenvolvimento utilizando a arquitetura multicamadas J2EE, pode se tornar muito difícil. Segundo Alur (2002), uma boa maneira de adquirir experiência em projeto é pela utilização de padrões de projeto (*design patterns*) que se constituem em um moderno e importante mecanismo para a elaboração de projetos orientados a objetos.

Sendo assim, neste trabalho é apresentada a arquitetura de um sistema de auxílio à matrícula de alunos, utilizando-se uma arquitetura multicamadas baseado no J2EE e seguindo alguns padrões de projeto específicos para esta arquitetura.

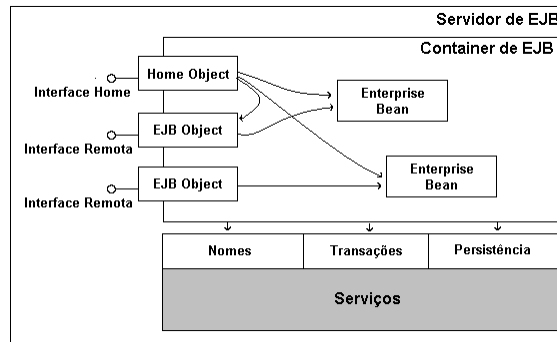
## 2 Java 2 Enterprise Edition (J2EE)

Segundo Sun (2002) *Java 2 Enterprise Edition* (J2EE) usa um modelo de aplicações multicamadas distribuído. As tecnologias que fazem parte desta plataforma de desenvolvimento são os *Enterprise JavaBeans*, que são componentes que implementam regras de negócio e os *Servlets* e os *JavaServer Pages*, que são classificados como componentes *Web*.

### 2.1 Enterprise Javabeans

De acordo com Roman (2002), *Enterprise JavaBeans* (EJB) são especificamente usados para resolver problemas de negócio. O padrão *EJB* é uma arquitetura de componentes que podem ser implantados em um ambiente distribuído. Ele especifica um contrato entre os componentes e os servidores de aplicação, de modo que um componente *EJB* pode ser implantado em qualquer servidor de aplicações que suporte *EJB*. Os *EJB* também referidos como *enterprise beans*, segundo Sun (2002), são escaláveis, transacionais e multi-usuários.

Segundo Seshadri (1999), os elementos definidos pela arquitetura de *EJB* são ilustrados na Figura 1.



Fonte: Baseado em Jubin (1999) e Roman (2002).

**Figura 1** – Principais elementos da arquitetura de EJB

Os elementos desta arquitetura são:

- servidor de *EJB*: gerencia um ou mais *containers* de *EJB* e provê os serviços de suporte, como gerenciamento de transações, persistência e acesso aos clientes. Também disponibiliza um serviço de nomes para os clientes localizarem os *enterprise beans* distribuídos através da rede (Jubin, 1999).
- container de *EJB*: provê um ambiente em que o *EJB* possa ser executado e chamado remotamente pelos clientes.
- home object* e *remote home interface*: para adquirir uma referência ao objeto o cliente deve chamar uma *fábrica* de objetos. A especificação chama esta fábrica de *home object* que é uma implementação da *remote home interface* que expõe para os clientes os métodos disponíveis para criar, remover e localizar os *EJB* (Roman, 2002).
- EJB object* e *remote interface*: descreve os métodos de negócio, que podem ser chamados remotamente e que estão disponíveis no *bean*. Esta interface deve ser disponibilizada pelo desenvolvedor do *bean*.

## 2.2 Servlets

Segundo Hunter (2001) um *servlet* é uma extensão genérica de um servidor, isto é, uma classe Java que pode ser carregada dinamicamente para expandir a funcionalidade de um servidor. Geralmente os *servlets* são usados em servidores *web*, onde eles funcionam como aplicações CGI (*Common Gateway Interface*), recebendo requisições e gerando respostas.

A classe do *servlet* executa no servidor *web* por meio de um *container* de *servlets*. O servidor *web* mapeia um conjunto de URL (*Universal Resource Locator*) para um *servlet*. Quando o servidor recebe uma requisição para uma destas URL, ele repassa a requisição para o *servlet*, que gera uma resposta para o cliente. Geralmente a resposta é um documento HTML ou XML. A grande vantagem do *servlet* em relação ao *applet* é que o *servlet* executa no servidor, por isso, ele não depende da compatibilidade entre os vários navegadores de Internet (Goodwill, 2001).

## 2.3 Javaserwer Pages

Muitas aplicações *Web* produzem primariamente páginas HTML dinâmicas que, quando acessadas, mudam somente os dados, e não a sua estrutura básica. Os *servlets* não fazem distinção entre o que é código de formatação HTML e o que são dados. A tecnologia de *JavaServer Pages* (JSP), entretanto, difere deste modelo de programação. Uma página JSP é primariamente um documento que especifica conteúdo dinâmico, ao invés de um programa que produz conteúdo. As

páginas JSP fornecem uma alternativa “centrada em documentos”, ao contrário dos *servlets* que são classes para criar conteúdo dinâmico.

Sun (2002) define uma página JSP como um documento contendo HTML estático, com algumas marcações para incluir dados ou para executar alguma lógica embutida na própria página JSP. Goodwill (2001) explica que as páginas JSP são uma extensão de *servlets*. De fato, o servidor transforma a página JSP em um *servlet* que irá gerar o conteúdo da página (Hunter, 2001).

### 3 Padrões de Projeto

Os padrões de projeto, também conhecidos como *Design Patterns*, visam uma melhor reutilização de software. Estes padrões tornam mais fácil reutilizar projetos e arquiteturas bem sucedidas. Eles ajudam a escolher alternativas de projeto que tornam um sistema reutilizável e a evitar alternativas que comprometam a reutilização. Segundo Marinescu (2002), um padrão de projeto é definido como uma solução desenvolvida utilizando boas práticas para um problema comum que ocorre várias vezes. Um padrão de projeto documenta e explana um problema importante que pode ocorrer no projeto ou implementação de uma aplicação e então discute a melhor solução prática para o problema.

Em termos de orientação a objetos, padrões de projeto identificam classes, instâncias, seus papéis, colaborações e a distribuição de responsabilidades. São, portanto, descrições de classes e objetos que se comunicam, implementados a fim de solucionar um problema comum em um contexto específico (Schneide, 1999). As vantagens de se utilizar padrões em um projeto são listadas por Alur (2002):

- a) foram testados: refletem a experiência e conhecimento dos desenvolvedores que utilizaram estes padrões com sucesso em seu trabalho;
- b) são reutilizáveis: fornecem uma solução pronta que pode ser adaptada para diferentes problemas quando necessário;
- c) são expressivos: formam um vocabulário comum para expressar grandes soluções sucintamente;
- d) facilitam o aprendizado: reduzem o tempo de aprendizado de uma determinada biblioteca de classes. Isto é fundamental para o aprendizado dos desenvolvedores novatos;
- e) diminuem retrabalho: quanto mais cedo são usados, menor será o retrabalho em etapas mais avançadas do projeto.

Neste trabalho são apresentados padrões de projeto que descrevem a estrutura de uma aplicação e outros que descrevem os elementos do projeto. O ponto em comum entre eles é que foram desenvolvidos e aplicados especificamente à plataforma de desenvolvimento *J2EE*.

O catálogo de padrões de projeto aqui apresentados foi dividido de acordo com as camadas a que eles pertencem. A camada de apresentação contém os padrões relacionados aos *Servlets* e páginas *JSP*. Os padrões da camada de negócios são relacionados à tecnologia de *EJB* e os padrões da camada de integração estão relacionados à comunicação com recursos e sistemas externos.

Os padrões da camada de apresentação, apresentados por Alur (2002) e utilizados neste trabalho são:

- a) *intercepting filter*: intercepta solicitações e respostas, e aplica algum filtro;
- b) *front controller*: fornece um controlador centralizado para manter a lógica de processamento que ocorre na camada de apresentação e que, erroneamente é colocado nas visões (*view*);

- c) *view helper*: encapsula a lógica que não esteja relacionada à formatação da apresentação em componentes auxiliares;
- d) *composite view*: cria uma visão através da composição de outras visões;
- e) *service to worker*: combina um componente distribuidor (*dispatcher*) com os padrões *Front Controller* e *View Helper*. Neste padrão a responsabilidade de recuperar o conteúdo para apresentar é do controlador;
- f) *dispatcher view*: semelhante ao padrão *Service to Worker*, mas, neste padrão a recuperação do conteúdo é responsabilidade das visões.

Os padrões da camada de negócios utilizados foram (Alur, 2002).

- a) *business delegate*: reduz o acoplamento entre o cliente e a camada de negócios;
- b) *data transfer object*: facilita o intercâmbio de dados entre as camadas;
- c) *data transfer object factory*: facilita e centraliza a criação de *data transfer objects*;
- d) *session facade*: fornece uma interface unificada para um sistema distribuído;
- e) *composite entity*: representa uma prática mais recomendada para criar *entity beans* de granulação grossa, agrupando os objetos dependentes em um único *entity bean*;
- f) *value list handler*: gerencia o processamento e armazenamento em *cache* de consultas que retornam uma grande quantidade de informações;
- g) *service locator*: encapsula a complexidade de localização e criação de serviços de negócio e localiza os objetos *Home* dos *EJB*.

Os padrões da camada de integração descritos por Alur (2002) são:

- a) *data access object*: abstrai e encapsula o acesso aos dados;
- b) *service activator*: facilita o processamento assíncrono para *Message-driven beans*.

As descrições detalhadas destes padrões podem ser encontradas em Alur (2002), (Marinescu, 2002) e Sorroche (2002).

## 4 Arquitetura do Sistema

Para exemplificar a arquitetura de um caso real de um sistema desenvolvido utilizando padrões de projeto e J2EE é apresentado um sistema de auxílio à matrícula desenvolvido por Sorroche (2002). O sistema em questão tem por finalidade apoiar o acadêmico em seu processo de matrícula, gerando sugestões de horários.

Para a visualização da arquitetura do trabalho desenvolvido são apresentados neste artigo apenas os diagramas de classe e o diagrama de estados. Os diagramas de sequência podem ser visualizados em Sorroche (2002).

Para o desenvolvimento deste trabalho foram utilizadas as seguintes ferramentas:

- a) *Together*: ferramenta case *UML*, que dá suporte a *EJB* e padrões de projeto;
- b) servidor de aplicações *J2EE JBoss*: suporta a execução de *EJB* e *servlets*;
- c) *JDeveloper*: ferramenta de desenvolvimento para Java;
- d) *Oracle*: é o banco onde estão todas as tabelas do sistema acadêmico de graduação.

### 4.1 Diagrama de Classes

As classes definidas para este sistema estão separadas em pacotes, sendo eles: *ejb*, *cliente*, *dao*, *modelo*, *busca* e *web* (Figura 2).

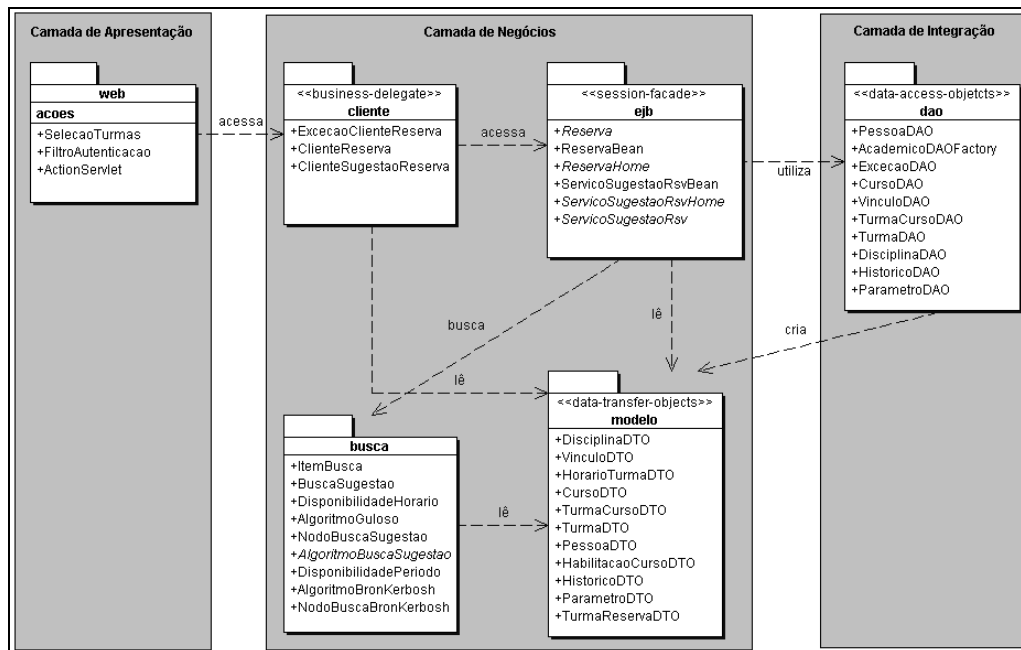


Figura 2 – Estrutura de pacotes

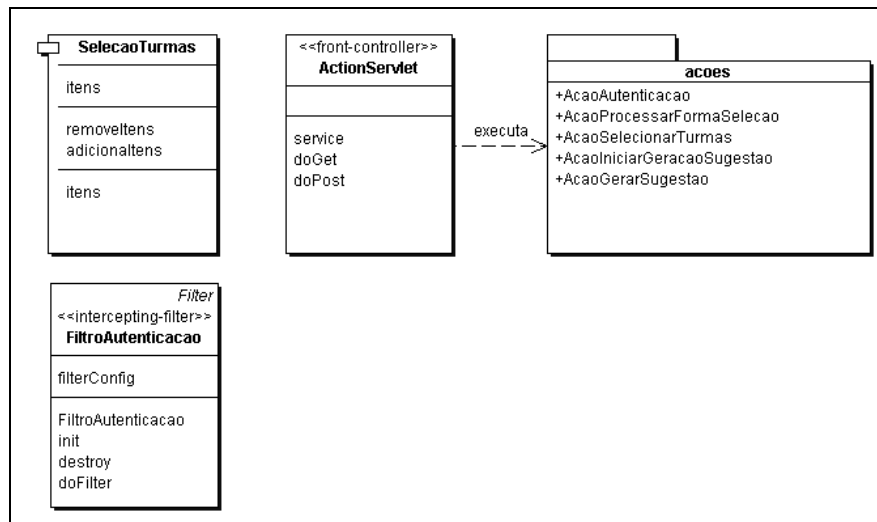
Cada um destes pacotes pertence a uma camada do sistema e contém os padrões de projeto para estas camadas:

- pacote *web*: contém a implementação de um *Intercepting Filter*, que verifica a cada solicitação se o usuário efetuou a autenticação e, um *Front Controller* que age como ponto inicial de contato para efetuar o processamento da lógica na camada *web*;
- pacote *cliente*: neste pacote estão definidos os *Business Delegate*, que são classes que representam os objetos de negócio distribuídos. Estas classes ficam do lado do cliente e simplificam o acesso aos *EJB*;
- pacote *ejb*: contém a definição dos *EJB* que formam a camada de *Session Facade* do sistema. Os *EJB* definidos neste pacote são uma fachada para a reserva de vaga e para o serviço de elaboração de sugestões de matrícula;
- pacote *busca*: contém a implementação dos algoritmos de busca e elaboração de sugestões;
- pacote *dao*: contém os objetos de acesso a dados (padrão *Data Access Objects*), que extraem e encapsulam o acesso para as tabelas do sistema acadêmico;
- pacote *modelo*: os objetos definidos neste pacote encapsulam os dados das tabelas do sistema acadêmico (padrão *Data Transfer Object*), que são passados da camada de negócios para a camada de apresentação.

A seguir serão descritos os pacotes mais detalhadamente e serão apresentados os diagramas de classes para cada pacote.

#### 4.1.1 Pacote WEB

Neste pacote estão as classes utilizadas na camada web juntamente com as páginas *JSP* e os *servlets*. A Figura 3 apresenta o diagrama de classes deste pacote.

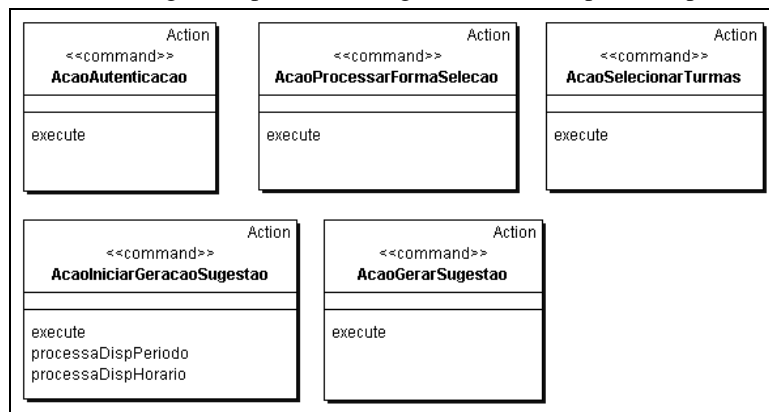


**Figura 3** – Diagrama de classes do pacote *web*

A função de cada classe é descrita a seguir:

- SeleccionTurmas*: contém as disciplinas selecionadas para gerar sugestões;
- FiltroAutenticacao*: é o *Intercepting Filter* para a aplicação que verifica, em todos os acessos, se o usuário efetuou a autenticação;
- ActionServlet*: é o *servlet* que processa a lógica de controle da camada web. Este *servlet* se adequa ao padrão *Front Controller* e processa a sua lógica através das ações (padrão *Command*). Este *servlet* é definido na API *Struts* (Burns, 2002).

As ações ou comandos executados pelo *ActionServlet*, estão definidos no pacote *acoes*, que é um sub-pacote de *web*. A Figura 4 apresenta o diagrama de classes para este pacote.



**Figura 4** – Diagrama de classes para o pacote *acoes*

As classes deste pacote são descritas a seguir:

- AcaoAutenticacao*: é executada para autenticar o aluno no sistema validando o seu código e senha, informados na tela de autenticação;
- AcaoProcessarFormaSelecao*: seleciona as disciplinas que o aluno ainda não obteve aprovação e para as quais já possui os pré-requisitos;

- c) *AcaoSelecionarTurmas*: seleciona as disciplinas que o aluno escolheu e armazena na sessão para uso posterior na geração das sugestões;
- d) *AcaoIniciarGeracaoSugestoes*: inicia uma nova busca de sugestões com as disciplinas selecionadas;
- e) *AcaoGerarSugestoes*: após iniciada uma busca, esta ação comanda a busca de sugestões para gerar as próximas sugestões.

#### 4.1.2 Pacote Cliente

Este pacote contém os *business delegate* que abstraem o acesso aos dois *EJB* definidos anteriormente (*ReservaBean* e *ServicoSugestaoRsvBean*), para os clientes (Figura 5).

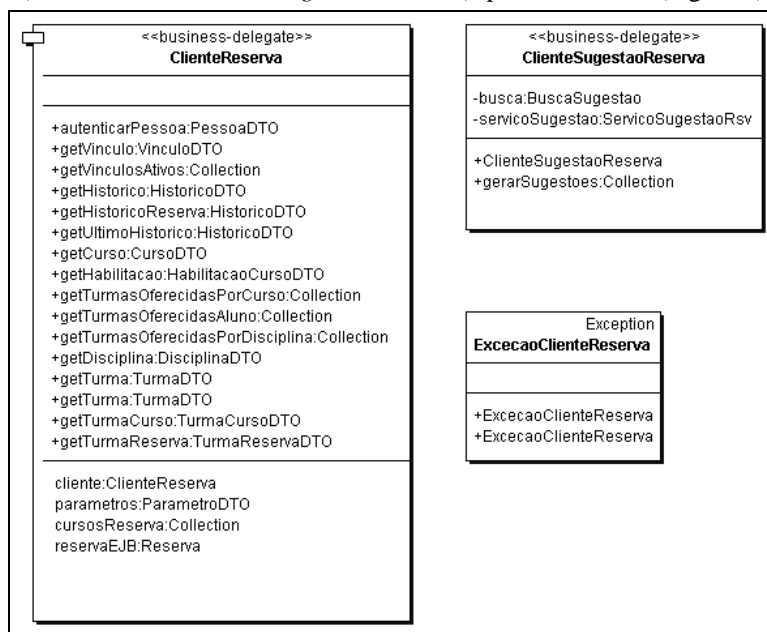


Figura 5 – Diagrama de classes do pacote *cliente*

Estas classes têm a seguinte função:

- a) *ClienteReserva*: é o *business delegate* do *EJB ReservaBean*. Os clientes acessam esta classe que por sua vez delega todas as chamadas de método para o *EJB*;
- b) *ClienteSugestaoReserva*: fornece acesso ao *EJB ServicoSugestaoRsvBean*;
- c) *ExcecaoClienteReserva*: exceção que pode ser gerada pelos métodos dos *business delegate*.

#### 4.1.3 Pacote EJB

Este pacote contém os *session beans* que formam a camada de *Session Facade* do sistema. Seu diagrama de classes é apresentado na Figura 6.

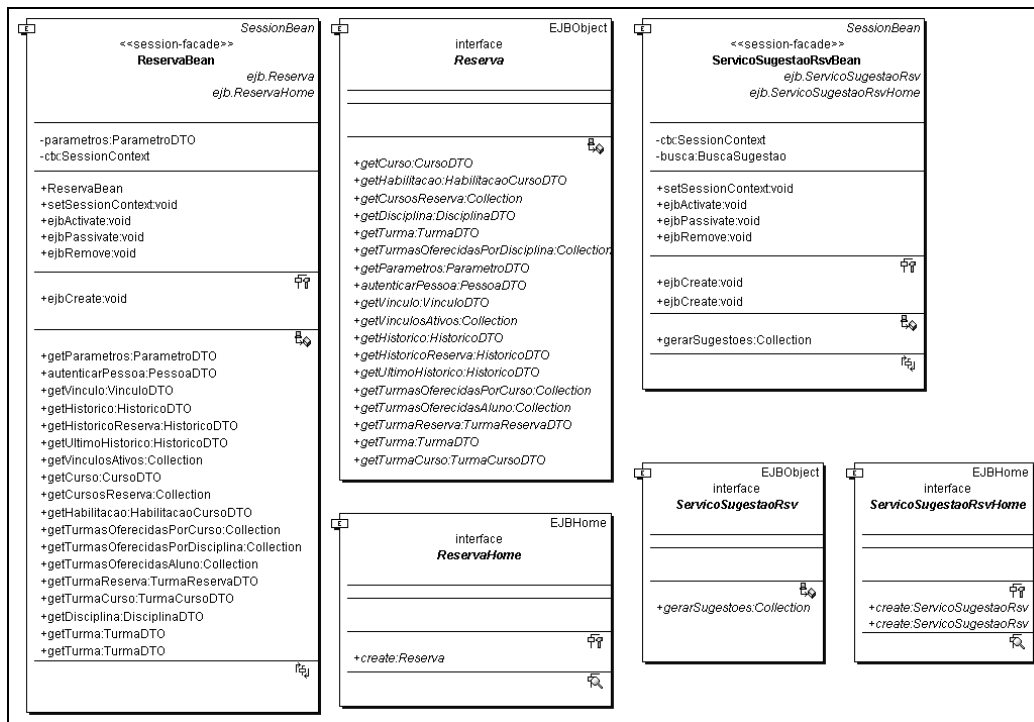


Figura 6 – Diagrama de classes do pacote *ejb*

As classes possuem as seguintes finalidades:

- ReservaBean*: é a classe de implementação do *EJB* que provê os serviços para a reserva de vaga. Este *EJB* contém os métodos para obter a lista de disciplinas oferecidas, autenticar um aluno e outras funções utilitárias;
- Reserva*: é a *remote interface* do *ReservaBean*;
- ReservaHome*: é a *home interface* do *ReservaBean*;
- ServicoSugestaoRsvBean*: é a classe de implementação do *EJB* que provê o serviço de geração das sugestões de reserva de vaga;
- ServicoSugestaoRsv*: é a *remote interface* do *ServicoSugestaoRsvBean*;
- ServicoSugestaoRsvHome*: é a *home interface* do *ServicoSugestaoRsvBean*.

#### 4.1.4 Pacote Busca

Neste pacote está a especificação do algoritmo de elaboração das sugestões. A Figura 7 apresenta o diagrama de classes deste pacote.



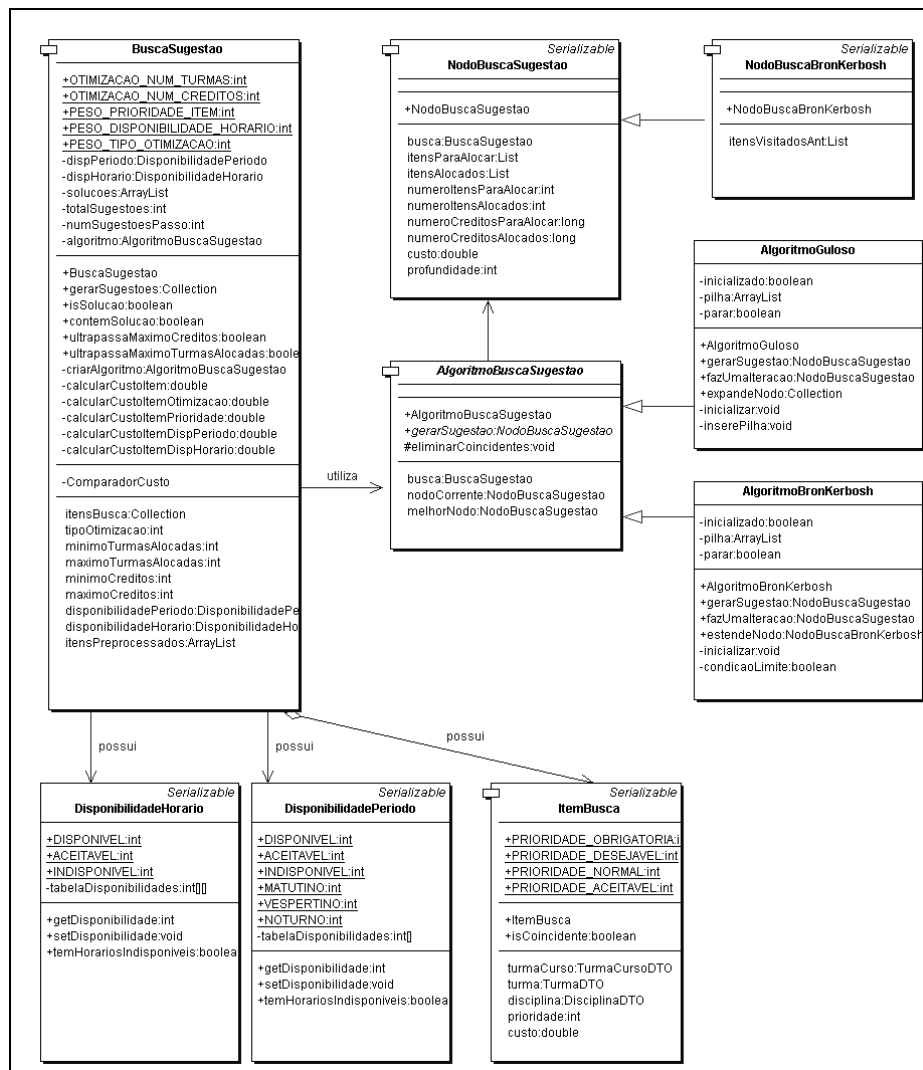


Figura 7 – Diagrama de classes do pacote *busca*

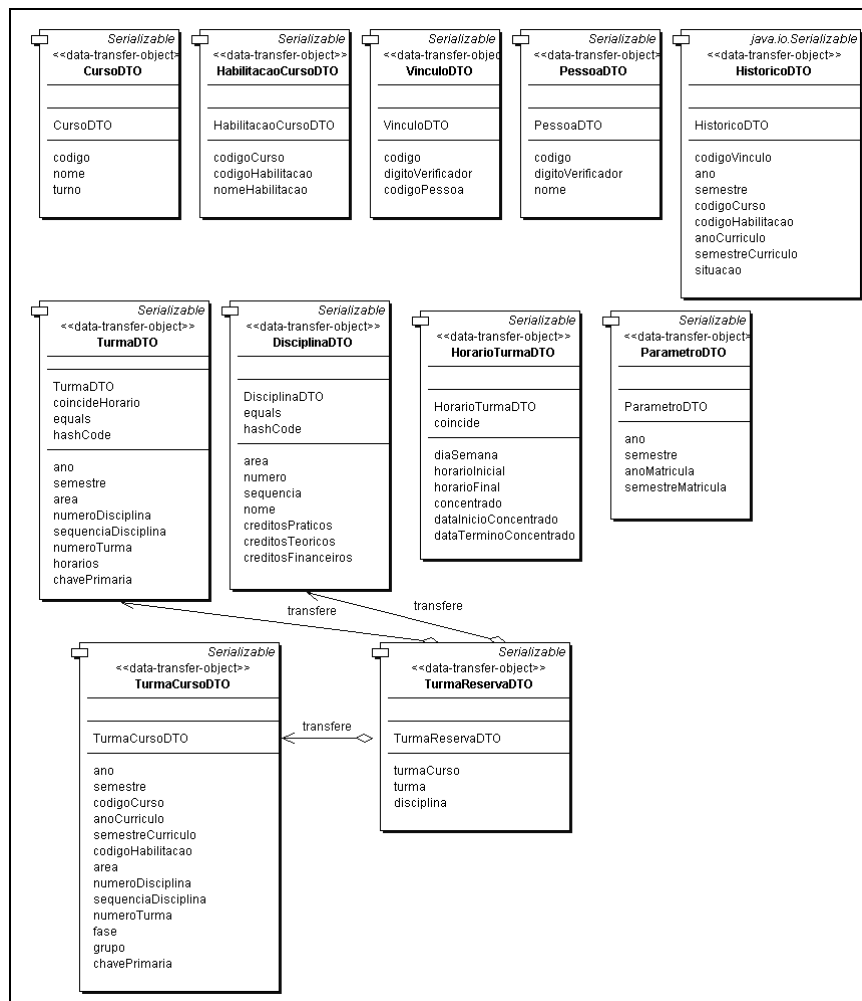
Estas classes têm as seguintes finalidades:

- BuscaSugestao*: representa uma busca por sugestões. Ela recebe um conjunto de objetos *ItemBusca*, as restrições, e a disponibilidade de horários do aluno e devolve um conjunto de sugestões;
- DisponibilidadeHorario*: representa a disponibilidade do aluno por horário;
- DisponibilidadePeriodo*: representa a disponibilidade do aluno por período (matutino, vespertino, noturno);
- ItemBusca*: representa uma disciplina ou turma com a prioridade atribuída pelo aluno e que será pesquisada na busca. Cada *ItemBusca* pode ser considerado como um vértice em um grafo que é pesquisado;
- AlgoritmoBuscaSugestao*: esta é uma superclasse abstrata de um algoritmo de busca de sugestões;

- f) *AlgoritmoGuloso*: é a implementação do algoritmo guloso para a geração de sugestões;
- g) *AlgoritmoBronKerbosh*: é a implementação do algoritmo de Bron e Kerbosh para a geração de sugestões;
- h) *NodoBuscaSugestao*: representa o estado da busca a cada iteração, o algoritmo guloso utiliza esta implementação;
- i) *NodoBuscaBronKerbosh*: é uma especialização de *NodoBuscaSugestao* para o algoritmo de Bron e Kerbosh, que contém um conjunto para armazenar os itens que já foram visitados.

#### 4.1.5 Pacote Modelo

Neste pacote estão os *Data Transfer Objects*. Estes objetos servem para trocar informações entre a camada de negócios e os clientes. A Figura 8 apresenta o diagrama de classes deste pacote.



**Figura 8** – Diagrama de classes do pacote *modelo*

O detalhamento da descrição destas classes pode ser obtido em Sorroche (2002).

#### 4.1.6 Pacote DAO

Neste pacote estão definidos todos os *Data Access Objects*. Estes objetos abstraem o acesso às tabelas do banco de dados acadêmico. O diagrama de classes deste pacote é apresentado na Figura 9.

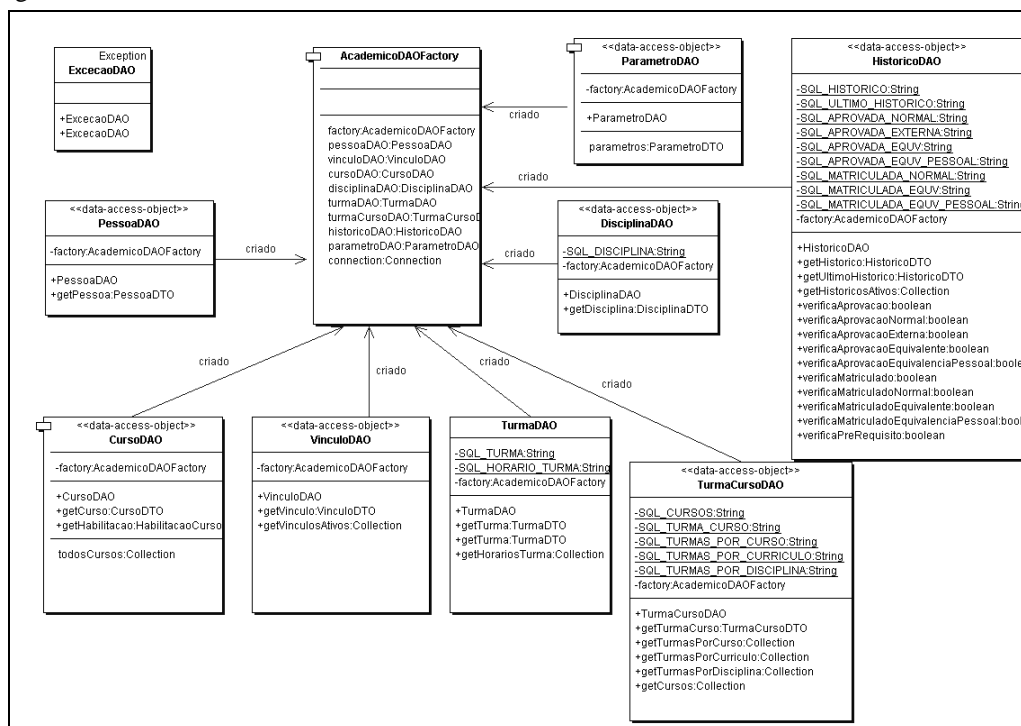


Figura 9 – Diagrama de classes do pacote *dao*

O detalhamento da descrição destas classes pode ser obtido em Sorroche (2002).

#### 4.2 Diagrama de Estados

O diagrama de estados apresentado na Figura 10 mostra a sequência de passos que o aluno irá seguir para gerar as sugestões. Neste diagrama as telas ou páginas JSP do sistema foram representadas como estados. Há também o estado “Gerando Sugestões”, que é o estado em que o sistema encontra-se elaborando as sugestões de reserva de vaga através do algoritmo de busca.

Neste fluxo, o aluno inicialmente se autentica no sistema e vai para uma tela onde informa as opções de seleção das disciplinas que deseja incluir na geração das sugestões. Estas opções visam incluir somente disciplinas da grade curricular do curso do aluno que ele ainda não obteve aprovação e para as quais já possua os pré-requisitos. Feita a seleção das disciplinas é apresentado um resumo para que o aluno possa definir prioridades para estas disciplinas e informar as restrições. Neste momento o aluno pode iniciar a geração das sugestões ou adicionar mais disciplinas do seu curso ou de outros cursos. O aluno pode, ainda, incluir disciplinas que façam parte da grade curricular do seu curso, mas, que estão sendo oferecidas em horários diferentes. Depois de geradas as sugestões, o aluno escolhe uma delas e efetua a sua reserva de vaga, terminando assim o processo.

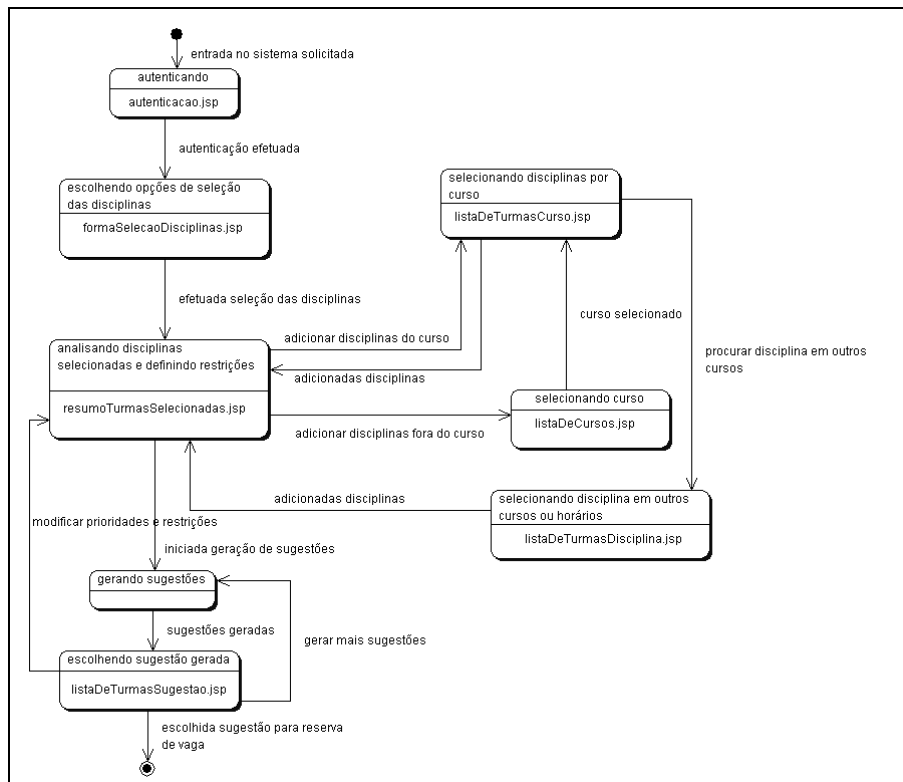


Figura 10 – Diagrama de estados

### 4.3 Definição da Interface do Usuário

As telas do sistema foram definidas de acordo com o padrão de projeto *Composite View*, uma *template* para as páginas *JSP*. Esta *template* é uma página *JSP* que especifica o layout de todas as páginas do sistema. A *template* define cinco seções em uma página e utiliza *custom tags* para incluir outras páginas *JSP* nestas seções. A Figura 11 apresenta o layout produzido pela *template*. A interface completa do sistema pode ser visualizada em Sorroche (2002).



 <b>FURB - Universidade Regional de Blumenau</b> Reserva de Vaga Seção de Desenvolvimento de Sistemas																																																																														
<b>Rogério Sorroche</b> Pessoa/Vínculo: 14482 / 9820204 Curso: 20 - Ciências da Computação																																																																														
<div> Selecção Automática Adicionar Disciplinas do Curso Adicionar Disciplinas Fora do Curso Geração de Sugestão Sair </div>																																																																														
<table border="1"> <thead> <tr> <th colspan="12">Disciplinas Selecionadas para Gerar Sugestão</th> </tr> <tr> <th>Curso</th> <th>Fase</th> <th>Disciplina</th> <th>HA</th> <th>Seg</th> <th>Ter</th> <th>Qua</th> <th>Qui</th> <th>Sex</th> <th>Sáb</th> <th>Prioridade</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>20</td> <td>8</td> <td>SIS.0016.02.001</td> <td>Análise e Projeto de Sistemas II</td> <td>4</td> <td></td> <td></td> <td></td> <td>1/4</td> <td>Normal</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>20</td> <td>8</td> <td>CMP.0039.00.002</td> <td>Projetos de Pesquisa em Ciências da Computação</td> <td>2</td> <td></td> <td>12/13</td> <td></td> <td></td> <td>Normal</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>20</td> <td>9</td> <td>CMP.0036.00.002</td> <td>Trabalho de Conclusão de Curso - TCC</td> <td>20</td> <td></td> <td>12/15</td> <td></td> <td></td> <td>Normal</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>20</td> <td>9</td> <td>CMP.0036.00.001</td> <td>Trabalho de Conclusão de Curso - TCC</td> <td>20</td> <td></td> <td>1/4</td> <td></td> <td></td> <td>Normal</td> </tr> </tbody> </table>												Disciplinas Selecionadas para Gerar Sugestão												Curso	Fase	Disciplina	HA	Seg	Ter	Qua	Qui	Sex	Sáb	Prioridade	<input checked="" type="checkbox"/>	20	8	SIS.0016.02.001	Análise e Projeto de Sistemas II	4				1/4	Normal	<input checked="" type="checkbox"/>	20	8	CMP.0039.00.002	Projetos de Pesquisa em Ciências da Computação	2		12/13			Normal	<input checked="" type="checkbox"/>	20	9	CMP.0036.00.002	Trabalho de Conclusão de Curso - TCC	20		12/15			Normal	<input checked="" type="checkbox"/>	20	9	CMP.0036.00.001	Trabalho de Conclusão de Curso - TCC	20		1/4			Normal
Disciplinas Selecionadas para Gerar Sugestão																																																																														
Curso	Fase	Disciplina	HA	Seg	Ter	Qua	Qui	Sex	Sáb	Prioridade																																																																				
<input checked="" type="checkbox"/>	20	8	SIS.0016.02.001	Análise e Projeto de Sistemas II	4				1/4	Normal																																																																				
<input checked="" type="checkbox"/>	20	8	CMP.0039.00.002	Projetos de Pesquisa em Ciências da Computação	2		12/13			Normal																																																																				
<input checked="" type="checkbox"/>	20	9	CMP.0036.00.002	Trabalho de Conclusão de Curso - TCC	20		12/15			Normal																																																																				
<input checked="" type="checkbox"/>	20	9	CMP.0036.00.001	Trabalho de Conclusão de Curso - TCC	20		1/4			Normal																																																																				
<b>Legenda:</b> C: horário em regime concentrado.																																																																														
 FURB Blumenau, SC - Seção de Desenvolvimento de Sistemas																																																																														

Figura 11 – Definição de uma *template* para páginas do sistema

Utilizando esta *template*, o conteúdo das seções cabeçalho, identificação, menu-superior e rodapé, foram definidos somente uma vez para todo o sistema, como páginas *JSP*, sendo reutilizadas várias vezes.

## 5 Conclusões

A junção das tecnologias de padrões de projeto e J2EE em um mesmo sistema apresentou-se totalmente apropriada. Mais especificamente no sistema em questão, o uso de padrões de projeto ajudou a reduzir a complexidade e promoveu uma grande reutilização no desenvolvimento. No caso de um sistema de porte médio, conforme verificado na arquitetura apresentada neste trabalho, estas características ficam ainda mais realçadas, uma vez que na própria especificação das classes é realizada a sua relação com os padrões de projeto.

A plataforma de desenvolvimento J2EE apresentou uma série de vantagens para o desenvolvimento de aplicações que necessitam de escalabilidade, disponibilidade e performance. Também destaca-se a portabilidade, uma vez que o protótipo foi desenvolvido numa plataforma *Windows NT* e posto em produção numa plataforma *Linux* sem que qualquer alteração fosse feita.

Enfim, uma metodologia que agrupe padrões de projeto e J2EE pode ser um diferencial de produtividade para o desenvolvimento de soluções empresariais.

## 6 Referências Bibliográficas

ALUR, Deepak; CRUPI, John; MALKS, Dan. **Core j2ee patterns**: as melhores práticas e estratégias de design. Tradução Altair Dias Caldas de Moraes, Cláudio Belleza Dias, Guilherme Dias Caldas Moraes. Rio de Janeiro: Campus, 2002.

BURNS, Ed; HUSTED, Ted; MCCLANAHAN, Craig R. **Struts user guide**, [2002?]. Disponível em: < <http://jakarta.apache.org/struts/userGuide/index.html>>. Acesso em: 15 nov. 2002.

GOODWILL, James. **Developing java servlets**. Indianápolis: Sams Publishing, 2001.

HUNTER, Jason; CRAWFORD, William. **Java servlet programming**. Cambridge: O'Reilly, 2001.

JUBIN, Henri; FRIEDRICHS, Jürgen; TEAM, Jalapeño. **Enterprise javabeans by example**. New Jersey: Prentice Hall PTR, 1999.

MARINESCU, Floyd. **EJB design patterns**: advanced patterns, processes, and idioms. New York: John Wiley & Sons, 2002.

ROMAN, Ed; AMBLER, Scott; JEWELL, Tyler. **Mastering enterprise javabeans**. New York: John Wiley & Sons, 2002.

SCHNEIDE, Ricardo Luiz. **Design patterns**. Rio de Janeiro, maio 1999. Disponível em: < [http://www.dcc.ufrj.br/~schneide/PSI\\_981/gp\\_6/design\\_patterns.html](http://www.dcc.ufrj.br/~schneide/PSI_981/gp_6/design_patterns.html)>. Rio de Janeiro, nov. 1999. Acesso em: 03 out 2002.

SESHADRI, Govind. **Enterprise java computing**: applications and architecture. Cambridge: Sign Books, 1999.

SORROCHE, Rogério. **Sistema de auxílio à matrícula de alunos utilizando java 2 enterprise edition**. 2002. 108 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SUN, Sun Microsystems. **Designing enterprise applications with the j2ee platform enterprise edition**. [S.l.], 2002b. Disponível em: <[http://java.sun.com/blueprints/guidelines/designing\\_enterprise\\_applications\\_2e/index.html](http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/index.html)>. Acesso em: 26 ago. 2002.