Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Eletrônica e de Computação

PROSSYS – Product Search System – Sistema de busca de produtos via Web

Autor: _____
Victor Gutmann

Orientador: _____
Prof. Sérgio Barbosa Villas-Boas, Ph. D.

Examinador: _____
Prof. Flávio Luis de Mello, D. Sc.

Examinador: _____
Prof. Aloysio de Castro Pinto Pedroza, D. Sc.

DEL

Fevereiro de 2010

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

AGRADECIMENTO

Dedico este trabalho aos meus pais, que sempre me apoiaram durante todo o percurso de minha vida, acompanhando todos os meus passos. Agradeço também a todos os meus amigos, que me deram animo e forças quando mais precisei. Alem deles, devo agradecer ao povo brasileiro, que possibilitou a minha formação nesta universidade publica.

RESUMO

O projeto é um sistema Java Web para cadastro e consulta de produtos em geral. Ele possui toda a estrutura necessária para a inclusão e a manutenção de empresas e estabelecimentos interessados em divulgar seus produtos, como, também, meios para consultas de preços por parte de clientes. Os objetivos envolvidos na realização deste projeto são o estudo das ferramentas utilizadas e, juntamente com os conhecimentos adquiridos ao longo do curso, a apresentação de uma forma de estruturar e codificar um sistema envolvendo banco de dados, internet e diferentes atores utilizando o programa.

Palavras-Chave: java, banco de dados, web, hibernate, struts.

ABSTRACT

The project is a Java Web system for registration and consultation products in general. It has all the structure needed for the inclusion and maintenance of companies and stores interested in advertising their products, and also search methods for customers. The goals involved in making this project are the study of the tools used and, together with the knowledge gained throughout the course, to provide a way to structure and codify a system involving database, internet and different actors using the program.

Key-words: java, database, web, hibernate, struts.

SIGLAS

API – Application Programming Interface

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

IDE – Integrated Development Environment

JSP – JavaServer Pages

JVM – Java Virtual Machine

SGBD – Sistema Gestor de Base de Dados

UFRJ – Universidade Federal do Rio de Janeiro

XML – eXtensible Markup Language

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 – Tema

O tema do projeto é a realização de um sistema web que possibilite maior controle e facilidade para divulgação de produtos à venda por empresas e fornecedoras, além de permitir uma busca mais simples e prática desses produtos ofertados aos seus clientes. Sendo assim, pretende-se estudar, modelar e codificar um software de uma forma mais clara, correta, eficiente e que a sua manutenção e atualização seja fácil e simples de ser executada.

1.2 – Delimitação

As entradas de dados, seja pelas empresas ou pelos clientes, são, de um modo geral, verificadas e validadas pelo sistema. Distorções de preços ou outras informações incorretas cadastradas pelas empresas serão identificadas através de denúncias feitas pelos usuários do sistema.

Os principais meios de busca são por nome e tipo do produto, podendo ser ordenados por preço, localização, empresa e estabelecimento.

Apesar da arquitetura do software ser projetada para isolar a camada de banco de dados, o sistema somente é testado no MySQL.

1.3 – Justificativa

Cada vez mais aumenta o número de pessoas que possuem acesso à internet, ao mesmo tempo em que várias empresas disputam entre si por uma fatia do mercado para vender seus produtos e serviços.

Também podemos afirmar que a quantidade de pessoas que consultam sites de busca de preços é grande. Esses sites, muitas vezes, não oferecem uma ferramenta prática e rápida de se consultar produtos ou não possuem opiniões dos usuários sobre determinado produto ou empresa.

Com o objetivo de dar suporte a esse cenário, será contruída uma ferramenta que auxilie tanto as empresas que disponibilizam suas mercadorias quanto os compradores na hora de decidir em qual lugar comprar.

1.4 – Objetivos

O sistema pretende facilitar a procura e anúncios de produtos via web, por ser um meio bastante difundido e prático.

Com esse sistema pretende-se criar um método sistemático, rápido e eficiente para anúncio e procura de produtos, facilitando tanto o anúncio, por atingir um maior número de clientes com um menor custo e maior controle, quanto a procura, por fornecer ao cliente facilidade, comodidade e eficiência na sua procura.

1.5 – Metodologia

A metodologia de trabalho é um dos pontos fortes do projeto. Como ele se propõe a ser um sistema de fácil manutenção e com possíveis atualizações no futuro, é necessário que a metodologia empregada seja bem definida e coerente. Como a linguagem é Java, orientada a objetos, e trata-se de um sistema *web* onde a orientação a objetos tem-se demonstrado mais efetiva, foi escolhida uma metodologia orientada a objetos, baseada na metodologia Objectory.

A arquitetura que será utilizada no software será definida pelo padrão MVC (Model-View-Controller). Por ser um sistema grande, será preciso separar as camadas de dados (Model) e o layout (View). Essa separação é feita para que alterações feitas no layout não alterem a parte da manipulação de dados e vice-versa.

A camada de dados utilizará uma ferramenta chamada Hibernate, que é um framework para fazer o mapeamento objeto-relacional. O Hibernate facilita o mapeamento dos atributos entre o banco de dados (que no projeto será o MySQL) e o modelo objeto da aplicação por meio dos "Java Annotations".

Já a camada de layout será apresentada por páginas JSP's (JavaServer Faces), que permitem criar páginas web dinâmicas.

Para conseguir essa separação entre as camadas, introduz-se uma outra camada que serve de controle. É o Controller que direciona e controla todas as ações executadas pelo sistema, informando como e onde uma determinada ação deve ser executada.

No projeto, essa camada controladora será desevolvida com o auxilio do framework Apache Struts.

1.6 – Descrição

No capítulo 2 é feita uma pequena introdução aos sistemas e motores de busca e a ramificação para os sistemas de busca de produtos na Web. Os fundamentos e as justificativas do uso das bibliotecas e ferramentas no projeto também são apresentados.

O capítulo 3 explica toda a estrutura do projeto e como ele foi implementado.

O capítulo 4 faz uma análise dos resultados obtidos do projeto.

Finalmente, o capítulo 5 apresenta as conclusões, verificando se os objetivos propostos foram alcançados e sugestões para futuras versões do projeto.

Capítulo 2

Fundamentação

2.1 – Desenvolvimento de Sistemas de Busca

Antes dos sistemas atuais de busca de produtos na Web surgirem, já existiam softwares que faziam buscas mais genéricas, como localizar determinada palavra na internet ou pesquisar assuntos específicos de interesse dos usuários.

2.1.1 – Sistemas e Motores de Busca

Um motor de busca é um programa com o objetivo de auxiliar a procura de informações armazenadas na Web, dentro de uma rede empresarial ou de um computador pessoal. Ele permite que um usuário solicite determinado conteúdo de acordo com algum critério especial, geralmente uma palavra ou frase, e uma lista de referências que combinam com o dado critério resultante é mostrado.

Como exemplo, podemos comparar um motor de busca com uma biblioteca, onde os livros (informações) são catalogados e registrados por categorias, nome, código, etc. A diferença entre os dois é que o catálogo do motor de busca começa em branco, sem nenhum livro. Ao se realizar uma consulta, a lista de ocorrências de assuntos é criada em poucos segundos por meio de softwares, conhecidos como spiders, que vasculham toda a rede em busca de ocorrências de um determinado assunto em uma página.

Os motores de busca se “alimentam” de índices atualizados para funcionarem de forma eficiente e rápida. Eles geralmente referem-se ao serviço de busca Web, procurando informações na rede pública da Internet. Outros tipos de sistemas incluem motores de busca para empresas, as Intranets, motores de busca pessoais e motores de busca móveis. Cada um desses ambientes podem possuir forma e relevância diferentes, fazendo com que as operações entre eles possuía resultados divergentes.

2.1.2 – Sistemas de Busca de Produtos

A motivação e fundamentação para o projeto se dá pela seguinte afirmação: a maioria dos consumidores pesquisa produtos em diversos sites antes de realizar uma compra. Porém, muitas vezes, a compra não é realizada online. Ou seja, o consumidor procura o fornecedor mais próximo da sua casa ou do seu local de trabalho.

No Brasil, segundo dados do Comitê Gestor do Brasil (Núcleo de Informação e Coordenação - NIC), 24% dos usuários da Internet, a cada três meses, são usuários novos. Por serem usuários novos, muitos deles não se sentem seguros à fazer compras diretamente pela Internet.

Há dois tipos de buscas para fazer compras: as realizadas pela Web, em sites como Yahoo e Google, e as realizadas em sites especializados em busca e comparação de preços de produtos, como o BuscaPé, BondFaro, Boa Dica, etc.

Nos sites de busca na Web, a pesquisa geralmente é acompanhada de anúncios que os lojistas publicam nos resultados para palavras-chave relacionadas aos seus produtos. Por exemplo, quando um consumidor procurar por "brinquedos", aparecerão na página de resultado tanto os sites de maior relevância associados a essa palavra quanto os anúncios patrocinados.

Os sites que comparam produtos e preços mostram informações sobre o produto buscado e a lista de lojas com os seus preços. A forma que os lojistas são cobrados varia de acordo com os sites, mas, geralmente, o lojista paga um custo fixo cada vez que o usuário clica em seu endereço Web e é direcionado para o site.

Por isto as buscas devem incluir informações relevantes (preço, marcas, ofertas) e prover um número de respostas que o usuário se disponha a selecionar. Os varejistas devem poder aproveitar as buscas sem resultados para oferecer produtos bastante vendidos na área do produto procurado.

2.2 – Linguagem, Bibliotecas e Ferramentas Usadas no Projeto

O projeto é escrito em Java e utiliza o NetBeans como compilador. As informações são salvas num banco de dados MySQL, com a persistência de dados feita pelo Hibernate. A

interface com usuário é por um browser (no caso do projeto, o browser testado foi o Mozilla Firefox), sendo as páginas JSP's codificadas com a ajuda do framework Struts. Todas essas ferramentas são gratuitas, diminuindo o custo de produção do projeto. As descrições e vantagens dessas ferramentas são realizadas a seguir.

2.2.1 – Java

Java é uma linguagem de programação orientada a objeto, que teve seu início de desenvolvimento na década de 90 pela Sun Microsystems. Diferentemente da maioria das linguagens existentes, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode e depois executado por uma maquina virtual. Bytecode é uma forma intermediária entre o código Java e o código de máquina. Essa característica faz com que os programas escritos em Java sejam independentes de plataforma, podendo ser executado em qualquer sistema que possua uma JVM. A estrutura da tecnologia Java pode ser resumida pela Figura 2.1.

Entre outras características de Java, podemos destacar as seguintes:

- **Recursos de Rede:** possui grande biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP;
- **Sintaxe:** similar à Linguagem C/C++;
- **Internacionalização:** **s**uporta nativamente caracteres Unicode;
- **Garbage Collector:** desalocação de memória automática por processo de coletor de lixo.

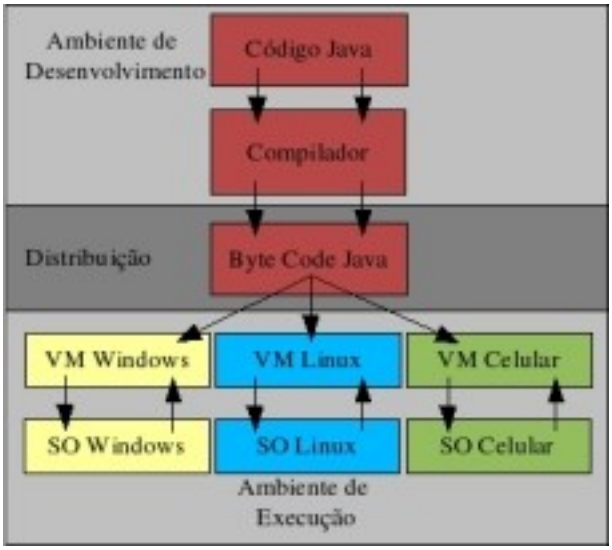

Figura 2.1 – Compilador e Interpretador Java

2.2.2 – NetBeans

A IDE NetBeans é um ambiente de desenvolvimento multiplataforma, que auxilia programadores a codificar, compilar, depurar e instalar aplicações. Foi arquitetada de forma a simplificar o desenvolvimento e aumentar a produtividade, pois reúne em uma única ferramenta todas estas funcionalidades. É integralmente escrita em Java, mas dá suporte a outras linguagem de programação, como o C, C++, Ruby e PHP, além de dar suporte a linguagens de marcação como XML e HTML.

O NetBeans fornece uma base sólida para a criação de projetos e módulos pois possui um grande conjunto de bibliotecas, módulos e API´s. Também há uma documentação vasta e bem organizada, fazendo com que o desenvolvedor tenha ferramentas para escrever seu software de maneira mais rápida.

Por ser escrito em Java, NetBeans é independente de plataforma e funciona em qualquer sistema operacional que seja compatível com a máquina virtual Java (JVM).

Alguns dos seus principais recursos são:

- **Editor de código fonte integrado:** rico em recursos para aplicações Web (Servlets e JSP);
- **Visualizador de classes integrado ao de interfaces:** gera automaticamente o código dos componentes de forma organizada, facilitando o entendimento do sistema em geral;
- **Plugins para Subversion:** Subversion é um sistema de controle de versão que permite trabalhar com diversas versões de arquivos organizados em um diretório e localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando se manipulou os arquivos;
- **CSS:** funcionalidades para editar folhas de estilos como destaques, recursos de auto-completar, análise de código;
- **Servidores web acoplados:** o NetBeans pode ser configurado para incluir aplicações Java web em servidores automaticamente;
- **Suporte a banco de dados:** possui ferramentas para manutenção de banco de dados no próprio ambiente de desenvolvimento.

2.2.3 – Banco de Dados

O banco de dados escolhido para armazenar os dados do sistema foi o MySQL. Ele é um SGBD que utiliza a linguagem SQL como interface. Entre suas características principais, estão:

- **Portabilidade:** dá suporte para praticamente todas as plataformas atuais;
- **Compatibilidade:** existência de drivers para diversas linguagens de programação (Delphi, Java, C/C++, Python, Perl, PHP, ASP, Ruby);
- **Facilidade de uso:** simples e prático de ser usado;
- **Ferramentas complementares:** possui ferramentas com interface gráfica como o MySQL Administrador e MySQL Workbench. O primeiro, como o nome diz, serve para administrar os bancos de dados, podendo criar tabelas, inserir dados, etc. Já o outro programa auxilia a criação e desenvolvimento de bancos de dados, já que é

possivel criar tabelas e relacionamentos a partir de um Diagrama Entidade Relacionamento. Os dois programas foram utilizados no projeto;

- **Desempenho:** possui um ótimo desempenho e estabilidade.

O desempenho do banco de dados é fundamental para aplicações que exijam uma grande quantidade de dados, que é o caso do projeto em questão. Em um artigo escrito por professores da Universidade Federal de Pernambuco (UFPE) é feita a comparação entre os dois bancos de dados gratuitos mais famosos (MySQL e PostgreSQL). A figura 2.2 mostra os resultados gerais obtidos.

Tabela 1 – Sumário Executivo

Tamanho do banco	512MB		1GB	
Módulo	**PostgreSQL**	**MySQL**	**PostgreSQL**	**MySQL**
Carga e Estrutura	14min	39min	45min	1h23min
Mono-Usuário	27min	8min	57min	19min
Multi-Usuário	1h06min	50min	3h43min	1h34min

Figura 2.2 – Desempenho do MySQL e PostegreSQL

Podemos observar que o desempenho do PostgreSQL foi superior ao MySQL somente no módulo de carga e estrutura. Essa diferença se dá principalmente na criação de índices. Já nos módulos mono-usuário (somente um usuário acessando o banco de dados) e multi-usuário (vários usuários tendo acesso ao banco) o MySQL teve um desempenho bem superior, chegando a ser 3 vezes mais rápido em algumas situações. Portanto, para o projeto, foi escolhido como banco de dados o MySQL.

2.2.4 – Hibernate

O Hibernate é um framework para o mapeamento objeto-relacional escrito na linguagem Java e, também, disponível em .Net com o nome NHibernate. Esta biblioteca facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, através do uso de arquivos XML ou a partir de Java

Annotations para estabelecer esta relação. No projeto foi decidido usar Java Annotations, pois as relações entre atributos das tabelas do banco de dados e dos objetos estão definidos dentro do próprio código, não precisando acessar um arquivo XML externo.

Sua principal característica é a transformação das classes em Java em tabelas de dados e dos tipos de dados Java para os da SQL. O Hibernate gera as chamadas SQL e faz com que o desenvolvedor não tenha o trabalho manual de converter os dados resultates, mantendo o programa portável para quaisquer bancos de dados SQL apesar de causar um pequeno aumento no tempo de execução.

2.2.5 – Apache Struts

Apache Struts é um framework de código aberto para criar aplicações Web em Java. A diferença entre aplicações Web de sites convencionais e das aplicações Web feitas em Java é que a última pode gerar páginas dinâmicas, ou seja, pode gerar páginas com conteúdo variável, interagindo com o usuário. Essa interação se dá através de lógicas de negócio para personalizar uma resposta.

Aplicativos da Web baseados em JSP, muitas vezes, misturam código de banco de dados, código de design da página e código de controle de fluxo. Isso faz com que o programa seja difícil de ser compreendido e sua manutenção se torna complicada.

Uma forma de separar os códigos em módulos dentro de um software é usar um padrão de aquitetura chamado Model-View-Controller (MVC), que será explicado no capítulo seguinte. O Struts é projetado para ajudar os desenvolvedores a criar aplicações web que utilizam uma arquitetura MVC e possui três características principais:

- Controle de pedidos são mapeados internamente pela aplicação por uma URI padrão;
- Controle de respostas, que transfere a lógica de negócio para outro recurso que completará o pedido;
- Biblioteca de tags que ajuda os desenvolvedores a criarem páginas dinâmicas.

Capítulo 3

Implementação

O projeto utiliza um padrão de arquitetura de software chamado Model-View-Controller (MVC). Este padrão separa a camada de layout da camada de dados e da lógica de negócio. Podemos observar o comportamento da arquitetura MVC através do modelo simplificado abaixo:

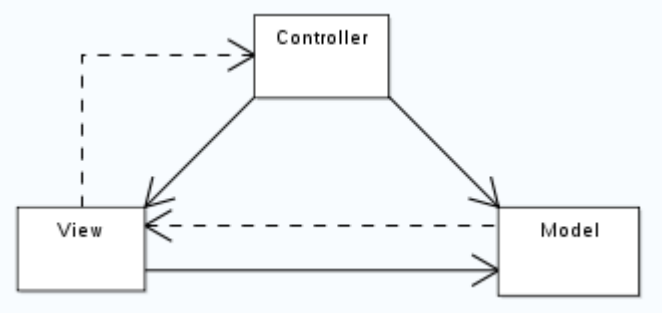

Figura 3.3 - Modelo da arquitetura MVC

Os componentes da arquitetura MVC estão descritos abaixo:

- **Model:** manipula as estruturas de dados existentes no sistema que podem ser, por exemplo, arquivos XML ou bancos de dados;
- **View:** é o componente responsável pela interface com o usuário, ou seja, é ele que apresenta as informações e dados necessários para a iteração entre o usuário e o sistema;
- **Controller:** tem por objetivo controlar o fluxo das atividades e a lógica do sistema. Funciona como um intermediário entre os componentes View e Model. O primeiro informa ao Controller o que foi solicitado pelo usuário e este, se necessário, busca os dados do componente View.

Possuindo esta base inicial de conhecimento sobre a arquitetura MVC, podemos explicar e descrever todos os módulos do projeto e como eles se aplicam nessa arquitetura.

3.1 – Decomposição

3.1.1 – Decomposição em módulos

O sistema está dividido em 6 módulos, segundo o modelo da figura abaixo:

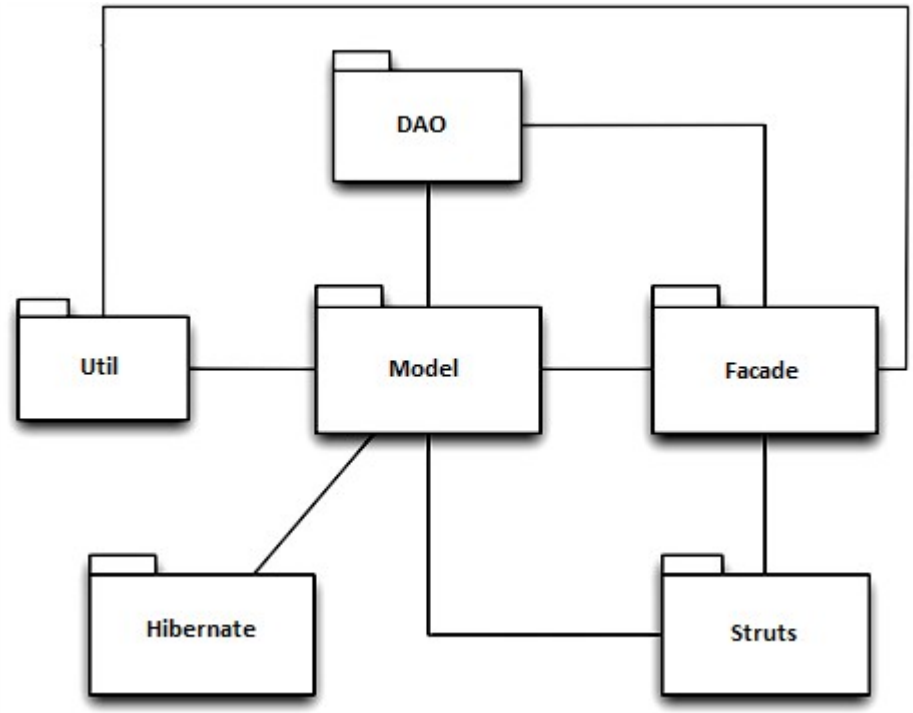

Figura 3.4 - Modelo dos módulos

Cada um desses módulos desempenha função fundamental e única no sistema e são explicados abaixo:

- **Model:** define as classes básicas do sistema, modelando as entidades que existem no banco de dados como cliente, produto, etc.;
- **Facade:** é responsável pela implementação dos casos de uso, como, por exemplo, procurar produto por nome e por tipo;

- **DAO:** é responsável pela manipulação das tabelas do banco de dados, fazendo inclusões, exclusões, alterações, consutas, etc.;
- **Struts:** trata das camadas View e Controller da arquitetura MVC, sendo responsável pela interface com o usuário e captura de dados, além de controlar o fluxo da lógica do sistema como um todo;
- **Hibernate:** este módulo é responsável pela persistência do banco de dados.
- **Util:** possui classes com ferramentas diversas necessárias para o sistema, como encriptador de senhas, variáveis globais, etc.

3.1.2 – Decomposição de Dados

O sistema é composto por 11 classes de dados, ilustradas pelo diagrama de classes (Figura 3.3) e descritas a seguir:

- **User:** superclasse que representa um usuário do sistema, que pode ser uma empresa, estabelecimento, administrador ou cliente;
- **Administrator:** classe que representa um admistrador. Herda da classe User;
- **Customer:** classe que representa um cliente. Herda da classe User;
- **Company:** classe que representa uma empresa. Herda da classe User;
- **Store:** classe que representa um estabelecimento. Herda da classe User;
- **Product:** classe que representa um produto. Possui atributos para guardar informações como nome, descrição, tipo de produto, etc.
- **Type:** classe que representa um tipo de produto. Possui atributos para guardar informações como nome, descrição e lista com os tipos de produtos que estão incluídos nele. Por exemplo, se o tipo de produto for Informática, essa lista de tipos poderá conter Monitores, Notebooks, Processadores, etc.;
- **Price:** classe associativa entre Product e Store. Um produto pode pertencer a vários estabelecimentos e possuir preços diferentes entre si.
- **PricePk:** classe que define os atributos correspondentes à chave primária composta de Price. A existência dessa classe se deve ao Hibernate, pois, para

fazer a persistência de dados em uma tabela com chave primária composta, é preciso criar uma nova classe com os atributos que fazem parte dela.

- **Rate:** classe associativa entre Customer e Product. Um produto pode receber avaliação de vários clientes e um cliente pode atribuir notas a mais de um produto.
- **RatePk:** classe que define os atributos correspondentes à chave primária composta de Rate. A existência dessa classe se deve ao Hibernate, pelo mesmo motivo explicado na classe PricePk.

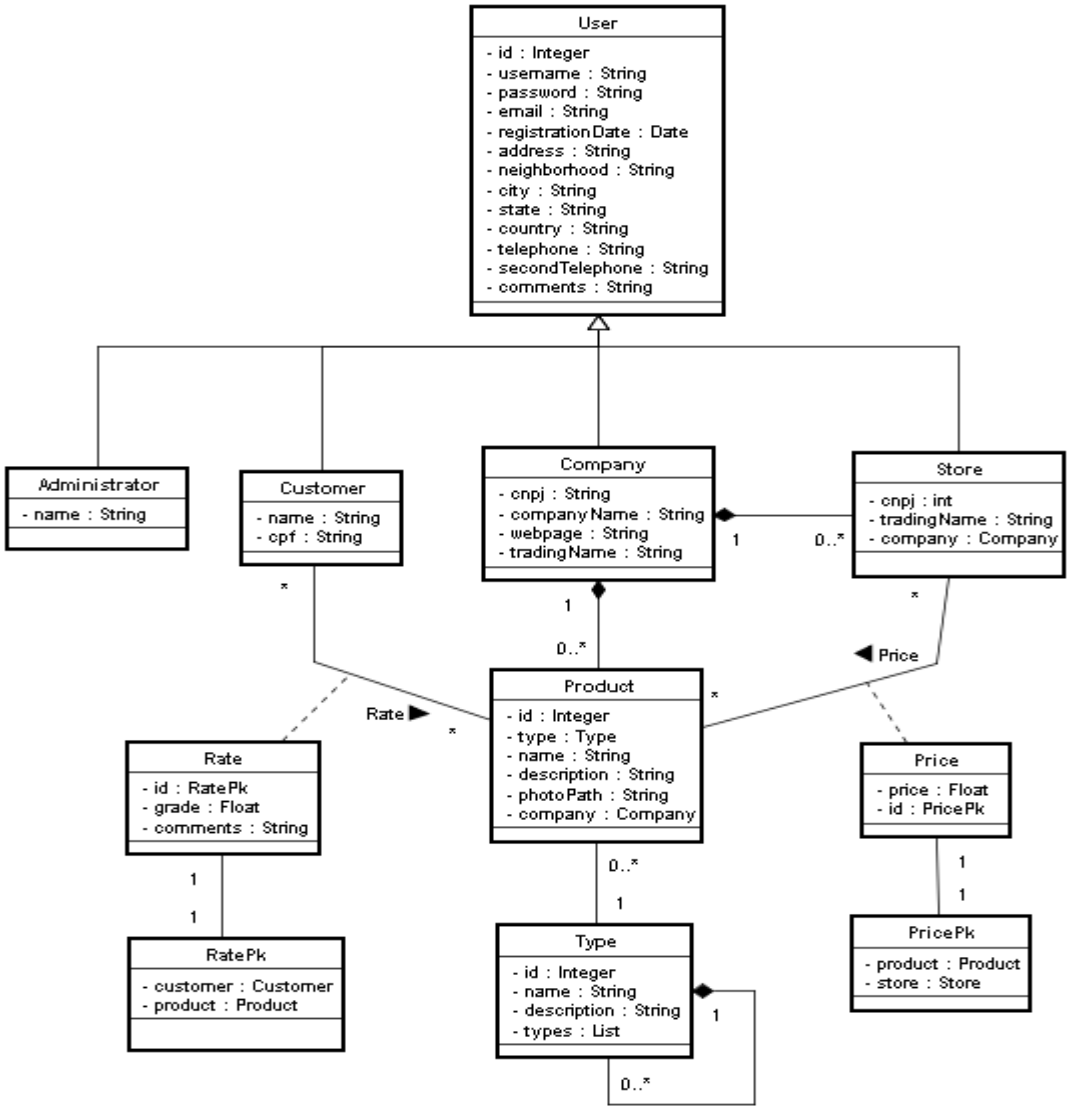

Figura 3.5 - Diagrama de Classes

3.2 – Descrição das Dependências

3.2.1 – Dependência entre Módulos

O módulo Model é utilizado por todos os módulos do sistema, pois é ele quem define as classes básicas do mesmo. O módulo Hibernate fornece suporte ao Model, fazendo a persistência do banco de dados.

Já o módulo DAO utiliza as classes de Model para fazer operações nas tabelas do banco de dados.

O módulo Facade, responsável pela implementação dos casos de uso, utiliza as classes dos módulos DAO e Util em seu código.

Finalmente, o módulo Struts utiliza as classes do módulo Facade para implementar os métodos chamados pelas páginas web.

Pode-se perceber que o sistema segue a estrutura MVC (Model-View-Controller). Os módulos Hibernate, Model e DAO fazem parte do componente Model; os módulos Util, Facade e Struts são integrantes do componente Controller e, por fim, as páginas web, que também utilizam ferramentas do Struts, fazem parte do componente View.

3.2.2 – Dependência entre Dados

O relacionamento entre os dados do sistema pode ser visto através do Diagrama Entidade Relacionamento (DER), ilustrado na Figura 3.4. A partir do diagrama, podemos listar as seguintes dependências:

- **Company, Administrator e Customer:** possuem uma coluna que é referenciada pelo identificador da tabela User;
- **Store:** além de ter uma coluna que é referenciada pelo identificador da tabela User, a tabela Company possui a coluna id_company, que é referenciada pelo identificador da tabela Company. Isso significa que cada estabelecimento pertence a uma empresa;

- **Product:** possui a coluna id_user, que é referenciada pelo identificador da tabela Company. Também possui a coluna id_type, que é referenciada pelo identificador da tabela Type;
- **Rate:** possui a coluna id_product, que é referenciada pelo identificador da tabela Product. Também possui a coluna id_user, que é referenciada pelo identificador da tabela Customer;
- **Price:** possui a coluna id_store, que é referenciada pelo identificador da tabela Store. Também possui a coluna id_product, que é referenciada pelo identificador da tabela Product;
- **Type_Association:** possui as colunas id_type_father e id_type_son, ambas referenciadas pelo identificador da tabela Type.

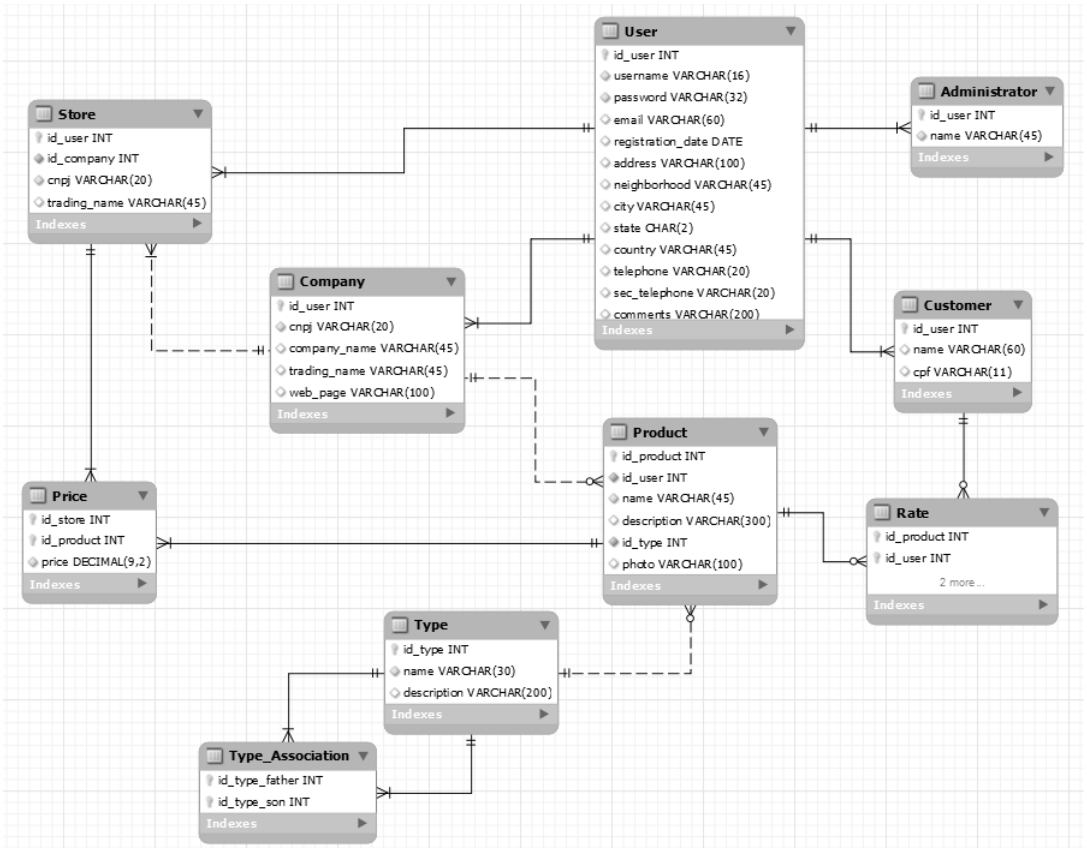

Figura 3.6 - Diagrama Entidade Relacionamento (DER)

3.3 – Projeto Detalhado dos Módulos

3.3.1 – Detalhamento do Módulo Model

A lógica das classes do modelo são bem simples. Só existem métodos getters e setters para manipular os seus atributos. Todos os getters possuem java Annotations para que o Hibernate relacione os atributos da classe com as colunas da tabela no banco de dados.

3.3.2 – Detalhamento do Módulo DAO

As classes contidas no módulo DAO tem a finalidade de manipular e fazer as operações no banco de dados. Cada cada tipo de usuário, que são os atores nos Diagramas de Casos de Uso (apresentados no item 3.4) possui uma classe DAO correspondente. Por exemplo, a classe StoreDAO possui métodos que manipulam dados relacionados aos estabelecimentos no banco de dados.

Como existem muitas classes e muitos métodos neste módulo, faremos uma abordagem bem genérica da lógica de um método das classes através do algoritmo a seguir:

```
operacao  (Objeto obj) {
        se (obj = vazio) {
                lançar exceção de argumento inválido;
        }
        tentar{
                abrir conexão com banco de dados;
                executar query;
        }
        pegar Exceção (exception) {
                lançar exceção de persistência (exception);
        }
        fechar conexão;
}
```

3.3.3 – Detalhamento do Módulo Facade

As classes contidas no módulo Facade utilizam os métodos do DAO para implementar os casos de uso do sistema. Estão dividas pelo tipo de usuário: Administrador, Empresa, Estabelecimento e Cliente.

Em geral, as funções implementadas neste módulo são simplesmente chamar os métodos do DAO e possuem o seguinte algoritmo:

```
metodo (Objeto obj) {
        se (obj = vazio) {
        lançar exceção de argumento inválido;
        }
        tentar {
                operacaoDAO(obj);
        }
        pegar Exceção de persistencia(exception) {
                lançar exceção de controle (exception);
        }
}
```

3.3.4 – Detalhamento do Módulo Struts

As classes do módulo Struts estão dividas em funcionalidades de cada tipo de usuário (Empresa, Estabelecimento, Administrador, Cliente).

Todas elas possuem mtos métodos getters e setters para guardar em seus atributos valores passados no browser e tamanhos permitidos dos campos.

Além dos getters e setters, cada classe possui métodos específicos para executar ações que foram especificadas pelos casos de uso. Esses métodos simplesmente pegam os campos passados pela página Web, preparam esses campos e passam para as classes do módulo Facade.

3.3.5 – Detalhamento do Módulo Hibernate

Não está no escopo do projeto saber a lógica utilizada pelo Hibernate. Somente utilizaremos suas ferramentas para fazer a persistência do banco de dados.

Como exemplo, em cada método get das classes do módulo Model, inclui-se uma anotação @Column (name = “nome”), fazendo a correlação do atributo nome da classe ao campo da tabela no banco de dados.

3.3.6 – Detalhamento do Módulo Util

O módulo Utilitários possui três classes: uma para encriptar senha, outra com constantes globais do sistema e a última somente para testes durante o desenvolvimento do projeto.

A classe de encriptar senha simplesmente chama um método de uma biblioteca q faz encriptação MD5 de String. A classe de constantes globais, como o próprio nome diz, são constantes utilizadas em todo o projeto. Todos os atributos são do tipo public static String ou public static Integer.

3.4 – Requisitos Funcionais

Neste item cada requisito funcional do sistema será detalhadamente especificado. Essa especificação será feita através de Diagrama de Casos de Uso, Especificação dos Casos de Uso, Diagrama de Atividades e Diagr

3.4.1 – Diagramas de Casos de Uso

Já que o projeto possui muitos casos de uso, eles foram divididos em quatro diagramas de casos de uso (Figuras 3.5 a 3.8) , cada um com atores diferentes.

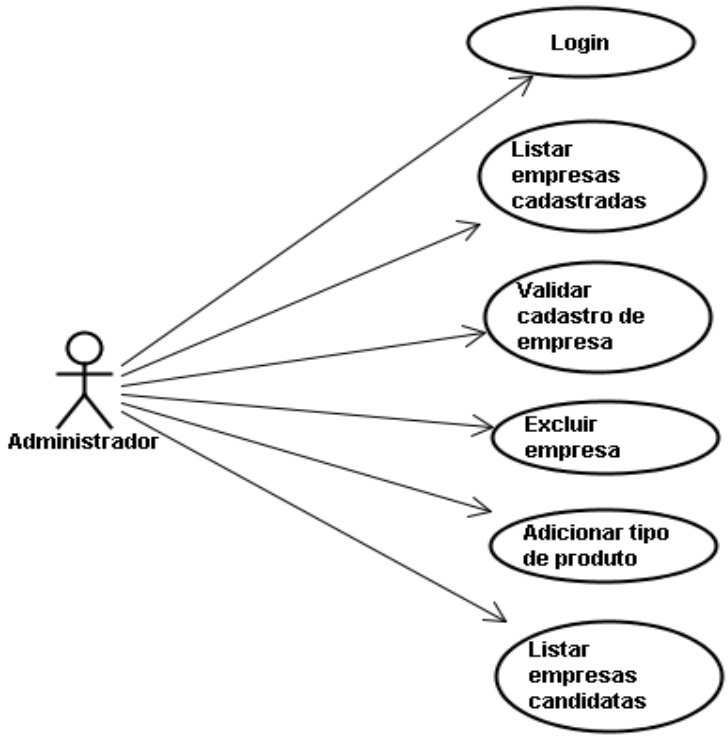

Figura 3.7 - Casos de uso do administrador

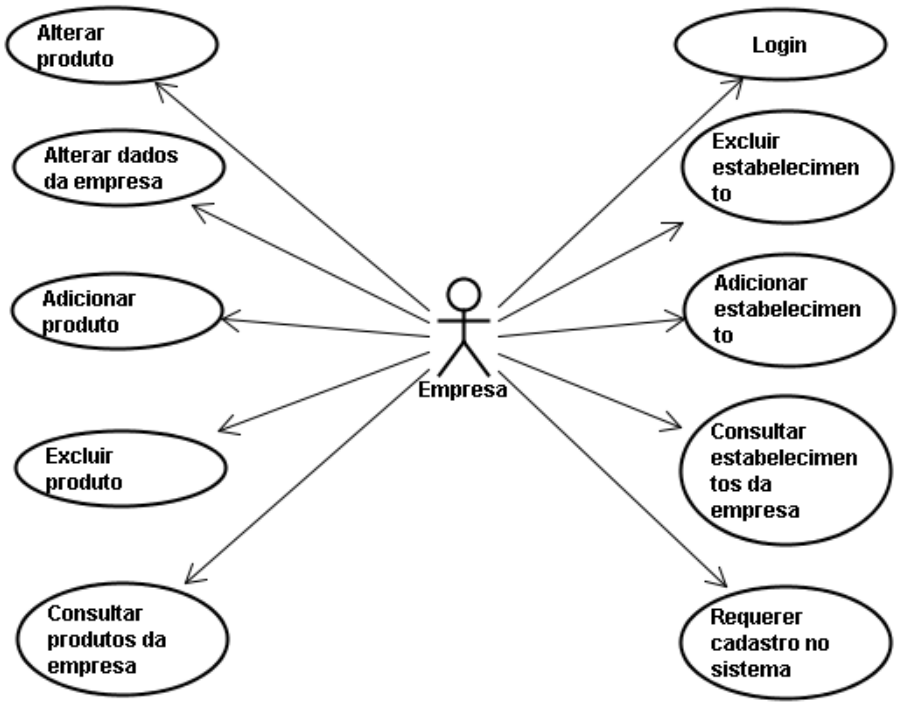

Figura 3.8 - Casos de uso da empresa

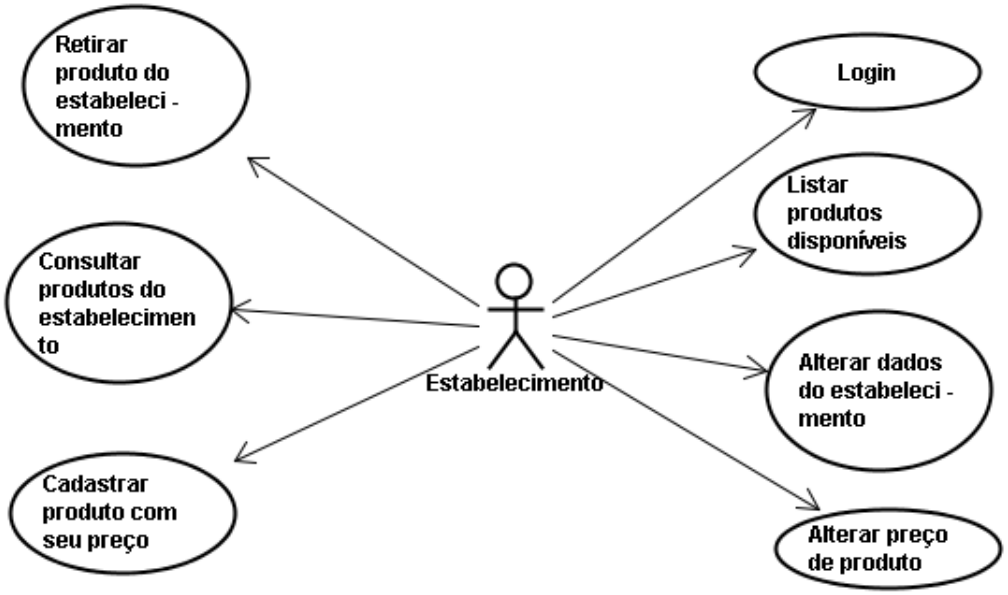

Figura 3.9 - Casos de uso do estabelecimento

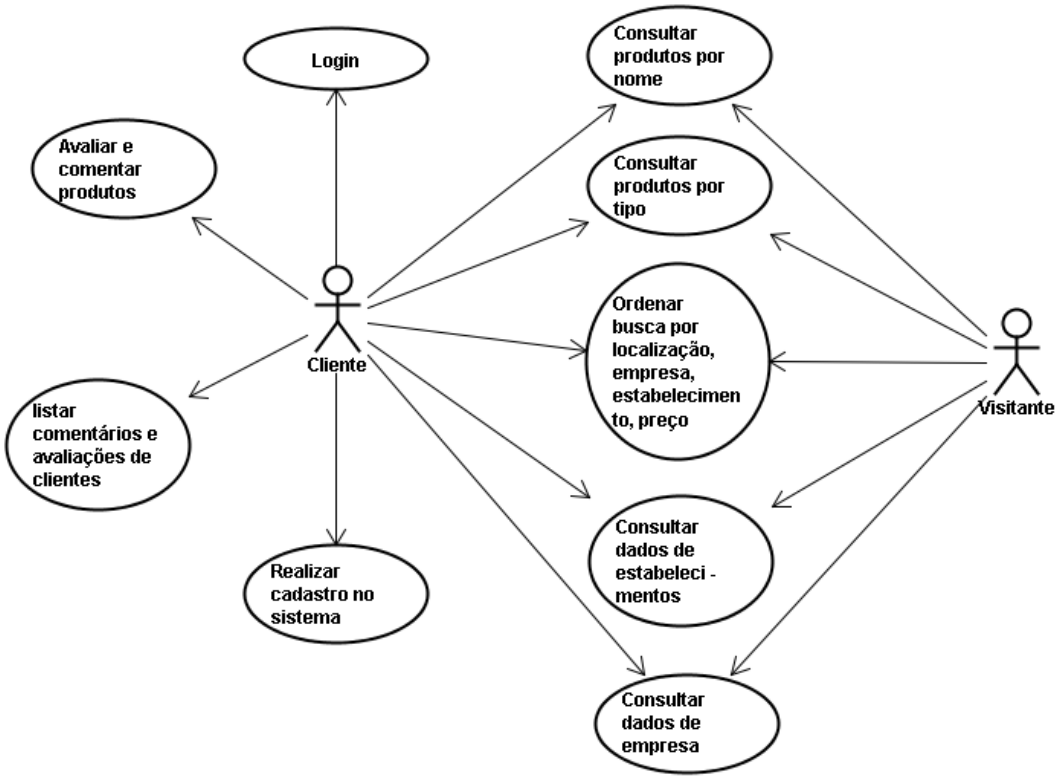

Figura 3.10 - Casos de uso de cliente e visitante

3.4.2 – Especificação dos Casos de Uso

Neste item será feita a especificação dos casos de uso de cada uma das funcionalidades apresentadas no diagrama de casos de uso:

- **Título:** Login.
  Objetivo: permitir que o administrador, empresa, estabelecimento ou cliente tenham acesso ao sistema.
  Pré-condições: nenhuma.
  Casos de uso associados: nenhum.
  Pós-condições: nenhuma.
  Prioridade: alta.
  Fluxo principal: (1) usuário insere username e senha; (2) sistema informa que o login foi efetuado com sucesso e permite acesso às funcionalidades correspondentes a seu cadastro.
  Fluxo alternativo: (1) sistema informa que o username ou senha estão incorretos.

- **Título:** Listar empresas candidatas.
  Objetivo: permitir que o administrador tenha acesso à uma lista de empresas em espera de aprovação para entrar no sistema.
  Pré-condições: administrador deve estar logado.
  Casos de uso associados: nenhum.
  Pós-condições: nenhuma.
  Prioridade: alta.
  Fluxo principal: (1) clicando na opção de listar empresas candidatas, o usuário é levado à página onde essa lista é mostrada.
  Fluxo alternativo: nenhum.

- **Título:** Validar cadastro de empresa.

  Objetivo: permitir que o administrador consulte dados da empresa que se candidatou a entrar no sistema e valide seu cadastro.

  Pré-condições: administrador deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) sistema mostra todos os dados cadastrais da empresa; (2) sistema informa que a empresa foi incluída ou recusada com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao incluir ou recusar a empresa.

- **Título:** Listar empresas cadastradas.

  Objetivo: permitir que o administrador tenha acesso a uma lista contendo todas as empresas já registradas no sistema.

  Pré-condições: administrador deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: baixa.

  Fluxo principal: (1) administrador clica no item para listar as empresas cadastradas; (2) sistema leva o usuário para a página onde as empresas cadastradas estão listadas.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao listar empresas cadastradas.

- **Título:** Excluir empresa.

  Objetivo: permitir que o administrador excluir uma empresa cadastrada.

  Pré-condições: administrador deve estar logado.

  Casos de uso associados: Listar empresas cadastradas

  Pós-condições: nenhuma.

  Prioridade: média.

Fluxo principal: (1) administrador clica no link para excluir ao lado do nome fantasia da empresa; (2) sistema informa que empresa foi removida com sucesso.

Fluxo alternativo: (1) sistema informa que ocorreu um erro ao excluir empresa.

- **Título:** Adicionar tipo de produto.

  Objetivo: permitir que o administrador incluir tipos de produtos para que as empresas possam classificar seus produtos.

  Pré-condições: administrador deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) administrador seleciona onde o novo tipo se localiza dentro da estrutura de tipos e digita nome e descrição do tipo; (2) sistema informa que o tipo de produto foi adicionado com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao adicionar tipo de produto.

- **Título:** Requerer cadastro no sistema.

  Objetivo: permite que empresas se cadastrem no sistema.

  Pré-condições: nenhuma.

  Casos de uso associados: Validar cadastro de empresa.

  Pós-condições: confirmação de cadastro pelo administrador.

  Prioridade: alta.

  Fluxo principal: (1) empresa insere todos os dados pedidos na página; (2) sistema informa que o cadastro foi realizado com sucesso e dirá para o mesmo aguardar a confirmação de sua inclusão no sistema pelo administrador.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao requerer cadastro no sistema.

- **Título:** Alterar dados da empresa.
  Objetivo: permite que a empresa mude alguns de seus dados.
  Pré-condições: empresa deve estar logada.
  Casos de uso associados: nenhum.
  Pós-condições: nenhuma.
  Prioridade: baixa.
  Fluxo principal: (1) empresa insere alterações no formulário; (2) sistema informa que a alteração dos dados da empresa foi feita com sucesso.
  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao alterar dados da empresa.

- **Título:** Adicionar produto.
  Objetivo: permite que a empresa cadastre um produto.
  Pré-condições: empresa deve estar logada.
  Casos de uso associados: nenhum.
  Pós-condições: nenhuma.
  Prioridade: alta.
  Fluxo principal: (1) empresa insere os dados indicados no formulário; (2) sistema informa que o produto foi adicionado com sucesso.
  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao adicionar produto.

- **Título:** Alterar produto.
  Objetivo: permite que a empresa mude dados de um produto.
  Pré-condições: empresa deve estar logada.
  Casos de uso associados: nenhum.
  Pós-condições: nenhuma.
  Prioridade: média.

Fluxo principal: (1) empresa insere as alterações no formulário; (2) sistema informa que o produto foi alterado com sucesso.

Fluxo alternativo: (1) sistema informa que ocorreu um erro ao alterar dados do produto.

- **Título:** Excluir produto.

  Objetivo: permite que a empresa retire um produto de sua lista.

  Pré-condições: empresa deve estar logada.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: média.

  Fluxo principal: (1) empresa escolhe uma empresa a ser excluída; (2) sistema informa que o produto foi excluído com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao excluir o produto.

- **Título:** Consultar produtos da empresa.

  Objetivo: permite que a empresa consulte todos os produtos cadastrado por ela.

  Pré-condições: empresa deve estar logada.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: média.

  Fluxo principal: (1) empresa seleciona a opção de listar produtos cadastrados; (2) sistema exibe a lista de produtos.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao listar produtos cadastrados pela empresa.

- **Título:** Consultar estabelecimentos da empresa.

  Objetivo: permite que a empresa consulte todos os seus estabelecimentos cadastrados.

Pré-condições: empresa deve estar logada.

Casos de uso associados: nenhum.

Pós-condições: nenhuma.

Prioridade: baixa.

Fluxo principal: (1) empresa seleciona a opção de listar estabelecimentos cadastrados; (2) sistema exibe a lista de estabelecimentos.

Fluxo alternativo: (1) sistema informa que ocorreu um erro ao listar estabelecimentos da empresa.

- **Título:** Adicionar estabelecimento.

  Objetivo: permite que a empresa adicione um estabelecimento.

  Pré-condições: empresa deve estar logada.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: média.

  Fluxo principal: (1) empresa insere os dados indicados no formulário; (2) sistema informa que o estabelecimento foi adicionado com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao adicionar estabelecimento.

- **Título:** Excluir estabelecimento.

  Objetivo: permite que a empresa remova um estabelecimento de sua lista de estabelecimentos.

  Pré-condições: empresa deve estar logada.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: baixa.

  Fluxo principal: (1) empresa escolhe empresa a ser excluída; (2) sistema informa que o estabelecimento foi excluído com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao excluir estabelecimento.

- **Título:** Alterar dados do estabelecimento.

  Objetivo: permite que o estabelecimento mude alguns de seus dados.

  Pré-condições: estabelecimento deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: baixa.

  Fluxo principal: (1) estabelecimento insere alterações no formulário; (2) sistema informa que a alteração dos dados do estabelecimento foi feita com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao alterar dados do estabelecimento.

- **Título:** Consultar produtos do estabelecimento.

  Objetivo: permite que o estabelecimento consulte todos os seus produtos cadastrados.

  Pré-condições: estabelecimento deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: média.

  Fluxo principal: (1) estabelecimento seleciona a opção de listar produtos cadastrados; (2) sistema mostra uma lista com todos os produtos cadastrados do estabelecimento.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao consultar produtos do estabelecimento.

- **Título:** Alterar preço de produto.

  Objetivo: permite que o estabelecimento altere o preço de seu produto.

  Pré-condições: estabelecimento deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

Fluxo principal: (1) estabelecimento seleciona o produto desejado; (2) sistema mostra todos os dados sobre o produto e permite que o estabelecimento altere seu preço; (3) sistema informa que o preço do produto foi alterado com sucesso.

Fluxo alternativo: (1) sistema informa que ocorreu um erro ao alterar preço do produto.

- **Título:** Retirar produto do estabelecimento.

  Objetivo: permite que o estabelecimento excluia o produto de seu estabelecimento.

  Pré-condições: estabelecimento deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: média.

  Fluxo principal: (1) estabelecimento seleciona o produto desejado; (2) sistema informa que o produto foi removido com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao remover o produto.

- **Título:** Listar produtos disponíveis.

  Objetivo: permite que o estabelecimento consulte todos os produtos de sua empresa que ainda não estão cadastradas pelo estabelecimento.

  Pré-condições: estabelecimento deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: média.

  Fluxo principal: (1) estabelecimento seleciona a opção de listar produtos disponíveis; (2) sistema mostra uma lista com todos os produtos disponíveis para o estabelecimento cadastrar.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao consultar produtos disponíveis.

- **Título:** Cadastrar produto com seu preço.

  Objetivo: permite que o estabelecimento adicione um produto incluindo seu preço.

  Pré-condições: estabelecimento deve estar logada.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) estabelecimento seleciona um produto a ser adicionado; (2) estabelecimento define um preço para o produto; (3) sistema informa que produto foi incluido no estabelecimento com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao adicionar produto no estabelecimento.

- **Título:** Consultar produtos por nome.

  Objetivo: permite que um cliente ou visitante consulte produtos pelo seu nome ou parte dele.

  Pré-condições: nenhuma.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) cliente ou visitante digita um nome ou parte dele para fazer a consulta; (2) sistema retorna uma lista de produtos com o nome digitado pelo usuário.

  Fluxo alternativo: nenhum.

- **Título:** Consultar produtos por tipo.

  Objetivo: permite que um cliente ou visitante consulte produtos pelo seu tipo.

  Pré-condições: nenhuma.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

Prioridade: alta.

Fluxo principal: (1) cliente ou visitante escolhe um tipo de produto; (2) sistema retorna uma lista de produtos com o tipo desejado.

Fluxo alternativo: nenhum.

- **Título:** Ordenar busca por localização, empresa, estabelecimento, preço.

  Objetivo: permite que um cliente ou visitante ordene o resultado da pesquisa por localização, empresa, estabelecimento ou preço.

  Pré-condições: nenhuma.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) cliente ou visitante escolhe um tipo de ordenação; (2) sistema retorna uma lista de produtos ordenados pelo que foi pedido.

  Fluxo alternativo: nenhum.

- **Título:** Consultar dados de estabelecimento.

  Objetivo: permite que um cliente ou visitante consulte dados sobre o estabelecimento.

  Pré-condições: nenhuma.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) cliente ou visitante seleciona o estabelecimento desejado; (2) sistema mostra os dados sobre o estabelecimento.

  Fluxo alternativo: nenhum.

- **Título:** Consultar dados de empresa.

  Objetivo: permite que um cliente ou visitante consulte dados sobre a empresa.

  Pré-condições: nenhuma.

Casos de uso associados: nenhum.

Pós-condições: nenhuma.

Prioridade: alta.

Fluxo principal: (1) cliente ou visitante seleciona a empresa desejada; (2) sistema mostra os dados sobre a empresa.

Fluxo alternativo: nenhum

- **Título:** Realizar cadastro no sistema.

  Objetivo: permite que um visitante se torne cliente através de um cadastro.

  Pré-condições: nenhuma.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) visitante informa todos os dados pedidos; (2) sistema informa que o cadastro foi realizado com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao cadastrar cliente.

- **Título:** Listar comentários e avaliações de clientes.

  Objetivo: permite que um cliente veja todas as avaliações e notas dos outros clientes sobre um determinado produto.

  Pré-condições: cliente deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) cliente seleciona o produto desejado; (2) sistema lista todas as avaliações e notas sobre o produto.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao listar avaliações.

- **Título:** Avaliar e comentar produto.

  Objetivo: permite que um cliente avalie e dê nota para o produto.

  Pré-condições: cliente deve estar logado.

  Casos de uso associados: nenhum.

  Pós-condições: nenhuma.

  Prioridade: alta.

  Fluxo principal: (1) cliente seleciona o produto desejado; (2) cliente faz comentários e atribui uma nota ao produto; (3) sistema informa que a avaliação foi inserida com sucesso.

  Fluxo alternativo: (1) sistema informa que ocorreu um erro ao inserir avaliação.

3.4.3 – Diagrama de Atividades

O Diagrama de Atividades do sistema é ilustrado pela Figura 3.9. Podemos observar que cada tipo de usuário (empresa, estabelecimento e administrador) possui um menu diferenciado, contendo as ações permitidas para cada um deles.

Todos os usuários, sendo cadastrados ou não, podem fazer consulta de produtos. Porém, somente um cliente cadastrado poderá avaliar os produtos. Essa necessidade de ser cadastrado para poder votar acontece porque, caso contrário, uma pessoa poderia votar diversas vezes no mesmo produto, contribuindo com uma distorção na média das notas.

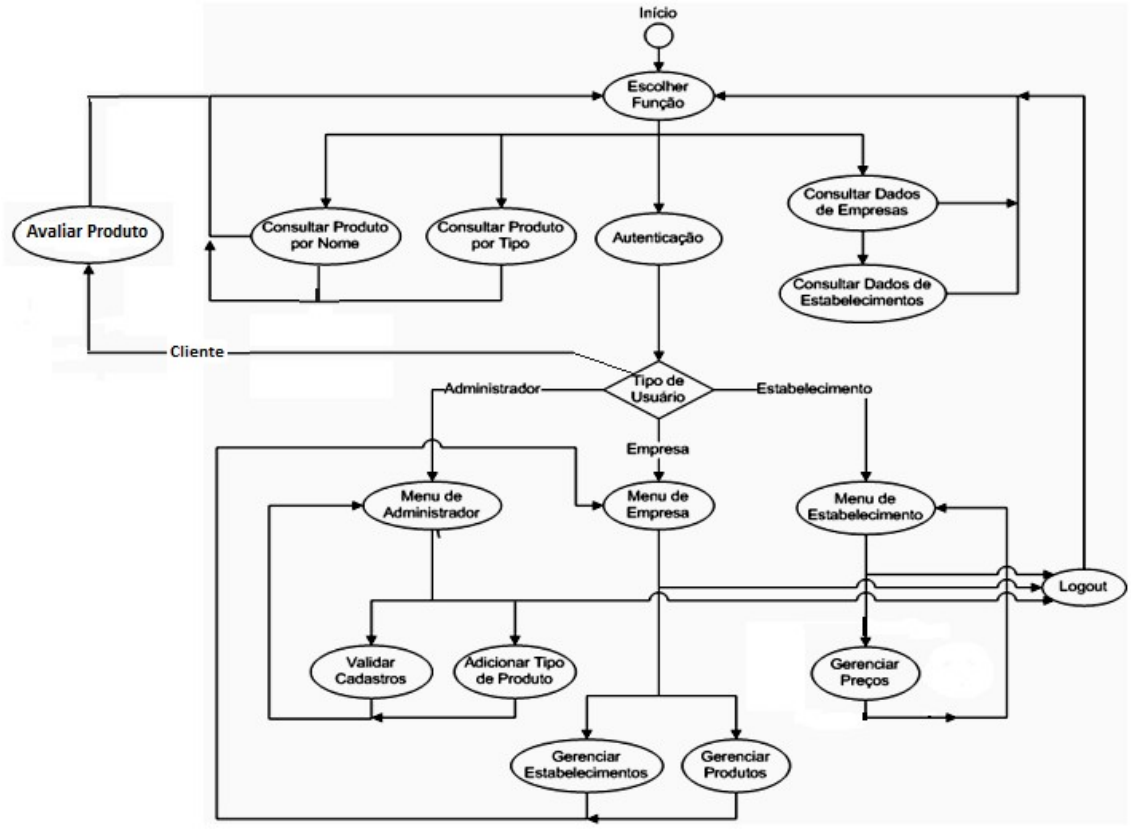

Figura 3.11 - Diagrama de Atividades

3.4.4 – Diagrama de Sequência

O diagrama de sequência representa a sequência de processos, ou seja, a sequência de mensagens passadas entre os objetos num software. O diagrama dá ênfase à ordenação temporal em que as mensagens são trocadas entre esses objetos.

Sendo assim, o diagrama de sequência faz com que os casos de uso sejam entendidos de forma aprofundada e completa. Como o projeto possui muitos casos de uso, não seria viável fazer um diagrama de sequência para cada um deles. Mas, como todos eles são parecidos na forma em que os processos são chamados, podemos utilizar um caso de uso como exemplo. Pela importância e relevância, escolheu-se fazer o diagrama com a busca de produtos pelo nome e é mostrado na Figura 3.10.

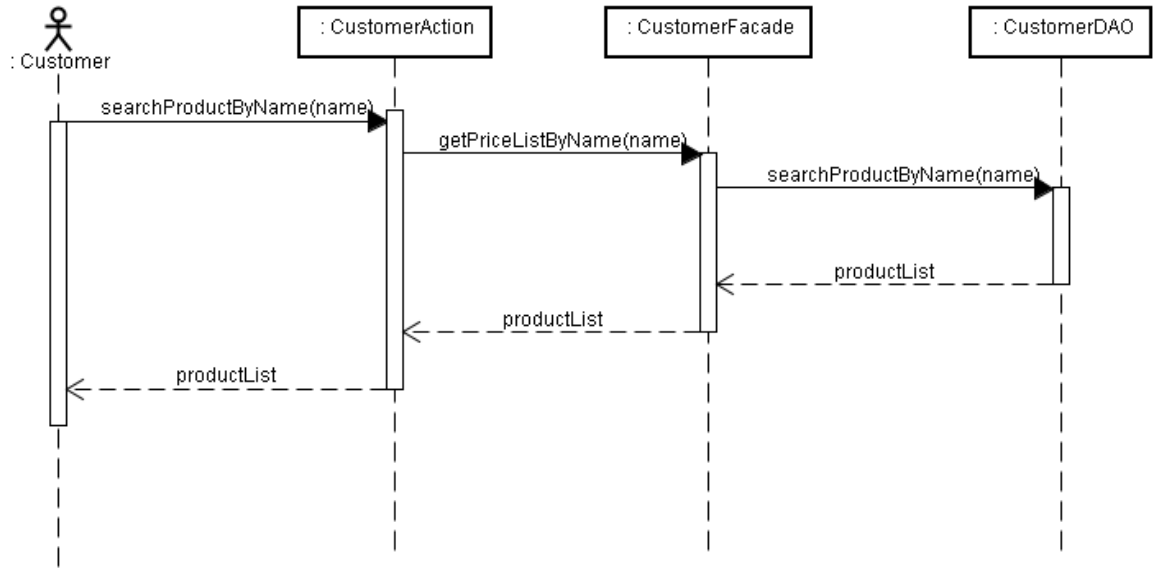

Figura 3.12 - Diagrama de Sequência da busca de produtos por nome

O diagrama de sequência para este caso de uso mostra que o cliente, ao solicitar a busca de produtos por nome, faz com que o sistema crie um objeto CustomerAction. Esse objeto contém um método chamado searchProductByName, que instancia um outro objeto, o CustomerFacade. Esse último possui um método chamado getPriceListByName, que instancia um outro objeto chamado CustomerDAO. Finalmente, este objeto possui um método chamado searchProductByName, que faz a busca e retorna a lista de produtos. Essa lista é repassada para os outros objetos até chegar ao Cliente.

Capítulo 4

Análise de Resultados

Para fazer a análise de resultados do projeto, o sistema foi testado por algumas pessoas, escolhidas aleatoriamente, para tentar descobrir possíveis erros de lógica ou bugs. Durante o desenvolvimento do projeto, algumas falhas foram detectadas e corrigidas sempre com sucesso. Os erros mais comuns encontrados enquanto o sistema estava em desenvolvimento eram de validação de campos e direcionamento de respostas para páginas erradas. Após todo esse processo de testes não pode-se confirmar a inexistência de erros de lógica ou bugs, porém, o sistema se mostrou estável e com respostas dentro do esperado.

Todos os casos de uso propostos foram implementados e funcionam de acordo com o especificado. O desempenho dos casos de uso, em função de gasto de memória, processamento e o tempo de execução, não foi medido por estar fora do escopo definido no projeto. O único ponto que vale a pena ser ressaltado, em função do desempenho dos casos de uso, é o modo em que o sistema está rodando: debug ou normal. No modo debug o sistema fica em torno de 60% mais lento, segundo medições feitas no próprio computador pessoal (Dell Inspiron 1525, processador Core 2 Duo, 3MB de Cache, 2 GB de memória RAM e 120 GB de HD). Esse percentual pode variar em máquinas diferentes.

Outra questão importante a ser mencionada sobre o sistema é a possibilidade de balanceamento de carga. O sistema foi inicialmente projetado para uma quantidade pequena e média de usuários, algo em torno de 20 a 30 mil acessando o site diariamente. Sendo assim, um único servidor com um processador de 2 a 4 núcleos já seria suficiente para realizar as requisições solicitadas por todos os usuários. Caso a quantidade de usuários aumente, será necessário um estudo sobre a quantidade de servidores e como eles estariam ligados na rede, bem como implementar o controle de fluxo e repartição das requisições.

Uma característica que faz com que informações erradas ou desnecessárias não sejam enviadas e processadas pelo servidor, diminuindo o consumo de processamento, é a validação de campos por Javascript. Antes de um formulário ser enviado, funções em

Javascript são executadas para verificar se todos os campos obrigatórios foram preenchidos e se não há nenhum erro de informação dentro deles.

A interface com o usuário é simples, sem muitos recursos. Esse fato foi relatado pela maioria das pessoas que testou o sistema. Porém, há grande facilidade para se implementar páginas mais modernas a partir das que já existem, pois, como já foi dito, o sistema foi projetado segundo o padrão de arquitetura MVC, que torna o software mais limpo e fácil de se fazer manter. Este trabalho de melhorar o visual das páginas pode ser feito por um web designer.

Também pode-se afirmar que o sistema é flexível em relação aos tipos de produtos anunciados. O administrador possui o direito de estruturar os tipos de acordo com sua vontade, fazendo do sistema um site de busca de produtos genérico.

Sendo assim, podemos concluir que o projeto obteve os resultados esperados, dando destaque à facilidade de se obter informações sobre os produtos e as opiniões dos usuários sobre os primeiros e, também, da praticidade de comparar preços e consultar dados sobre as empresas e estabelecimentos que estão vendendo esses produtos.

Capítulo 5

Conclusão

5.1 – Avaliação

Ao longo do desenvolvimento do projeto pudemos aprender e aprofundar conhecimentos sobre ferramentas importantes que auxiliam o desevolvedor na hora de implementar sistemas Web. Este fato contribuiu para que os objetivos do projeto fossem atingidos:

- **O sistema pretende facilitar a procura e anúncios de produtos via Web, por ser um meio bastante difundido e prático:** o sistema realmente foi desenvolvido para ser um sistema Web e pode ser utilizado por qualquer pessoa que tenha acesso à internet.

- **Com esse sistema pretende-se criar um método sistemático, rápido e eficiente para anúncio e procura de produtos, facilitando tanto o anúncio, por atingir um maior número de clientes com um menor custo e maior controle, quanto a procura, por fornecer ao cliente facilidade, comodidade e eficiência na sua procura:** esse objetivo também foi atingido. Conseguimos implementar o sistema com ferramentas que facilitaram o desenvolvimento do projeto de forma estruturada e prática de se manter. Também pudemos perceber a praticidade para os usuários finais pois, em poucas etapas, os serviços pedidos são efetuados.

5.2 – Adições Futuras

Sugestões de melhorias e adições futuras são descritas a seguir:

- **Melhorar a interface gráfica das páginas Web:** uma das coisas que chama mais atenção em um sistema é sua interface gráfica. O projeto atual não deu atenção especial neste ponto, fazendo com que as páginas não tenham um apelo visual interessante.
- **Permitir que uma empresa ou estabelecimento tenha mais de uma conta disponível:** o sistema atualmente só permite uma conta por empresa e por estabelecimento. Como, em cada um deles, pode existir mais de uma pessoa encarregada de administrar os produtos e preços, o ideal seria criar vários logins para cada entidade.
- **Permitir ao administrador excluir comentários de clientes:** caso algum cliente escreva comentários ofensivos, seria razoável o administrador poder retirar esses tipos de comentários.
- **Permitir ao cliente consultar produtos por diversos parâmetros:** o presente sistema permite ao cliente consultar produtos por nome e tipo e ordenar o resultado por preço, localização, nome da empresa ou estabelecimento. Pode-se adicionar mais filtros a essa pesquisa, como faixa de preços, média das notas dos consumidores para determinados produtos, etc.
- **Implementar funcionalidades focadas no estabelecimento:** uma sugestão de funcionalidade seria emitir estatística sobre os produtos do estabelecimento, fornecendo a quantidade de produtos em estoque, histórico de vendas e outras informações relevantes.
- **Integrar serviços deste sistema com outros sistemas:** essa integração é baseada na arquitetura orientada a serviços, na qual uma rotina é disponibilizada como serviço em uma rede e todos os sistemas compatíveis podem compartilhar estes serviços. Um meio de se implementar a arquitetura

é através de web services, uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes.

Bibliografia

[1] *Biblioteca Hibernate Red Hat Inc.* https://www.hibernate.org/ (Primeiro acesso em Agosto de 2009).

[2] *MySQL, The world's most popular open source database.* http://www.mysql.com/ (Primeiro acesso em Agosto de 2009).

[3] *Apache Struts, The Apache Software Foundation.* http://struts.apache.org/ (Primeiro acesso em Agosto de 2009).

[4] *Java, Developer Resources for Java Technology.* http://java.sun.com/ (Primeiro acesso em Agosto de 2009).

[5] *Compilador Java Netbeans, Sun Microsystems Inc.* http://netbeans.org/ (Primeiro acesso em Agosto de 2009).

[6] *Tomcat, The Apache Software Foundation.* http://tomcat.apache.org/ (Primeiro acesso em Agosto de 2009).

[7] *JSP (Java Server Pages), Sun Microsystems Inc.* http://java.sun.com/products/jsp (Primeiro acesso em Agosto de 2009).

[8] *Wikipedia - MySQL.* http://en.wikipedia.org/wiki/MySQL (Primeiro acesso em Agosto de 2009).

[9] *Wikipedia – Apache Struts.* http://en.wikipedia.org/wiki/Apache_Struts (Primeiro acesso em Outubro de 2009).

[10] *Wikipedia – Caso de uso.* http://pt.wikipedia.org/wiki/Caso_de_uso (Primeiro acesso em Novembro de 2009).

[11] *Wikipedia – Diagrama de Sequência.* http://pt.wikipedia.org/wiki/Diagrama_de_sequ%C3%AAncia (Primeiro acesso em Novembro de 2009).

[12] *Wikipedia – Motor de busca.* http://pt.wikipedia.org/wiki/Motor_de_busca (Primeiro acesso em Dezembro de 2009).

[13] *Comitê Gestor da Internet no Brasil – Pesquisa Sobre o Uso das Tecnologias da Informação e da Comunicação no Brasil 2008*. http://www.cetic.br/tic/2008/index.htm (Primeiro acesso em Dezembro de 2009).

[14] *Universidade Federal de Pernambuco – Comparativo de Desempenho entre Bancos de Dados de Código Aberto*, disponível no site http://www.upf.br/erbd/download/15997.pdf (Primeiro acesso em Dezembro de 2009).