Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Graduação em Ciências da Computação

Uma Aplicação Web para Avaliação do Corpo Docente Universitário

Ricardo Kendy Horigome
Thiago Veiga Leffa Behenck

Florianópolis
2007

RICARDO KENDY HORIGOME
THIAGO VEIGA LEFFA BEHENCK

Uma Aplicação Web para Avaliação do Corpo Docente Universitário

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina, como requisito parcial para a obtenção do grau de Bacharel em Ciências da Computação.

Prof. José Mazzucco Junior - Orientador

Florianópolis
2007

Uma Aplicação Web para Avaliação do Corpo Docente Universitário

Por

RICARDO KENDY HORIGOME
THIAGO VEIGA LEFFA BEHENCK

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina, como requisito parcial para a obtenção do grau de Bacharel em Ciências da Computação.

Prof. José Mazzucco Junior - Orientador

_____

Orientador: Prof. José Mazzucco Junior, UFSC.

_____

Membro: Prof. Leandro J. Komosinski, UFSC.

_____

Membro: Prof. José Francisco Fletes, UFSC.

Dedicatória

Dedicamos às nossas famílias, entes queridos e amigos, pela compreensão dos momentos ausentes e dedicação no que puderam ajudar para que esse trabalho pudesse ser desenvolvido.

Agradecimentos

Agradecemos aos mestres que colaboraram com este trabalho, proporcionando uma excelente base teórica e agregando maior valor ao nosso trabalho.

RESUMO

Este trabalho tem como finalidade a implementação de uma aplicação Web para avaliação do Corpo Docente Universitário, mais precisamente, do curso de Ciências da Computação da Universidade Federal de Santa Catarina. A aplicação a ser desenvolvida utilizará tecnologias atuais e permitirá de forma dinâmica gerar relatórios estatísticos sobre o desempenho, de cada membro do Corpo Docente, durante o semestre. Os dados estatísticos serão coletados por uma Interface Web, em cujos conteúdos estarão definidas questões referentes ao desempenho semestral dos professores, de forma que essas questões serão respondidas pelos próprios alunos do curso. Com a análise dos relatórios gerados, poderão ser tomadas atitudes, pelo Departamento de Informática e Estatística, a fim de melhorar a qualidade de ensino.

Palavras-chave: Avaliação, Corpo Docente, Aplicação Web, Relatórios Estatísticos.

ABSTRACT

This academic work has as purpose the implementation of an Web Application for evaluation of the University Faculty, more necessarily, of the course of Computer Science of the Federal University of Santa Catarina. The application to be developed it will use current technologies and it will allow, in dynamic way, to generate statistical reports about the performance, of each member of the Faculty, during the semester. The statistical data will be collected by an Web Interface, whose content will be referring questions to the semester performance of the professors, thus that questions will be answered by the proper students of the course. With the analysis of the generated reports, attitudes could be taken, by Department of Computer science and Statistics, in order to improve the quality of education.

Key words: Evaluation, Faculty, Web Application, Statitical Reports.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

JSP – *Java Server Pages*

HTML – *Hyper Text Murkup Language*

XML – *eXtensible Markup Language*

EJB – *Enterprise Java Beans*

JDBC – *Java Data Base Connectivity*

XSLT - *eXtensible Stylesheet Language Transformation*

POJO – *Plain Old Java Object*

HTTP - *HyperText Transfer Protocol*

DOM - *Document Object Model*

MVC – *Model-View-Controler*

AJAX - *Asynchronous JavaScript Technology and XML*

J2EE – Java *2 Enterprise Edition*

WWW – *World Wide Web*

W3C – *World Wide Web Consortium*

DTD - *Document Type Definition*

JPG - *Joint Photographic Group*

SUMÁRIO

1. INTRODUÇÃO

A melhoria da qualidade de ensino é um objetivo amplamente almejado, principalmente pelas Universidades. Dessa forma, a avaliação feita pelos alunos colabora para o crescimento pessoal e profissional do docente. Segundo Sousa [SOUSA], pesquisas extensivas sobre vários aspectos a respeito do uso das respostas dos alunos para avaliar o desempenho do professor, têm contribuído para a compreensão do processo ensino/aprendizagem. Seria errado considerar a hipótese que todos os professores possuem uma didática e metodologia de ensino impecáveis, caso contrário de nada adiantaria uma discussão sobre a avaliação dos mesmos como uma forma de melhorar a qualidade de ensino.

A avaliação do corpo docente de uma Universidade é de grande importância para a melhoria dessa qualidade, beneficiando tanto os estudantes como a Instituição. A necessidade de implantar um sistema de avaliação dos professores do curso de Ciências da Computação da Universidade Federal de Santa Catarina é evidente, pois é a forma direta de obter a opinião dos alunos do curso sobre cada professor, mantendo o anonimato dos alunos e obtendo seu grau de satisfação.

Uma maneira eficaz de se realizar uma avaliação é através de uma aplicação Web, pois pode-se obter mais informações em um curto espaço de tempo. Aliando as tecnologias atuais somado à propagação da Internet como meio de comunicação, o problema em questão torna-se viável junto à implementação desse projeto, já que com tais tecnologias é possível garantir maior confiabilidade do sistema, melhor desempenho, modularidade e independência de sistemas operacionais. Além disso, o acesso à informação torna-se mais eficiente, dado que as ferramentas a serem utilizadas, são apropriadas para demonstrações de dados estatísticos.

Consequentemente, a função principal desse sistema de avaliação é de apenas prover os dados de forma coerente e organizada, com a finalidade de facilitar à tomada de decisão do Departamento. Cabe a ele decidir sobre as atitudes a serem tomadas para cada docente e assim agregar maior valor ao sistema de ensino, visando sempre à melhoria da educação.

1.1. OBJETIVOS

1.1.1. Objetivo Geral

Aplicar os conhecimentos de tecnologia da informação para fornecer suporte às tomadas de decisões, no que diz respeito à avaliação do corpo docente, focando na análise de dados, a fim de proporcionar ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina fontes de informações estatísticas visando à melhoria da qualidade de ensino. Utilizar ferramentas livres e gratuitas e implementar uma aplicação de baixo custo, com o intuito de expandir o sistema de avaliação de docentes, para benefício tanto dos alunos como da própria instituição.

1.1.2. Objetivos Específicos

Primeiramente, modelar uma base de dados coerente e eficaz para armazenar os dados provenientes da avaliação em questão. Posteriormente, utilizar-se dos conhecimentos das tecnologias Web para desenvolver uma aplicação a fim de obter dados, através da avaliação do corpo docente, onde os alunos irão avaliá-los através de um questionário específico. Finalmente, pôr em prática os conhecimentos estatísticos na análise dos dados coletados, disponibilizando-os em formas de gráficos e histogramas, facilitando assim a visualização dos resultados das avaliações.

1.2. JUSTIFICATIVA

A capacidade de se expressar dos professores juntamente com sua didática e relacionamento com os alunos são importantes aspectos a serem considerados para aumentar a aprendizagem do aluno. Os docentes têm recebido muito pouca preparação para o desempenho de suas funções, podendo-se citar principalmente o relacionamento interpessoal. É preciso que

esta preparação mereça um cuidado especial e que os administradores também recebam o benefício de uma melhor preparação, pois a percepção que se tem é a de que para melhorar o ensino é preciso trabalhar tanto com professores, como administradores e com alunos. Acredita-se que dado o necessário suporte institucional é responsabilidade do professor a devida quantidade e qualidade de melhoria do processo ensino/aprendizado. Segundo Sousa [SOUSA] é extremamente importante aumentar a consciência do professor a respeito de sua própria filosofia, objetivos, estilos de ensino, e também aumentar a sua familiaridade e flexibilidade no emprego de métodos alternativos para alcançar seus objetivos. Isso tudo tem que ter por base um ambiente centrado no aluno, onde o professor é incentivado a se interessar desde a preparação das aulas e metodologias de ensino à interação com seus alunos. Além disso, é necessário ressaltar que os professores não mudam sua maneira de ensinar a menos que se esforcem para isto, que tenham oportunidade e sejam ajudados a encontrar meios para manter novas atitudes, comportamentos e métodos.

1.3. MOTIVAÇÃO

A busca pela melhoria da qualidade de ensino é hoje a principal alavanca para o surgimento de inovações e à introdução de mudanças. Tal fato faz com que as Universidades busquem aprimorar esta qualidade, de modo que haja um consenso sobre a necessidade de se oferecer aos alunos a melhor qualidade de ensino possível, bem como formação de caráter pessoal e profissional. Contudo, o maior empecilho de hoje ainda é o mesmo há tempos, ou seja, saber quais as estratégias a serem empregadas para prover um ensino de qualidade aos alunos.

O uso das respostas dos alunos para avaliar o desempenho do professor é um importante aspecto que tem contribuído para a compreensão dos processos de ensino e aprendizagem. Sendo assim, o significado da avaliação dos docentes além de contribuir para o seu próprio crescimento profissional e pessoal, dá suporte para a avaliação da qualidade desses

processos, e consequentemente colabora com o emprego de melhorias. Além disso, fatores como questões éticas devem ser amplamente consideradas, de modo a preservá-las, e cuja influência colabora com a valorização do professor.

1.4. DESCRIÇÃO DO PROBLEMA

No curso de Ciências da Computação da Universidade Federal de Santa Catarina há um grande índice de desistências dos alunos, segundo consta na relação de formandos frente ao número de alunos ingressantes. Tal fato, revela dois principais motivos para que isso ocorra, sendo eles, a falta de afinidade do aluno com o curso e desapontamento dos alunos com a qualidade do curso. Dentre esses motivos o segundo é o mais importante, pois é dever do Departamento de Informática e Estatística prover um ensino de qualidade. Ultimamente tem-se ouvido muitas reclamações dos alunos referentes às matérias ministradas por alguns docentes, tal fato pode ser facilmente comprovado por qualquer aluno da graduação, pois faz parte de seu cotidiano. A insatisfação dos alunos mostra que a qualidade de ensino não está adequada, porém muitos se sentem oprimidos ou mesmo tímidos em reivindicar a melhoria de qualidade nas aulas ministradas.

O sistema de avaliação do corpo docente vem facilitar e estreitar a relação dos alunos com o curso, pois permitirá que o aluno opine sobre o desempenho de cada professor, mantendo o anonimato daquele que opinou. Dessa maneira, os alunos irão se sentir mais seguros e confiantes em sugerir, criticar e apontar falhas no sistema de ensino de cada professor. A partir dessas avaliações pode-se gerar estatísticas e compará-las com avaliações de semestres anteriores, cabendo ao Departamento interpretar essas informações e tomar as medidas cabíveis.

2. BASE DE DADOS E CAMADA DE PERSISTÊNCIA

Neste capítulo serão apresentados alguns conceitos básicos sobre banco de dados relacional bem como a apresentação do conceito de Camada de Persistência. A modelagem da base de dados dará uma noção geral de como a aplicação tratará os dados e a maneira que estarão armazenados, já a Camada de Persistência mostrará as tecnologias empregadas na integração entre a aplicação *Web* com a base de dados.

2.1. BASE DE DADOS

2.1.1. Definição

De acordo com Date [2003], uma base de dados é um conjunto de dados com uma estrutura regular que organiza informações. Normalmente agrupa informações utilizadas para um mesmo fim. De maneira mais técnica, uma base de dados é formada por um conjunto de registros que são armazenados fisicamente em *hardware*. Para efetuar consultas, inserções, alterações e remoções de dados, são utilizados os sistemas gerenciadores de banco de dados, ou SGBD, que é o *software* responsável pela manipulação desses dados.

Os registros, ou tuplas, geralmente estão divididos em atributos, são os atributos que dão as características ao registro e o tipo de informação ao qual se refere. Dessa maneira, alguns registros podem conter dados brutos ou então referenciar outros registros, a esse referenciamento é dado o nome de relacionamento de dados.

Para se compreender uma base de dados, precisa-se primeiro classificá-la. Uma base de dados pode ser classificada através do seu modelo de dados, cujos principais tipos são:

- Modelo Navegacional
- Modelo Relacional
- Modelo Orientado a Objetos

2.1.2. Base de Dados Relacional

Segundo Ramakrishnan [2003], uma base de dados relacional é um conjunto de relações com nomes distintos. Uma relação é uma tabela com linhas e colunas e constitui-se em um conjunto de linhas únicas e distintas, e é composta por duas partes:

- Esquema: especifica o nome da relação e o nome e o tipo de cada coluna.
- Instância: é o conjunto de registros que compõem a tabela em um dado momento, é variável e deve obedecer ao esquema da relação.

De uma maneira mais informal, uma base de dados relacional armazena todos os dados em tabelas separadas, ao invés de armazená-los em uma única tabela principal. Sendo assim, para que os dados possam ser aplicados ao contexto em que serão utilizados, precisam estar relacionados entre si, ou seja, as tabelas que os contém devem estar relacionadas. Para distinguir os dados de uma tabela são utilizadas chaves que são campos da base de dados e especificam um único registro. O fato de uma base de dados relacional manter os dados em tabelas distintas e relacionadas conforme a necessidade, permite a criação de bases de dados maiores e com desempenho de acesso aos dados mais eficiente que um modelo centrado em uma única tabela.

2.1.3. Modelo Entidade Relacionamento (MER)

De acordo com Cougo [1999], o modelo de Entidade Relacionamento descreve o mundo como: “...cheio de coisas que possuem características próprias e que se relacionam entre si”. As coisas em questão podem ser pessoas, conceitos, eventos, etc. Essas coisas são entidades que possuem características próprias, sendo que cada uma delas possui uma instância distintamente identificada. Cada entidade possui duas características principais: atributos e relacionamentos. Os atributos são características que toda a instância de um tipo possui, contudo podem variar entre as instâncias. Eles dão características às informações referentes à uma entidade. As

entidades podem se relacionar entre si, sendo esta a principal característica do modelo de Entidade-Relacionamento, pois permite que informações de entidades sejam relacionadas segundo certas restrições.

Os tipos de relação utilizados no Modelo Entidade-Relacionamento são:

- Relação 1..1: Relação um para um. Indica que as tabelas possuem relacionamento único entre si. Em um contexto hipotético por exemplo, uma pessoa está relacionada com apenas um cargo.
- Relação 1..N: Relação um para muitos. Indica que uma instância de uma entidade está relacionada com várias instâncias de outra entidade. Em um contexto hipotético por exemplo, um curso possui várias disciplinas.
- Relação N..N: Relação muitos para muitos. Indica que várias instâncias de uma entidade se relacionam com várias instâncias de uma outra entidade. Em um contexto hipotético por exemplo, várias pessoas estão relacionadas com vários projetos nas quais ela trabalham.

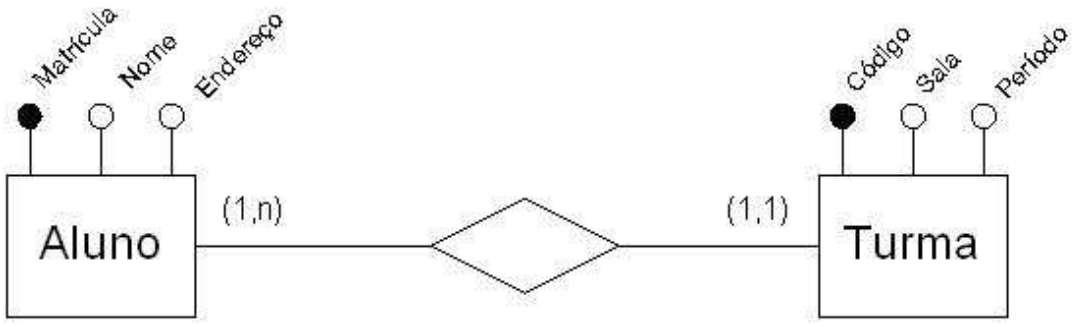

Figura 1: Exemplo de Diagrama Entidade Relacionamento

2.2. CAMADA DE PERSISTÊNCIA

2.2.1. Persistência de dados com Hibernate

Neste tópico, serão apresentados, de forma sucinta, alguns conceitos sobre Hibernate. O Hibernate é um *framework* de persistência de dados de alta performance e serviços de consultas. Permite o desenvolvimento de classes persistentes seguindo o padrão de orientação a objetos, incluindo associação, herança, polimorfismo, composições e coleções. Além disso, permite expressar consultas em seu próprio formato, o HQL, bem como SQL nativo. Uma de suas principais características é a persistência de objetos em banco de dados relacionais de maneira transparente, através de qualquer tipo de aplicação Java, seja ela baseada em um sistema *Web* ou mesmo em um sistema *Desktop*. Tal característica é mais conhecida como mapeamento, onde classes Java representam tabelas do banco de dados relacional, simplificando a camada de persistência de dados e aumentando seu nível de abstração.

Algumas outras características que fazem do Hibernate o *framework* de persistência de dados mais utilizado no âmbito Java:

- Mapeamento Flexível: permite mapear as classes Java da melhor maneira possível, conforme as necessidades do projeto e utilizando estratégias de mapeamento.
- Facilidade nas consultas e manuseio de dados: as consultas são expressas de maneira amigável e a paginação é aplicada ao nível do banco de dados, usando-se um dialeto SQL específico para cada tipo. Neste caso utiliza-se o JDBC, uma biblioteca específica para acesso ao banco de dados, que irá efetuar a conexão com o banco utilizando seu próprio dialeto.
- Facilidades no uso de Metadados: Metadados são utilizados para facilitar o mapeamento, tal como mapeamentos baseados no uso de anotações.
- Nível de sessão: Cada sessão possui seu próprio conjunto de objetos instanciados e utiliza-se de *cache* para resolver referências circulares e requisições repetidas.

- Performance: utiliza um sistema de inicialização de conjuntos de objetos conforme a necessidade. Possui uma arquitetura de alta escalabilidade.
- Integração: Possui suporte ao EJB 3.0, JBoss e arquiteturas J2EE.

Figura 2 – Representação da Arquitetura do Hibernate.

2.2.2. Mapeamento Objeto/Relacional com JPA

JPA é um acrônimo para *Java Persistence API*, é uma API de persistência de POJOs para mapeamento objeto/relacional, que contém uma completa especificação ao suporte do uso de metadados de anotações na linguagem Java [7]. Inspirado em frameworks de mapeamentos Objeto/Relacional tal como o Hibernate, o JPA utiliza anotações para mapear os objetos para uma base de dados relacional. Tais objetos são conhecidos como entidades.

Entidades JPA são POJOs que não estendem e nem implementam classes. Com o JPA não é necessário utilizar-se de arquivos XML para descrever esses mapeamentos, como é feito sem o uso dessa tecnologia, onde é necessário criar arquivos de mapeamento referentes a cada POJO que representa uma tabela de uma base de dados relacional. A seguir, exemplo básico de um POJO mapeado com JPA:

```
@Entity
@Table(name = "cliente")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)

public class Cliente {

  @Id
  @GeneratedValue
  private Long id;
  @Column(name = "telefone", length = 15)
  private String telefone;
  @Column(name = "e_mail")
  private String email;
  // contrutores, métodos getters e setters
}
```

Neste exemplo, a classe mapeada com JPA é identificada com a anotação **@Entity**, que significa que ela é uma entidade mapeada para uma tabela da base de dados relaciona. A anotação **@Id** dá a característica de chave primária da tabela ao atributo id. Já a anotação **@Column**, significa que o atributo logo abaixo é uma coluna da tabela correspondente e que possui suas características, como o nome físico que consta na tabela, tipo da informação, tamanho máximo da informação, no caso de ser uma *String*, etc.

A anotação **@Table** indica que o POJO a ser mapeado corresponde à tabela do nome passado e a anotação **@Inheritance** indica o tipo de estratégia utiliza para o relacionamento entre as tabelas, que neste caso é do tipo TABLE_PER_CLASS, onde cada classe é mapeada para uma tabela separada.

3. CONCEITOS BÁSICOS DE PROGRAMAÇÃO WEB

Este capítulo visa apresentar alguns conceitos e tecnologias necessárias para a compreensão do trabalho desenvolvido.

3.1. HTML

Criada por Tim Berners-Lee em meados de 1990 o HTML (HyperText Markup Language, ou Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada para produzir páginas na Web.

3.2. Protocolo HTTP

HTTP é a sigla em língua inglesa de HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto) e é um protocolo da camada de Aplicação do modelo OSI, utilizado para transferência de dados na rede mundial de computadorers, a World Wide Web. O mesmo é utilizado para transferir dados de imagens, sons e texto [WIKIPEDIA 2007].

Este protocolo se comunica na linguagem HTML, contudo, para haver comunicação, com o servidor da página Web, teremos de utilizar comandos próprios do mesmo, os quais não são em linguagem HTML.

Este protocolo tem uma relevância por ser o mais utilizado na rede mundial de computadores, principalmente em sistemas cliente-servidor que são a grande maioria. De forma simplificada os computadores clientes, em geral computadores mais simples, realizam requisições, que são codificadas segundo este protocolo, à servidores que geralmente são computadores mais robustos que realizam maiores processamento, estes por sua vez processam a requisição e devolvem uma resposta ao cliente.

3.3. XML

O acrônimo XML significa Linguagem de Marcação Extensível (*eXtensible Markup Language*). Esta linguagem utiliza marcações para

descrever e transportar dados de maneira padrão, e atualmente é recomendada pela W3C como um padrão desde 1998. Atualmente essa linguagem utiliza *Document Type Definition* (DTD) ou XML *Schema* para se descrever as regras de validação e definições da linguagem do documento XML [W3Schools XML].

Devido à facilidade de uso e a diversidade de funções disponíveis o XML tem sido amplamente utilizado para armazenar dados compartilhados em diversos programas e no transporte de informações pela internet.

3.4. JSP

Java Server Pages ou JSP é uma tecnologia orientada à criação de páginas Web com programação em Java. Com JSP podemos criar aplicações Web que são executadas em vários servidores *Web*, de múltiplas plataformas, já que Java é em essência uma linguagem multi-plataforma. As páginas JSP são compostas de código HTML/XML misturados com *tags*, estrutura de marcação que possui breves instruções contendo uma marca de início e outra de fim, especiais para programar scripts de servidor com a sintaxe Java. A tecnologia JSP dá maior dinamicidade ás páginas HTML, através de expressões JSP como: *tags*, *scripts* e expressões regulares, além de possibilitar ao programador a criação de novas *tags* que podem ser reutilizadas em outras páginas como bibliotecas.

Os arquivos com extensão “.jsp” incluem, dentro da estrutura de uma pagina HTML, as sentenças Java a serem executadas no servidor. Antes que os arquivos sejam funcionais, o JSP realiza uma fase de tradução dessa página em um servlet, implementado em um arquivo class (Byte codes de Java). Esta fase de tradução se realiza habitualmente quando se recebe a primeira solicitação da página JSP, embora exista a opção de pré-compilar em código para evitar esse tempo de espera na primeira vez que um cliente solicita a página.

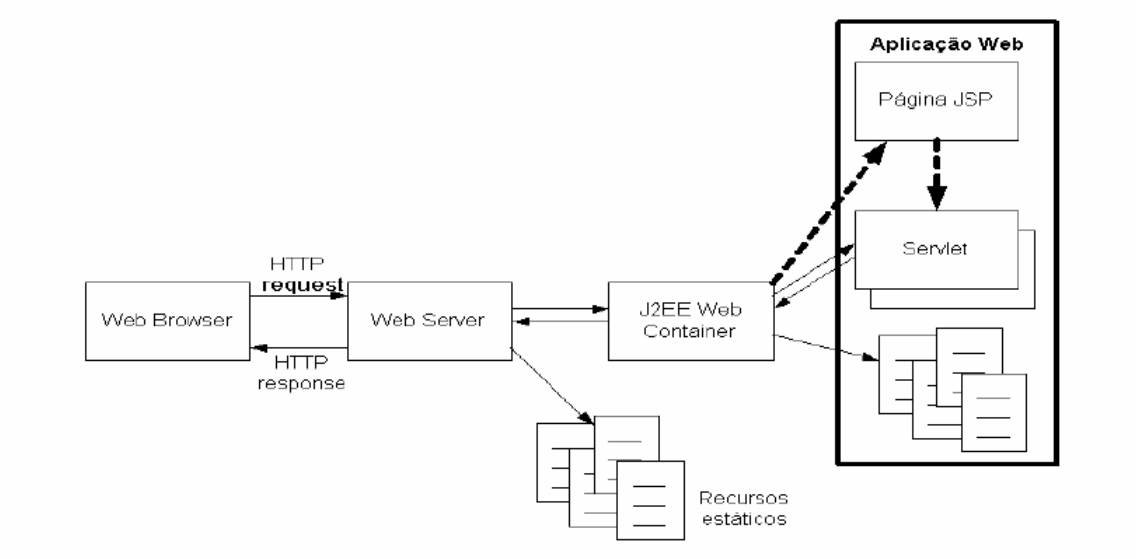

Figura 3 - Interação entre os componentes em uma Aplicação Web

3.5. DOM

O Modelo de Objeto de Documento (*Document Object Model*) é uma plataforma que permite o acesso e alteração de seus dados, estrutura e estilo dinamicamente por programas e scripts. Sua estrutura é baseada na estrutura de árvore e atualmente é a forma padrão da W3C para armazenar os dados de uma página Web [W3C DOM].

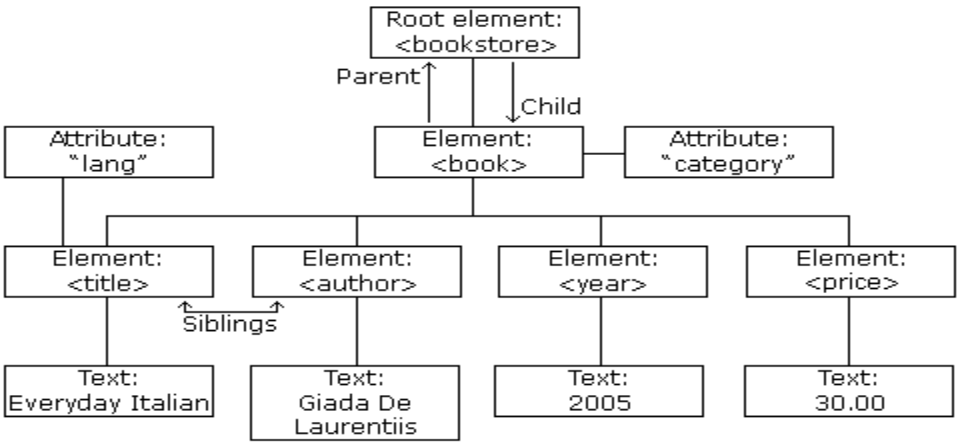

Figura 4 – Representação da arvore de elementos do DOM – fonte: [W3Schools DOM]

Atualmente existem duas API's para o DOM:

- **DOM *core*:** se restringe ao controle e suporte para alterações no documento.

- **DOM HTML:** fornece suporte à linguagem HTML, no sentido de controlar quais operações podem ou devem ser realizadas no documento.

3.6. Javascript

JavaScript é uma linguagem de programação criada pela Netscape em 1995, que a princípio se chamava LiveScript, para atender, principalmente, às seguintes necessidades:

- Validação de formulários no lado cliente (programa navegador);
- Interação com a página. Javascript tem sintaxe semelhante a do Java, mas é totalmente diferente no conceito e no uso.

Possui as seguintes características:

- Oferece tipagem dinâmica - tipos de variáveis não são definidos.
- É interpretada, ao invés de compilada.
- Possui ótimas ferramentas padrões para listagens (como as linguagens de script, de modo geral).
- Oferece bom suporte às expressões regulares (característica também comum em linguagens de script).

Sua união com o CSS é conhecida como DHTML. Usando o Javascript, é possível modificar dinamicamente os estilos dos elementos da página em HTML.

Por essas características, esta linguagem vem sendo amplamente utilizada em programação Web, principalmente por conseguir modificar páginas HTML de forma dinâmica, sem a necessidade de reconstrução da página inteira. Apesar das melhorias causadas por esta tecnologia, ela traz maiores dificuldades na hora da manutenção do código pois geralmente mistura parte da lógica de negócios nas páginas, componentes da camada de visão, o que não é recomendado segundo o modelo MVC.

3.7. Arquitetura MVC

A arquitetura MVC (*Model-View-Controller*) visa realizar a separação do software em três camadas, modelo, visão e controle, a fim de tornar o código mais legível e reutilizável [SUN MVC].

Cada camada tem as seguintes funções:

- **Modelo**: é a representação lógica do programa, é responsável pela execução de tarefas referentes à lógica de negócios e alteração do estado do modelo.
- **Visão**: representa a interface com o usuário, é responsável pela obtenção correta dos dados e pela sua visualização e compreensão.
- **Controle**: realiza a conexão entre o modelo e a visão, sendo responsável pelo gerenciamento dos eventos, interpretando as ações dos usuários na interface para o modelo, onde são realizadas as alterações e, retorna uma visão adequada.

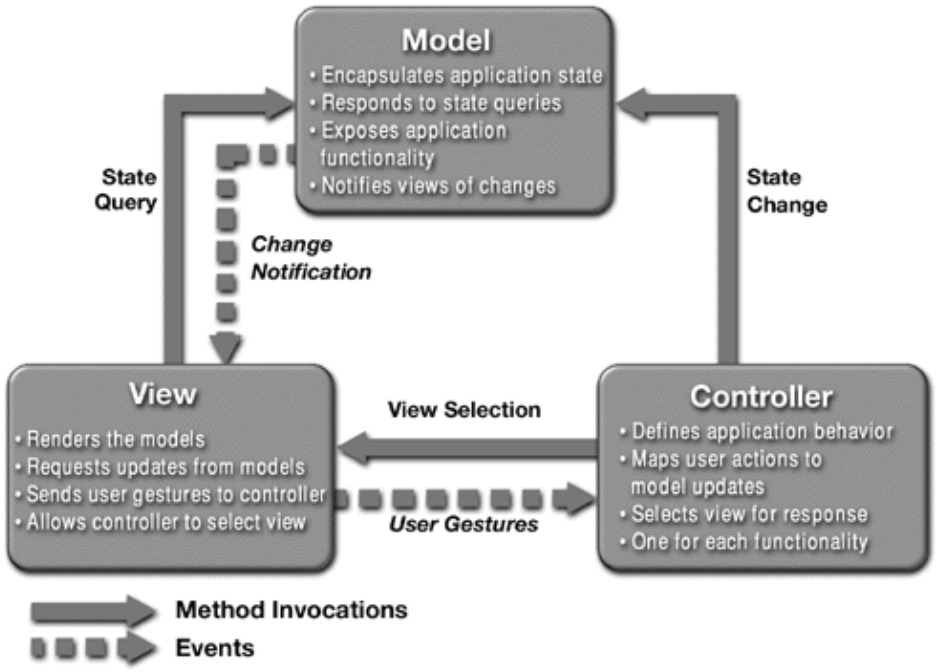

Figura 5 – Modelo MVC - fonte: [SUN MVC].

3.8. AJAX

O termo Asynchronous JavaScript Technology and XML (AJAX) surgiu a partir de conjunto de tecnologias, principalmente Javascript e XML, que se integram para realizar requisições assíncronas ao servidor a partir de navegadores com a finalidade de tornar a visualização das páginas HTML mais agradável. Esta tecnologia era pouco utilizada até que a empresa Google™, uma das maiores empresas em TI do mundo, começou a utilizar o AJAX em seus aplicativos Web.

Utilizando a tecnologia Javascript em uma página HTML pode-se realizar uma requisição assíncrona, que pode retornar dados no formato de documento XML, HTML, textual ou de notação de objetos Javascript (JSON), estes que podem ser utilizados para a atualização da página via DOM [SUN AJAX].

Abaixo temos as aplicações mais comuns do AJAX.

- Validação em tempo real: dados de um formulário que necessitam de validação no servidor podem ser submetidos isoladamente enquanto o usuário continua as suas tarefas na página.
- Auto complemento: parte do formulário é completado de acordo com o que o usuário digita em um campo.
- Carregamento sobre demanda: de acordo com dados ou eventos gerados no cliente a página HTML pode popular alguns dados de modo a carregar as páginas mais rapidamente.
- Controle de interfaces avançadas: componentes mais avançados como: árvores, tabelas, calendários e barras de progresso podem interagir melhor sem a necessidade de recarregar toda a página.
- Atualização de dados constantes: aplicações que necessitam de atualizações constantes como tabelas de estoque, pontuação, tempo entre outras não precisam atualizar todos os dados, constantemente, realizando apenas um “pooling” nos dados que devem ser atualizados.
- *Submit* parcial: um formulário pode ser parcialmente submetido, não necessitando da atualização completa da página.

3.8.1. Funcionamento

Abaixo será mostrada a figura do ciclo de vida de uma requisição AJAX com seus passos descritos.

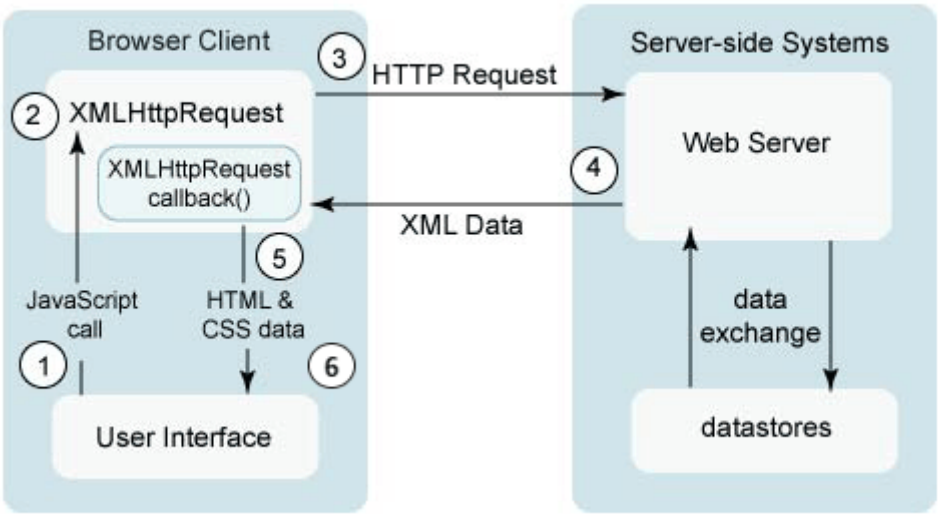

Figura 6 - Ciclo de vida de uma requisição AJAX - Fonte:[SUN AJAX]

Passo 1: O evento é gerado, acionando as funções em Javascript.

Passo 2: Um componente XMLHttpRequest é gerado, levando consigo a função que deve ser invocada no servidor.

Passo 3: O componente XMLHttpRequest é lido e sua função é executada no servidor.

Passo 4: Ocorrem as validações no servidor, caso haja necessidade das mesmas.

Passo 5: A resposta é montada no formato XML contendo os resultados da função executada no passo 3 e enviada ao cliente.

Passo 6: A resposta gerada no servidor chega ao cliente, que ativa a função em Javascript para verificar se não houve nenhum problema, caso não haja erros os dados são processados e o DOM é atualizado.

Passo 7: com o DOM do HTML atualizado, as alterações são exibidas ao usuário.

3.8.2. Vantagens

- Redução do tráfego de dados: segundo estudos, já foi comprovado que o uso do AJAX pode reduzir os dados transferidos na rede já que não há uma necessidade de se recarregar a página por completo [CRANE&PASCARELLO], como demonstra o gráfico abaixo:

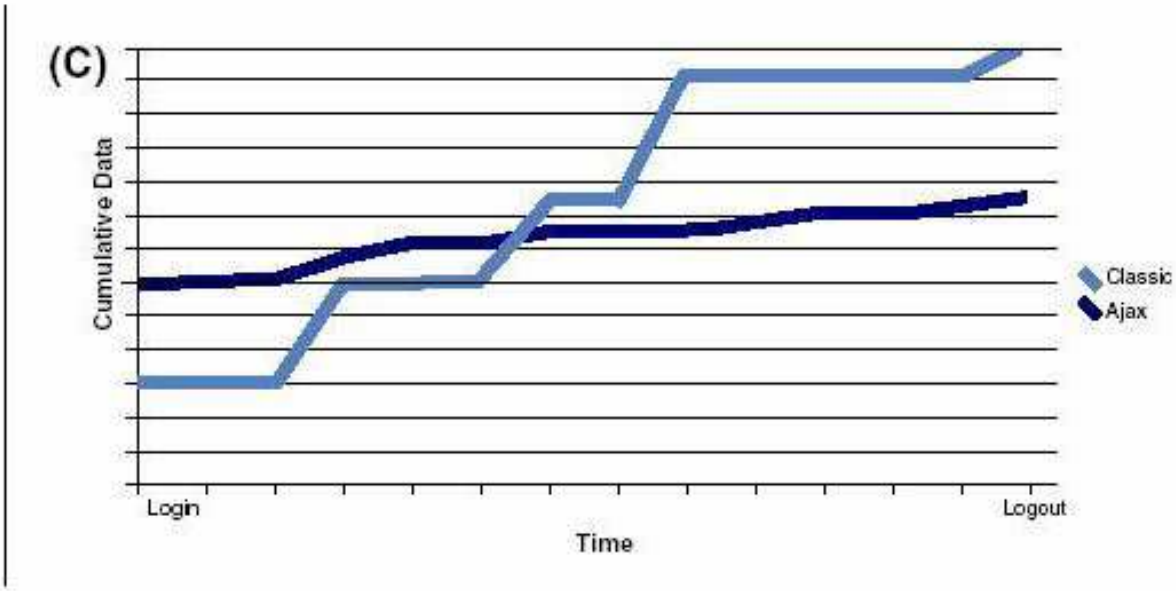

Figura 7 - Comparação do fluxo de dados entre o Cliente e o Servidor.
Fonte: CRANE&PASCARELLO, 2006.

- Maior interatividade e conforto do usuário: por estar sempre trocando informações com o servidor, o cliente se mantém atualizado e não gera um grande fluxo de dados como visto no gráfico acima, o que reduz o tempo de espera do usuário e permite uma interação mais constante e agradável.

3.8.3. Desvantagens

- Falhas nos botões voltar, favoritos e recarregar: a dinamicidade que o AJAX permite não foi prevista nos navegadores que estavam acostumados à páginas estáticas com requisições síncronas, assim os botões citados acima salvam apenas o endereço URL da página

atual, porém o AJAX possibilita alterar o DOM da mesma página HTML, causando a impressão de estar em outra página, o que na realidade não ocorre.

- Aumento da utilização de Javascript: por se tratar de uma parte da tecnologia AJAX, o aumento do uso de Javascript é inevitável, o que não é muito desejado devido a vários fatores já comentados como: transferência da lógica de negócios para visão o que dificulta a manutenção, a compatibilidade entre navegadores, que se torna problemática já que Javascript possui um comportamento diferente em cada navegador, segurança por parte da lógica ser transferida para o cliente e brechas na segurança, que podem ser abertas mais facilmente.

3.8.4. Conclusão

A tecnologia AJAX vem como uma solução para o tempo excessivo despendido pelos usuários, desde o envio de requisições ao tempo de espera para recebimento dos dados. Dessa forma, novos horizontes estão se abrindo para a utilização de páginas mais complexas e atrativas. Contudo, nota-se que ainda há alguns problemas que dificultam o uso desta tecnologia, como a implementação não unificada do Javascript e os navegadores, que ainda não possuem compatibilidades padronizadas e alternativas para problemas como a perda de URL.

4. JAVA SERVER FACES

4.1. Introdução

Java Server Faces é um *framework* baseado na estrutura MVC para desenvolvimento de aplicações *Web* em java, que teve o apoio da SUN, e foi desenvolvido e lançado oficialmente em 2002. Seu modelo de programação se diferencia dos demais *frameworks* como Struts pois é dirigido por eventos. [GEARY&HORSTANN, 2005]

O JSF participa principalmente nas camadas de visão e controle, possuindo uma vasta biblioteca de *tags*, para auxiliar na construção das páginas Web, e um *front-controller* que juntamente com os arquivos de configuração realiza o controle dos eventos no sistema.

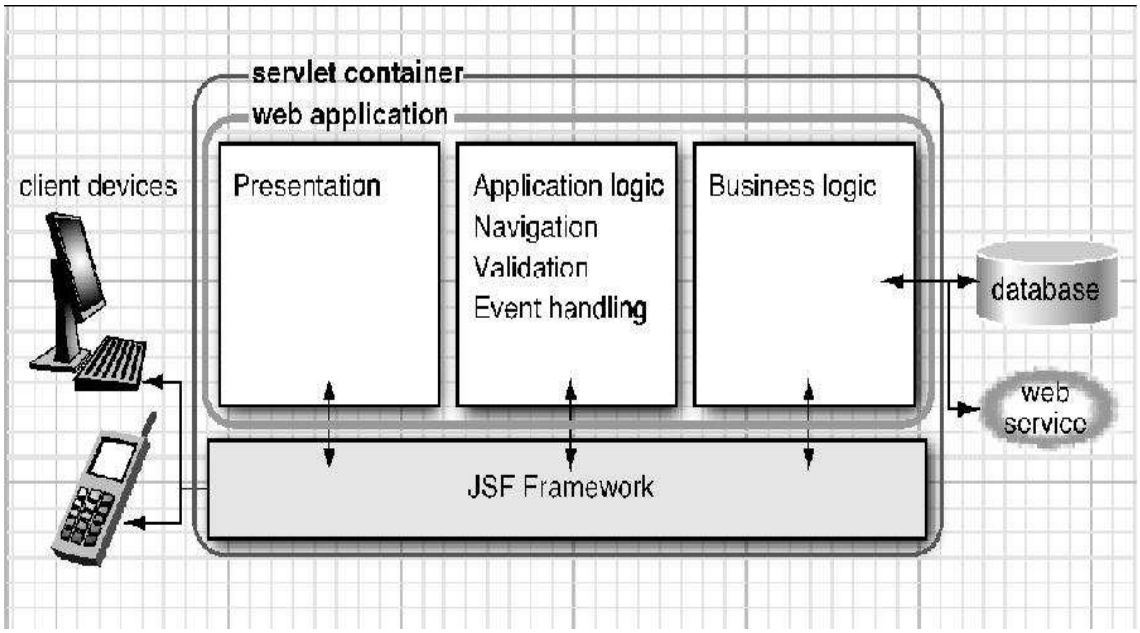

Figura 8 - Visão geral do *Framework* Java Server Faces –
Fonte: [GEARY&HORSTANN, 2007]

4.2. Componentes

4.2.1. FacesServlet

Tem o papel de controlador-frontal (*front-controller*) do *framework*, interceptando as requisições e acionando o modelo, de acordo com as configurações de navegação descritas no arquivo faces-config.xml.

4.2.2. Backing Beans

São classes que expõe objetos de negócios para a camada de apresentação, representando de forma parcial ou total um formulário. Geralmente possuem atributos e seus respectivos métodos “get” e “set” para que o framework possa ter acesso aos atributos.

4.2.3. Faces-config.xml

É o principal arquivo de configurações do *framework*, contendo as equivalências necessárias para a conexão entre o modelo e a visão, neste arquivo são declarados basicamente os Backing Beans juntamente com suas classes java equivalentes e o escopo em que os Beans serão armazenados, os quais podem ser:

Request: escopo de requisição, dura apenas no curto período da requisição HTTP até a resposta do mesmo, utilizado em requisições mais rotineiras.

Session: escopo de sessão o Bean é mantido na sessão do navegador (HttpSession) até que a sessão expire por tempo de inatividade ou seja fechada através do método *invalidate*, geralmente é usado para tarefas longas como por exemplo, um carrinho de compras.

Application: escopo de aplicação, persiste enquanto durar a aplicação Web, é compartilhado por todas as requisições e todas as sessões.

Exemplo:

```
<managed-bean>
        <managed-bean-name>user</managed-bean-name>
        <managed-bean-class>com.corejsf.UserBean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Leitura: o Bean que possui o nome “user” está associado à classe contida em “com.corejsf.UserBean” e possui escopo de sessão. Além das configurações dos Backing Beans são configuradas as regras de navegação que serão utilizadas no FacesServlet, especificando-se a página que gera o evento, seguido dos casos de navegação que indicarão o s*tring* de retorno da requisição e a página que deverá ser carregada.

Exemplo:

```
  <navigation-rule>
        <from-view-id>/index.jsp</from-view-id>
        <navigation-case>
         <from-outcome>login</from-outcome>
        <to-view-id>/welcome.jsp</to-view-id>
        </navigation-case>
  </navigation-rule>
```

Leitura: se na página contida no local “/index.jsp” alguma ação retornar o *string* “login” então vá para página contida em “/welcome.jsp”

4.2.4. Expression Language

Os componentes JavaServer faces das páginas Web podem ter acesso às propriedades e funções dos Backing Beans através de expressões especiais do *framework*, as *expression languages*, onde geralmente se começa com o nome do Bean seguido do nome do atributo ou função como a expressão a seguir:

```
#{turma.professor.nome}
```

Que equivale a seguinte operação no modelo:

```
Turma.getProfessor.getNome();
```

4.3. Funcionamento

Os passos de codificação e decodificação são importantes para uma melhor compreensão do funcionamento do JSF, para compreender o que de fato ocorre com as *tags* do *framework*, e poder tirar um melhor proveito.

Codificação: após a codificação das páginas utilizando JSF, estas necessitam ser codificadas para um formato em que o navegador possa compreender, ou seja, em formato HTML, passo que *framework* realiza sempre que uma nova página é requisitada. Após a leitura do arquivo, é montada uma árvore de componentes, em que cada componente é tratado por seu *handler*, gerando o código HTML, e da iteração desta árvore são gerados identificadores para cada um de modo que o *framework* possa associar e reconhecer seus componentes equivalentes no modelo.

Decodificação: após a exibição da página, sempre que forem enviados dados do formulário, o *framework* precisa reconhecer os campos de forma a gerar os Beans corretamente. Ele realiza esta tarefa pelo processo inverso ao anterior, reconhecendo seus identificadores (que chegam da requisição na forma de *strings* em pares ID/Valor, como: “id0=valor0&id1=valor1”), nos componentes e verificando na árvore de componentes.

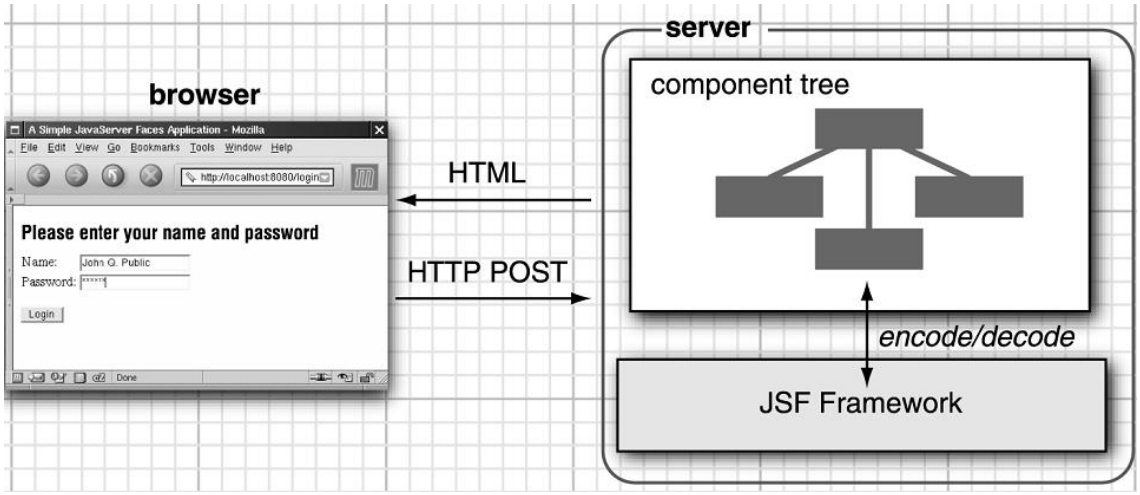

Figura 9 - Codificação / Decodificação - fonte: [GEARY&HORSTANN, 2007]

4.3.1. Ciclo de Vida

Aqui serão descritas as seis etapas do ciclo de vida de uma requisição ao JSF, da requisição até a resposta. Abaixo segue a ilustração do ciclo, onde as linhas sólidas representam o fluxo normal, completo, e as linhas tracejadas os fluxos alternativos.

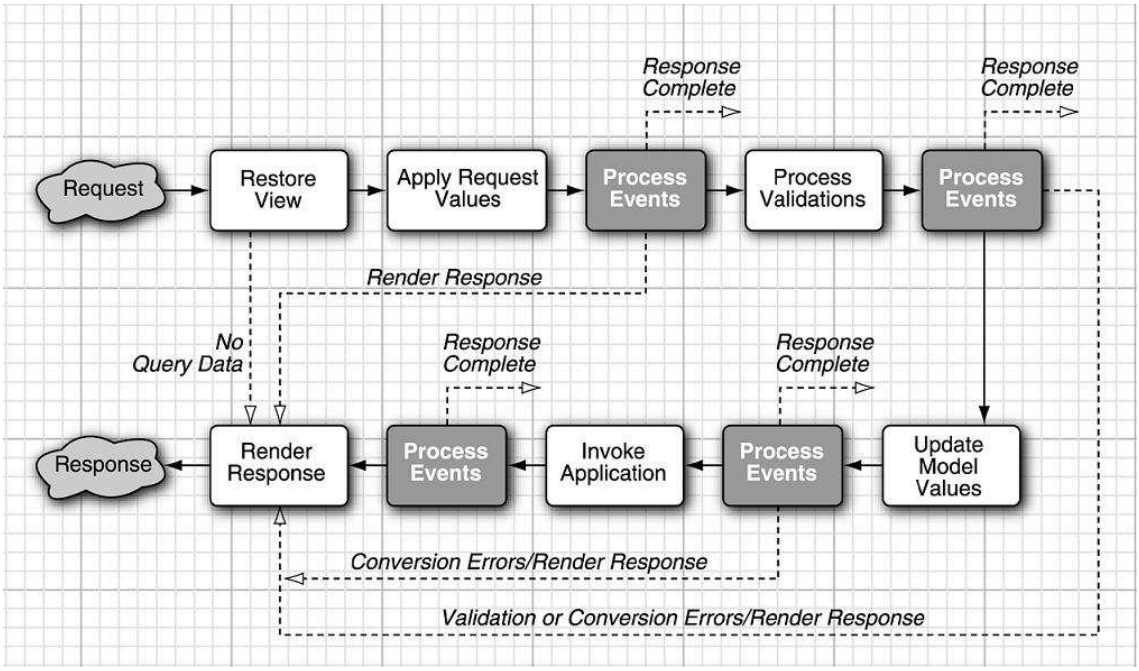

Figura 10 - Ciclo de vida de uma requisição JSF - fonte: [GEARY&HORSTANN, 2007]

Fase 1: Restaurar visão – Restore View

Recupera a árvore de componentes ou monta a mesma caso seja a primeira vez em que a página é exibida, salvando esta num objeto do tipo FacesContext.

Fase 2: Aplicar Valores de requisição – Apply Request Values

Esta etapa será realizada apenas se houver algum dado requisitado, caso contrário o próximo passo será renderizar a Resposta. Nesta fase os dados são armazenados localmente de acordo com a árvore de componentes, onde são realizadas as conversões dos dados em *string* da requisição para os tipos declarados nos Beans. Além disso, é montada uma fila de eventos que podem

ser executados após cada, seguindo diretamente para fase de renderizar a resposta ou mesmo terminar o processamento caso alguma exceção ocorra.

Caso o atributo “immediate” seja *true*, então a validação e os eventos associados ao componente serão antecipados para essa fase.

Fase 3: Processar validações – Process Validations

A última etapa consiste na montagem da página de retorno de acordo com os processos anteriores. Os valores submetidos e com seus tipos corretos sofrem uma nova validação, agora pelo método indicado no atributo “validator” onde usualmente são realizadas validações sintáticas. Caso ocorra algum erro em alguma conversão um evento de erro de validação é gerado para desviar o fluxo ao último passo, retornando assim a página com seus valores para que o usuário possa corrigir os dados inválidos.

Fase 4: Atualizar valores do modelo – Update Model Values

Após a validação dos dados os Beans associados à página são atualizados de acordo com a árvore de componentes através dos métodos set.

Fase 5: Invocar aplicação – Invoke Application

Neste passo o modelo já tem seus dados atualizados, então é executado o método vinculado à propriedade “action” do componente que gerou o evento, tal método retorna uma *string* que é passada a um *handler* de navegação. O *handler* é responsável por encontrar a página seguinte, definida pelas regras de navegação no arquivo faces-config.xml.

Fase 6: Renderizar resposta – Render Response

Na última fase ocorre a montagem da página resultante de acordo com os processos das fases anteriores.

4.4. Vantagens

- Possui inúmeros componentes para auxiliar na construção de interfaces.
- Facilidade para criação de novos componentes.
- Bom suporte de várias IDEs tais como eclipse e Netbeans.
- Modelo de programação orientado a eventos.
- Suporte a internacionalização.
- Associa os eventos do lado cliente com os manipuladores dos eventos do lado servidor (os componentes de entrada possuem um valor local representando o estado no lado servidor).

4.5. Desvantagens:

- Ainda não é um *framework* maduro no mercado.
- Só possui requisições síncronas.
- Aumento de requisições ao servidor.

4.6. Conclusão

O *framework* JSF que se foca principalmente nas camadas de visão e controle traz consigo um maior conforto aos desenvolvedores, por ser direcionado a eventos, assemelhando-se à programação *Desktop*, que juntamente com as suas *taglibs* facilitam a criação de novos componentes.

A programação orientada a eventos do *framework* causa um aumento de fluxo considerável à aplicação já que a cada evento gerado é feita uma submissão do formulário completo ao servidor, causando processamentos desnecessários o que acarreta num maior tempo de resposta ao usuário.

Finalizando, foi constatado que o JSF traz grandes melhorias na produção, por facilitar as tarefas de quem vai desenvolver, porém não é tão vantajoso ao usuário se não for usado de forma correta, pois pode vir a degradar o servidor pelo uso excessivo de eventos na busca de soluções mais fáceis.

5. CONCEITOS BÁSICOS DE ESTATÍSTICA

5.1. Definição

Segundo Bussab [BUSSAB], a estatística é uma área do conhecimento que utiliza teorias probabilísticas para explicação de eventos, estudos e experimentos. Tem por objetivo obter, organizar e analisar dados, determinar as correlações que apresentem, tirando delas suas conseqüências para descrição e explicação do que passou e previsão e organização do futuro.

A estatística é na prática uma ciência de desenvolvimento de conhecimento humano através do uso de dados empíricos. Baseia-se na teoria estatística, um ramo da matemática aplicada. Na teoria estatística, a aleatoriedade e incerteza são modeladas pela teoria da probabilidade. Algumas práticas estatísticas incluem, por exemplo, o planejamento, a sumarização e a interpretação de observações. Porque o objetivo da estatística é a produção da melhor representação possível da informação a partir dos dados disponíveis.

5.2. Medidas-Resumo

5.2.1. Mediana

É a realização que ocupa a posição central da série de observações, quando são ordenadas em ordem crescente. Assim, se as cincos observações de uma variável forem 3, 4, 7, 8 e 8, a mediana é o valor 7, correspondendo à terceira observação. Quando o número de observações for par, usa-se como mediana a média aritmética das duas observações centrais. Acrescentando-se o valor 9 à serie acima a mediana será (7 + 8) / 2 = 7,5.

5.2.2. Média Aritmética

Corresponde à soma das observações dividida pelo número delas, por exemplo, 3, 4, 7, 8 e 8: (3 + 4 + 7 + 8 +8) / 5 = 6.

5.2.3. Quartil

É qualquer um dos três valores que divide o conjunto ordenado de dados em quatro partes iguais, e assim cada parte representa um quarto da amostra ou população. Os quartis são divididos em:

- primeiro quartil ou quartil inferior: é o valor que corresponde aos 25% da amostra ordenada, também denominado como 25º percentil.
- segundo quartil ou mediana: é o valor onde se encontra 50% da amostra ordenada, denominado também como 50º percentil.
- terceiro quartil ou quartil superior: valor onde se encotram 25% dos valores mais elevados, também denominado como 75º percentil.

5.3. Gráfico de Setores

Mais conhecido como gráfico de Pizza, destina-se a representar a composição, usualmente em porcentagem, de partes de um todo. Consiste num círculo de raio arbitrário, representando o todo, dividido em setores, que correspondem a partes de maneira proporcional. A figura abaixo representa um gráfico de setores [BUSSAB].

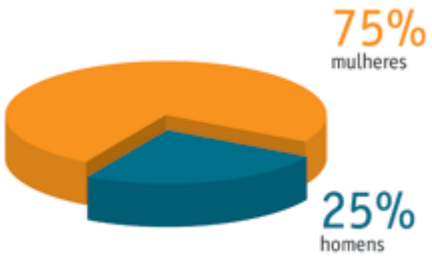

Figura 11 - Exemplo de gráfico de setores.

5.4. Gráfico em Barras

O gráfico em barras consiste em construir retângulos ou barras em que uma das dimensões é proporcional à magnitude a ser representada, sendo a outra arbitrária, porém igual para todas as barras. Essas barras são dispostas

paralelamente umas às outras, horizontal ou verticalmente, como no exemplo a seguir [BUSSAB].

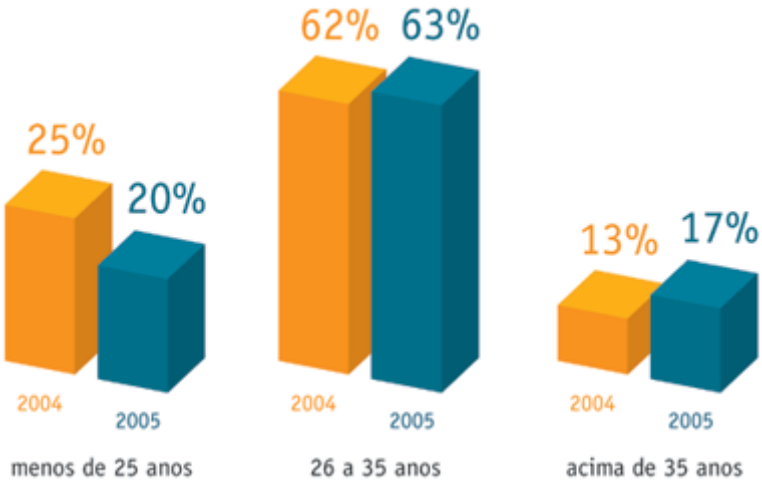

Figura 12 - Exemplo de gráfico em barras vertical.

5.5. *Box Plot*

Para se construir um Box Plot, consideramos um retângulo onde estão representadas a mediana e os quartis. A partir do retângulo, para cima, segue uma linha até o ponto mais remoto que não exceda LS = q3 + (1,5)dq, chamado *limite superior*. De modo similar, da parte inferior LI = q1 – (1,5)dq, chamado limite inferior. Os valores compreendidos entre esses dois limites são chamados *valores adjacentes*. As observações que estiverem acima do limite superior (LS) ou abaixo do limite inferior (LI) estabelecidos serão chamados *pontos exteriores* e representados por pontos. Essas são as observações destoantes das demais e podem ou não ser o que se chama de *outliers* ou valores atípicos.

O *Box Plot* passa uma idéia da posição, dispersão, assimetria, caudas e dados discrepantes. A posição central é dada pela mediana e a dispersão por dq. As posições relativas do primeiro, segundo e terceiro quartis dão uma noção de assimetria da distribuição. Os comprimentos das caudas são dados pelas distâncias entre o retângulo, que contém a maior concentração dos dados, e os valores extremos. A seguir o exemplo de um *Box Plot*.

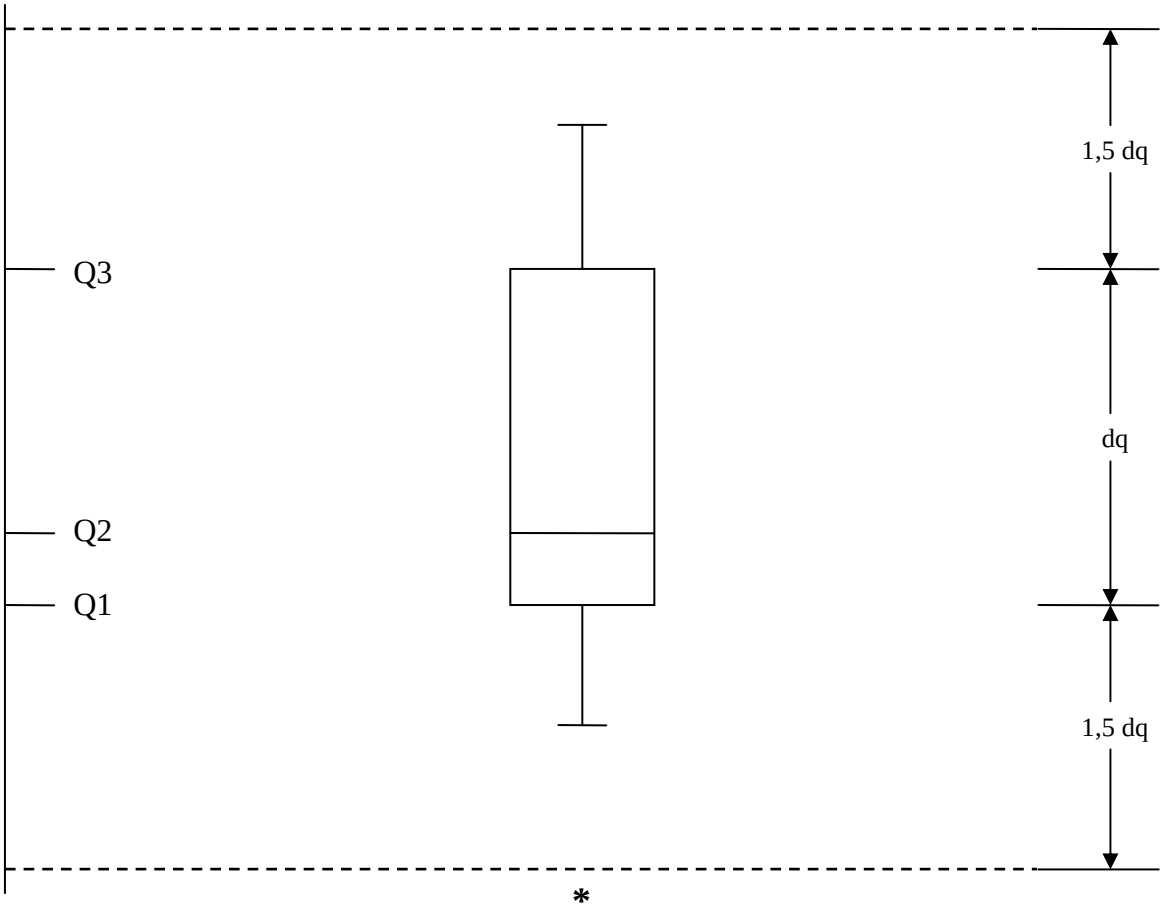

Figura 13 - *Box Plot* [BUSSAB].

5.6. Escala de Atitude

O fato de avaliar tanto com o critério de uma escala (sentimento do aluno) qualitativa e ao mesmo tempo atribuindo uma nota, permite analisar a coerência da resposta do aluno utilizando indicadores estatísticos (entre eles a correlação entre respostas, no caso nota/sentimento, através do coeficiente de contigência no caso da escala). É uma forma de validar as respostas dadas, ou seja, se elas estão coerentes, por exemplo, se um aluno atribui um sentimento: Concorda Parcialmente, e uma nota: 3 à uma determinada questão, isto implica que há divergência nesta resposta. O importante no trabalho é obter um alto grau de confiabilidade e de validade a partir do sistema de avaliação.

6. DESENVOLVIMENTO DO PROJETO

Este capítulo tem como objetivo descrever claramente os passos aplicados para o desenvolvimento do projeto de Avaliação do Docente do Departamento de Informática e Estatística. Além disso, este capítulo tratará desde a modelagem da base de dados ao desenvolvimento da camada de apresentação, aliando teoria e prática, de modo a compreender a utilização das tecnologias atuais com o intuito de solucionar os requisitos desejados do sistema de avaliação do docente.

Primeiramente, foram apresentados no capítulo 2 os conceitos básicos referentes à base de dados e à camada de persistência, responsáveis pelo armazenamento dos dados, no contexto da aplicação.

No capítulo 3, foram indicados os conceitos referentes à Programação para *Web*, necessários para a compreensão do modo como são tratadas as informações a fim de obter os dados desejados. Há ainda conceitos utilizados há algum tempo, mas que estão sendo aplicados largamente às tecnologias atuais tal como o Ájax.

No capítulo 4, é apresentada uma síntese do *framework* utilizado, o JSF, para o contexto da Programação para *Web*. Como este âmbito é muito vasto e possui muitas alternativas de tecnologias, este capítulo tem por finalidade mostrar o funcionamento do JSF, as vantagens e as desvantagens na sua utilização.

Por fim, no capítulo 5, são passados os conceitos básicos de estatística empregados no projeto. A finalidade deste capítulo é mostrar na prática a real utilidade dos gráficos na interpretação dos dados, repassando a informação de maneira amigável. No escopo do projeto, a geração dos gráficos e conseqüentemente a sua interpretação é de suma importância e deve ser de boa confiabilidade, pois irá demonstrar o desempenho do professor perante sua turma. Segundo o pensador chinês Confúcio, “Uma imagem vale por dez mil palavras”.

6.1. Definição do Escopo

Este trabalho teve como base para definição do escopo o Departamento de Informática e Estatística, que necessita, há algum tempo, de um sistema de avaliação dos docentes com a finalidade de melhorar a qualidade de ensino. O INE é composto atualmente por 58 professores, sendo 31 doutores, 24 mestres e 3 com títulos de especialização, contudo esses títulos de formação não são garantias na qualidade de ensino. Este projeto tem como foco a avaliação dos docentes por parte dos discentes, através de uma aplicação Web prática e objetiva, visando obter dados para auxiliar a tomada de decisão da coordenadoria do INE frente ao desempenho do professor na disciplina na qual ele leciona.

6.2. Análise de Requisitos

6.2.1. Requisitos Não-Funcionais

- Utilização de ferramentas gratuitas e/ou OpenSource.
- Suporte a multiplataforma.
- Independência de Banco de Dados.

6.2.2. Requisitos Funcionais

- Aluno e Coordenador são identificados no sistema.
- A validação do usuário será feita pelo seu nome de usuário.
- Páginas iniciais diferentes para cada tipo de usuário.
- O aluno pode avaliar o professor durante um período estipulado, nas disciplinas às quais estiver matriculado.
- O coordenador poderá ter acesso a todos os dados estatísticos de todas as disciplinas do curso em que coordena, gerando gráficos para cada turma de uma disciplina.
- O coordenador do curso poderá estipular a data de início e fim das avaliações.

6.3. Modelagem da Base de Dados

O banco de dados utilizado no Projeto foi o PostGreSQL, versão 8.2. O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados. [POSTGRES]. Já para a modelagem da base de dados foi utilizada a ferramenta Aqua Data Studio, versão 4.5, gratuita para desenvolvimento sem fins lucrativos.

A modelagem foi feita com base na necessidade de armazenamento dos dados. Existem treze tabelas, das quais dez contém os dados propriamente ditos, cujas tabelas são: pessoa, aluno, professor, curso, turma, disciplina, questão, questionário, semestre e resposta. E três tabelas que explicitam o relacionamento entre tabelas, que são: rel_turma_aluno (relacionamento entre turma e aluno), rel_curso_prof (relacionamento entre curso e professor) e rel_turma_professor (relacionamento entre turma e professor). A figura abaixo mostra o diagrama Entidade-Relacionamento da base de dados modelada para o escopo do projeto.

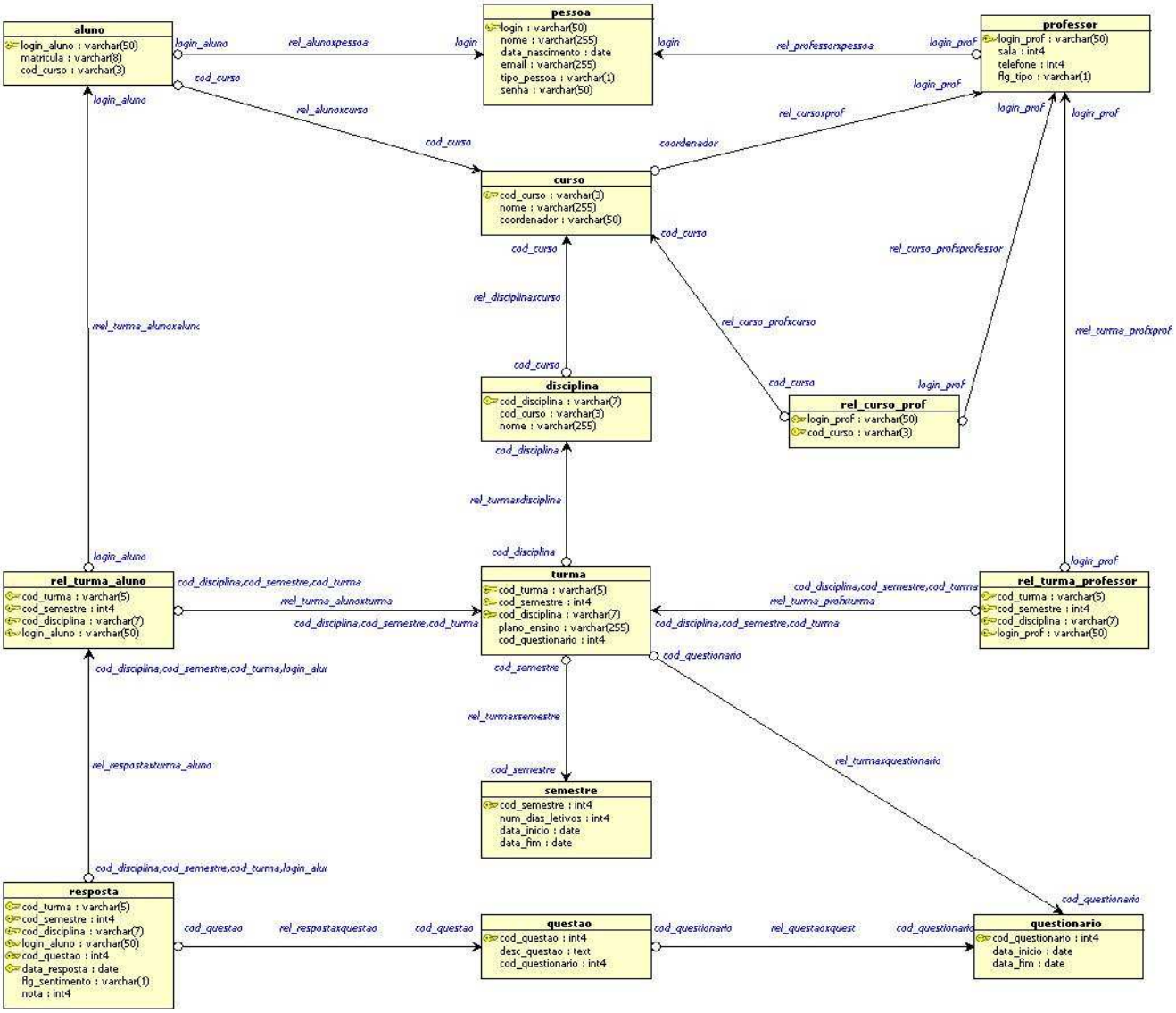

Figura 14 - Diagrama Entidade-Relacionamento

6.4. Camada de Persistência

6.4.1. Persistência de Dados

Para o processo de persistência de dados, foi utilizado o *framework* Hibernate, versão 3.2. As bibliotecas utilizadas foram: hibernate3.jar, o driver JDBC para o banco de dados PostGresSQL, postgresql-8.2-504.jdbc3.jar e a biblioteca de tratamento de conexões, proxool-0.9.0RC2. A configuração do Hibernate é feita através do arquivo hibernate.cfg.xml, e este deve conter as classes que estão mapeadas. Por exemplo:

```
<mapping class="br.ufsc.inf.sigrad.pojo.Aluno"/>
```

Além disso, devem ser configurados alguns parâmetros, tais como, o *pool* de conexões, indicando qual será o arquivo XML conterá as informações para que o Hibernate possa se conectar, e o dialeto que irá indicar qual sintaxe deve utilizar, dependendo do tipo do banco de dados, nesse caso o PostGreSQL.

6.4.2. Mapeamento Objeto/Relacional com JPA

Para efetuar o mapeamento Objeto/Relacional teve-se que criar os POJOs referentes à cada tabela do banco de dados. Foram mapeadas apenas as tabelas que não são de relacionamento. Ou seja, foram criados nove POJOs, que estão representados pelas seguintes classes em Java: Pessoa.java, Aluno.java, Professor.java, Curso.java, Disciplina.java, Questão.java, Questionário.java, Turma.java, Semestre.java e Resposta.java. A classe Resposta.java apesar de representar uma tabela de relacionamento, entra no processo de mapeamento, pois contém um atributo que é um dado propriamente dito, e não apenas uma referência a um outro dado.

No caso das tabelas que contém chaves primárias compostas, o mapeamento deve utilizar uma outra estrutura. Nesse caso, as chaves primárias serão atributos de uma classe que irá representar as chaves compostas como sendo um único objeto. No mapeamento com JPA essa característica é conhecida com EmbeddedId, ou seja, haverá uma anotação na classe que indicará que as chaves compostas serão formadas por um objeto que conterá essas chaves, como no exemplo a seguir:

- Turma.java

```
@Entity
@Table(name="turma")
public class Turma implements Serializable{
        @EmbeddedId
        private TurmaPK pk;
  // outros atributos, métodos getters e setters
```

- TurmaPK.java

```
@Embeddable
public class TurmaPK implements Serializable{
```

```
        @Column(name="cod_turma", length=5, nullable=false)
        private String codTurma;

        @ManyToOne(targetEntity=Semestre.class)
    @JoinColumn(name="cod_semestre",     referencedColumnName="cod_semestre",
nullable=false)
        private Semestre semestre;

    @ManyToOne(targetEntity=Disciplina.class)
    @JoinColumn(name="cod_disciplina",     referencedColumnName="cod_disciplina",
nullable=false)
        private Disciplina disciplina;
  // outros atributos, métodos getters e setters
```

Este exemplo mostra que a chave primária de Turma é constituída por três atributos: o código da Turma, o Objeto Semestre e o Objeto Disciplina. Para o caso dos atributos que são objetos, o mapeamento irá reconhecer automaticamente o identificador do Objeto, que deverá estar mapeado, indicando qual é o seu atributo identificador. O exemplo abaixo, indica o mapeamento correto para o Objeto Disciplina:

```
@Entity
@Table(name="disciplina")
public class Disciplina implements Serializable{

@Id
@Column(name="cod_disciplina", length=7, nullable=false)
private String codDisciplina;
// outros atributos, métodos getters e setters
```

6.5. Camada de Negócios

A camada de negócios consiste em processar no servidor as informações provenientes da camada de apresentação e/ou da camada de persistência. Esse processamento é acionado através de uma requisição feita pelo cliente.

No escopo deste trabalho, a camada de negócios irá tratar eventos gerados na camada de apresentação, situada em um cliente, visto que o modelo empregado é do tipo cliente-servidor. Além disso, também acessará a

camada de persistência para recuperação de dados ou mesmo alteração e exclusão.

A camada de negócios deste projeto é composta por classes Java, Backing Beans, que representam as ações provenientes das suas respectivas páginas. Por exemplo: Quando um cliente efetuar *login* no sistema, a camada de negócios, mais precisamente a classe LoginBean.java, irá receber uma requisição e validará se o usuário existe e se a senha passada por este usuário condiz com a registrada no banco de dados, e então permitirá o acesso à página caso a senha for válida, caso contrário é mostrada uma mensagem de erro. A seguir o diagrama de caso de uso, as ações que os usuários podem executar.

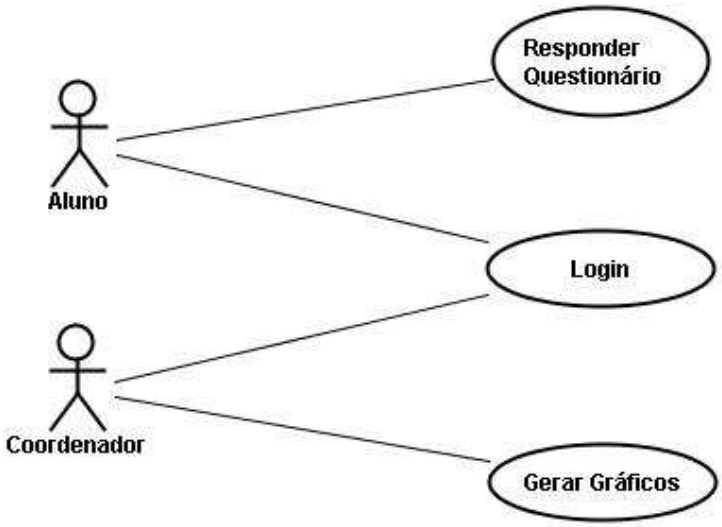

Figura 15 - Casos de Uso

6.5.1. Ação Login

Nesta ação, o usuário passa como parâmetros um *login* e uma senha, para poder se autenticar no servidor. Logo após que o usuário clicar no botão entrar, é enviada uma requisição ao servidor que chamará a classe LoginBean.java. Esta classe contém os métodos de validação e os métodos “getters” e “setters”. A validação do usuário é feita com base nos parâmetros passados. Quando a requisição chega ao Backing Bean LoginBean, ele acessa a base de dados e recupera o POJO Pessoa de acordo com o *login* fornecido. Sendo assim, o POJO Pessoa irá conter um atributo senha caso exista o *login* passado. Então, a senha enviada pelo usuário é comparada à senha contida no

POJO e se forem equivalentes o usuário é redirecionado para sua página inicial, de acordo com o tipo de pessoa, que pode ser aluno ou coordenador. Caso o POJO Pessoa não possa ser recuperado, ou seja, retorne nulo, é enviada uma resposta ao usuário contendo uma mensagem de erro de autenticação.

6.5.2. Ação Responder Questionário

A ação responder questionário está restritamente relacionada aos usuários que são alunos. A partir do momento que um usuário é autenticado como aluno, sua página inicial conterá uma lista de disciplinas as quais está matriculado. Dessa forma, o usuário irá escolher uma disciplina para responder o questionário. Caso o questionário escolhido já tenha sido respondido é enviada uma mensagem de erro. Caso contrário, o usuário é redirecionado à página de questionários referentes à disciplina escolhida. Este processo, é requisitado ao HomeAlunoBean que irá recuperar as disciplinas do aluno autenticado ao carregar a página inicial do aluno. Após escolher a disciplina e clicar no botão questionário, o Backing Bean responsável buscará no banco de dados se o aluno em questão já respondeu ao questionário.

Ao carregar a página de questionário, o Backing Bean QuestionarioBean irá consultar na base de dados as questões associadas à turma a qual o aluno pertence. Com base nas questões recuperadas do banco de dados e informações referentes à disciplina, tal como nome e código da disciplina, é montada a página para que possa ser efetivada a avaliação. Para cada questão está relacionado uma lista que expressa sentimento do aluno, com os possíveis valores: Discordo totalmente, Discordo parcialmente, Concordo parcialmente e Concordo totalmente. Além da lista de sentimento do aluno, também está relacionada à cada questão um campo de seleção onde pode ser escolhida uma nota entre 0 e 10. Para cada questão esses valores devem ser obrigatoriamente preenchidos, caso contrário o QuestionarioBean enviará uma mensagem notificando a falta de preenchimento de algum campo necessário para a finalização da avaliação. Após a finalização, as respostas fornecidas pelo usuário serão armazenadas no banco de dados, que registrará a data que foi respondido o questionário.

6.5.3. Ação Gerar Gráficos

A ação gerar gráficos é restritamente aplicada ao coordenador, após a sua autenticação no sistema. Ao carregar a página inicial do coordenador, o Backing Bean HomeProfessorBean irá buscar no banco de dados, as disciplinas do curso o qual ele coordena. Para efetuar a consulta aos gráficos de avaliação, devem ser passados parâmetros de consulta que irão filtrar os dados na geração dos gráficos. Esses parâmetros são: Data Inicial (referente à data de início da avaliação), Data Final (referente à data final da avaliação), Disciplina (uma lista que já vem preenchida com as disciplinas relacionadas ao curso), Turma (lista que é preenchida conforme a disciplina escolhida) e Semestre (semestre em que o questionário foi respondido). Após preenchidos os parâmetros de consulta, a ação gerar gráficos chamada pelo usuário é enviada ao HomeProfessorBean que se encarregará de recuperar os dados no banco de dados e retornará um arquivo de imagem JPG com o gráfico selecionado, caso exista dados, caso contrário o Backing Bean notificará o usuário que não existem dados. Os tipos de gráficos que podem ser escolhidos são:

- Gráfico em Barras
- Gráfico em Setores (mais conhecido como Gráfico de Pizza)
- *Box Plot*

Se o gráfico selecionado for o de Setores, habilitará o campo Questão, onde o usuário poderá especificar que a geração do gráfico seja feita para determinada questão, no caso para o número da questão.

6.6. A biblioteca JFreeChart

O JFreeChart é uma biblioteca gratuita inteiramente desenvolvida em Java, utilizada para facilitar o desenvolvimento de gráficos com qualidade profissional. Esta biblioteca foi utilizada neste projeto, pois trata-se de uma ferramenta de fácil utilização e por fornecer a geração de vários tipos de gráficos, tais como o Gráficos de Setores, em Barras e *Box Plot*. Os gráficos a seguir foram gerados com esta biblioteca no projeto em questão.

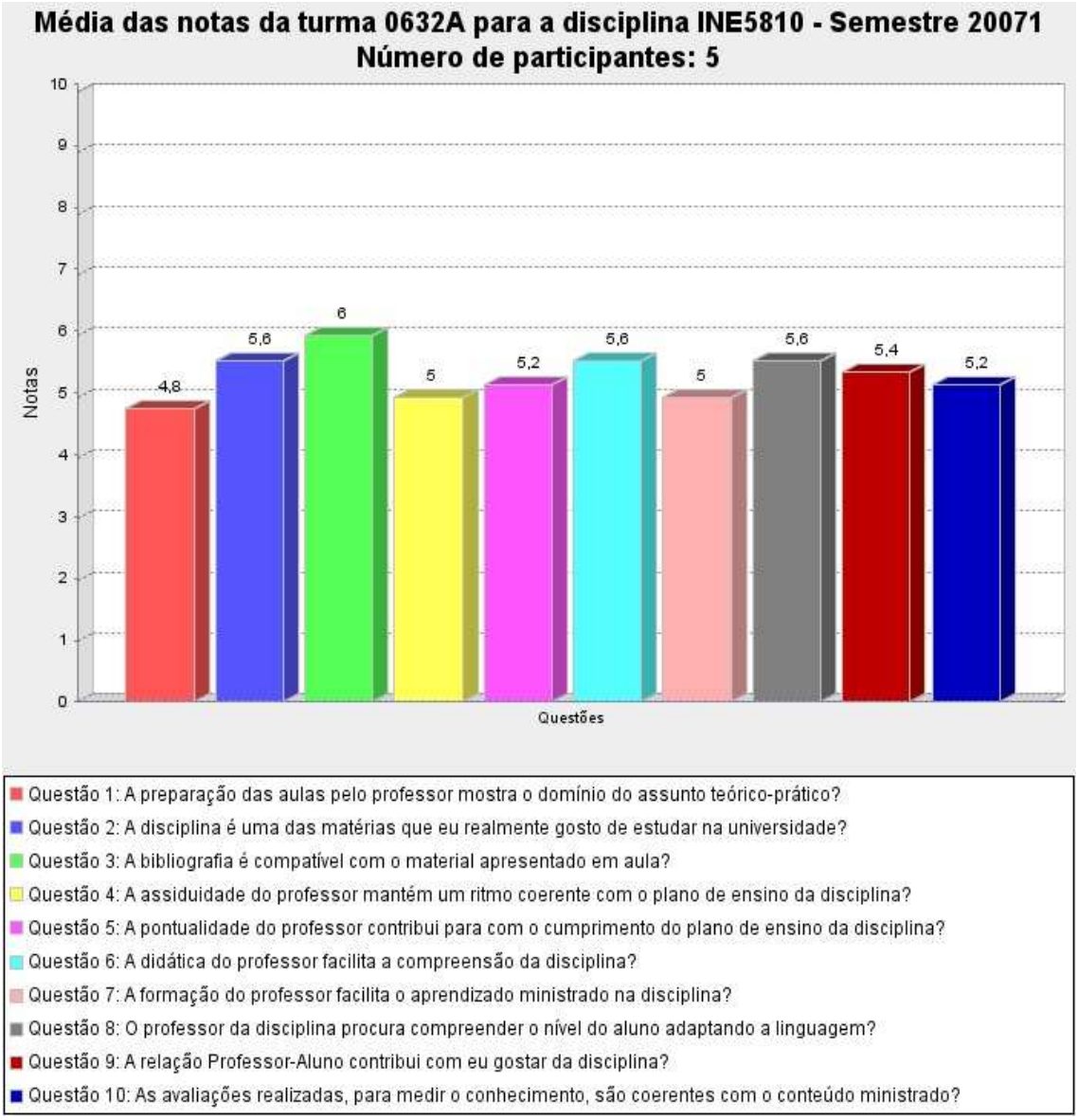

Figura 16 - Exemplo de Gráfico em Barras gerado pela aplicação

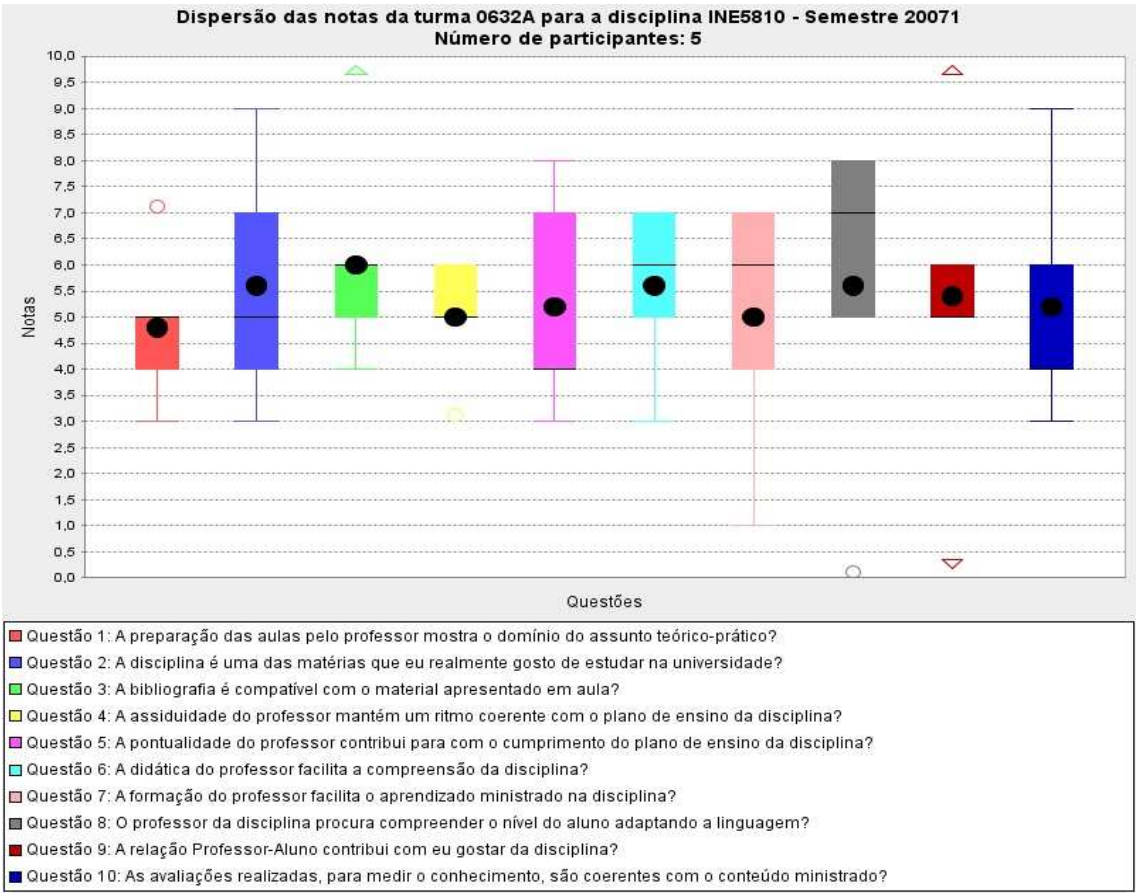

Figura 17 - Exemplo de *Box Plot* gerado pela aplicação

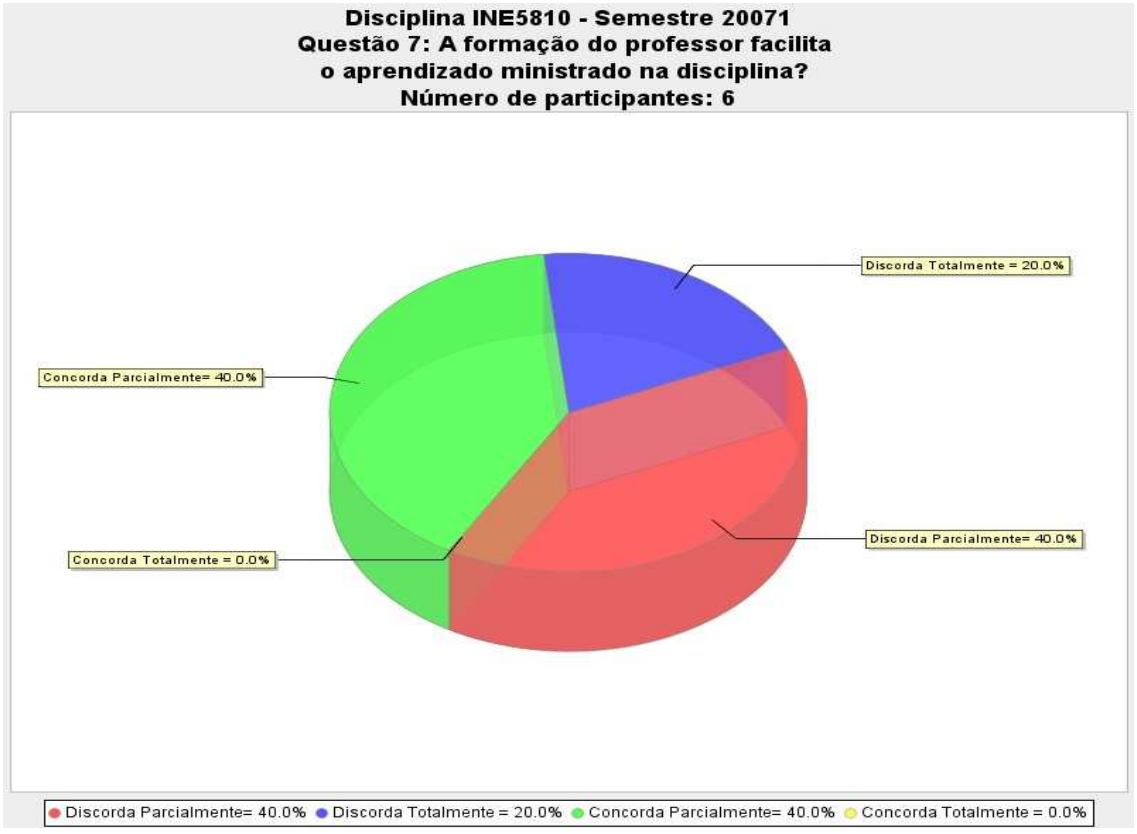

Figura 18 - Exemplo de Gráfico de Setores gerado pela aplicação

7. CONCLUSÃO

O presente trabalho se propõe a disponibilizar de uma forma prática e coerente um questionário de avaliação do corpo docente do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina. Em testes reais realizados com alunos, a aplicação mostrou-se eficiente e confiável, gerando corretamente os gráficos relacionados a cada turma de uma disciplina. Contudo, deve-se ressaltar que a divulgação de tal avaliação é de extrema importância, já que os conceitos de estatística podem ser mais bem aplicados em um maior volume de dados, trazendo uma representação considerável dos dados coletados.

A conscientização dos alunos sobre a importância de se responder corretamente o questionário é imprescindível para o sucesso da avaliação. Todavia, deve-se respeitar ainda a coerência das respostas, dada a liberdade dos discentes considerando-se a sua não-obrigatoriedade. Além do mais, o anonimato pode estimular o discente a divergir suas respostas no momento da avaliação, o que causa discrepância dos dados dificultando a estimativa de desempenho do docente.

Finalmente, a interpretação dos gráficos gerados é de responsabilidade do coordenador, visto que para o escopo deste trabalho, o mesmo utilizará os gráficos para averiguar o desempenho do docente, focando sempre na melhoria da qualidade de ensino, afinal um dos anseios das Universidades públicas é atualmente buscar sempre a melhor formação do discente.

7.1. Sugestões para Trabalhos futuros

A idéia original deste trabalho era fazer um sistema integrado da graduação, onde o aluno poderia acessar um portal contendo informações referentes ao curso, desde suas disciplinas do semestre até as avaliações dos professores. Contudo, devido à complexidade e disponibilidade de tempo para realização de todos os requisitos desejados, seu escopo foi reduzido apenas à avaliação do corpo docente. Sendo assim, algumas sugestões para trabalhos futuros seriam:

- Desenvolvimento do portal do aluno, onde estariam disponíveis planos de ensino das disciplinas bem como as notas das turmas em cada semestre.
- Integração dos dados junto ao CAGR, Controle Acadêmico da Graduação.

Bibliografia

1. Ramakrishnan, Database management systems, Ramakrishnan, Gehrke, McGraw Hill, 2003.

2. Cougo, Paulo. Modelagem Conceitual e Projeto de Banco de Dados, Campus, Rio de Janeiro. 1999.

3. Barbieri, Carlos. Modelagem de Dados. IBPI Press, Rio de Janeiro 1994.

4. Date, C. J. An Introduction to Database Systems, 8ª Edição. Addison-Wesley Longman. 1999.

5. Hibernate. **Relational Persistence for Java and .NET.** Disponível em: <http://www.hibernate.org>. Acesso em: 15 de Maio de 2007.

6. JPA. **Java Persistence API FAQ**. Disponível em: <http://java.sun.com/javaee/overview/faq/persistence.jsp>. Acesso em: 18 de Maio de 2007.

7. WIKIPEDIA 2007, Wikkipedia http. Disponível em: <http://pt.wikipedia.org/wiki/HTTP >. Acesso em: Março de 2007.

8. GEARY&HORSTANN, 2005 GEARY, David. CAY, Horstmann. Core JavaServer Faces. Alta Books. 2005. p.1-168.

9. GEARY&HORSTANN, 2007 GEARY, David. CAY, Horstmann. Core JavaServer Faces. Disponível em: <http://www.horstmann.com/corejsf/>. Acesso em: abril de 2007.

10. W3Schools XML, W3SChools, Introduction to XML. Disponível em: <http://www.w3schools.com/xml/xml_whatis.asp>. Acesso em abril de 2007.

11. W3C DOM, **The World Wide Web Consortium, Document Object Model (DOM)**. Disponível em: <http://www.w3.org/DOM/>. Acesso em: Maio de 2007.

12. SUN MVC, SUN Microsystens, **Java BluePrints Model-View-Controller**. Disponível em <http://java.sun.com/blueprints/patterns/MVC-detailed.html>. Acesso em: abril de 2007.

13. SUN AJAX, artigo: SUN AJAX, SUN Microsystens, Asynchronous JavaScript Technology and XML (Ajax) With the Java Platform. Disponivel em: http://java.sun.com/developer/technicalArticles/J2EE/AJAX/. Acesso em: abril de 2007.

14. CRANE&PASCARELLO, CRANE, Dave; PASCARELLO, Eric. Ajax in Action. Greenwich. 2006. 650 p.

15. SOUSA, E.C.B.M. de. A Importância da Avaliação de Docentes para seu próprio crescimento profissional e para a melhoria da qualidade de ensino. In: *Anais do V Encontro Brasileiro sobre ensino de Engenharia Química*, Itatiaia, RJ.

16. BUSSAB, W. de O. Estatística Básica. [S.l.]: Saraiva, 2003.