Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

FÁBIO AUGUSTO VICENTE

DESENVOLVIMENTO DE APLICATIVOS WEB PARA GERENCIAMENTO DE SERVIÇOS

Assis
2010

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

DESENVOLVIMENTO DE APLICATIVOS WEB PARA GERENCIAMENTO DE SERVIÇOS

FÁBIO AUGUSTO VICENTE

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de Curso de Bacharelado em Ciência da Computação, analisado pela seguinte comissão examinadora:

Orientadora: Profa. Dra. Marisa Atsuko Nitto

Analisador : Prof. Dr. Almir Rogério Camolesi

Assis
2010

FICHA CATALOGRÁFICA

VICENTE, Fábio Augusto
Desenvolvimento de aplicativo *Web* para gerenciamento de serviços / Fábio Augusto Vicente. Fundação Educacional do Município de Assis – FEMA – Assis, 2010.
45p.
Orientadora: Profa. Dra. Marisa Atsuko Nitto.
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Java. 2. Gerenciamento de serviços. 3. Web.

CDD:001.6
Biblioteca da FEMA.

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

FÁBIO AUGUSTO VICENTE

DESENVOLVIMENTO DE APLICATIVOS WEB PARA GERENCIAMENTO DE SERVIÇOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de Curso de Bacharelado em Ciência da Computação.

Orientadora: Profa. Dra. Marisa Atsuko Nitto

Área de Concentração: Informática

Assis
2010

DEDICATÓRIA

Dedico este trabalho á minha mãe Maria de Fátima da Silva Vicente, a meu pai Valdecy Vicente e aos meus irmãos, que sempre estiveram comigo e me motivaram a enfrentar as dificuldades encontradas no decorrer do curso.

AGRADECIMENTOS

Agradeço primeiro a Deus pela força e proteção que me acompanharam durante todo o meu curso e nas constantes viagens;

Á Professora Doutora Marisa Atsuko Nitto pela orientação deste trabalho e que sempre transmitiu estímulos sábios e objetivos;

Ao grupo de professores do curso de Bacharelado em Ciência da Computação, pela dedicação demonstrada em todas as aulas e pelo companheirismo durante esses anos;

A Fundação Educacional do Município de Assis por ter possibilitado a realização deste trabalho;

Aos amigos da faculdade, amigos de infância, colegas de ônibus, parentes, a minha namorada Daniela, que direta e indiretamente me ajudaram nesta caminhada, que foram longas, mas que será fundamental para meu futuro profissional e pessoal.

“É justamente a possibilidade de realizar um sonho que torna a vida interessante”.

Paulo Coelho (O Alquimista)

RESUMO

Uma necessidade atual das organizações é a disponibilização do maior período possível dos serviços da área de Tecnologia da Informação e a possibilidade de gerência de alguns destes serviços de forma imediata, independente e de fácil gestão podendo ser executada por um colaborador da própria empresa permitindo rapidez na administração dos serviços disponibilizados pelos servidores da rede da organização. Este trabalho apresenta a especificação e implementação de um aplicativo Web para o controle e gerenciamento de pedidos de compra utilizando os conceitos da tecnologia Java Server Faces e Java. A aplicação web é um conjunto de programas que utiliza a web, através de um navegador (Internet Explorer, Firefox, Chrome, entre outros), como ambiente de execução. Em outras palavras é um software que pode ser acessado de qualquer micro que tenha acesso a Internet.

A demonstração de um caso de uso, as tecnologias, os aspectos técnicos, os benefícios, sua arquitetura e a segurança utilizada nesta solução formam a base deste trabalho.

Palavras-chave: Web; Gerenciamento de serviço; Java.

ABSTRACT

A current need of the organizations is to provide the longest possible service in the area of Information Technology and the ability to run some of these services in a prompt, independent and easy way to manage. It can be performed by an employee of the company itself in allowing rapid administration of service by an organization's network servers. This work presents the specification and implementation of a Web application for tracking and managing purchase orders using the concepts of Java and Java Server Faces technology. The web application is a set of programs that use the web through a browser (Internet Explorer, Firefox, Chrome, etc.), as the execution environment. In other words this software can be accessed from any computer that has Internet access.

The demonstration of a use case, the technologies, the technical aspects, its benefits, and the security architecture used in this solution form the basis for this one.

Keywords: Web; Service Management; Java.

LISTA DE ILUSTRAÇÕES

Figura 1 – Código Monolítico.....21
Figura 2 – Código em Duas Camadas 22
Figura 3 – Código em Três Camadas 23
Figura 4 – Interação entre os Componentes do Padrão MVC.....24
Figura 5 – Modelagem do Projeto 27
Figura 6 – Caso de Uso Funcionário.....28
Figura 7 – Caso de Uso Cotação 29
Figura 8 – Caso de Uso Gerente.....29
Figura 9 – Diagrama de Classes.....30
Figura 10 – Diagrama de Sequência Funcionário 31
Figura 11 – Diagrama de Sequência Cotação.....32
Figura 12 – Diagrama de Sequência Gerente.....32
Figura 13 – Camada do Modelo no Projeto.....33
Figura 14 – Camada da Visão no Projeto 34
Figura 15 – Camada do Controlador no Projeto.....34
Figura 16 – Interface Login.....35
Figura 17 – Interface Inicial Funcionário 36
Figura 18 – Interface Cadastro de Pedido.....37
Figura 19 – Interface Inicial Usuário com Pedidos Realizados.....38
Figura 20 – Interface Inicial Cotação.....38
Figura 21 – Interface Cotar Pedido 39
Figura 22 – Interface Inicial Gerente 39
Figura 23 – Interface Aprovar Pedido.....40
Figura 24 – Interface Gerência.....41
Figura 25 – Interface Cadastro de Usuário 41
Figura 26 – Interface Cadastro de Departamento 42

SUMÁRIO

1 - INTRODUÇÃO 13

1.1 - OBJETIVO 14

1.2 - MOTIVAÇÃO 14

1.3 - JUSTIFICATIVA 14

2 - FUNDAMENTAÇÃO TEÓRICA BÁSICA 15

2.1 - LINGUAGEM DE PROGRAMAÇÃO JAVA 15

2.2 - ECLIPSE 16

2.3 - JAVA SERVER PAGES (JSP) 16

2.4 - JAVA SERVER FACES (JSF) 17

2.5 - HIBERNATE 18

2.6 - TOMCAT 19

2.7 - APACHE MYFACES 19

2.8 - BANCO DE DADOS POSTGRESQL 20

3 – PROGRAMAÇÃO EM CAMADAS 21

3.1 - PADRÃO MODELO VISUALIZAÇÃO CONTROLE (MVC) 23

3.1.1 - Modelo (Model) 24

3.1.2 – Visão (View) 25

3.1.3 – Controlador (Controller) 26

4 - DESENVOLVIMENTO DO APLICATIVO 26

4.1 - DEFINIÇÃO DO PROBLEMA 26

4.2 - MODELAGEM DO PROBLEMA 27

4.3 - ESPECIFICAÇÃO DO APLICATIVO 28

4.3.1 - Casos de Uso 28

4.3.2 - Diagrama de Classes 30

4.3.3 - Diagramas de Seqüência 31

4.3.3.1 - Diagrama de Seqüência Funcionário 31

4.3.3.2 - Diagrama de Seqüência Cotação 32
4.3.3.3 - Diagrama de Seqüência Gerente 32
4.4 – CAMADAS DO APLICATIVO 33
4.5 – IMPLEMENTAÇÃO DO APLICATIVO 35
4.5.1 – Implementação da Autenticação do Usuário (login) 35
4.5.2 - Implementação do Caso de Uso “Funcionário” 36
4.5.3 - Implementação do Caso de Uso “Cotação” 38
4.5.4 - Implementação do Caso de Uso “Gerente” 39

5 - CONCLUSÃO 42

6 - REFERÊNCIAS BIBLIOGRÁFICAS 43

1 – INTRODUÇÃO

O computador tornou-se uma ferramenta essencial em todos os setores das empresas, sejam elas grandes corporações ou até mesmo de médio e pequeno porte. Com isso as redes de computadores seguem em um crescimento constante, atingindo mais pessoas. Um fator marcante que contribui para o desenvolvimento de tal processo é o crescente número de usuários da internet, que passa por um processo de disseminação acelerado. Em função deste crescimento surgem necessidades quanto ao controle de diversas tarefas que devem ser executadas.

Geralmente as empresas possuem vários departamentos, cada uma delas com necessidades específicas, podendo haver grande diferença em relação às necessidades dos usuários, de software instalado aos recursos dos equipamentos, aos sistemas operacionais e até a arquitetura de hardware. Sendo assim, quanto maior e mais heterogênea uma rede, maior a necessidade de uma ferramenta que ajude a gerenciar estes equipamentos (Bambineti, 2008).

Os recursos que a Internet possui são ilimitados e necessita de uma excelente infra-estrutura Web, além de um acompanhamento constante e especializado para manter-se à frente no concorrido mercado da grande rede mundial. No mundo de negócios atual, o conteúdo Web de qualquer empresa possui um papel fundamental para seu sucesso, pois aumenta a abrangência de marketing, de vendas, e da interação de todos os setores da empresa, independente do espaço físico que ela se encontra (Endel a, 2010).

1.1 – OBJETIVO

O objetivo deste trabalho é adquirir conhecimento das técnicas e ferramentas que envolvem o desenvolvimento de aplicativos Web, tendo em vista que é uma área muito promissora. Esta aplicação envolve conhecimento de comunicação Web, pois proporciona agilidade aos usuários de determinada empresa que dependem de serviços de outros setores da mesma. E para alcançar este objetivo será desenvolvido um aplicativo de controle e gerenciamento de pedidos de compra utilizando as tecnologias de *Java Server Faces* e Java.

1.2 – MOTIVAÇÃO

A grande motivação é poder utilizar os conhecimentos adquiridos com este tema no mercado de trabalho. Uma grande vantagem do Aplicativo *Web* reside no fato de que seus usuários não dependem do ambiente físico de determinada empresa para acessar e utilizar suas funcionalidades basta ele utilizar um micro que tenha acesso a Internet.

1.3 – JUSTIFICATIVA

Com este trabalho pretendo adquirir muito conhecimento na área de desenvolvimento de Aplicativos Web e Java, já que no mercado atual a aplicação via Internet é tendência mundial. Além disso, é uma oportunidade de conhecer teoricamente as ferramentas adequadas para cada uma das tecnologias existentes.

2 – FUNDAMENTAÇÃO TEÓRICA BÁSICA

Neste capitulo será feita uma descrição teórica das principais tecnologias e ferramentas que serão utilizadas no desenvolvimento do aplicativo *web* de gerenciamento e controle de pedidos de compras.

2.1 – LINGUAGEM DE PROGRAMAÇÃO JAVA

Java começou na verdade como um projeto da Sun que objetivava o estudo e a análise de interações entre dispositivos eletrônicos e computadores, não exatamente para o desenvolvimento de uma linguagem para aplicativos embarcados. Nesse tempo a linguagem tinha o nome-código Green e foi idealizada por James Gosling. Contudo, a Sun perdeu a disputa para outra empresa e o projeto Green estava marcado para ser descontinuado. Entretanto, eis que surge algo novo, com a explosão da Internet em 1993, os criadores do projeto observaram um grande potencial em utilizar a linguagem para o desenvolvimento de páginas Web com o chamado conteúdo dinâmico. Com o objetivo de tirar proveito desse mercado o projeto Green sofre algumas mudanças e adaptações, se chamando Oak e em 1995 é lançado com o nome Java. Assim em 1995 a Sun faz a divulgação da linguagem no evento SunWorld e a disponibiliza oficialmente em 1996, ano da primeira JavaOne. Java está presente em todos os lugares, desde telefones celulares até TVs e computadores. A linguagem Java é uma das mais difundidas e conta com inúmeros recursos pela Web (Endel b, 2010).

Sistemas Java se dividem em várias partes: um ambiente, a linguagem, a interface de programas aplicativos (*Applications Programming Interface - API*) Java e várias bibliotecas de classes (Deitel e Deitel., 2001).

2.2 – ECLIPSE

É a IDE (*Integrated Development Environmente*) Java focada no desenvolvimento de aplicações de software e ferramentas. Possui como principais características a forte orientação ao desenvolvimento baseado em plug-ins e o amplo suporte ao desenvolvedor. Esta poderosa IDE é um software que possui um conjunto de funcionalidades embutidas, com o objetivo de prover um modo mais fácil e interativo de construir e manipular seus programas. De acordo com (Java Free, 2010), entre estas funcionalidades geralmente figuram:

- Um editor de texto com facilidades especialmente desenhadas para a linguagem;
- Um compilador (e um interpretador, no caso Java e outras linguagens interpretadas);
- Um editor gráfico, que permite a criação e edição da interface gráfica do programa a ser desenvolvido;
- Um *debugger*, que consiste em detectar os bugs do código para serem corrigidos. Ele possibilita um monitoramento mais elegante do funcionamento do seu programa, facilitando a remoção dos erros.

2.3 - JAVA SERVER PAGES (JSP)

O JSP é uma tecnologia voltada para o desenvolvimento de aplicações para web, seu diferencial é a capacidade de gerar páginas dinâmicas, através de uma sintática familiar a do HTML (*Hypertext Markup Language*). Como o HTML é utilizado para o desenvolvimento de conteúdo estático, o JSP mescla suas “tags” com as “tags” do HTML, incluindo o conteúdo dinâmico em meio ao conteúdo estático. Portanto o JSP é um documento de texto contendo HTML e Java. Por ter como base a linguagem de programação Java, o JSP possui a vantagem da portabilidade de plataforma, que permite sua execução em diversos sistemas operacionais e possibilita o

desenvolvedor produzir aplicações que acessem a banco de dados, manipule arquivos no formato de texto, dentre outros recursos (Todd, 2003).

2.4 - JAVA SERVER FACES (JSF)

As tecnologias voltadas para o desenvolvimento de aplicações web têm mudado constantemente. JSF é uma tecnologia que incorpora características de um framework MVC (*Model-View-Controller*) para web e de um modelo de interfaces gráficas baseado em eventos. Como é baseado no padrão de projeto MVC, uma de suas melhores vantagens é a clara separação entre a visualização e o modelo (Endel c, 2010).

O padrão MVC divide uma aplicação em três camadas: modelo, visualização e controle. O modelo é responsável por representar os objetos de negócio, manter o estado da aplicação e fornecer ao controlador o acesso aos dados. Já a visualização representa a interface com o usuário, sendo responsável por definir a forma como os dados serão apresentados e encaminhar as ações dos usuários para o controlador. A camada de controle é responsável por fazer a ligação entre o modelo e a visualização, além de interpretar as ações do usuário e as traduzir para uma operação sobre o modelo, onde são realizadas mudanças e, então, gerar uma visualização apropriada. O controle no JSF é composto de um *servlet* denominado *FacesServlet*, por arquivos de configuração e por um conjunto de manipuladores de ações e observadores de eventos. O *FacesServlet* é responsável por receber requisições da web, em seguida tais requisições são redirecionadas para o modelo, para então remeter uma resposta. Os arquivos de configuração são responsáveis por realizar associações e mapeamentos de ações e pela definição de regras de navegação. Já os manipuladores de eventos são responsáveis por receber os dados vindos da camada de visualização, acessar o modelo, e então devolver o resultado para o *FacesServlet* (Endel c, 2010).

O modelo no JSF representa os objetos de negócio e executa uma lógica de negócio ao receber os dados da camada de visualização. Já a camada de visualização é composta por *component trees* (hierarquia de componentes UI), tornando possível unir um componente ao outro para formar interfaces mais complexas (Endel c,

2010). Segundo (Endel c, 2010), *Java Server Faces* oferece ganhos no desenvolvimento de aplicações web por diversos motivos:

- Permite que o desenvolvedor crie UIs através de um conjunto de componentes UIs predefinidos;
- Fornece um conjunto de tags JSP para acessar os componentes;
- Reusa componentes da página;
- 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);
- Fornece separação de funções que envolvem a construção de aplicações web.

Embora *Java Server Face*s forneça “tags” JSP para representar os componentes em uma página, ele foi projetado para ser flexível, sem limitar-se a nenhuma linguagem markup em particular, nem a protocolos ou tipo de clientes (Endel c, 2010).

2.5 - HIBERNATE

O Hibernate é uma tecnologia ambiciosa que tem como finalidade resolver os problemas de persistência em Java, ela age mediando à interação da aplicação com o banco de dados relacional, com isso o desenvolvedor fica mais livre para se concentrar no problema em si. Com o Hibernate não é necessário que você siga todas as regras específicas e padrões de projeto, quando escreve sua lógica empresarial e classes de persistência, sendo assim esta tecnologia se integra suavemente com a maioria das novas e já existentes aplicações e não requerem drásticas modificações nas mesmas (Bauer, King, 2005).

Hibernate é uma ferramenta de mapeamento objeto/relacional para Java, que transforma os dados tabulares de um banco de dados em um grafo de objetos definido pelo desenvolvedor. Com o uso da tecnologia do Hibernate, o desenvolvedor deixa de escrever muito do código de acesso a banco de dados e SQL, acelerando a velocidade do seu desenvolvimento de uma forma considerável.

O Hibernate é indicado para sistemas que contam com um modelo rico, onde a maior parte da lógica de negócios fica na própria aplicação Java, dependendo pouco de funções específicas do banco de dados (Bauer, King, 2005).

2.6 - TOMCAT

Servidor Web Java, o Tomcat é a implementação das tecnologias de *Java Servlet* e Java Server Pages. Desenvolvido em ambiente aberto e participativo, recebe a colaboração de desenvolvedores de todo o mundo, o Tomcat não pode ser considerado um servidor de aplicação completo, pelo fato de não preencher os requisitos necessários (Tomcat, 2010).

O Tomcat por ser robusto, pode ser utilizado mesmo em um ambiente de produção, ele tem a capacidade de atuar como servidor web, ou pode funcionar integrado a um servidor web dedicado como, por exemplo, o Apache. Quando utilizado como servidor web, o Tomcat provê um servidor web HTTP (*Hiper Text Transfer Protocol*) puramente em Java, que possui ferramentas para configuração e gerenciamento, o que também pode ser feito editando manualmente arquivos de configuração formatados em XML. Para disponibilizar as aplicações nesse servidor, é preciso seguir um padrão rígido e bem estabelecido, garantindo que uma mesma aplicação possa ser também montada em outro servidor de aplicação de forma transparente (Tomcat, 2010).

2.7 - APACHE MYFACES

O projeto Apache MyFaces ou simplesmente MyFaces, foi desenvolvido para dar sustentação ao padrão JSF e é um *framework* para o desenvolvimento de aplicações web. O MyFaces é um projeto da Apache Software Foundation que vem crescendo rapidamente. Além do MyFaces Core, que é a implementação do JSF em si, estão surgindo vários subprojetos que fornecem componentes adicionais, trazendo suporte a Ajax, vinculação com dados, entre outras funcionalidades importantes no desenvolvimento web (Camolesi, 2009).

2.8 - BANCO DE DADOS POSTGRESQL

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (SGBDOR) baseado no POSTGRES, Versão 4.2, desenvolvido no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. O projeto POSTGRES, liderado pelo Professor Michael Stonebraker, foi patrocinado pelas seguintes instituições: Defense Advanced Research Projects Agency (DARPA); Army Research Office (ARO); National Science Foundation (NSF); e ESL, Inc (Endel d, 2010).

O PostgreSQL descende deste código original de Berkeley, possuindo o código fonte aberto. Fornece suporte às linguagens SQL92/SQL99 além de outras funcionalidades modernas (Endel d, 2010).

O POSTGRES foi o pioneiro em muitos conceitos objetos-relacional que agora estão se tornando disponíveis em alguns bancos de dados comerciais. Os Sistemas de Gerenciamento de Bancos de Dados Relacionais (SGBDR) tradicionais suportam um modelo de dados que consiste em uma coleção de relações com nome, contendo atributos de um tipo específico. Nos sistemas comerciais em uso, os tipos possíveis incluem número de ponto flutuante, inteiro, cadeia de caracteres, monetário e data. É largamente reconhecido que este modelo não é adequado para aplicações futuras de processamento de dados. O modelo relacional substituiu com sucesso os modelos anteriores em parte devido à sua simplicidade. De acordo com (Endel d, 2010), o PostgreSQL oferece um substancial poder adicional, devido à incorporação dos conceitos mostrados abaixo de uma forma que os usuários podem facilmente estender o sistema:

- Herança;
- Tipos de dados;
- Funções;

e outras funcionalidades que fornecem poder e flexibilidades adicionais:

- Restrições;
- Gatilhos;
- Regras;

- Integridade da transação.

Estas funcionalidades colocam o PostgreSQL dentro da categoria de bancos de dados referida como objeto-relacional. Repare que isto é diferente daqueles referidos como orientados a objetos que, em geral, não são muito adequados para dar suporte às linguagens de banco de dados relacionais tradicionais. Portanto, embora o PostgreSQL possua algumas funcionalidades de orientação a objetos, está firmemente ligado ao mundo dos bancos de dados relacionais. Na verdade, alguns bancos de dados comerciais incorporaram recentemente funcionalidades nas qual o PostgreSQL foi o pioneiro (Endel d, 2010).

3 – PROGRAMAÇÃO EM CAMADAS

Nos primórdios da computação, quando um aplicativo era executado em uma única máquina, era comum encontrar sistemas monolíticos contendo todas as funcionalidades do aplicativo em uma única camada, onde sua manutenção e atualização eram extremamente complexas (Neto et al., 2008). A figura 1 mostra um código monolítico.

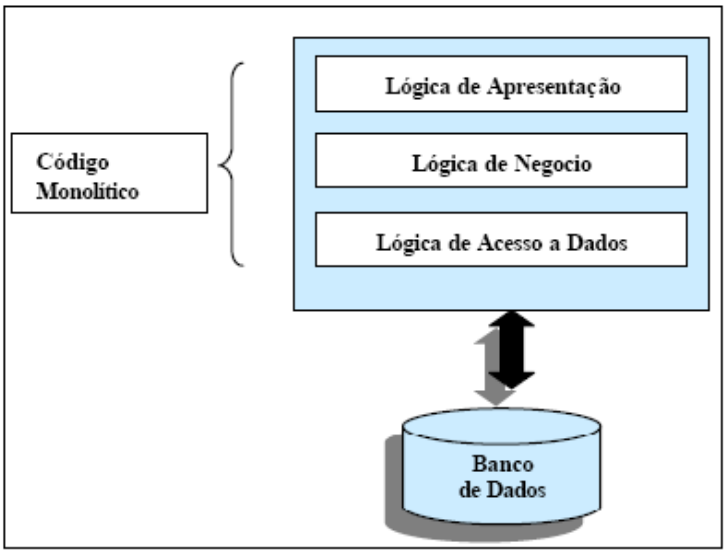

Figura 1: Código Monolítico. Fonte: (Neto et al., 2008).

Com o objetivo de se manter vários aplicativos e uma única base de dados, a arquitetura monolítica evoluiu para uma arquitetura com duas camadas onde a lógica de acesso aos dados estava separada do restante da aplicação, permitindo assim que diversos programas possam acessar a mesma base de dados. Apesar da evolução, os sistemas ainda eram potencialmente monolíticos, pois a lógica da interface e a lógica de negócios permaneceram reunidas em uma única camada, como demonstra a figura 2.

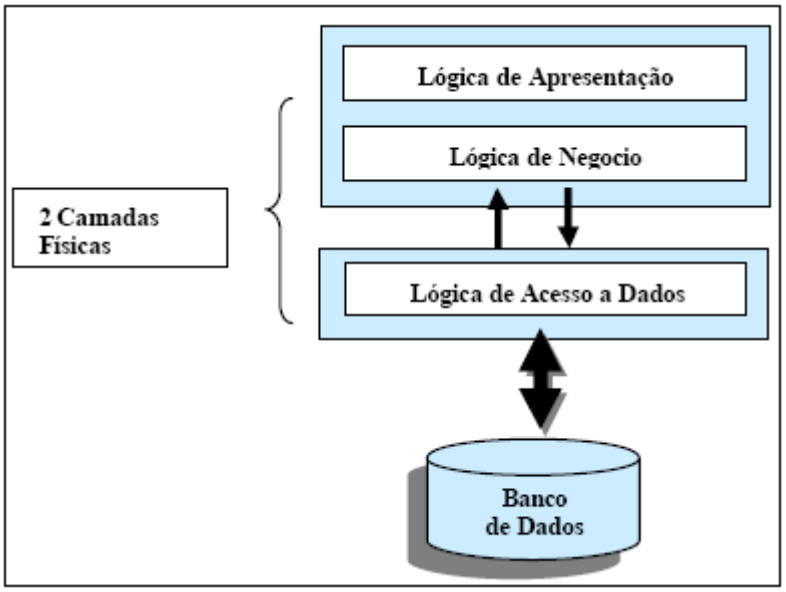

Figura 2: Código em Duas Camadas. Fonte: (Neto et al., 2008).

Com o surgimento da Internet, a arquitetura com duas camadas teve que ser modernizada, pois o tempo necessário para carregar todos os componentes da regra de negócio no cliente em um aplicativo *Web* era extremante elevado, tornando assim o sistema inviável. Devido aos problemas de eficiência, a arquitetura com duas camadas foi substituída por uma arquitetura de três camadas.

A arquitetura em três camadas envolve a separação das funcionalidades do aplicativo, com o objetivo de separar a lógica de apresentação, a lógica de negócio e a lógica de acesso a dados.

Essa separação torna o sistema mais flexível, pois qualquer alteração em uma determinada camada não influi nas demais, desde que o mecanismo de comunicação entre elas permaneça inalterado (Neto et al., 2008).

A figura 3 demonstra um código em três camadas:

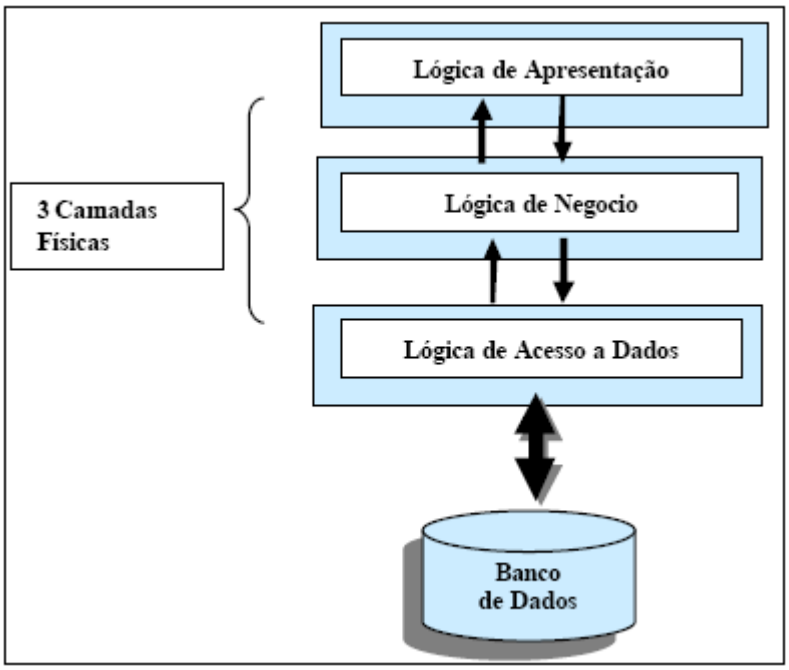

Figura 3: Código em Três Camadas. Fonte: (Neto et al., 2008).

3.1 – PADRÃO MODELO VISUALIZAÇÃO CONTROLE (MVC)

O MVC é um padrão de projeto arquitetural baseado em outros padrões como o *Observer*, *Composite* e *Strategy*. O MVC possibilita a separação de um projeto em múltiplas camadas, das quais fazem parte: Modelo (Model), Visão (View) e Controlador (Controller) (Campos et al., 2009).

A figura 4 mostra as interações entre os componentes do padrão MVC.

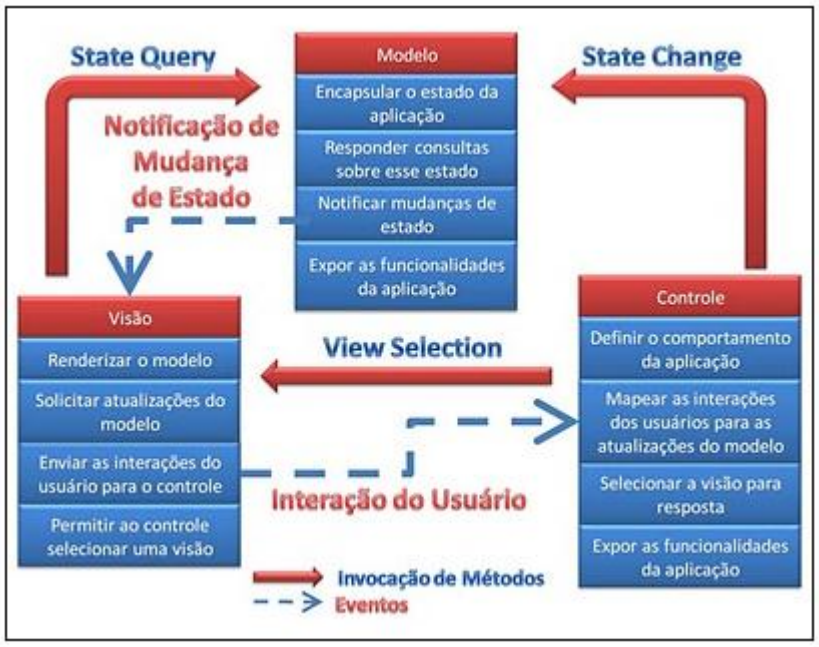

Figura 4: Interação entre os Componentes do Padrão MVC. Fonte: (Endel e, 2010).

Será feita uma descrição das principais características dos componentes do padrão MVC. Os componentes do padrão MVC são: Modelo (*Model*), Visão (*View*) e Controlador (*Controller*).

3.1.1 – Modelo (*Model*)

A camada do modelo é designada como o núcleo da aplicação que funciona implementando toda a lógica do negócio e as operações que governam e modificam tais dados (Sun Microsystems, 2008). Com isso, o modelo é todas as classes e funcionalidades em si. O *model* representa e manipula, seguindo as regras de negócio, os dados do sistema (Campos et al., 2009).

O modelo encapsula o estado e comportamento da aplicação além de ser o único componente do MVC que faz interface da aplicação frente à fonte de dados, que normalmente é representada pelo banco de dados da aplicação. Um conceito errado e frequente é que o banco de dados de uma aplicação é o próprio *model*. Na verdade, o banco de dados constitui a camada de persistência do modelo e os componentes *model* são os únicos a acessarem e executarem alterações no banco de dados (Campos et al., 2009).

Conforme (Souza, 2004), devido aos bancos serem, em sua esmagadora maioria, relacionais, é necessário no modelo existir um mapeamento dos objetos do *software* orientados a objeto, para as tabelas do banco de dados. Essa técnica é o Mapeamento Objeto-Relacional (ORM – *Object Relational Mapping*). Além desta técnica, há o padrão DAO (*Data Access Object* – Objeto de acesso a dados) que permite melhor acesso aos dados por parte dos objetos.

3.1.2 – Visão (View)

A visão é o componente da arquitetura MVC que é responsável pela apresentação, é a interface de representação do modelo, ou seja, trata-se da fronteira entre usuário e o sistema em si (Campos et al., 2009). Conforme (Reenskaug, 1979) a *view* pode exibir alguns atributos e ocultar outros, atuando como um filtro para os dados do modelo. É papel do controlador definir a *view* apropriada para a exibição da resposta obtida pela requisição feita.

No modelo tradicional do MVC, décadas após a concepção de Reenskaug, apesar de ainda com os ideais do mesmo, a *view* é composta por GUI's (Graphical Users Interface) enquanto que no MVC web a visualização das informações acontece através das páginas HTML (*HyperText Markup Language* – Linguagem de Marcação de Hipertexto) para informação estática e JSP, ASP, PHP, dentre outras, para informações dinâmicas. Outra diferença entre o MVC original e o *web* é que as *views* no original recebiam notificações do *model*, feitas por sua vez, através da implementação de interfaces que definem objetos observadores e observados. No caso de aplicações *web* estas notificações são sempre mediadas pelo controlador, principalmente pela necessidade do reuso (Campos et al., 2009).

É na visão que ocorrem todas as interações do usuário que serão tratadas pelo controlador para chamar os métodos apropriados no modelo. Este componente pode ser considerado o mais flexível do MVC, podendo ser facilmente alterado ou substituído. Novas visões também podem ser facilmente implementadas, sem afetar em nada a estrutura do sistema (Campos et al., 2009).

3.1.3 – Controlador (Controller)

Este componente controla a comunicação entre o modelo e a visão. Ele é considerado como a fronteira entre os outros dois componentes do MVC e sua finalidade é controlar interações que ocorram a partir do usuário que trabalha sobre elementos na camada de visão e descobre o que essa entrada significará para o modelo. Ou, ainda, do modelo em resposta às ações anteriores (Campos et al., 2009).

As entradas que esses componentes recebem são eventos de mouse, entradas de teclado, entre outras. Estes eventos por sua vez serão traduzidos em requisições de serviços para que outro componente possa tratá-lo. É o *controller* que define o comportamento da aplicação, mapeando as ações do usuário em atualizações e buscas no modelo, e também selecionando qual a *vie*w deve ser enviada como resposta (Campos et al., 2009). Em aplicações *web*, uma solução comumente adotada para a criação deste controle central é a implementação do padrão *Front-Controller*. Um objeto é criado e serve como ponto de entrada principal para todas as requisições que vem da *view*, este objeto mapeia a requisição para o tratador adequado. O *Struts* é um exemplo de *framework* que faz o papel de *Front-Controller* em aplicações Java para web (Campos et al., 2009).

4 – DESENVOLVIMENTO DO APLICATIVO

Neste capítulo será feita a definição e modelagem do problema, e elaborada a especificação do aplicativo proposto neste trabalho.

4.1 – DEFINIÇÃO DO PROBLEMA

O problema que será abordado neste projeto está relacionado com o desenvolvimento de um aplicativo *Web* que facilitará empresas que possuem

departamento de compras, já que seu objetivo principal é o controle e o gerenciamento de pedidos de compra. Serão três tipos de usuários:

- **Funcionário**: este deverá enviar seu pedido de compra ao usuário de cotação;
- **Cotação**: será responsável pela cotação dos pedidos recebidos, em seguida enviará os pedidos cotados para a aprovação do usuário do tipo gerente;
- **Gerente**: terá a responsabilidade de aprovar (ou não aprovar) os pedidos cotados.

Para fazer a especificação deste projeto será utilizada a metodologia orientada a objetos, a *Unified Modeling Language* (UML), usando como ferramenta o Jude.

4.2 – MODELAGEM DO PROBLEMA

O desenvolvimento deste projeto foi baseado conforme os três tipos de usuários envolvidos: funcionário, cotação e gerente. A figura 5 mostra uma visão geral da modelagem do problema.

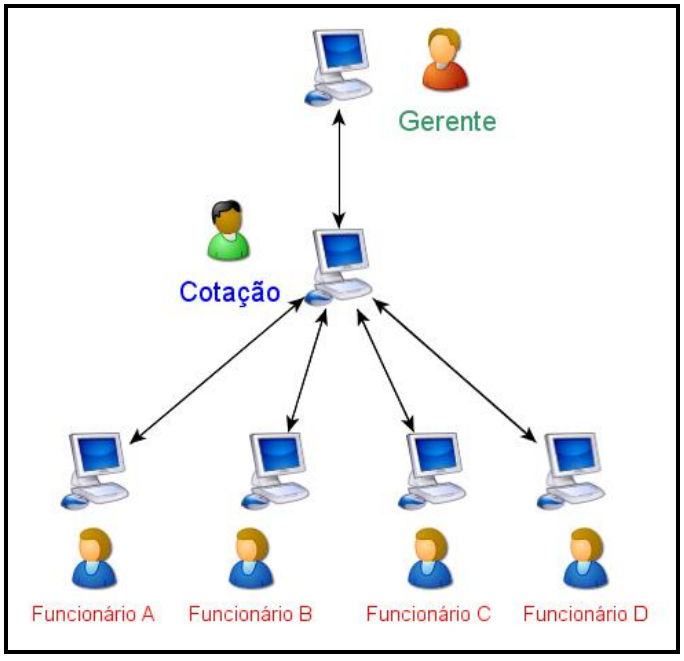

Figura 5: Modelagem do Projeto.

4.3 – ESPECIFICAÇÃO DO APLICATIVO

Será feita uma descrição da especificação do aplicativo desenvolvido neste projeto.

4.3.1 – Casos de Uso

O aplicativo possui três casos de usos:

- **Funcionário**: responsável em efetuar pedidos de compras e enviá-los para cotação;
- **Cotação**: responsável em efetuar a cotação dos pedidos recebidos, enviá-los para aprovação (ou não aprovação) e imprimir os pedidos aprovados;
- **Gerente**: responsável em aprovar (ou não aprovar) os pedidos cotados recebidos, cadastrar usuário e cadastrar departamento;

A figura 6 demonstra o caso de uso Funcionário:

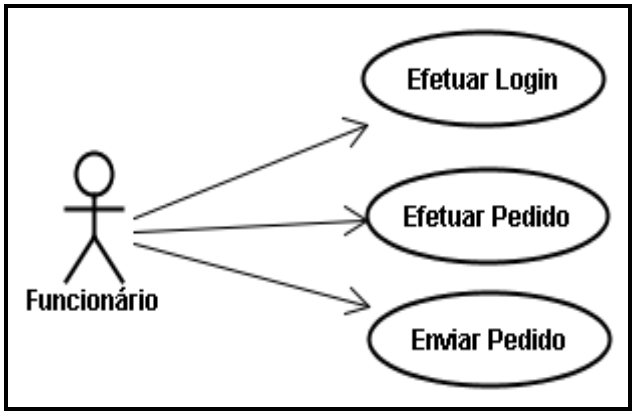

Figura 6: Caso de Uso Funcionário.

A figura 7 demonstra o caso de uso Cotação:

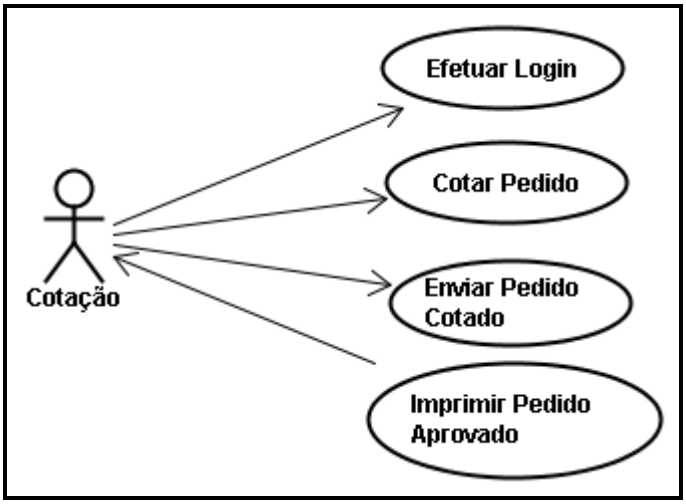

Figura 7: Caso de Uso Cotação.

A figura 8 demonstra o caso de uso Gerente:

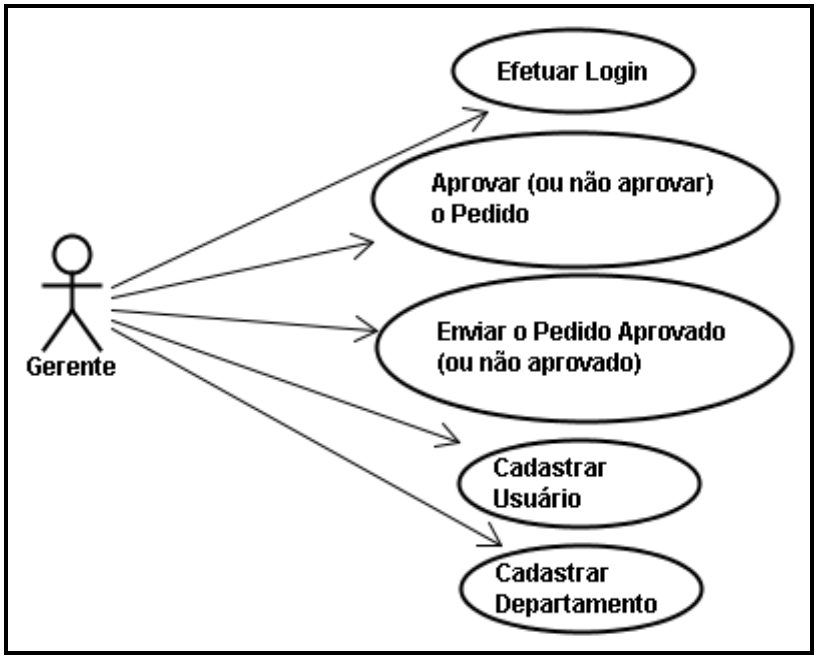

Figura 8: Caso de Uso Gerente.

4.3.2 – Diagrama de Classes

As classes utilizadas no projeto são:

a) **Usuario**: classe que contém todas as informações dos usuários do sistema e que possui os seguintes atributos: codigo, nome, username, senha e tipo (funcionário, cotação ou gerente);
b) **Departamento**: classe que contém as informações dos departamentos utilizados no sistema e que possui os seguintes atributos: codigo e nome;
c) **Pedido**: classe que contém as informações necessárias do pedido de compra e que possui os seguintes atributos: codigo, descricao, status (aguardando, cotado, aprovado ou cancelado), dataInclusao e dataConclusao;
d) **ItemPedido**: classe que contém as informações dos itens que constituem o pedido de compra e que possui os seguintes atributos: codigo, descricao, quantidade e valorUnitario.

O Diagrama de Classes está demonstrado na figura 9.

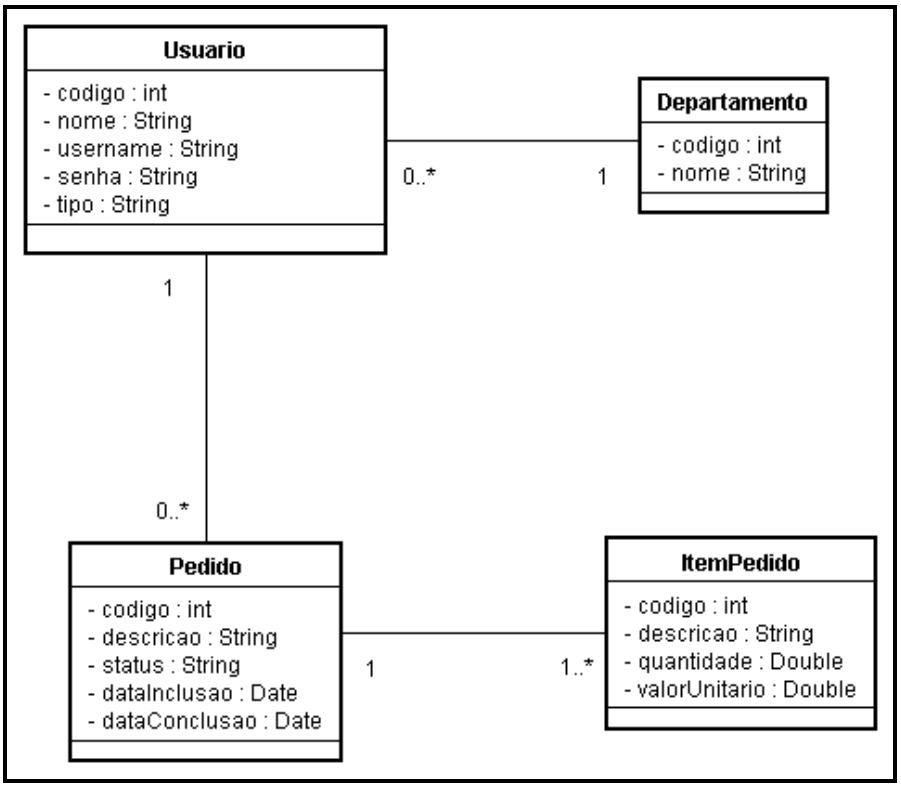

Figura 9: Diagrama de Classes.

Definido o diagrama de classes, serão definidas as ações do sistema, através do diagrama de sequência, onde se estabelece cada passo do sistema diante dos casos de uso existentes.

4.3.3 – Diagramas de Seqüência

Os diagramas de sequências representam como o próprio nome diz a sequência em que as ações ocorrem no aplicativo. Para cada caso de uso, há um diagrama de seqüência, conforme a descrição a seguir:

4.3.3.1 – Diagrama de Seqüência Funcionário

Este diagrama de seqüência é executado quando o usuário do tipo funcionário efetua seu pedido de compra. Este diagrama está demonstrado na figura 10.

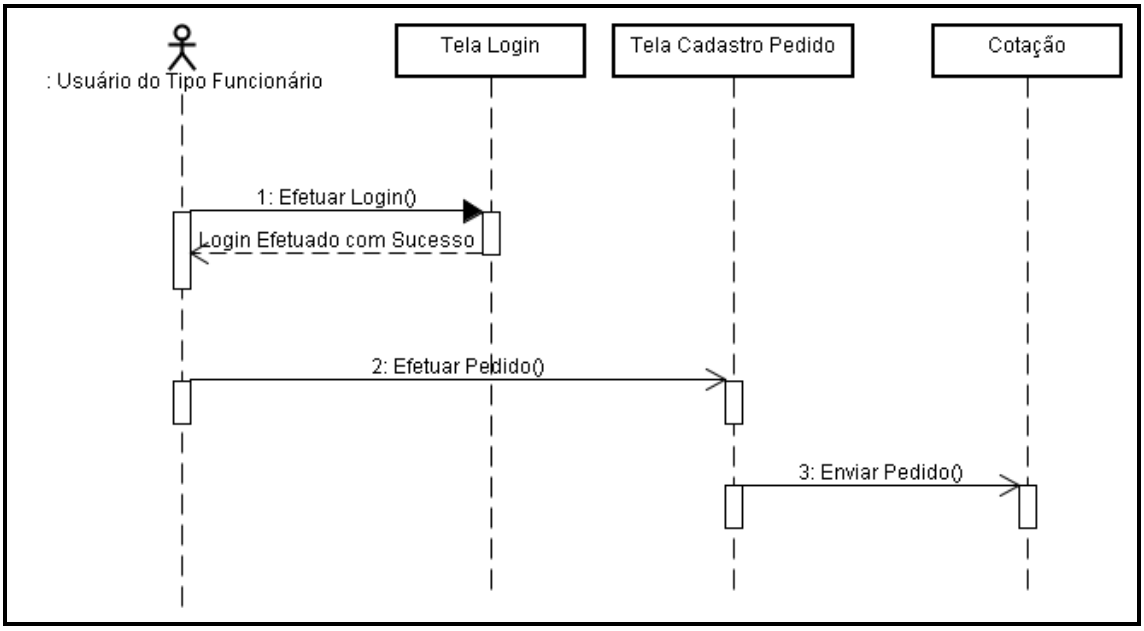

Figura 10: Diagrama de Sequência Funcionário.

4.3.3.2 – Diagrama de Seqüência Cotação

Este diagrama de seqüência é executado quando o usuário do tipo cotação efetuar sua rotina. Este diagrama está demonstrado na figura 11.

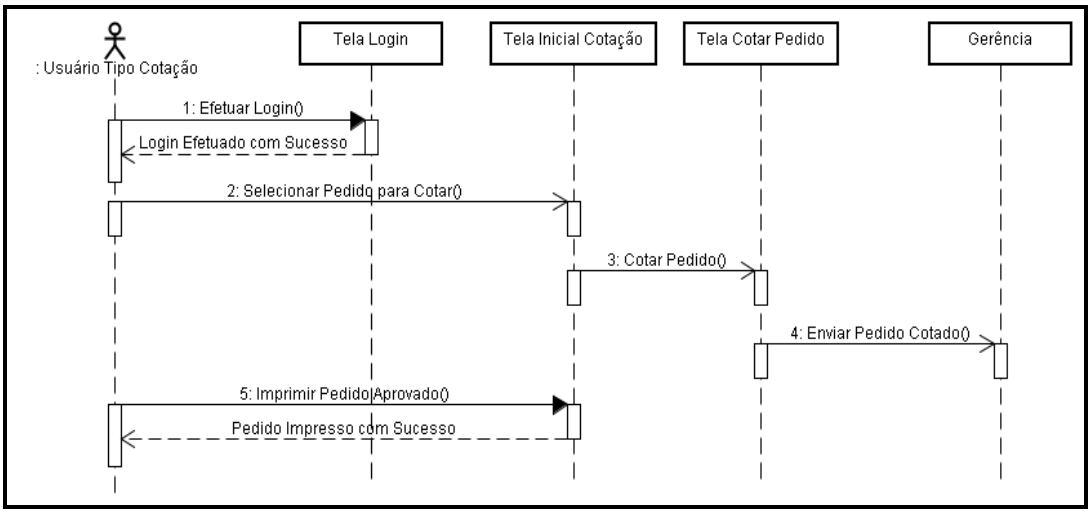

Figura 11: Diagrama de Sequência Cotação.

4.3.3.3 – Diagrama de Seqüência Gerente

Este diagrama de seqüência é executado quando o usuário do tipo gerente efetuar sua rotina. O diagrama de seqüência gerente está demonstrado na figura 12.

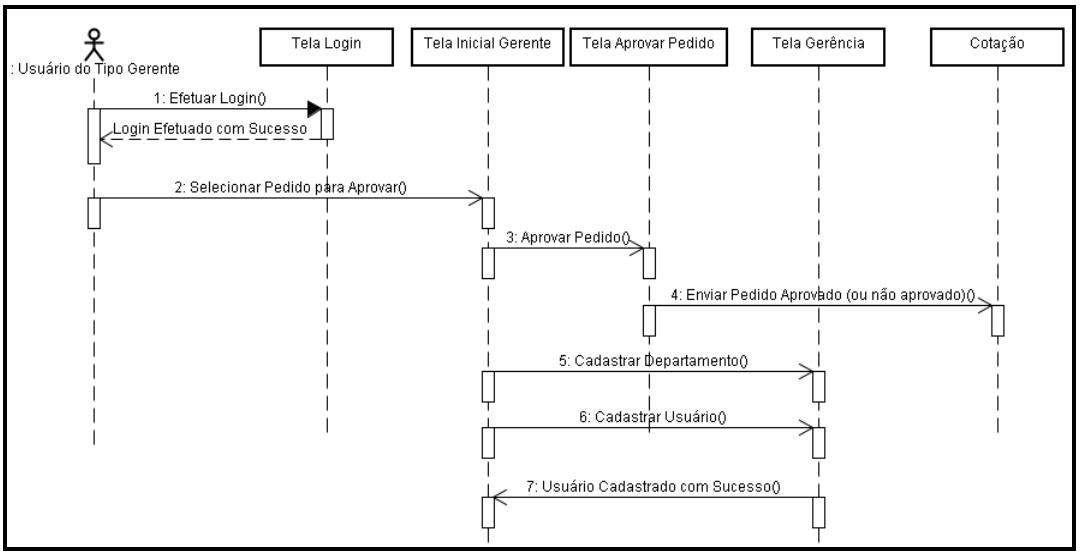

Figura 12: Diagrama de Sequência Gerente.

4.4 – CAMADAS DO APLICATIVO

Serão apresentadas as camadas do aplicativo *web* para o gerenciamento de serviços. Estas camadas envolvem as chamadas camadas do modelo, camadas da visão e a camada do controlador no projeto. A figura 13 mostra a camada do modelo no projeto (beans, dao).

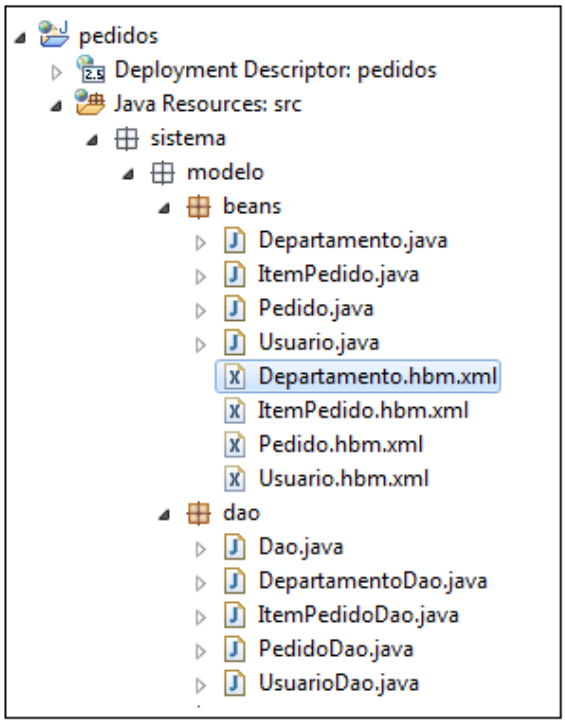

Figura 13: Camada do Modelo no Projeto

- **Departamento.java, ItemPedido.java, Pedido.java e Usuario.java**: classes que possuem os atributos utilizados na aplicação.
- **Departamento.hbm.xml, ItemPedido.hbm.xml, Pedido.hbm.xml e Usuario. hbm.xml**: técnica Mapeamento Objeto-Relacional.
- **Dao.java, DepartamentoDao.java, ItemPedidoDao.java, PedidoDao.java e UsuarioDao.java**: onde foi implementadas as quatro operações básicas do banco de dados (*insert*, *update*, *delete* e *select*).

A figura 14 mostra a camada da visão no projeto.

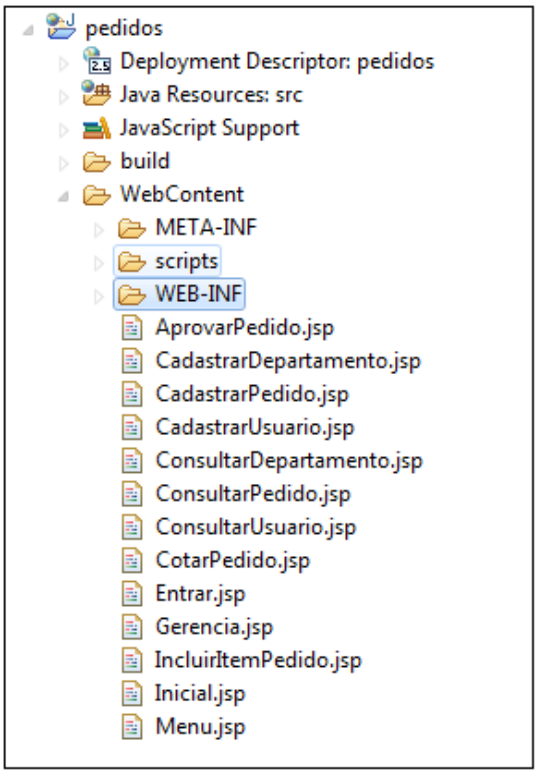

Figura 14: Camada da Visão no Projeto.

Uma observação importante é que no projeto a camada da visão está sempre nas páginas JSP. A figura 15 mostra a camada do controlador no projeto:

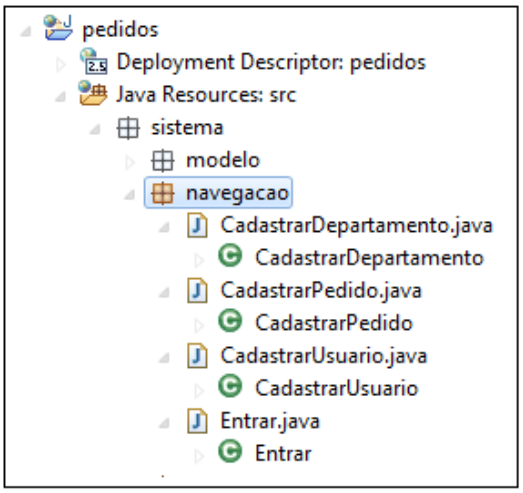

Figura 15: Camada do Controlador no Projeto.

No projeto a camada do controlador está na parte da navegação, onde estão implementadas as classes de controle das páginas JSP.

4.5 - IMPLEMENTAÇÃO DO APLICATIVO

Será apresentado à implementação do aplicativo *web* para gerenciamento de serviços. A implementação do aplicativo foi feita utilizando a IDE do Java Eclipse.

4.5.1 – Implementação da Autenticação de Usuário

O usuário que desejar acessar alguma entidade do aplicativo deverá efetuar o *login* da aplicação, que nada mais é do que a autenticação do usuário. O acesso das funcionalidades do aplicativo será de acordo com a permissão estabelecida pelo tipo de usuário (funcionário, cotação ou gerente). Agora se o usuário ainda não possuir acesso, vai ter que solicitar um *login* ao administrador que possui permissão de cadastrar um novo usuário, que no caso do sistema é o usuário do tipo gerente. A figura 16 mostra a interface do *login*.

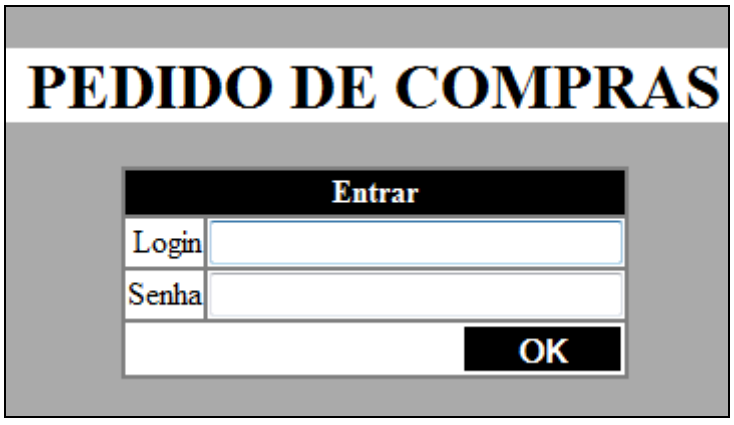

Figura 16: Interface Login.

Para um melhor entendimento sobre a execução do projeto será apresentada a execução dos casos de uso “Funcionário”, “Cotação” e “Gerente”.

4.5.2 – Implementação do Caso de Uso “Funcionário”

O caso de uso “Funcionário” ocorre quando um usuário do tipo funcionário efetua seu *login* para realizar um pedido de compra. A figura 17 mostra a interface inicial do sistema para o usuário do tipo funcionário.

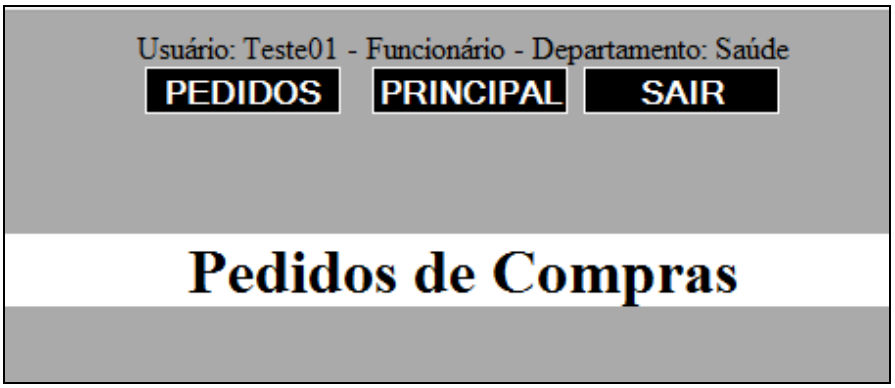

Figura 17: Interface Inicial Funcionário.

Para realizar um pedido o usuário deverá selecionar o botão “PEDIDOS”, e que mostrará uma nova interface para o cadastro do pedido, conforme mostra a figura 18.

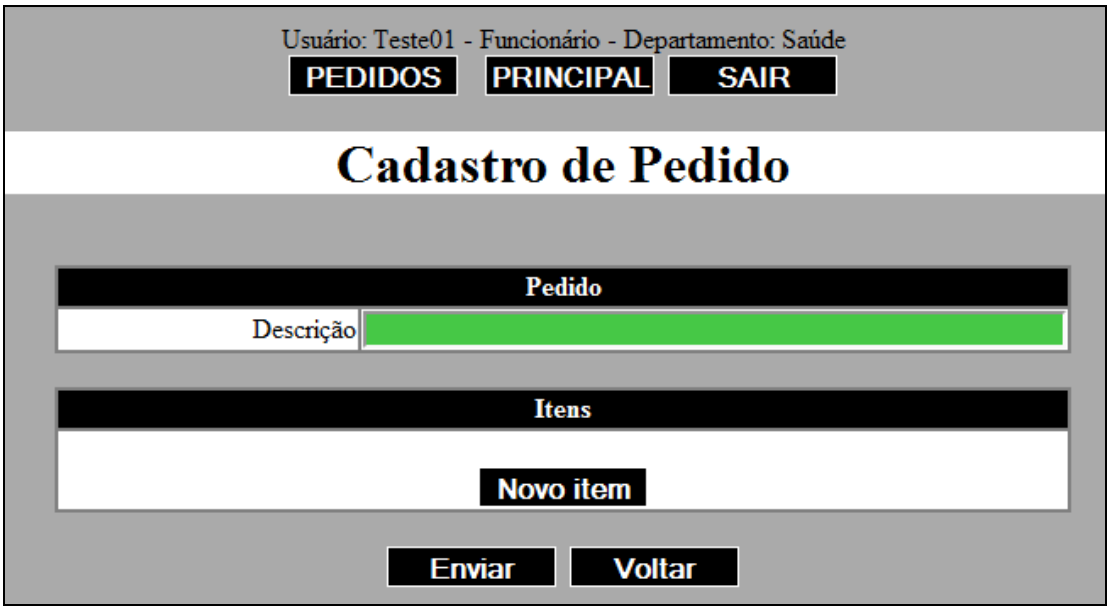

Figura 18: Interface Cadastro de Pedido.

A figura 18 mostra a interface de cadastro de pedido, onde o usuário deverá preencher a descrição do seu pedido (finalidade), em seguida clicar no botão “Novo item” para inserir os itens desejados. Para finalizar o pedido, o usuário deverá clicar no botão “Enviar” ou clicar no botão “Voltar” para o pedido ser cancelado.
Quando um determinado usuário tiver realizado alguns pedidos, ao abrir o aplicativo novamente, terá acesso a todos os seus pedidos realizados com seu respectivo status:

- **Aguardando**: o pedido ainda está aguardando cotação;
- **Aprovado:** o pedido foi aprovado para a compra;
- **Cancelado**: o pedido foi cancelado, com isso a compra não será realizada.

A figura 19 mostra um usuário que já possui alguns pedidos.

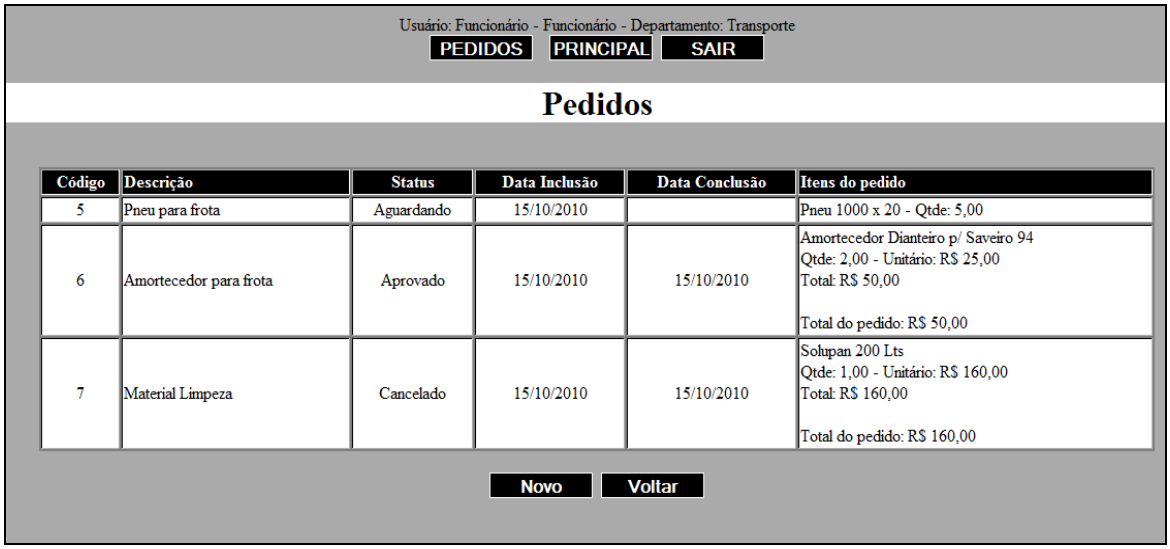

Figura 19: Interface Inicial Usuário com Pedidos Realizados.

4.5.3 – Implementação do Caso de Uso “Cotação”

O caso de uso “Cotação” ocorre quando o usuário do tipo cotação efetua seu *login* para realizar as cotações dos pedidos recebidos e imprimir os pedidos aprovados. A figura 20 mostra a interface inicial do sistema para o usuário do tipo cotação.

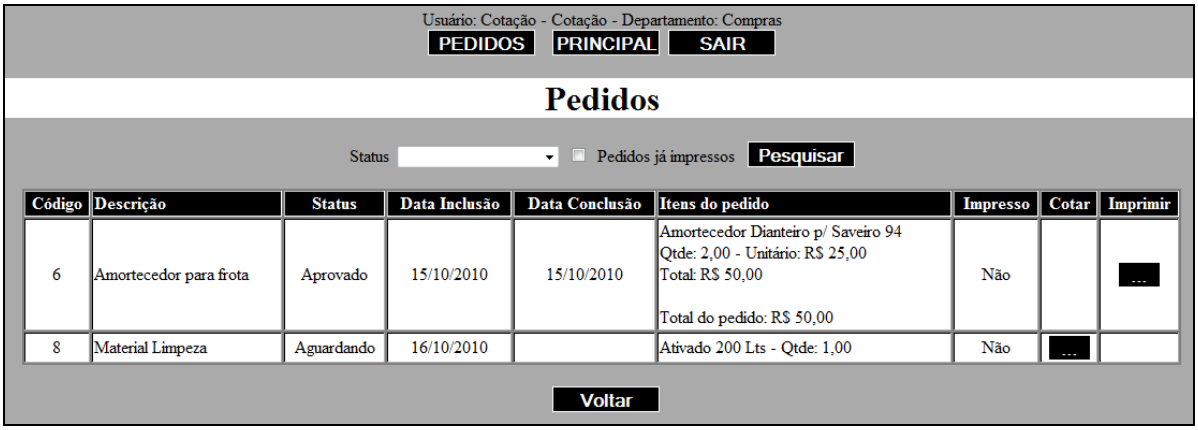

Figura 20: Interface Inicial Cotação.

Note que o usuário do tipo cotação só recebe os pedidos com *status* de aguardando ou aprovado. Os pedidos com status de aprovado deverão ser impressos, já os com *status* de aguardando deverão ser cotados. A figura 21 mostra a interface em que os pedidos são cotados.

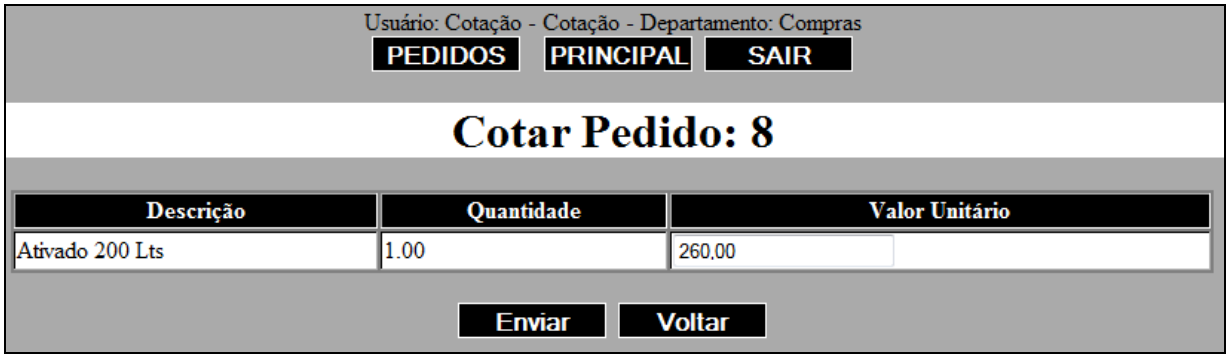

Figura 21: Interface Cotar Pedido.

Para realizar a cotação do pedido o usuário deverá preencher o valor unitário de cada item, em seguida deverá clicar no botão “Enviar” para finalizar o processo. Para poder cancelar a cotação, o botão “Voltar” deverá ser clicado.

4.5.4 – Implementação do Caso de Uso “Gerente”

O caso de uso “Gerente” ocorre quando o usuário do tipo gerente efetua seu login para aprovar (ou não aprovar) um pedido recebido ou cadastrar um novo usuário. A figura 22 mostra a interface inicial do sistema para o usuário do tipo gerente.

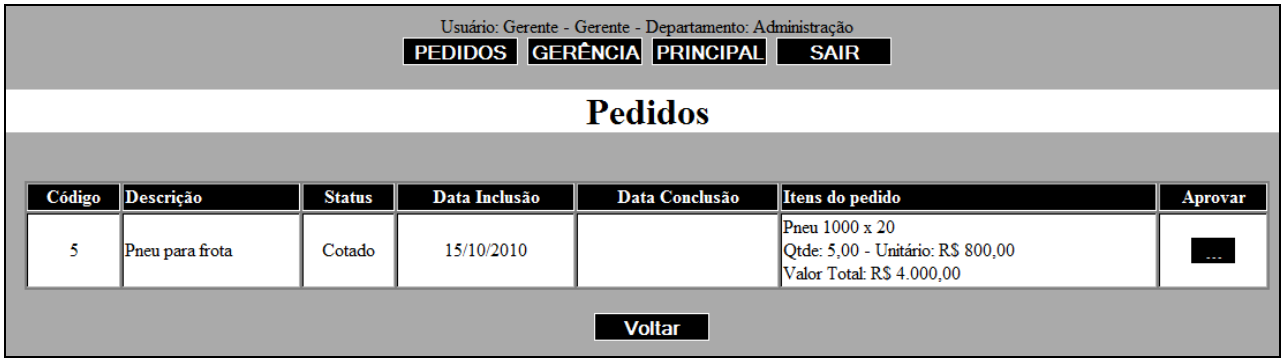

Figura 22: Interface Inicial Gerente.

Para poder aprovar um pedido o usuário deverá clicar no botão “PEDIDOS” da tela inicial, logo todos os pedidos recebidos serão mostrados, conforme na figura 22. Em seguida o usuário deverá clicar no botão “Aprovar”, para que a tela de aprovar pedido seja acessada.

A figura 23 mostra a tela para aprovar pedido.

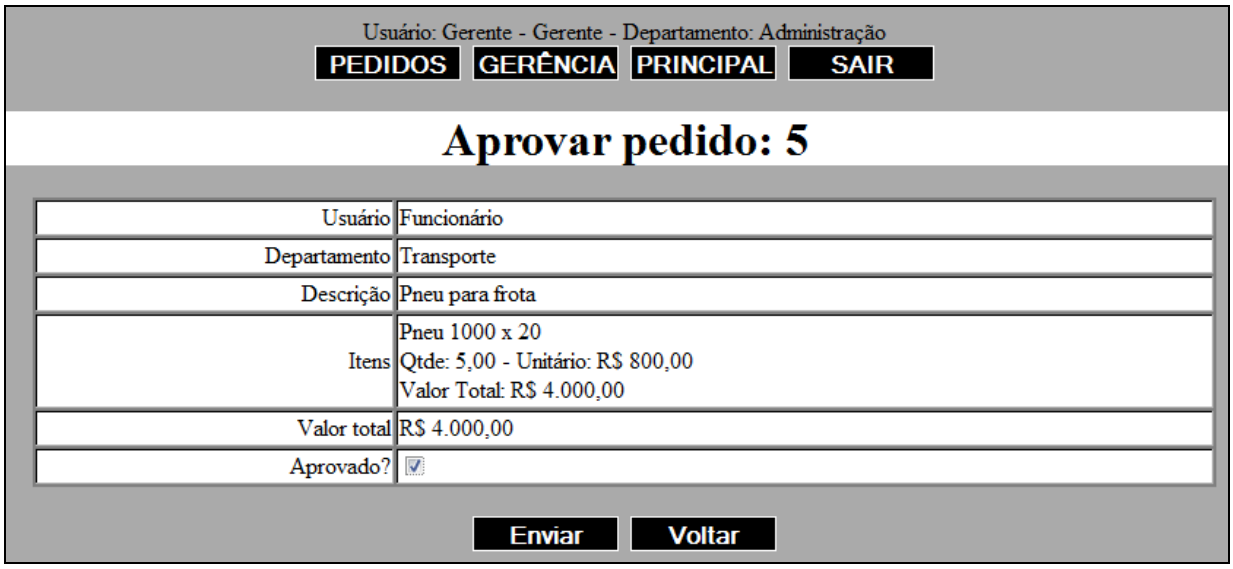

Figura 23: Interface Aprovar Pedido.

O pedido só será aprovado quando o usuário marcar a opção “Aprovado?” e em seguida clicar no botão “Enviar”. Para poder cancelar o pedido o usuário deverá clicar no botão “Enviar” sem marcar a opção “Aprovado?” ou clicar no botão “Voltar” para que todo o processo de aprovação seja cancelado.

Para poder cadastrar um novo usuário, o usuário do tipo gerente deverá clicar no botão “GERÊNCIA” da interface inicial. A figura 24 mostra a interface gerência.

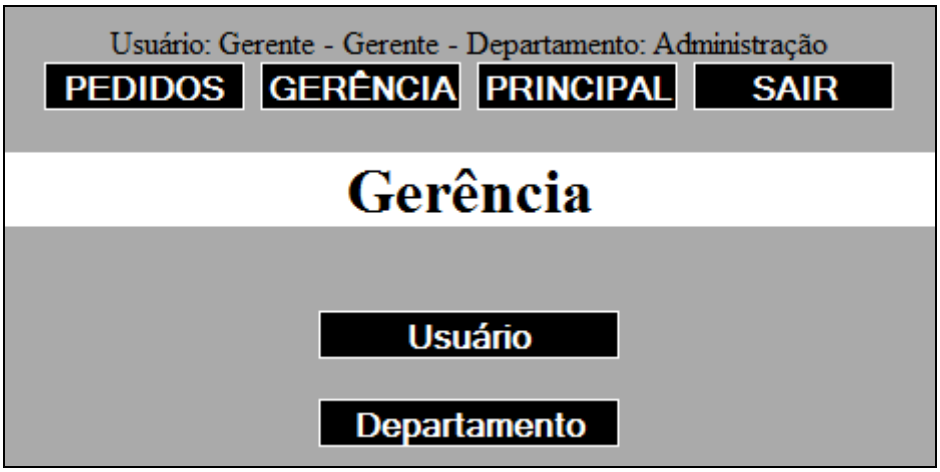

Figura 24: Interface Gerência.

Em seguida o usuário deverá clicar no botão “Usuário” para poder acessar a interface de cadastro. A figura 25 mostra a interface de cadastro de usuário.

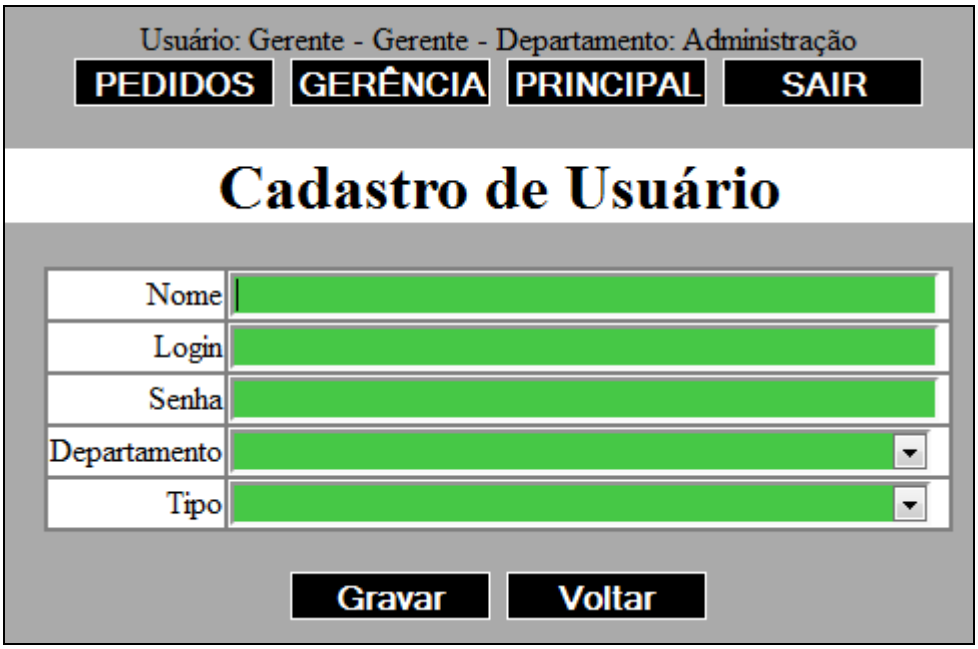

Figura 25: Interface Cadastro de Usuário.

Para finalizar o cadastro de um novo usuário, o usuário do tipo gerente deverá preencher todos os campos e em seguida clicar no botão “Gravar”, para cancelar toda a operação basta clicar no botão “Voltar”. Se o departamento do novo usuário não estiver cadastrado no sistema o usuário deverá retornar na tela gerência e clicar no botão “Departamento”, conforme a figura 26.

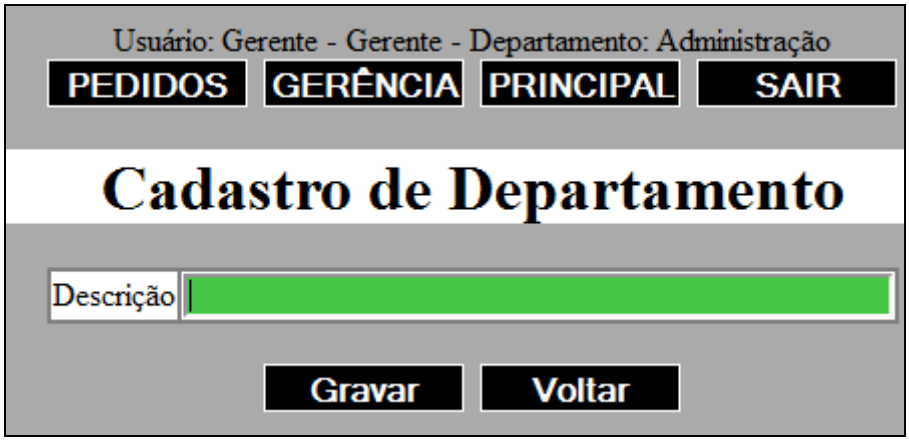

Figura 26: Interface Cadastro de Departamento.

Para finalizar o cadastro do novo departamento, o usuário deverá preencher a descrição e em seguida clicar no botão “Gravar” ou para cancelar o cadastro clicar no botão “Voltar”.

5 – CONCLUSÃO

O desenvolvimento de aplicações utilizando arquitetura de três camadas deixa o aplicativo mais flexível, pois separa as funcionalidades do mesmo, e isto facilita muito nas atualizações futuras.

A importância no desenvolvimento deste trabalho foi adquirir conhecimento das tecnologias relacionadas à Java na implementação de aplicativos Web, já que no mercado atual a aplicação via Internet é tendência mundial. Um das dificuldades encontradas foi na programação e na utilização destas tecnologias. Entretanto, considero satisfatório o conhecimento adquirido, pois foi possível implementar um aplicativo que será utilizado na empresa que trabalho. Esta aplicação servirá de base para implementar mais funcionalidades futuramente.

6 – REFERÊNCIAS BIBLIOGRÁFICAS

(Bambineti, 2008) Bambineti C.: **Sistema de Web Services para Inventário de Estações em Rede**, Trabalho de Graduação, Universidade Regional de Blumenau, Blumenau, 2008.

(Bauer, King, 2005) BAUER, Christian; KING, Gravin: **Hibernate in Action.** Grennwich: Manning Publication Co., 2005.

(Camolesi, 2009) CAMOLESI, I.: **Gerenciador de Projeto UML para Fábrica de Software.** 2009. 47p. Trabalho de Conclusão de Curso. Fundação Educacional do Município de Assis – FEMA/Instituto Municipal de Ensino Superior de Assis – IMESA.

(Campos et al., 2009) CAMPOS, Fernando A. Barbeiro; SANTUCI, Leonardo Balduino: **Estudo de Aplicabilidade do Padrão MVC.** 2009. 111p. Trabalho de Conclusão de Curso. Universidade de Franca.

(Deitel e Deitel., 2001) Deitel, H. M., Deitel, P. J.: **Java Como Programar**, trad. Edson Furmankiewicz, 3ª Edição, Editora Bookman, 2001.

(Endel a, 2010) **Sistemas Web.** Disponível em: <http://redt.com.br>. Acesso em 25 nov. 10.

(Endel b, 2010) **A história do Java**. Wikilivros – Livros abertos por um mundo aberto. Disponível em: <http://pt.wikibooks.org/wiki/Java/HistoriaJava>. Acesso em 16 jun. 10.

(Endel c, 2010) Pitanga, T.: **Java Server Faces: A mais nova tecnologia Java para desenvolvimento Web**. Disponível em:

<http://www.programefacil.com/programacao/java/apostilasjava.php>. Acesso em 14 set. 10.

(Endel d, 2010) The PostgreSQL Global Development Group. **Tutorial do PostgreSQL 7.3.2.** Disponível em: <http://www.scribd.com/doc/5088/Tutorial-PostgreSQL>. Acesso em 14 set. 10.

(Endel e, 2010) Padrão MVC. Disponível em: <http://erealityhome.wordpress.com/2007/11/20/padrao-mvc/>. Acesso em 01 dez. 10.

(Java Free, 2009) Eclipse. Disponível em: <http://www.javafree.uol.com.br/wiki/Eclipse>. Acesso em 16 jun. 10.

(Neto et al., 2008) NETO, Kanji Hara; NADALETE, Lucas G.; GENNARI, Fábio A.; FREITAS, Antônio A. Carneiro. **Desenvolvimento de Sistema Web utilizando arquitetura em Três Camadas e Applets**. In: Boletim Informativo. Centro Federal de Educação Tecnológica do Paraná – CEFET – PR, 9 p.

(Reenskaug, 1979) REENSKAUG, T. M. H. **MVC – Xerox Parc 1978-79**. 1979. Disponível em: <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>. Acesso em: 01 dez. 10.

(Souza, 2004) SOUZA, M. V. B. **Estudo Comparativo entre Frameworks Java para Construção de Aplicações Web.** 2004. 67 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal de Santa Maria, Santa Maria.

(Sun Microsystems, 2008) Model – View – Controller. Disponível em: <http://java.sun.com/blueprints/patterns/MVC.html>. Acesso em: 01 dez. 10.

(Todd, 2003) Todd, Nick; Szolkowski, Mark. **Java Server Pages: Guia do Desenvolvedor.** Campus, 2003.

(Tomcat, 2010) Tomcat. Disponível em: <http://tomcat.apache.org>. Acesso em 16 jun. 10.