

LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO
DO RIO GRANDE DO NORTE

ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE
CONTROLE DE ORDEM DE SERVIÇOS NA WEB

ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE
CONTROLE DE ORDEM DE SERVIÇOS NA WEB

SAULO CANTALICE MOREIRA

NATAL
2005

ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE DE ORDEM DE SERVIÇOS NA WEB

Monografia apresentada ao Curso de
Graduação em Sistemas de Informação
da Faculdade Natalense para o
Desenvolvimento do Rio grande do
Norte como requisito parcial para
obtenção do grau de Bacharel em
Sistemas de Informação

Orientador:

Fellipe Araújo Aleixo, MsC.

NATAL
2005



ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE
CONTROLE DE ORDEM DE SERVIÇOS NA WEB

Monografia apresentada ao Curso de
Graduação em Sistemas de Informação
da Faculdade Natalense para o
Desenvolvimento do Rio Grande do
Norte como requisito parcial para
obtenção do grau de Bacharel em
Sistemas de Informação

Catálogo na Publicação – Biblioteca da FARN
Setor de Processos Técnicos

Moreira, Saulo Cantalice

Análise e implementação de um sistema de controle de ordem de serviço na web
/ Saulo Cantalice Moreira. – Natal: FARN, 2005.
64 f.

Orientador: Felipe Araújo Aleixo (M.Sc.)

Monografia (Graduação do Bacharelado em Sistemas de Informação) –
Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

1. Sistema Operacional – Monografia. 2. Automação de Rotinas – Monografia. I.
Título.

RN/FARN/BC

CDU 004.451.46

83-26.550

**FARN
BIBLIOTECA**

Nº Reg. M000430

17, 05, 06

M

004.451.46

M8390

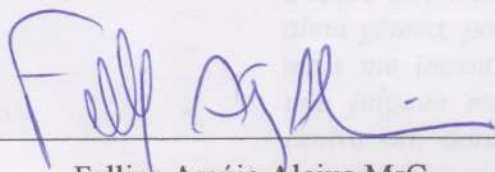
2005

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO
DO RIO GRANDE DO NORTE**

ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE DE ORDEM
DE SERVIÇOS NA WEB

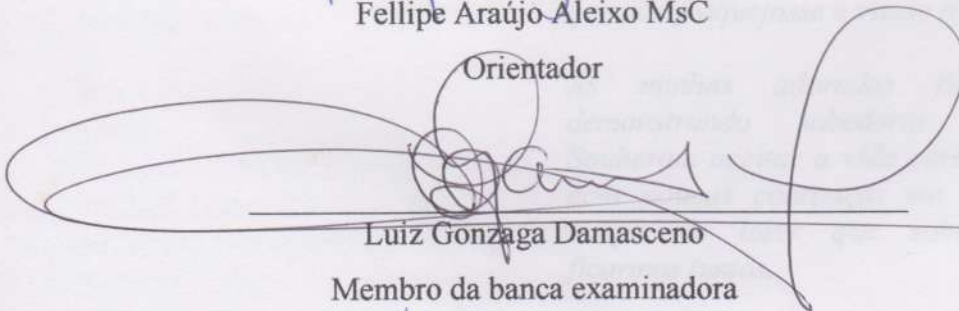
Saulo Cantalice Moreira

Monografia apresentada e aprovada em 17 de 12 de 2005, pela banca examinadora
composta pelos seguintes membros.



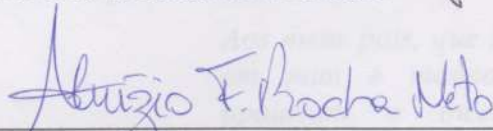
Fellipe Araújo Aleixo MsC

Orientador



Luiz Gonzaga Damasceno

Membro da banca examinadora



Aluizio Ferreira da Rocha Neto

Membro da banca examinadora

Natal, 17 de dezembro de 2005.

SUMÁRIO

RESUMO	01
ABSTRACT	02
1. INTRODUÇÃO	03
2. REFERENCIAL TEÓRICO	04
2.1. ORDEM DE SERVIÇO	04
2.2. INTERNET	05
2.3. WORD WIDE WEB	05
2.4. HTML	05
2.4.1. Tag	06
2.5. PROGRAMAÇÃO ORIENTADA A OBJETO (POO)	06
2.5.1. Classes	07
2.5.2. Objetos	07
2.5.3. Atributos e Métodos	07
2.5.4. Encapsulamento	08
2.5.5. Herança	08
2.5.6. Polimorfismo	08
2.6. UML - Unified Modeling Language	09
2.6.1. Diagrama de Casos de Uso	10
2.6.2. Diagrama de Colaboração	10
2.6.3. Diagrama de Classes	10
2.7. JAVA	11
2.7.1. Java Servlet	11
2.7.2. Java Server Pages	11
2.8. SQL	12
2.9. POSTGRES	12
2.10. ECLIPSE	12
2.11. MACROMEDIA DREAMWEAVER	12
2.12. APACHE TOMCAT	12
3. DESENVOLVIMENTO DO SISTEMA	13
3.1. OBJETIVO GERAL	13
3.2. OBJETIVOS ESPECÍFICOS	13
3.3. USUÁRIOS DO SISTEMA	13
3.4. FUNÇÕES DO SISTEMA	13
3.5. ATRIBUTOS DO SISTEMA	13
3.6. ARQUITETURA DO SISTEMA	13
3.7. CASOS DE USO	14
3.7.1. Descrição de Alto Nível dos Casos de Uso	16
3.7.2. Casos de Uso Expandido	18
3.7.3. Caso de uso - Cadastro Funcionário	31
3.7.3.1. Expansão do Caso de Uso	32
3.7.3.2. Modelo de Domínio	33
3.7.3.3. Diagrama de Sequência	33
3.7.3.4. Detalhes da Implementação	34
3.7.4. Caso de Uso - Cadastro de Ordem de Serviço	36

Dedico este trabalho a Estela, a minha alma gêmea, por ter sido a pessoa que mais me incentivou, nos momentos em que julgava mais difíceis, ela sempre estava ali, dando animo e força, para que não fraquejasse e viesse a desistir.

As minhas adoradas filhas, que demonstrando sabedoria e amor Souberam aceitar a vida corrida do pai com poucas cobranças em tão pouco tempo de lazer que sobrava para ficarmos juntos.

Aos meus pais, que sempre acreditaram em mim e mesmo distante, sempre apoiavam e incentivavam o meu trabalho.

SUMÁRIO

RESUMO.....	01
ABSTRACT	02
1. INTRODUÇÃO	03
2. REFERENCIAL TEÓRICO.....	04
2.1. ORDEM DE SERVIÇO	04
2.2. INTERNET	05
2.3. WORD WIDE WEB	05
2.4. HTML	05
2.4.1. Tag	06
2.5. PROGRAMAÇÃO ORIENTADA A OBJETO (POO)	06
2.5.1. Classes	07
2.5.2. Objetos	07
2.5.3. Atributos e Métodos	07
2.5.4. Encapsulamento	08
2.5.5. Herança	08
2.5.6. Poliformismo	08
2.6. UML – Unified Modeling Language	09
2.6.1. Diagrama de Casos de Usos	10
2.6.2. Diagrama de Colaboração	12
2.6.3. Diagrama de Classes	13
2.7. JAVA	15
2.7.1. Java Servlet	16
2.7.2. Java Server Pages –	16
2.8. SQL	17
2.9. POSTGRES	17
2.10. ECLIPSE	18
2.11. MACROMEDIA DREAMWEAVER	18
2.12. APACHE TOMCAT	19
3. MODELAGEM DO SISTEMA	20
3.1. OBJETIVO GERAL	20
3.2. OBJETIVOS ESPECÍFICOS	20
3.3. USUÁRIOS DO SISTEMA	20
3.4. FUNÇÕES DO SISTEMA	21
3.5. ATRIBUTOS DO SISTEMA.....	22
3.6. ARQUITETURA DO SISTEMA	22
3.7. CASOS DE USO	24
3.7.1. Descrição de Alto Nível dos Casos de Uso	26
3.7.2. Casos de Uso Expandido	31
3.7.3. Caso de uso - Cadastro Funcionário	31
3.7.3.1. Expansão do Caso de Uso	32
3.7.3.2. Modelo de Domínio	33
3.7.3.3. Diagrama de Sequência	33
3.7.3.4. Detalhes da Implementação	34
3.7.4. Caso de Uso – Cadastro de Ordem de Serviço	36

3.7.4.1. Expansão do Caso de Uso	36
3.7.4.2. Modelo de Domínio	38
3.7.4.3. Diagrama de Seqüência	39
3.7.4.4. Detalhes da Implementação	39
3.7.5. Caso de uso – Cadastro de Orçamento	41
3.7.5.1. Expansão do Caso de Uso	42
3.7.5.2. Modelo de Domínio	43
3.7.5.3. Diagrama de Seqüência	44
3.7.5.4. Característica da Implementação	44
3.7.6. Modelo de dados	47
4. CONCLUSÃO	48
5. REFERÊNCIAS	49
ANEXOS	51

A informatização das rotinas de entrada de abertura de Ordem de Serviço da empresa TECLYBRA Informática Ltda. O sistema permitirá o acompanhamento desde a entrada, até a entrega do equipamento ao cliente solicitante de execução dos serviços de conserto e manutenção. O sistema será acessado a partir da Internet, facilitando assim o uso e reduzindo custos de manutenção, já que a empresa terá funcionários atuando como técnicos residentes distantes nas empresas dos clientes, e isso permitirá obter novos pedidos de serviços e gerenciar-las a partir de um equipamento conectado a Internet, verificando o seu andamento. O sistema foi desenvolvido usando a tecnologia Java e JSP, fazendo uso do banco de dados PostgreSQL.

Palavras-chaves: Ordem de Serviço, JSP (Java Server Pages), HTML (Hypertext Markup Language), Java.

RESUMO

Este trabalho visa apresentar um sistema computacional que tem por finalidade, a automatização das rotinas de controle de abertura de Ordem de Serviço da empresa TECLYBRA Informática Ltda. O sistema permitirá o acompanhamento desde a entrada, até a entrega do equipamento ao cliente solicitante da execução dos serviços de conserto e manutenção. O sistema será acessado a partir da Internet, facilitando assim o uso e reduzindo custos de manutenção, já que a empresa tem funcionários atuando como técnicos residentes diretamente nas empresas dos clientes, e isto permitirá abrir novas ordens de serviços e gerenciá-las a partir de um equipamento conectado a Internet, verificando o seu andamento. O sistema foi desenvolvido usando a tecnologia Java e JSP, fazendo uso do banco de dados Postgres.

Palavras-chaves: Ordem de Serviço. JSP (*Jave Server Pages*). HTML (*Hipertext Markup Language*). Java.

1. INTRODUÇÃO

ABSTRACT

A crescente expansão, uso e divulgação da Internet tem se tornado um grande atrativo para as empresas modernizarem e investirem em novos Sistemas de Informações (SI) ou em Tecnologia da Informação (TI). Essas tecnologias levam as empresas serem mais competitivas objetivando atender princípios básicos da administração, como redução de custos.

This work aims at to present a computational system that has for purpose, the automatization of the routines of control of opening of Administrative order of company TECLYBRA Ltda. The system will allow the accompaniment of the entrance ties the delivery of the equipment to the solicitant customer of the execution of the services of fixes and maintenance. The system will make use of the Internet, thus facilitating the use and reducing costs of maintenance, since the company has resident employees acting as technician directly in the companies of the customers, and this it will allow to open new administrative orders and to manage them from a connected equipment the Internet, verifying its course. The system was developed using the technology Java and JSP, making use of the database Postgres.

Resumo: Este trabalho apresenta um sistema computacional que tem por finalidade, a automatização das rotinas de controle de abertura de ordem administrativa da empresa TECLYBRA Ltda. O sistema permitirá o acompanhamento das etapas de entrega do equipamento ao cliente solicitante da execução dos serviços de reparos e manutenção. O sistema utilizará a Internet, facilitando o uso e reduzindo custos de manutenção, uma vez que a empresa possui funcionários atuando como técnicos residentes diretamente nas empresas dos clientes, e isso permitirá a abertura de novas ordens administrativas e o gerenciamento delas a partir de um equipamento conectado à Internet, podendo também verificar o andamento de uma ordem de serviço.

A empresa, por possuir funcionários atuando como técnicos residentes diretamente nas empresas dos clientes, terá a facilidade de abrir novas ordens de serviços e gerenciá-las a partir de um equipamento conectado à Internet, podendo também verificar o andamento de uma ordem de serviço.

Como o sistema será baseado na Web, será possível a utilização do sistema a partir de qualquer lugar, desde que esse lugar esteja conectado à Internet. Para isso permite não só um uso de forma distribuída pelos funcionários da empresa, mas também permite aos clientes um acompanhamento das suas solicitações sem a necessidade dos mesmos se deslocar até a empresa.

2. REFERENCIAL TEÓRICO

1. INTRODUÇÃO

A crescente expansão, uso e divulgação da Internet tem se tornado um grande atrativo para as empresas modernizarem e investirem em novos Sistemas de Informações (SI) ou em Tecnologia da Informação (TI). Essas tecnologias levam as empresas serem mais competitivas objetivando atender princípios básicos da administração, como redução de custos, agilidade nas suas ações e satisfação dos seus clientes.

Pensando nisso, este projeto busca a redução de custos da empresa Teclybra Informática, tornando-a mais ágil na prestação de serviços de manutenção dos equipamentos de seus clientes.

Com o aproveitamento do parque de equipamentos existente na empresa, não haverá necessidade de novas aquisições, já que está sendo utilizado como equipamento hospedeiro do sistema, um servidor de arquivos existente e para o acesso das estações de trabalho será necessário apenas um navegador de Internet.

A empresa, por possuir funcionários atuando como técnicos residentes diretamente nas empresas dos clientes, terá a facilidade de abrir novas ordens de serviços e gerenciá-las a partir de um equipamento conectado a Internet, podendo também verificar o andamento de uma ordem de serviço.

Como o sistema será baseado na Web, será possível a utilização do sistema a partir de qualquer lugar, desde que esse lugar esteja conectado a Internet. Este fato permite não só um uso de forma distribuída pelos funcionários da empresa, mas também permite aos clientes um acompanhamento das suas solicitações sem a necessidade dos mesmos se deslocar até a empresa.

2. REFERENCIAL TEÓRICO

Para entender o uso do Sistema de Controle de Ordem de Serviço – SCOS, faz-se necessário abordar alguns conceitos envolvidos direta ou indiretamente neste trabalho tais como: Ordem de serviço, Internet, Linguagem Java, JSP entre outros.

2.1. Ordem de Serviço

Procedimento de controle dos serviços de consertos de equipamentos, mantendo o registro e acompanhamento deste serviço. A operacionalização destes procedimentos consiste de:

- Abertura de ordem de serviços;
- Orçamento;
- Fechamento da ordem de serviço;
- Entrega de equipamento.

Abertura de Ordem de Serviço → o cliente ao chegar à recepção com o equipamento destinado à assistência técnica é realizado o cadastro do cliente, do equipamento e do defeito apresentado aparentemente, registrando os dados no sistema, e emitindo um comprovante de entrega do equipamento à empresa.

Orçamento → após a identificação do defeito apresentado no equipamento a ser consertado, é feito, o orçamento das peças e mão de obra necessária ao reparo e repassado esse valor para aprovação do cliente.

Fechamento da Ordem de Serviço → depois de realizado o serviço necessário ao conserto do equipamento, é efetuado o fechamento da ordem de serviço. Procedimento realizado por técnico responsável.

Entrega de Equipamento → após o fechamento da ordem de serviço é realizada a entrega do equipamento ao cliente, emissão do comprovante de entrega, recibo de pagamento, garantia do serviço e atualização dos dados no sistema. Procedimento realizado pelo atendente.

2.2. Internet

Internet é um meio de comunicação composto pela interligação de várias redes de computadores, que utilizam um protocolo comum (conjunto de regras possibilitando entendimento entre as partes envolvidas) permitindo o acesso de qualquer computador a outros, gerando uma infinita base de dados. (VA, 2003).

Rucker apud Gandeman (2004), define a Internet como "uma vasta coleção de grandes e pequenos computadores interligados em redes que se estendem pelo mundo inteiro".

2.3. Word Wide Web

Também chamada Web ou WWW, é o serviço da Internet que propicia a navegação dos usuários por hipertextos que utilizam várias mídias. Ela é um sistema de informações organizado de maneira a englobar todos os outros sistemas de informação disponíveis na Internet, na realidade o WWW faz parte do W3C que é um consorcio, onde são definidos os padrões da Web.

De acordo com Holzschlag e Kaiser (2002) World Wide Web Consortium (W3C) é:

Um consórcio de âmbito internacional dedicado a "conduzir a Web ao seu pleno potencial". É dirigido por Tim Berners-Lee, o inventor da Web. Fundado em 1994, o W3C tem mais de 400 organizações membro incluindo aí a Microsoft, América Online (proprietária da Netscape Communications), Apple Computer, Adobe, Macromedia, Sun Microsystems, e uma vasta gama de fabricantes de hardware e software, fornecedores de conteúdos, instituições acadêmicas e companhias de telecomunicações. O Consórcio está hospedado em três instituições de pesquisas - MIT nos EUA, INRIA na Europa e Keio University no Japão.

2.4. HTML

É uma linguagem de formatação utilizada para desenvolvimento de páginas Web. Pode ser utilizado qualquer editor de texto para escrever uma página HTMLr, pois não precisa compilar, baseia-se no uso de *tag's*. Necessitamos apenas, utilizar as *tag's* específicas para definir a formatação desejada e ao gravar o arquivo, utilizar a extensão do arquivo como ".html". O *browser* irá reconhecer essas *tags* como comandos e exibirá de acordo com formatação desejada.

2.4.1. Tag

Quando um browser exibe uma página Web, ele lê arquivo no formato texto da página e procura símbolos especiais denominados *tags*, que dizem como o texto ou a informação deve ser exibido. Por exemplo: um *tag* pode dizer que um texto deve ser mostrado em negrito, em itálico, com certo tipo de fonte, etc. (SILVA, 2005).

De acordo com Conallen (2003), HTML é uma linguagem de *tag* baseada no *Standard Generalized Markup Language* (SGML) que é usada para expressar o conteúdo e a formatação visual de uma página web.

Já Ramalho (1996, P.5) descreve HTML como sendo: “Uma linguagem especializada, dedicada à exibição e acesso de páginas Web. Consiste de texto comum e de códigos especiais chamados *tags* que, na verdade, são os comandos da linguagem HTML”.

2.5. Programação Orientada a Objeto (POO)

A proposta da programação orientada a objeto é representar o mais fielmente possível às situações do mundo real nos Sistemas Computacionais, permitindo fácil uma compreensão para o desenvolvimento de sistemas, utilizando estruturas de dados que simulem o comportamento dos objetos.

De acordo com Correia e Tafner (2001) “Orientação a Objeto consiste em considerar os Sistemas Computacionais como uma coleção de objetos que interagem de maneira organizada”.

Para Du Bois (2005) Programação Orientada a Objetos (POO) é: “Um paradigma de programação de computadores onde se usam classes e objetos, para representar e processar os dados em um programa de computador”.

A linguagem Java faz uso desse paradigma, tendo o conceito de classes e objetos bem definidos e utilizando as operações destes objetos. Na programação orientada a objetos encontramos alguns conceitos que devemos destacar para um melhor entendimento para o mundo acadêmico e aos usuários, são eles:

- Classes
- Objetos
- Atributos e métodos

- Encapsulamento
- Herança
- Polimorfismo

2.5.1. Classes

Quando temos um conjunto de atributos dando característica a um objeto e temos também métodos que permitam acessar esses atributos ou modificar seu estado definindo o seu comportamento, temos então o conceito de classe.

Booch, Rumbaugh e Jacobson (2000, p.49), afirmam que: "Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmo atributos, operações, relacionamentos e semânticas".

Du Bois (2005) reforça esta afirmação, utilizando um termo mais técnico, quando diz que:

Uma classe é um modelo, um tipo definido pelo usuário. As classes contem em uma única abstração dados (que são chamados de atributos) e funções (que são chamadas de métodos). Dessa maneira uma classe conte propriedades e operações de um conjunto de *objetos*.

2.5.2. Objetos

Du Bois (2005) define Objetos como sendo: "Instâncias de uma classe, ou seja, é a concretização da abstração definida em uma classe. A classe define o molde de alguma coisa, e o objeto é uma unidade que segue esse molde".

2.5.3. Atributos e Métodos

Atributos são as características de cada objeto. O objeto, por exemplo, tem atributos como o modelo, tamanho e etc. Fazendo uma analogia com o ser humano, teríamos raça, altura, cor da íris, nome, data nascimento e etc. Os objetos que constituem as classes além de atributos, possuem métodos que são à maneira com as quais esses atributos possam ser modificados.

2.5.3. “Podemos dizer que os atributos dos objetos são ‘variáveis’ ou ‘campos’ que armazenam os diferentes valores que as características dos objetos podem conter” (TAFNER e CORREIA, 2001).

2.5.4. Encapsulamento

Na classe os dados são relativos ao que queremos modelar e as operações necessárias para manipular estes dados, porém muitas vezes queremos apenas que os dados só possam ser acessados a partir de suas operações. Para alcançar estas características definimos o encapsulamento.

Segundo Figueiredo (2002, p.8) "Encapsulamento é a ocultação ou empacotamento de dados e procedimentos dentro do objeto".

2.5.5. Herança

Uma classe nova nem sempre é criada do zero. Ela pode "herdar" atributos e operações de outras já existentes.

De acordo com Furlan (1998, p.19):

Herança é a capacidade de um novo objeto tomar atributos e operações de um objeto existente, permitindo criar classes complexas sem repetir código. A nova classe simplesmente herda seu nível base de características de antepassado na hierarquia de classe.

2.5.6. Polimorfismo

É a capacidade de objetos diferentes reagirem segundo a sua função a uma ordem padrão. O comando "abre", por exemplo, faz um objeto entrar em ação, seja ele uma janela, uma porta ou uma tampa de garrafa.

Para Thompson (2000) “O polimorfismo é a capacidade de dois objetos de classes distintas tem de implementarem o mesmo método, mas de forma diferente”.

2.6. UML - *Unified Modeling Language*

No desenvolvimento de qualquer projeto se faz necessária uma completa descrição do problema e dos seus requisitos, no intuito de deixar claro o que seja o problema e como o projeto se propõe a solucioná-lo.

Ao conseguir descrever o problema através de vários modelos e documentos, estará sendo feita uma análise do problema.

Segundo Demarco (1989) "Análise é o estudo de um problema, que antecede a tomada de uma ação".

Quando se trata do desenvolvimento de um sistema computacional, o estudo focaliza uma área de trabalho específica, levando a especificação de um novo sistema, que terá como ação, a sua implementação.

Com a criação das linguagens de modelagem orientada a objetos, surgidas entre a metade da década de 1970 e o final da década de 1980, porém foi somente com a união dos métodos criados por Grady Booch, Ivar Jacobson e James Rumbaugh é que surgiu a UML (*Unified Modeling Language*).

A meta de UML é descrever qualquer tipo de sistema, em termos de diagramas e documentos para visualização, especificação, construção e documentação de sistemas.

De acordo com Booch, Rumbaugh e Jacobson (2000, p.13).

A UML (*Unified Modeling Language*) é uma linguagem padrão para a elaboração da estrutura de projetos de software. A UML poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que faça uso dos sistemas complexos de software.

Larman (2000, p. 38) reforça essa afirmação e acrescenta, UML é:

Uma linguagem para especificar, visualizar e construir artefatos de sistema de software. Ela é um sistema de notação (incluindo a semântica para suas notações) dirigida à modelagem de sistemas, usando conceitos orientados a objetos.

Na implementação de um sistema a UML divide o sistema em diagramas formados por nove tipos diferentes:

- Diagrama de classes;
- Diagrama de objetos;
- Diagramas de casos de uso;
- Diagramas de sequência;
- Diagramas de colaborações;

- Diagramas de estados;
- Diagramas de atividades;
- Diagramas de componentes;
- Diagrama de implantação.

Na elaboração deste projeto, foram abordados apenas três diagramas, que são essenciais para o desenvolvimento para análise de um sistema orientado a objetos. Que são:

- Diagrama de casos de uso;
- Diagrama de colaboração;
- Diagrama de classes.

2.6.1. Diagrama de Casos de Uso

Representa a interação dos humanos ou autômatos que interagem com o sistema com um propósito.

Os casos de usos são utilizados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado.

Um caso de uso envolve a interação de um ou mais atores com o sistema. Os atores podem ser humanos ou sistemas automatizados.

Um caso de uso é uma descrição de um conjunto de seqüências de ações, inclusive variantes que ocorrem em caso de erro, que um sistema executa para produzir um resultado de valor observável por um ator (BOOCH, RUMBAUGH E JACOBSON: 2000, P. 220).

Na figura 01, temos exemplo de um diagrama de caso de uso, em seguida a descrição de alto nível e o mesmo caso uso de forma expandida:

Figura: 01 – Exemplo de caso de uso de alto nível

O caso de uso de alto nível, mostra de forma geral como é feita a interação dos atores com o sistema.

Na representação do caso de uso expandido, é visualizada toda a seqüência de eventos necessários para completar o processo.

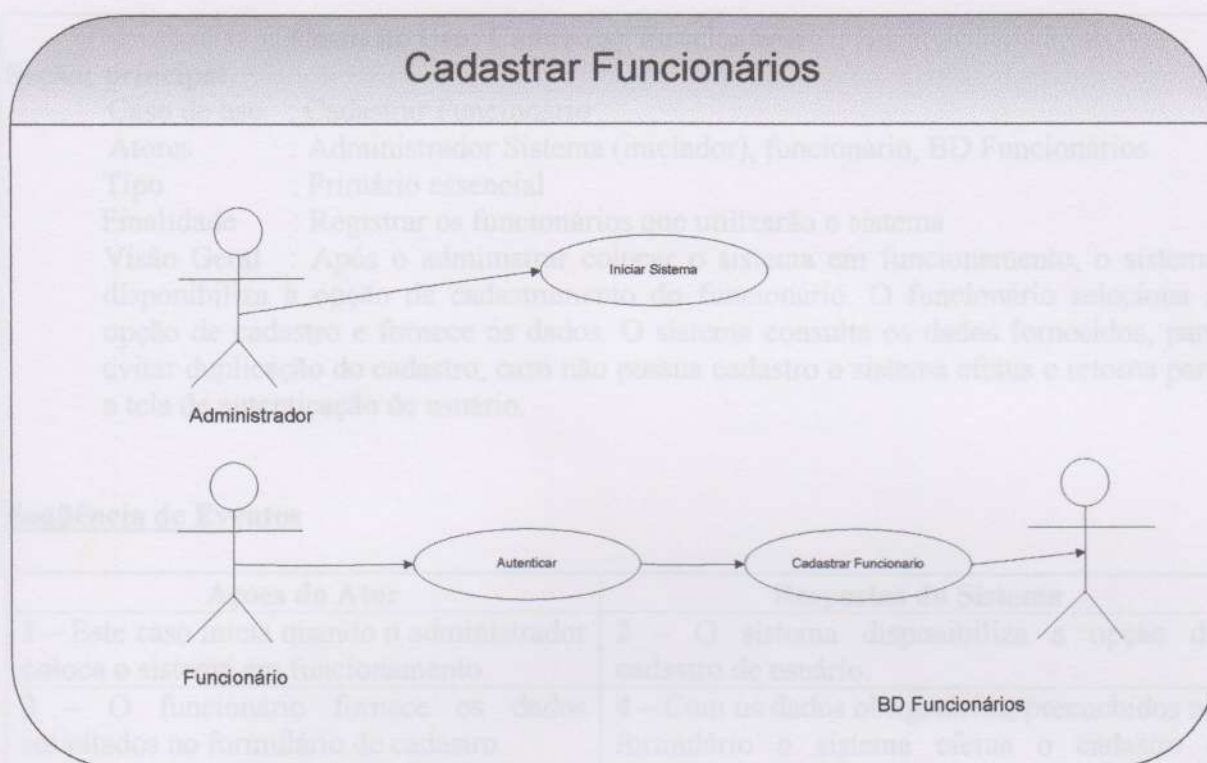


Figura 01 – Exemplo de diagrama de caso de uso

No diagrama de casos de uso na figura 01, temos a interação do ator (funcionário) com o sistema e a interação com o banco de dados do sistema, representado pelo ator (BD funcionários).

Caso de uso : **Cadastrar Funcionário**

Atores : Administrador (Iniciador), Funcionário.

Tipo : Primário

Descrição : Após o administrador colocar o sistema em funcionamento, o sistema disponibiliza a opção de cadastramento do funcionário. O funcionário seleciona a opção de cadastro e fornece os dados. O sistema consulta os dados fornecidos, para evitar duplicação do cadastro, caso não possua cadastro o sistema efetua e retorna para a tela de autenticação de usuário.

Figura. 02 – Exemplo de caso de uso de alto nível

O caso de uso de alto nível, mostra de forma geral como é feita a interação dos atores com o sistema.

Na representação do caso de uso expandido, é visualizada toda a sequência de eventos necessários para completar o processo.

Casos de Uso: Cadastrar Funcionário	
Seção: principal	
Caso de uso	: Cadastrar Funcionário
Atores	: Administrador Sistema (iniciador), funcionário, BD Funcionários
Tipo	: Primário essencial
Finalidade	: Registrar os funcionários que utilizarão o sistema
Visão Geral	: Após o administrar colocar o sistema em funcionamento, o sistema disponibiliza a opção de cadastramento do funcionário. O funcionário seleciona a opção de cadastro e fornece os dados. O sistema consulta os dados fornecidos, para evitar duplicação do cadastro, caso não possua cadastro o sistema efetua e retorna para a tela de autenticação de usuário.
<u>Seqüência de Eventos</u>	
Ações do Ator	Respostas do Sistema
1 – Este caso inicia quando o administrador coloca o sistema em funcionamento.	2 – O sistema disponibiliza a opção de cadastro de usuário.
3 – O funcionário fornece os dados solicitados no formulário de cadastro.	4 – Com os dados obrigatórios, preenchidos no formulário o sistema efetua o cadastro e retorna a tela de autenticação.
Seqüências alternativas de Eventos:	
Passo 4 → O funcionário já possui cadastro, o sistema emite mensagem informando: usuário cadastrado.	

Figura 03 – Exemplo de caso de uso expandido

2.6.2. Diagrama de Colaboração

Na construção do diagrama de colaboração são mostrados os objetos e os seus relacionamentos e as trocas de mensagens entre os objetos (vide figura 04).

Uma colaboração é uma sociedade de classes, interfaces e outros elementos que trabalham em conjunto para fornecer algum comportamento cooperativo maior do que a soma de todas as suas partes (BOOCH, RUMBAUGH E JACOBSON: 2000, P. 366).

Para Matos (2002, p.62) “O diagrama de colaboração fixa a atenção em como os objetos estão se organizando para efetuar uma tarefa. É como se uma cadeia de ajuda fosse efetivada com a intenção de que um desejo fosse alcançado”.

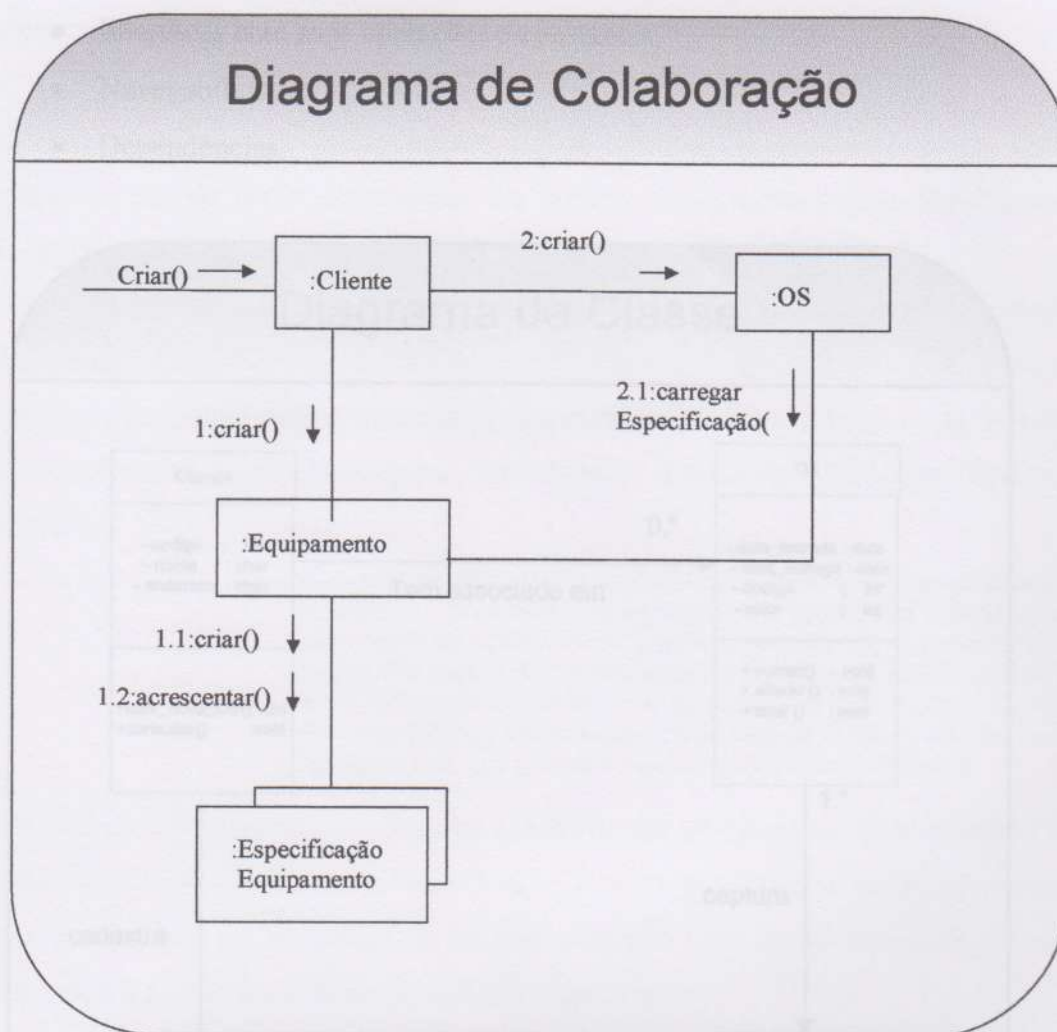


Figura 04 Exemplo de um diagrama de colaboração – Cadastrar ordem de serviço.

2.6.3. Diagrama de Classes

Os diagramas de classes são utilizados para dar uma visão estática do sistema, possibilitando suporte aos requisitos funcionais de um sistema, os serviços que o sistema deverá fornecer aos usuários finais.

Um diagrama de classe é um diagrama que mostra conjunto de classes, interfaces e seus relacionamentos. (BOOCH, RUMBAUGH E JACOBSON: 2000, P. 105).

De acordo com Larman, um diagrama de classes ilustra as especificações para as classes de software e de interfaces de uma aplicação. As informações típicas de um diagrama incluem:

- Métodos;
- Classes, associações e atributos;

- Interfaces com suas operações e constantes;
- Navegabilidade;
- Dependências.

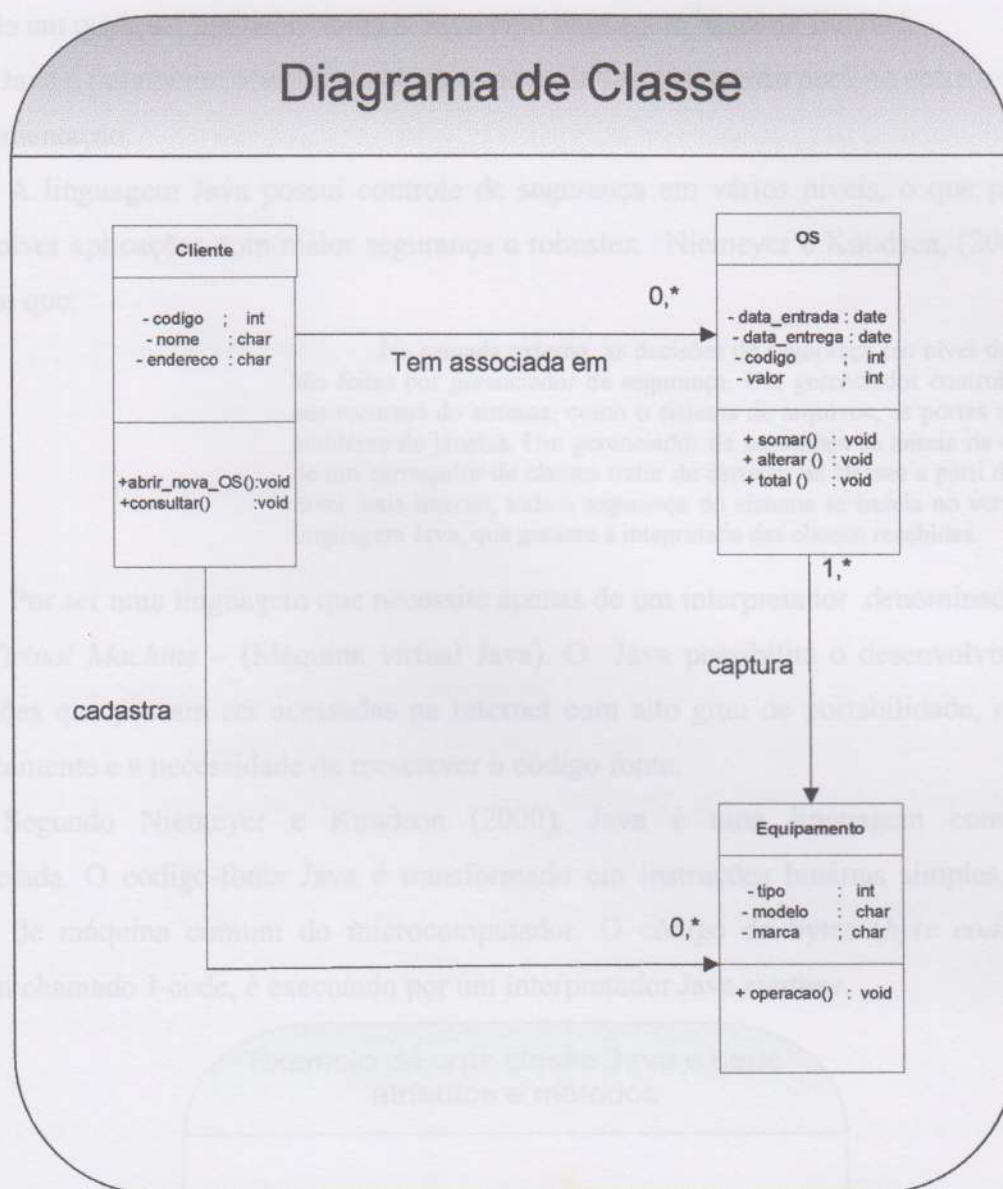


Figura 05 – Exemplo de diagrama de classes

No exemplo acima, figura 05, podem ser vistos as classes (Cliente, Os, Equipamento) com seus atributos (`-codigo`, `-data_entrada`, `-modelo`, etc), métodos (`+abrir_nova_OS()`, `+somar()`, `+operacao()`), e os seus relacionamentos (cadastra, captura).

Figura 06 – Exemplo de uma classe Java

2.7. Java

A linguagem Java é independente de plataforma de hardware e de sistemas operacionais. O fato de poder desenvolver um sistema em que seu código fonte possa ser utilizado em qualquer ambiente torna a Java uma linguagem bastante flexível.

Java é puramente orientada a objetos tornando um casamento perfeito entre o modelo e a implementação.

A linguagem Java possui controle de segurança em vários níveis, o que possibilita desenvolver aplicações com maior segurança e robustez. Niemeyer e Knudsen, (2000 p. 17) afirmam que:

Na camada externa, as decisões de segurança em nível de aplicação são feitas por gerenciador de segurança. Um gerenciador controla o acesso aos recursos do sistema, como o sistema de arquivos, as portas de rede e o ambiente de janelas. Um gerenciador de segurança se baseia na capacidade de um carregador de classes tratar de carregar as classes a partir da rede. No nível mais interno, toda a segurança do sistema se baseia no verificador da linguagem Java, que garante a integridade das classes recebidas.

Por ser uma linguagem que necessite apenas de um interpretador denominado JVM – *Java Virtual Machine* – (Maquina virtual Java). O Java possibilita o desenvolvimento de aplicações que possam ser acessadas na Internet com alto grau de portabilidade, reduzindo significativamente e a necessidade de reescrever o código fonte.

Segundo Niemeyer e Knudson (2000), Java é uma linguagem compilada e interpretada. O código-fonte Java é transformado em instruções binárias simples, como o código de máquina comum do microcomputador. O código de bytes (*byte code*) Java, também chamado J-code, é executado por um interpretador Java *runtime*.

Exemplo de uma classe Java e seus atributos e métodos

```

public class Automovel {
    public Automovel() {
        String modelo, cor;
        int ano;
        boolean estadoMotor = false;

        public void ligarMotor() {
            estadoMotor = true;
        }

        public String getModelo() {
            return modelo;
        }
    }
}

```

Atributos

Métodos

Figura 06 – Exemplo de uma classe Java

2.7.1. Java Servlet

O servlet nada mais é que uma classe Java que adiciona recursos de funcionalidade de um servidor web, possibilitando página de conteúdo dinâmico para interação com os clientes através do modelo de solicitação e resposta.

Um Servlet Java é uma classe Java que executa no lado do servidor que é ativado pelo servidor Web, para atender pedidos HTTP. Os Servlets são classes Java normais, traduzidos para *bytecode* independentes de plataforma e, por essa razão, são automaticamente portáteis para qualquer máquina na qual a linguagem Java esteja disponível. (NIRVA, SOLOMON e BASU.2001, P. 37)

Cacho (2003, p.22) afirma que:

Java Servlet é uma versão para servidor dos applets cliente, executados em um servidor web, sendo melhorias para substituir o antigo Comum Gateway Interface (CGI), uma das primeiras extensões para web simples, que tornavam possíveis a criação de conteúdo HTML dinâmico.

2.7.2. Java Server Pages (JSP)

O *Java Server Pages*, ou simplesmente JSP como é conhecido, é uma tecnologia desenvolvida para a Internet baseada na linguagem Java, que possibilita o uso da lógica das classes Java para implementação de conteúdo dinâmico nas páginas com a utilização de *tags* especiais.

Com o uso dessas *tags* é possível implementar códigos Java diretamente em páginas estáticas em HTML, tornando-as dinâmicas, o que possibilita uma interatividade com os usuários ao acessar a página web.

Conforme Cattel apud Cacho (2003, p.16), *Java Server pages* são baseados em servlets com o intuito de simplificar o desenvolvimento de conteúdo dinâmico na web. O JSP é um arquivo HTML, com chamadas às classes codificada em linguagem de programação para mostrar a parte dinâmica da página ou também código Java dentro de *tags* especiais no HTML ou XML.

2.8. SQL

SQL (*Structured Query Language* – Linguagem de Consulta Estruturada) é uma linguagem de consulta amigável em banco de dados, que foi criada pela IBM nos anos 70.

A versão original foi desenvolvida no laboratório de pesquisa da IBM San Jose (hoje o Centro de Pesquisa Almaden). Esta linguagem originalmente chamada Sequel foi implementada como parte do projeto System R, no início dos anos 70. Em 1986, o American National Standard Institute (ANSI) publicou um padrão SQL. A IBM publicou seu próprio padrão SQL, o *Systems Application Architecture Database Interface* (ROCHA NETO, 2001 p. 19).

A SQL possui diversos comandos que possibilitam realizar varias operações da álgebra relacional oriundos de instruções básicas como: INSERT UPDATE, DELETE e SELECT.

INSERT é responsável pela inserção de dados no banco de dados.

UPDATE é a instrução que atualiza os valores dos campos no banco de dados.

DELETE elimina, exclui informações do banco de dados.

SELECT é responsável pela busca de dados no banco de dados.

2.9. PostgreSQL

As informações manipuladas em um sistema necessitam estar armazenada em um local que possibilite facilidade de acesso e confiabilidade, pensando nisso para a implantação do Sistema de Controle de Ordem de Serviço - SCOS, foi o escolhido o banco de dados *POSTGRES*, por disponibilizar todas característica de funcionabilidade e segurança e por ter sido desenvolvido com a filosofia de software livre. O que representa uma significativa economia para a empresa, uma vez que não terá investimento em licenciamento.

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (SGBDOR), desenvolvido no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. O PostgreSQL oferece recursos adicionais pela incorporação dos conceitos de herança, tipos de dados e funções, tornando possível os usuários estenderem o sistema facilmente, além de possuir outras s funcionalidades que fornecem poder e flexibilidade adicionais tais como: restrições, gatilhos, regras, integridade da transação. Estas funcionalidades colocam o PostgreSQL dentro da categoria de banco de dados referida como objeto-relacional (TUTORIAL, 2005).

2.10. Eclipse

Para o desenvolvimento de aplicações com a tecnologia Java, existe diversos software para ambiente de desenvolvimento integrado (IDE - *integrated development environment*), alguns com custos elevados para o licenciamento e outros gratuitos, como é o caso do Eclipse.

A escolha do Eclipse se deu, por apresentar características que facilitam o desenvolvimento tais como:

- Oferecer amplos e inovativos recursos de produtividade, geração de código e muitos outros auxílios, atalhos e automação de desenvolvimento e escrita de código.
- Consistir em um projeto sério e ativo, bem organizado e coordenado, além do amplo apoio da comunidade e de grandes empresas e instituições.
- Possuir uma arquitetura de software aberta e extensível, permitindo que *plug-ins* sejam criados e facilmente integrados ao Eclipse; existe uma grande variedade de *plug-ins* disponíveis, gratuitos e comerciais, para adicionar e ampliar toda gama de características e recursos ao IDE.
- Ser software gratuito, livre e de código aberto.

2.11. Macromedia Dreamweaver

O Macromedia Dreamweaver é um editor de HTML profissional que projeta e gerencia sites e páginas da Web visualmente, facilitando o início do trabalho e fornecendo ferramentas de ajuda para aprimorar a experiência em projetos para páginas da Web. Inclui várias ferramentas e recursos para uso com HTML e JavaScript, um depurador JavaScript e editores de código, que possibilitam editar JavaScript, XML, JSP e outros documentos de texto diretamente no Dreamweaver.

Os recursos de edição visual do Dreamweaver também permitem adicionar rapidamente desenhos e funcionalidades às páginas, sem ter que codificar uma linha sequer.

2.12. Apache Tomcat DO SISTEMA

Para o uso de sistemas desenvolvidos em JSP, faz-se necessário o uso de um servidor de aplicações JAVA para Web. O servidor escolhido para o uso do SCOS é o Tomcat, por ser um servidor caracterizado como software livre e apresentar uma relação de performance e segurança bastante satisfatória.

O Tomcat é um servidor de aplicações Java para Web. É software livre e de código aberto desenvolvido dentro do conceituado projeto Jakarta e oficialmente endossado pela Sun como a Implementação de Referência (RI) para as tecnologias Java Servlet e Java Server Pages (JSP). O Tomcat é robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção (TUTORIAL, 2003).

- Criação e remoção de contas de funcionários;
- Criação e remoção de contas de clientes;
- Cadastro e remoção de equipamentos - acompanhamento dos equipamentos que deram entrada na oficina, podendo gerar estatística de atendimento por tipo de equipamento;
- Abertura e acompanhamento de Ordem de Serviço;
- Controle e listagem de produção das Ordens de Serviço atendidas por técnicos;
- Cadastro e remoção de peças utilizadas no conserto dos equipamentos;
- Gerar orçamento para o cliente - para cada ordem de serviços, o sistema gera um orçamento em função das peças utilizadas e serviços executados;
- Listagem das ordens de serviços atendidas - visualização na tela ou impressão da ordem de serviços;
- Listagem de mala direta do cadastro de clientes.

3.2. Usuários do Sistema

- Administrador do Sistema - Responsável por criar, alterar, consultar e apagar contas dos funcionários que utilizarão o sistema, além da manutenção e suporte, tendo total controle para manipulação do sistema.

3. MODELAGEM DO SISTEMA

3.1. Objetivo Geral

O sistema tem como objetivo a automatização das rotinas de controle de abertura de Ordem de Serviço da empresa TECLYBRA Informática Ltda., bem como o acompanhamento até a entrega do equipamento ao cliente que solicitou a execução de serviço de conserto e manutenção.

3.2. Objetivos Específicos:

- Criação e remoção de contas de funcionários;
- Criação e remoção de contas de clientes;
- Cadastro e remoção de equipamentos - acompanhamento dos equipamentos que deram entrada na oficina, podendo gerar estatística de atendimento por tipo de equipamento;
- Abertura e acompanhamento de Ordem de Serviço;
- Controle e listagem de produção das Ordens de Serviços atendidas por técnicos;
- Cadastro e remoção de peças utilizadas no conserto dos equipamentos;
- Gerar orçamento para o cliente - para cada ordem de serviços, o sistema gera um orçamento, em função das peças utilizadas e serviços executados;
- Listagem das ordens de serviços atendidas – visualização na tela ou impressão da ordem de serviços;
- Listagem de mala direta do cadastro de clientes.

3.3. Usuários do Sistema

- **Administrador do Sistema** - Responsável por criar, alterar, consultar e apagar contas dos funcionários que utilizarão o sistema, além da manutenção e suporte, tendo total controle para manipulação do sistema.

- **Recepcionistas** - Responsáveis pelo cadastro, alteração dos dados dos clientes, cadastro com emissão de recibo de entrega na abertura das ordens de serviço e emissão da garantia quando finalizada.
- **Técnicos** - Responsáveis por cadastrar peças, serviços e gerar orçamento da ordem de serviço, podendo alterar o status da ordem de serviço, em função dos serviços realizados na OS.

3.4. Funções do Sistema

Ref#	Função	Categoria
R1.1	Cadastrar usuários	Evidente
R1.2	Alterar dados dos usuários	Evidente
R1.3	Excluir usuários	Evidente
R1.4	Listar clientes	Evidente
R2.1	Cadastrar OS	Evidente
R2.2	Alterar dados OS	Evidente
R2.3	Imprimir OS	Evidente
R2.4	Fechar OS	Evidente
R2.5	Emitir comprovante de entrega OS	Evidente
R2.6	Aprovar orçamento	Evidente
R2.7	Cadastrar tipo OS	Evidente
R2.8	Alterar tipo OS	Evidente
R2.9	Excluir tipo OS	Evidente
R3.1	Cadastrar Técnico	Evidente
R3.2	Alterar dados dos técnicos	Evidente
R3.3	Excluir Técnico	Evidente
R4.1	Cadastrar Produtos/Serviços	Evidente
R4.2	Alterar produtos/serviços	Evidente
R4.3	Excluir produtos/serviços	Evidente
R4.2	Alterar produtos/serviços	Evidente
R5.1	Emitir extrato de cliente	Evidente
R5.2	Emitir extrato produção técnico	Evidente
R5.3	Emitir Relatório caixa diário	Evidente
R5.4	Cadastrar Orçamento	Evidente
R5.5	Prover mecanismo de armazenamento persistente	Oculto

3.5. Atributos do Sistema

Atributo	Detalhes e Restrições de Limite
Metáfora de Interface	Gráfica e Colorida(HTML) Formulários e Caixas de dialogo estilo WEB Maximizar a navegação por dispositivo apontador e facilidade com uso do teclado.
Plataforma de Sistema Operacional	Sistema Operacional com recursos de Servidor, utilizando o Container Tomcat versão 5.x e nas estações qualquer Sistema Operacional com um navegador (browser), com acesso a Internet.

3.6. Arquitetura do Sistema

A arquitetura utilizada para o sistema é baseada no modelo de três camadas, (figura 07) por ser um modelo que favorece as atualizações das regras do negócio, uma vez que a lógica do negócio fica no servidor.

Todo o acesso do cliente ao banco de dados é realizado de acordo com as regras contidas no servidor de aplicações. O cliente não tem acesso direto ao banco de dados, sem antes passar pelo servidor de aplicações. As três camadas são:

- **Camada de Apresentação** – é a parte que fica do lado cliente, acessada pelo navegador. Quando há necessidade de alterações na Interface do programa, é necessário apenas que seja reinicializado o servidor Web onde está armazenada as paginas, que sofreram modificações, não havendo necessidade de instalação no lado do cliente. Porém cabe ressaltar, que alterações na interface, são menos freqüentes do que alterações nas regras do negócio.
- **Camada de Negócios** - camada que contém os componentes de software que implementam as regras do negócio, as quais determinam de que maneira os dados serão processados. Esta camada foi deslocada para o servidor de aplicações. Desta maneira, quando uma regra do negócio for alterada, basta atualizá-la no servidor de aplicações. Após a atualização, todos os usuários passarão a ter acesso a nova versão, sem que seja necessário reinstalar o programa em cada um dos computadores da rede. Ao centralizar as regras do negócio em um servidor de aplicações, estamos facilitando a tarefa de manter a aplicação atualizada.

- **Camada de Persistência** - na camada de persistência temos, os componentes de software que interagem com o servidor de banco de dados, onde ficam armazenadas as informações necessárias para o funcionamento da aplicação. Essas informações são acessadas somente pelos componentes executados no servidor de aplicação e não diretamente pelo cliente.

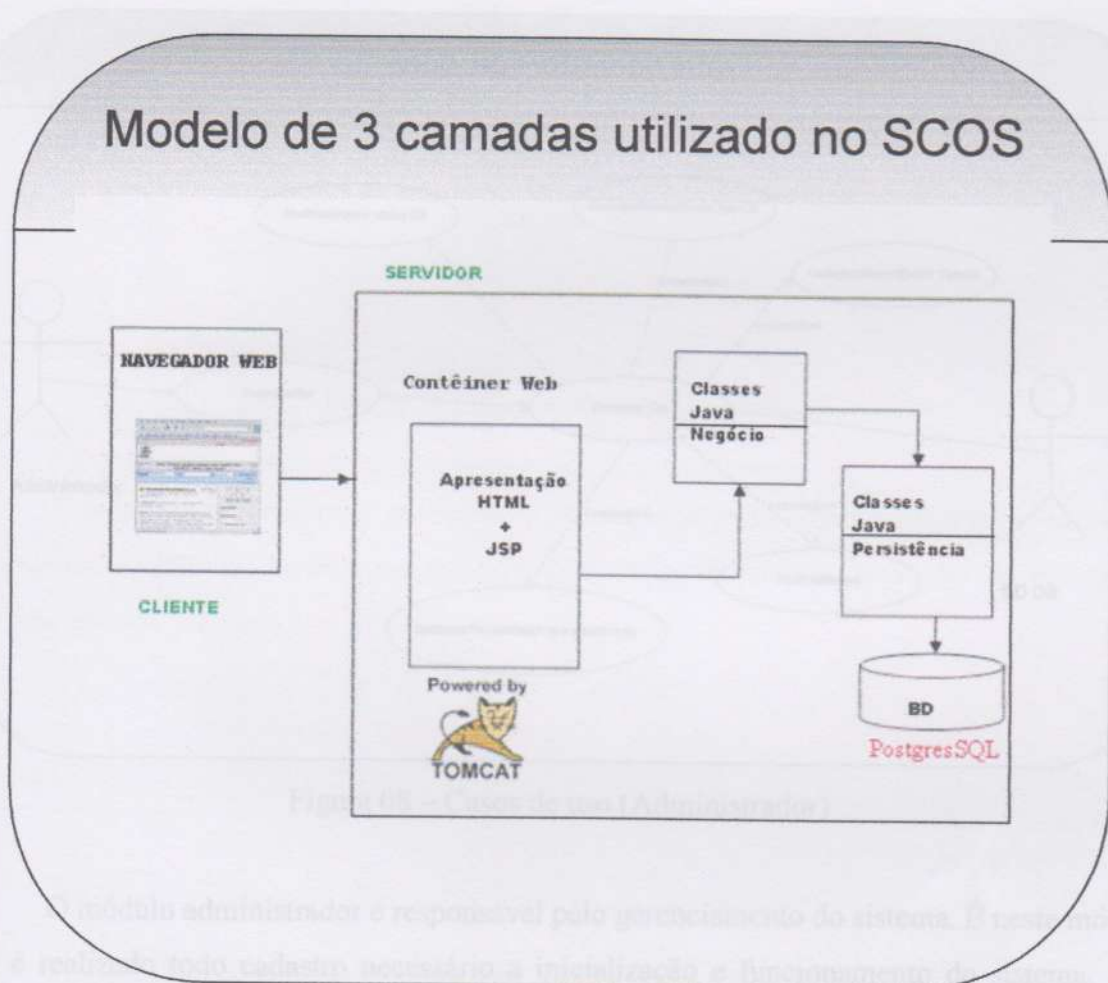


Figura 07 – Diagrama 3 camadas utilizada no desenvolvimento do SCOS

3.7. CASOS DE USO

O sistema de Ordem de serviços possui vários módulos, identificados pelos diagramas de casos de uso, como mostrados nas figuras 08, 09 e 10, tendo as descrições de cada módulos e seus aspectos funcionais modelados como segue:

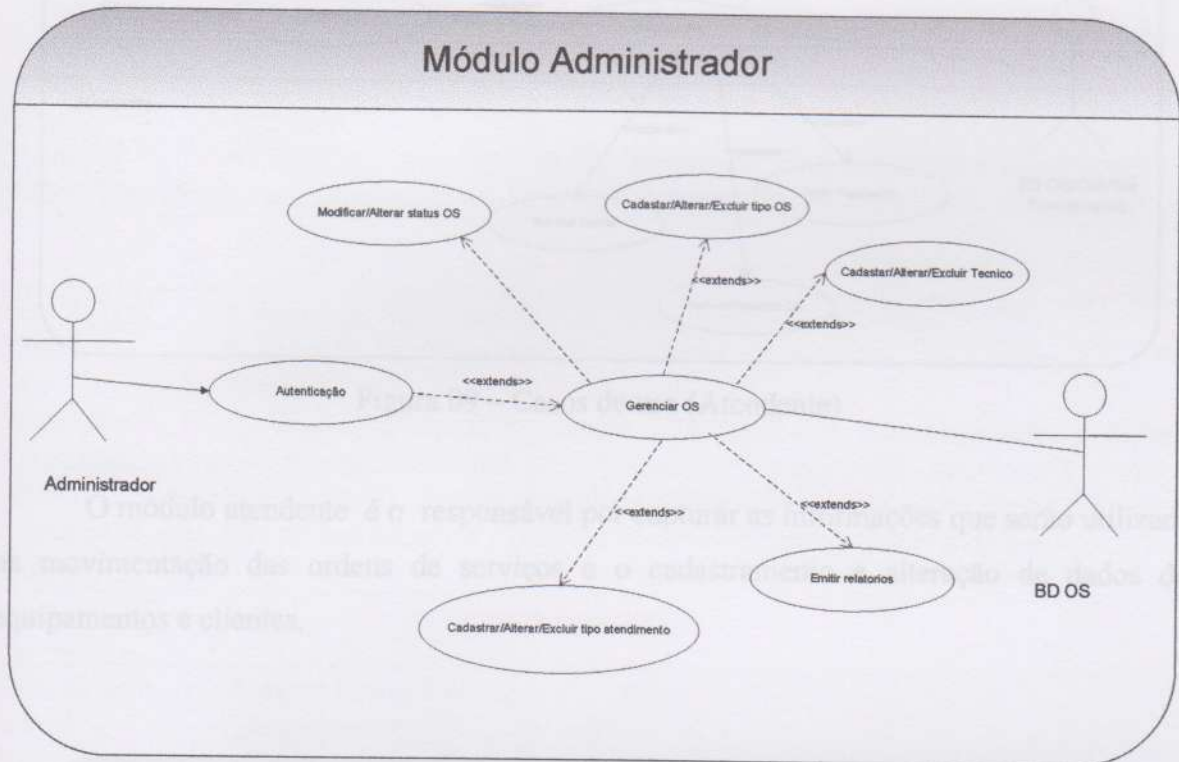


Figura 08 – Casos de uso (Administrador)

O módulo administrador é responsável pelo gerenciamento do sistema. É neste módulo que é realizado todo cadastro necessário a inicialização e funcionamento do sistema, bem como alterações dos parâmetros de configuração.

Figura 10 – Diagrama de casos de uso (Técnico)

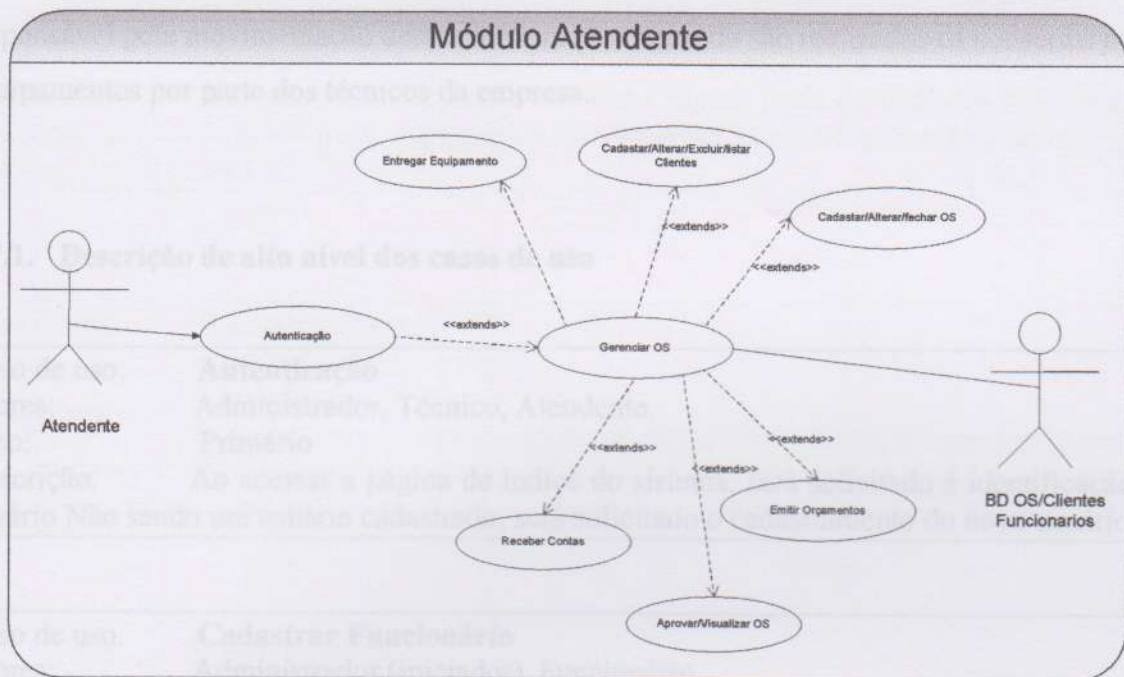


Figura 09 – Casos de uso (Atendente)

O módulo atendente é o responsável por capturar as informações que serão utilizadas na movimentação das ordens de serviços e o cadastramento e alteração de dados dos equipamentos e clientes.

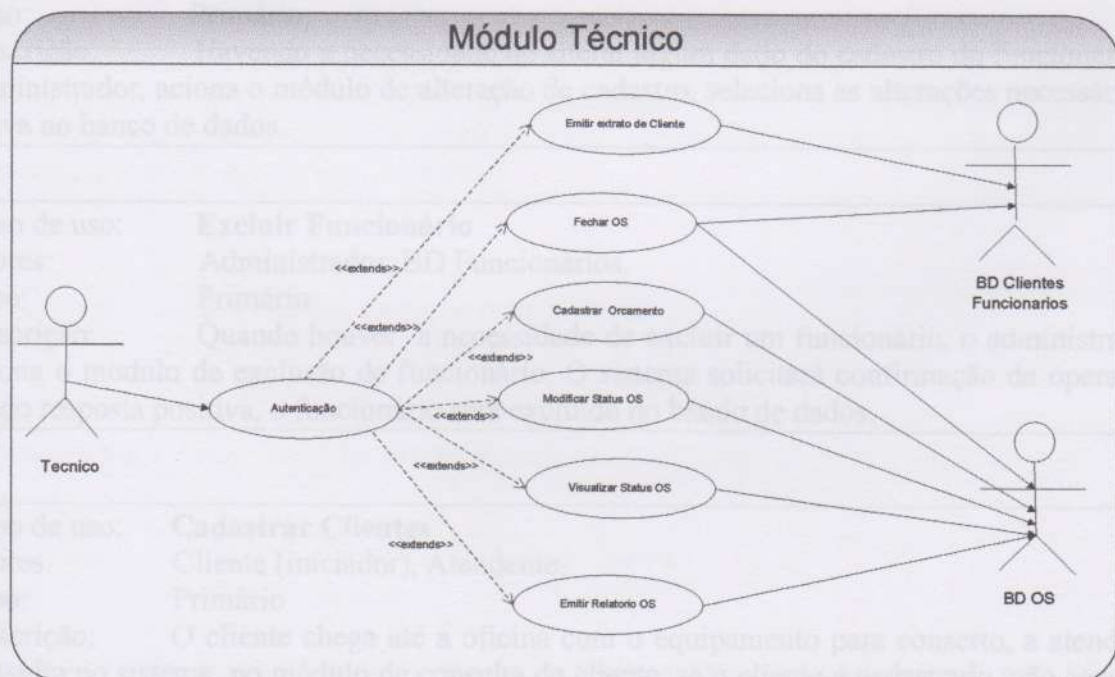


Figura 10 – Diagrama de casos de uso (Técnico)

Uma vez que as informações foram cadastradas nos módulos anteriores, este módulo é responsável pela movimentação dessas informações, quando são realizados os consertos nos equipamentos por parte dos técnicos da empresa.

3.7.1. Descrição de alto nível dos casos de uso

Caso de uso:	Autenticação
Atores:	Administrador, Técnico, Atendente.
Tipo:	Primário
Descrição:	Ao acessar a página de índice do sistema, será solicitado a identificação do usuário. Não sendo um usuário cadastrado, será solicitado o cadastramento do novo usuário.

Caso de uso:	Cadastrar Funcionário
Atores:	Administrador (iniciador), Funcionário.
Tipo:	Primário
Descrição:	Após o administrador colocar o sistema em funcionamento, o sistema disponibiliza a opção de cadastramento de funcionário. O funcionário seleciona a opção de cadastro e fornece os dados. O sistema consulta os dados fornecidos, para evitar duplicação do cadastro, caso não possua cadastro o sistema efetua e retorna para a tela de autenticação de usuário.

Caso de uso:	Alterar dados Funcionário
Atores:	Administrador, BD Funcionários.
Tipo:	Primário
Descrição:	Havendo a necessidade de alterar algum dado do cadastro do funcionário, o administrador, aciona o módulo de alteração de cadastro, seleciona as alterações necessárias e grava no banco de dados.

Caso de uso:	Excluir Funcionário
Atores:	Administrador, BD Funcionários.
Tipo:	Primário
Descrição:	Quando houver a necessidade de excluir um funcionário, o administrador, aciona o módulo de exclusão de funcionário. O sistema solicitará confirmação da operação, tendo resposta positiva, o funcionário será excluído do bando de dados.

Caso de uso:	Cadastrar Clientes
Atores:	Cliente (iniciador), Atendente.
Tipo:	Primário
Descrição:	O cliente chega até a oficina com o equipamento para conserto, a atendente consulta no sistema, no módulo de consulta de cliente, se o cliente é cadastrado, não sendo, a atendente solicita os dados do cliente e cadastra.

Caso de uso: **Alterar dados Cliente**
Atores: Atendente, BD Clientes.
Tipo: Primário
Descrição: Havendo a necessidade de alterar algum dado do cadastro de clientes, a atendente, aciona o módulo de alteração de cadastro, seleciona as alterações necessárias e grava no banco de dados.

Caso de uso: **Excluir Cliente**
Atores: Atendente, BD Clientes.
Tipo: Primário
Descrição: Quando houver a necessidade de excluir um cliente, a atendente, aciona o módulo de exclusão de cliente. O sistema solicitará confirmação da operação e autenticação do administrador, tendo resposta positiva, o cliente será excluído do banco de dados.

Caso de uso: **Listar Clientes**
Atores: Atendente, BD Clientes.
Tipo: Primário
Descrição: Quando houver a necessidade de uma visualização da listagem dos clientes cadastrados, a atendente, aciona o módulo de listagem clientes. O sistema pergunta, qual a forma de visualização: na tela ou na impressora. De acordo com a opção escolhida o sistema emite a listagem.

Caso de uso: **Cadastrar Ordem de Serviço**
Atores: Cliente (iniciador), Atendente, BD OS.
Tipo: Primário
Descrição: O cliente chega ate a oficina com o equipamento para conserto, a atendente recupera os dados do cliente no sistema, acionando o modulo de cadastro de OS, e informa ao sistema todos os dados interessados no equipamento o qual será consertada e a devida reclamação fornecida pelo cliente. Após os dados serem armazenados, o sistema emitir um comprovante de entrega do equipamento.

Caso de uso: **Alterar dados da Ordem de Serviço**
Atores: Administrador, BD OS.
Tipo: Primário
Descrição: Caso haja necessidade de alterar uma informação na ordem de serviço, o administrador, aciona o módulo de alteração da OS, e informa os dados a serem alterados.

Caso de uso: **Visualizar status da Ordem de Serviço**
Atores: Atendente, técnico.
Tipo: Primário
Descrição: Após usuário ser autenticado, o sistema permite a visualização do status da OS, sendo necessário, o usuário selecionar uma OS cadastrada no sistema.

Caso de uso: **Cadastrar orçamento**
Atores: Técnico, BD OS.
Tipo: Primário
Descrição: Uma vez que o técnico identificou o defeito e relacionou as peças e serviços necessários ao conserto, ele seleciona a opção de cadastrar orçamento no sistema, após sua autenticação, e cadastrar as peças e serviços.

Caso de uso: **Alterar orçamento**
Atores: Técnico, BD OS.
Tipo: Primário
Descrição: Após o orçamento está concluído, só é possível ser alterado pelo técnico quando houver uma alteração nas peças utilizada ou adição de um novo serviço a ser executado. Para isso o técnico seleciona a opção de alteração de orçamento no sistema e altera/acrescenta o serviço ou peça.

Caso de uso: **Emitir orçamento**
Atores: Atendente, BD OS.
Tipo: Primário
Descrição: Após o orçamento está concluído a atendente imprimir o orçamento e repassa para o cliente através de e-mail ou de fax, selecionando a opção de emitir orçamento do sistema.

Caso de uso: **Aprovar Ordem de Serviço**
Atores: Atendente, BD OS.
Tipo: Primário
Descrição: Após o orçamento está concluído a atendente entra em contato com o cliente e informa/envia o orçamento. Tendo uma resposta positiva do cliente, a atendente seleciona no sistema a opção de aprovar OS. O sistema altera o status da OS, de EM SERVIÇO, para APROVADA.

Caso de uso: **Fechar Ordem de Serviço**
Atores: Atendente, técnico. BD OS
Tipo: Primário
Descrição: Após a OS ter sido aprovada pelo cliente, e o técnico concluído o reparo no equipamento, o usuário (técnico ou atendente) seleciona a opção de fechamento da ordem de serviço. O sistema muda o status, removendo o equipamento da lista de serviço.

Caso de uso: **Recebimento de contas**

Atores: Atendente, BD OS

Tipo: Primário

Descrição: Quando o cliente é informado que o equipamento já está pronto e disponível para entrega e comparece a recepção para o acerto de contas, a atendente seleciona a opção de recebimentos de contas e seleciona a ordem de serviço em questão, atualizado o status para OS PAGA.

Caso de uso: **Entregar Equipamento**

Atores: Atendente, técnico, BD OS

Tipo: Primário

Descrição: Uma vez que foi feito o recebimento dos valores existente na ordem de serviço, pagos pelo cliente então a atendente seleciona no sistema a opção de entrega de equipamento e seleciona a OS quitada e altera o status para ENTREGUE, com isso não mais sendo permitido alteração na OS. O sistema em seguida emite comprovante de entrega e da garantia do serviço.

Caso de uso: **Emitir extrato do cliente**

Atores: Atendente, técnico, BD OS, Clientes.

Tipo: Primário

Descrição: O sistema permitir a emissão em tela ou na impressora, do extrato de movimento do cliente, mostrando todo o historio do(s) equipamento(s) que foi realizado conserto na oficina. Para isso os usuários selecionam o módulo de emissão de extrato e seleciona o cliente a ser gerado o extrato.

Caso de uso: **Emitir relatório de ordem de serviço**

Atores: Atendente, técnico, BD OS

Tipo: Primário

Descrição: O sistema permite a emissão na tela ou na impressora da relação de todas ordens de serviços, ordenadas por status ou por sequência. Podendo selecionar a impressão de um determinado status. Para isso os usuários selecionam o módulo de emissão de relatório e seleciona o a opção desejada.

Caso de uso: **Cadastrar tipo de ordem de serviço e tipo de atendimento**

Atores: Administrador, BD OS.

Tipo: Primário

Descrição: Para que o sistema possa disponibilizar, no cadastro de ordem de serviço, os tipos de ordem de serviço, é necessário o cadastro prévio, para isso o administrador seleciona o módulo de cadastro de tipos de OS e adiciona novos tipos (O sistema vem previamente cadastrado os seguintes tipos: INTERNA, A DOMICILIO, GARANTIA, AVULSO, CONTRATO).

Caso de uso: **Alterar tipo de ordem de serviço e tipo de atendimento**

Atores: Administrador, BD OS.

Tipo: Primário

Descrição: Quando houver a necessidade de fazer alteração no tipo de atendimento ou tipo de ordem de serviço, o administrador ativa o módulo de alteração de tipos de OS seleciona a opção de alterar, modifica e os dados e ao finalizar o sistema pede confirmação da alteração, tendo resposta positiva o sistema atualiza o banco.

Caso de uso: **Excluir tipo de ordem de serviço e tipo de atendimento**

Atores: Administrador, BD OS.

Tipo: Primário

Descrição: Quando houver a necessidade de excluir um tipo de atendimento ou tipo um de ordem de serviço, o administrador ativa o módulo de exclusão de tipos de OS, seleciona o tipo a ser excluído. O sistema pede confirmação da exclusão, tendo resposta positiva o sistema exclui o tipo do banco de dados.

Caso de uso: **Emitir caixa diário**

Atores: Administrador, BD OS.

Tipo: Primário

Descrição: O sistema permite a emissão, na tela ou na impressora, da listagem do caixa, mostrando todo o movimento de ordens de serviços quitadas durante o dia. Procedimento realizado sempre ao final do dia pelo administrador.

Caso de uso: **Prover mecanismo de persistência de dados**

Atores: Administrador, BD OS.

Tipo: Primário

Descrição: O sistema permite o agendamento de um horário para que inicie de forma automática uma copia do banco de dados.

Figura 11 – Detalhe do Caso de uso – Cadastrar Funcionários

3.7.2. Casos de uso expandidos

Na expansão dos casos de usos, destacam-se:

- Caso de uso Cadastro funcionários
- Caso de uso Cadastro de Ordem de Serviço
- Caso de uso Cadastro de Orçamento

3.7.3. Caso de uso - Cadastro de funcionários

É um caso de uso primário e essencial, pois se trata dos profissionais que mantêm o sistema alimentado com todas as atividades existentes na oficina,

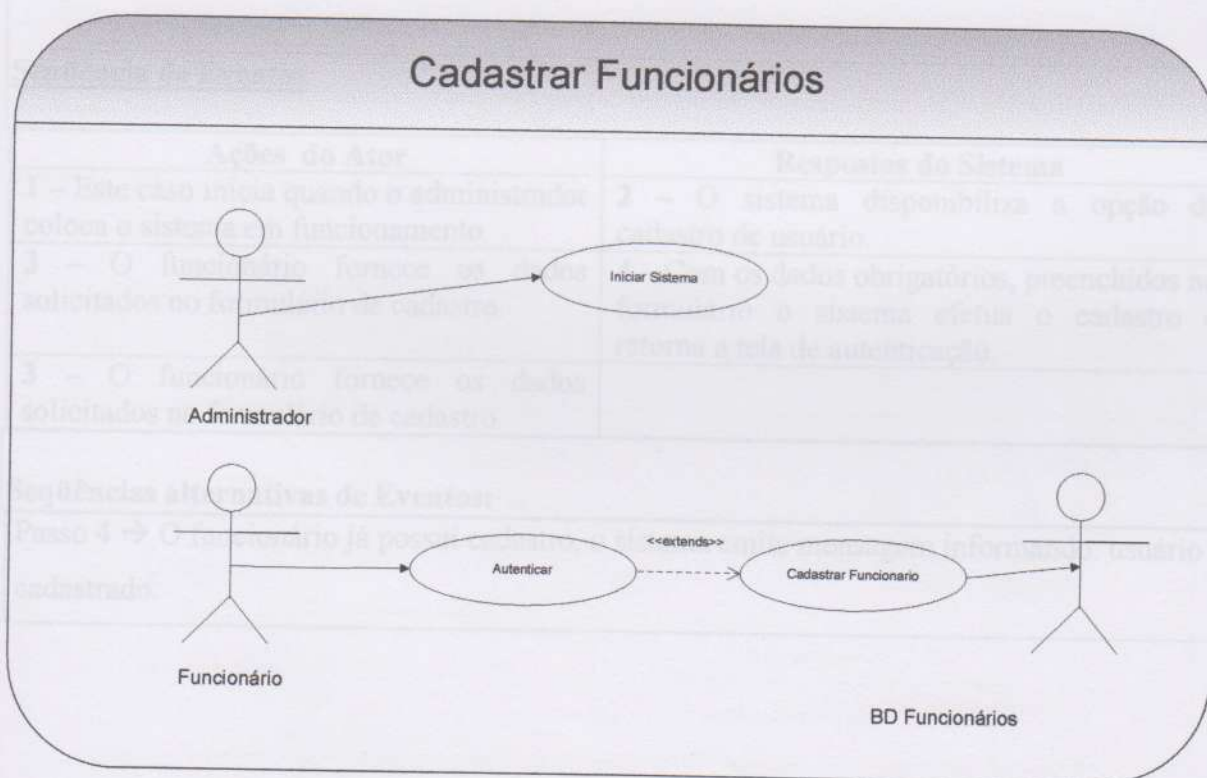


Figura 11 – Detalhe do Caso de uso – Cadastrar Funcionários

3.7.3.1. Expansão do caso de uso

O caso de uso Cadastro Funcionário possui a seguinte função:

Permitir a inclusão de registros (funcionários) que irão utilizar o sistema;

Casos de Uso: Cadastrar Funcionário

Seção: principal

Caso de uso : Cadastrar Funcionário
 Atores : Administrador Sistema (iniciador), funcionário, BD Funcionários
 Tipo : Primário essencial
 Finalidade : Registrar os funcionários que utilizarão o sistema
 Visão Geral : Após o administrar colocar o sistema em funcionamento, o sistema disponibiliza a opção de cadastramento do funcionário. O funcionário seleciona a opção de cadastro e fornece os dados. O sistema consulta os dados fornecidos, para evitar duplicação do cadastro, caso não possua cadastro o sistema efetua e retorna para a tela de autenticação de usuário.

Figura 12 - Modelo de domínio - caso de uso cadastrar funcionário

Seqüência de Eventos

Ações do Ator	Respostas do Sistema
1 – Este caso inicia quando o administrador coloca o sistema em funcionamento.	2 – O sistema disponibiliza a opção de cadastro de usuário.
3 – O funcionário fornece os dados solicitados no formulário de cadastro.	4 – Com os dados obrigatórios, preenchidos no formulário o sistema efetua o cadastro e retorna a tela de autenticação.
3 – O funcionário fornece os dados solicitados no formulário de cadastro.	

Seqüências alternativas de Eventos:

Passo 4 → O funcionário já possui cadastro, o sistema emite mensagem informando: usuário cadastrado.

Figura 13 - Diagrama de seqüência - Cadastrar Funcionário

3.7.3.2. Modelo de domínio

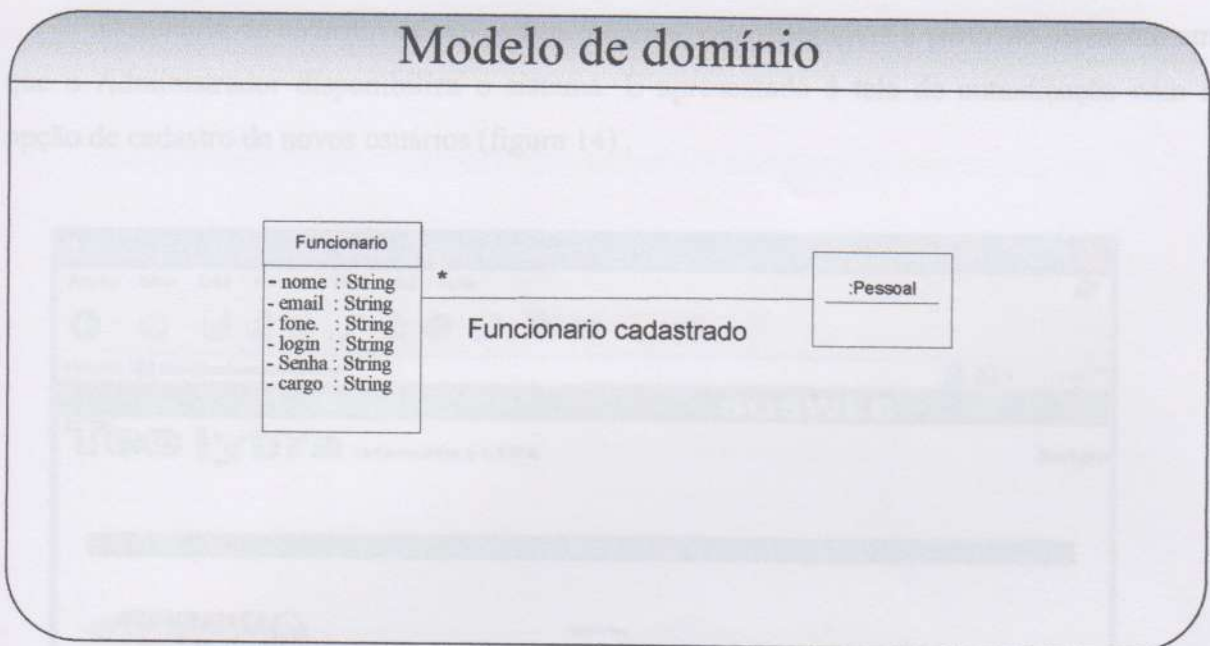


Figura 12 – Modelo de domínio – caso de uso cadastrar funcionário

3.7.3.3. Diagrama de Sequência.

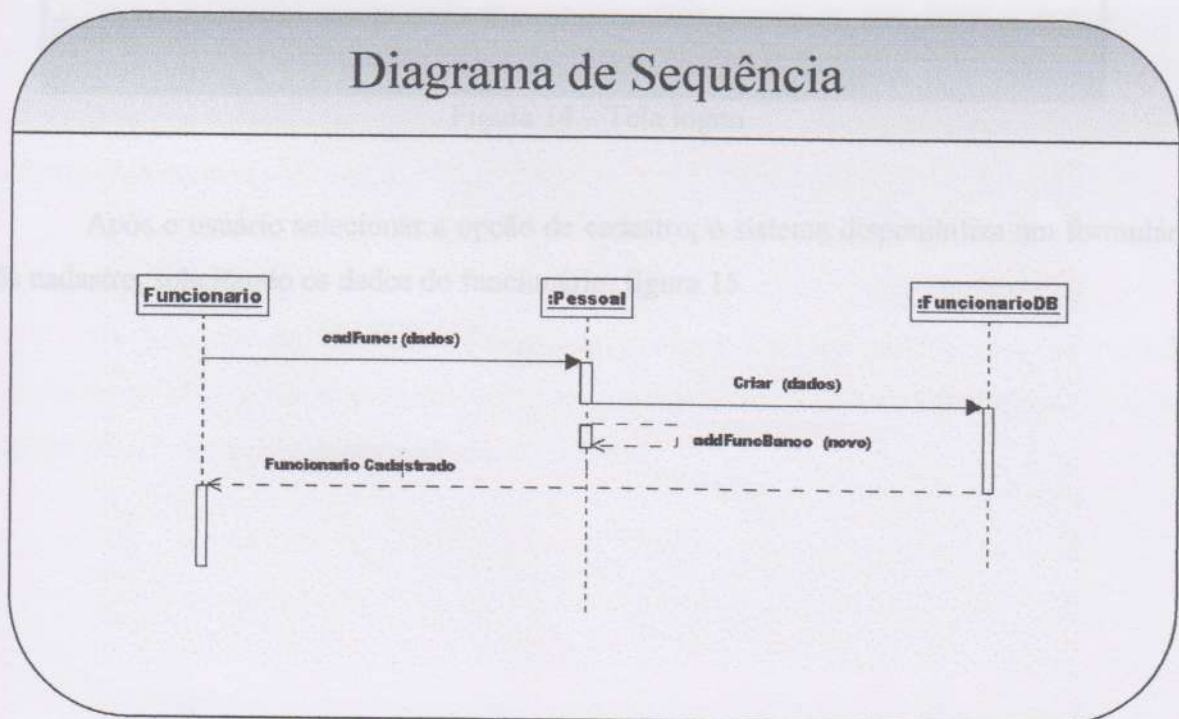


Figura 13 – Diagrama de sequência - Cadastrar funcionário

3.7.3.4. Detalhes de implementação

O módulo de cadastro de novos funcionários, está disponível a partir do momento em que o Administrador disponibiliza o sistema. É apresentado à tela de autenticação com a opção de cadastro de novos usuários (figura 14).

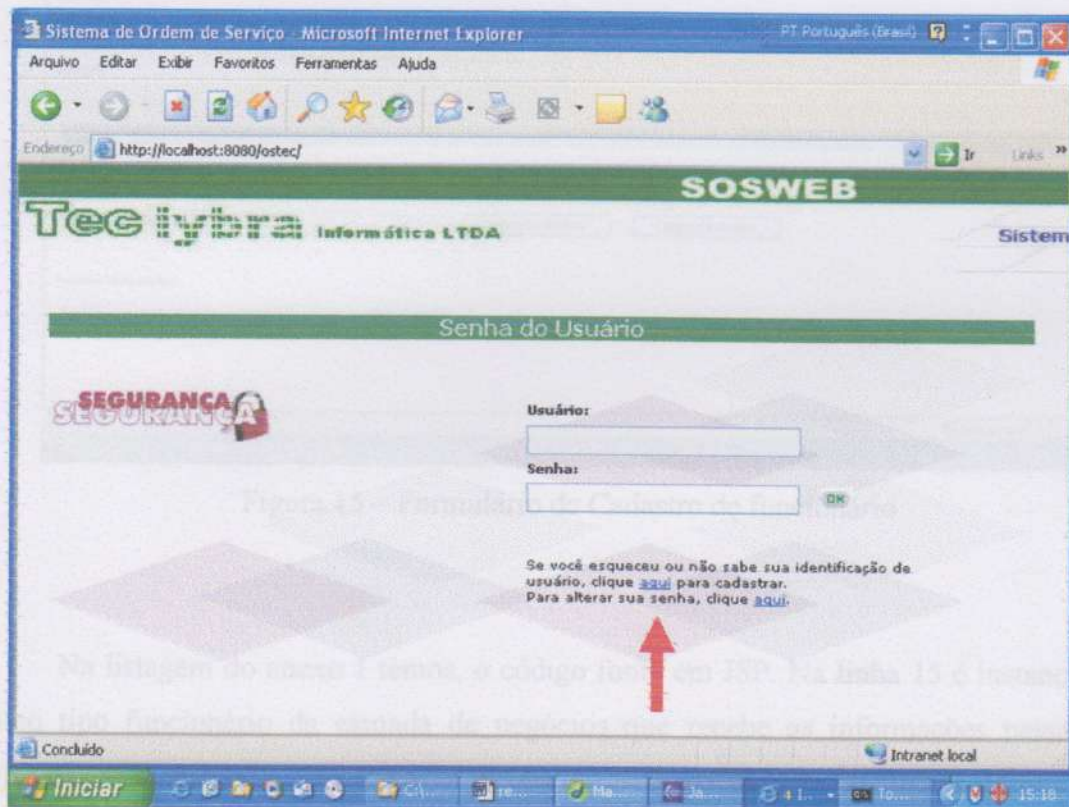


Figura 14 – Tela login

Após o usuário selecionar a opção de cadastro, o sistema disponibiliza um formulário de cadastro, solicitando os dados do funcionário, figura 15.

Sistema de Ordem de Serviço - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço http://localhost:8080/sosweb/

SOSWEB

Tec lybra Informática LTDA

Sistema Ordem de Serviço

Cadastrar Sair - Página Inicial

Nome: *
E-mail: *
Telefone:
Cargo:

Dados do Sistema

Login: *
Senha: *
Confirme a senha:

Envia os dados Limpa os dados

* Dados obrigatórios
Escolha uma senha fácil de ser lembrada, porém de difícil dedução

Concluído

Iniciar

Figura 15 – Formulário de Cadastro de funcionário

Na listagem do anexo 1 temos, o código fonte em JSP. Na linha 15 é instanciado um objeto tipo funcionário da camada de negócios que recebe as informações passada pelo SUBMIT da camada de interface, e na linha 23 é instanciado um novo objeto do tipo funcionário, da camada de dados, que possui um método responsável por armazenar as informações no banco de dados.

Na listagem do anexo 2, temos a definição da classe Funcionário com seus atributos e métodos.

Na listagem do anexo 3 é mostrada a classe FuncionarioDB, com os métodos responsáveis pelos acessos ao banco, seja para gravação, recuperação ou atualização de dados, representando a camada de persistência do sistema.

O caso de uso Cadastro de ordem de serviço possui a seguinte função: permitir a inclusão de registros (ordem de serviço) com todos os dados do equipamento e do cliente, que deu entrada na oficina, mantendo a associação dos técnicos que executaram os serviços de manutenção.

3.7.4. Caso de uso Cadastro de Ordem de Serviço

Este é o módulo mais abrangente do sistema. Ele é responsável pela consolidação de todas as principais funções do sistema.

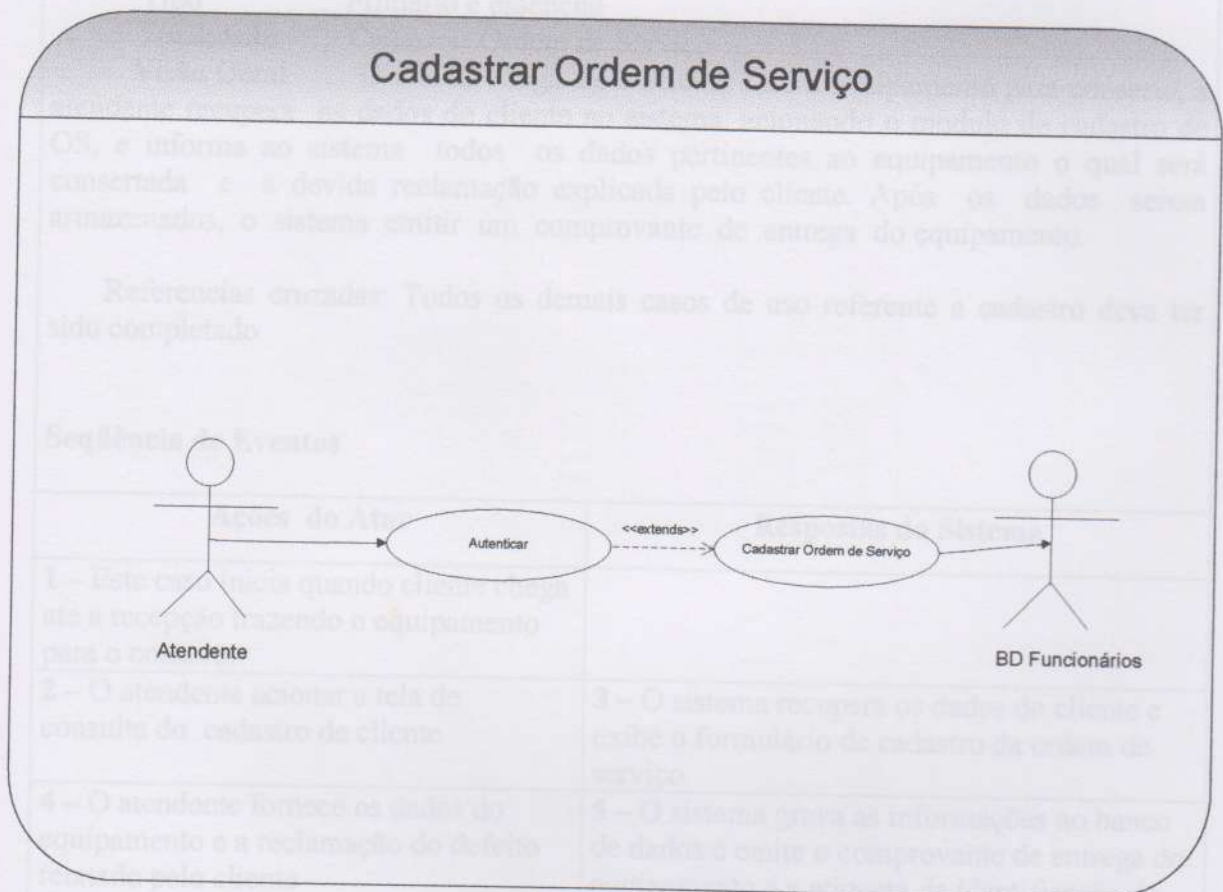


Figura 16 – Diagrama de caso de uso – Cadastrar ordem de serviço

3.7.4.1. Expansão do caso de uso

O caso de uso Cadastro de ordem de serviço possui a seguinte função: permitir a inclusão de registros (ordem de serviço) com todos os dados do equipamento e do cliente, que derem entrada na oficina, mantendo a associação dos técnicos que executaram os serviços de manutenção.

3.7.4.3. Modelo de domínio

Casos de Uso: Cadastrar Ordem de Serviço	
Seção Principal Caso de Uso : Cadastra Ordem de Serviço Atores : Cliente (iniciador), Atendente, BD OS. Tipo : Primário e essencial Finalidade : Cadastrar Ordem de Serviço Visão Geral : O cliente chega ate a oficina com o equipamento para conserto, a atendente recupera os dados do cliente no sistema, acionando o modulo de cadastro de OS, e informa ao sistema todos os dados pertinentes ao equipamento o qual será consertada e a devida reclamação explicada pelo cliente. Após os dados serem armazenados, o sistema emitir um comprovante de entrega do equipamento. Referencias cruzadas: Todos os demais casos de uso referente a cadastro deva ter sido completado	
Seqüência de Eventos	
Ações do Ator	Respostas do Sistema
1 – Este caso inicia quando cliente chega ate a recepção trazendo o equipamento para o conserto.	
2 – O atendente acionar a tela de consulta do cadastro de cliente.	3 – O sistema recupera os dados do cliente e exibe o formulário de cadastro da ordem de serviço.
4 – O atendente fornece os dados do equipamento e a reclamação do defeito relatado pelo cliente	5 – O sistema grava as informações no banco de dados e emite o comprovante de entrega do equipamento e a etiqueta de identificação do equipamento com o numero da ordem de serviço registrada para aquele equipamento.
Seqüências Alternativas de Eventos:	
Passo 3 → O cliente não possui cadastro então o sistema aciona o formulário de cadastro do cliente.	

3.7.4.2. Modelo de domínio

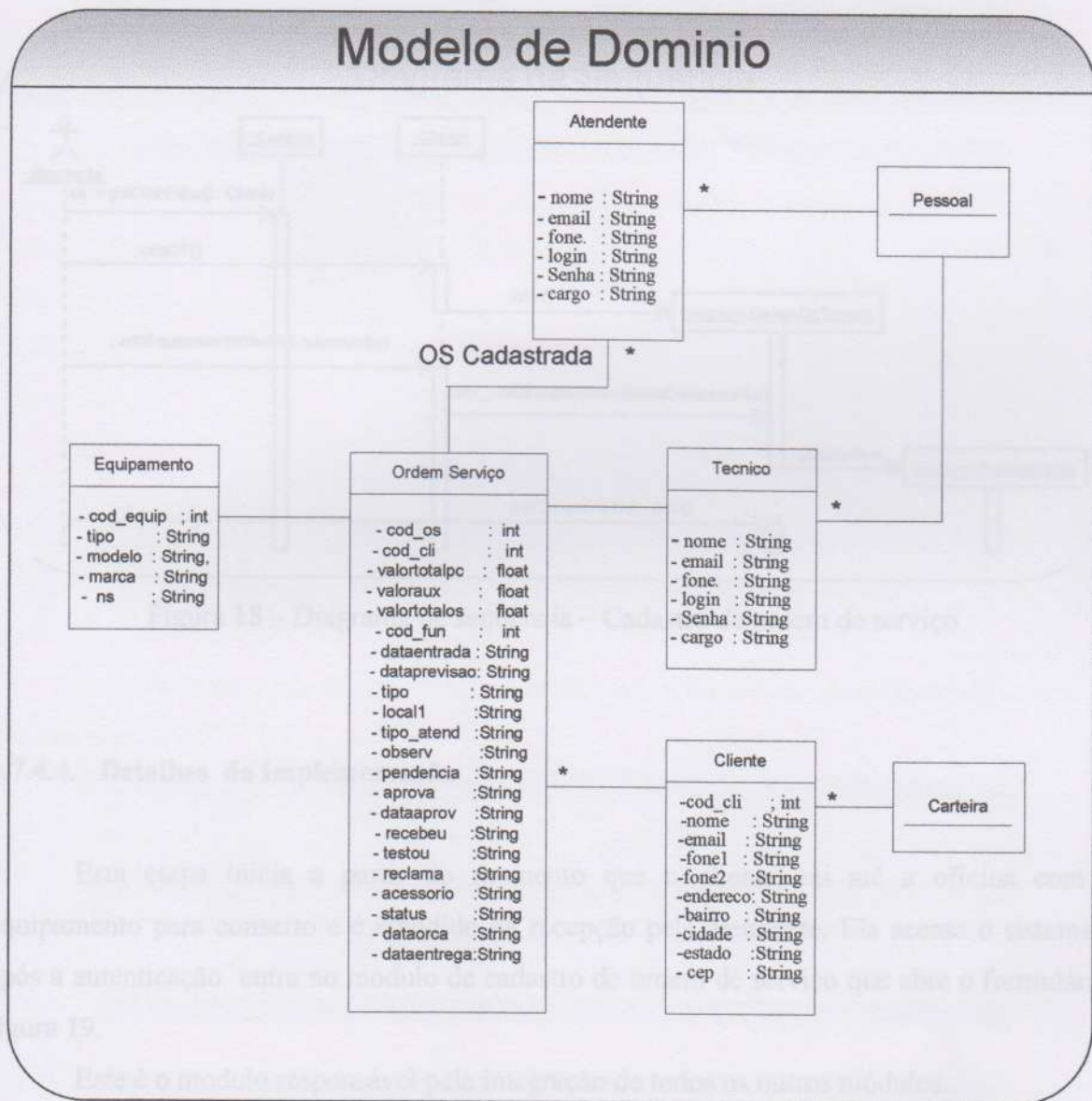


Figura 17 – Modelo de domínio caso de uso cadastrar OS

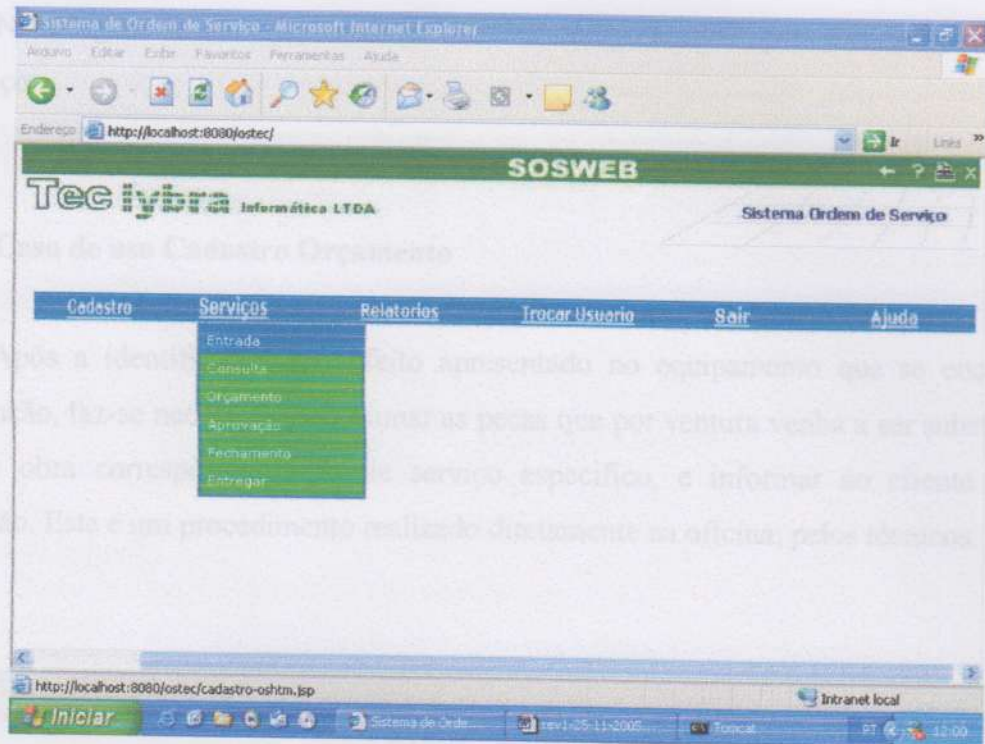


Figura 19 – Tela principal do cadastro de ordem de serviço

Ao selecionar a tela de cadastro, o sistema envia a formulário de cadastro de Ordem de Serviço, figura 20.

Figura 20 – Formulário de cadastro de ordem de serviço

3.7.5.1. Expansão do caso de uso

O caso de uso Cadastro de orçamento possui a seguinte função: Permitir a inclusão de peças e serviços necessários ao devido conserto dos equipamentos que se encontra em manutenção, como forma de quantificar valores que serão passados para o cliente tomador dos serviços e executá-los após sua aprovação.

Casos de Uso: Cadastrar Orçamento	
<u>Seção Principal</u> Caso de Uso : Cadastrar Orçamento Atores : Técnico, BD OS. Tipo : Primário e essencial Finalidade : Cadastrar Orçamento Visão Geral Uma vez que o técnico identificou o defeito e relacionou as peças e serviços necessários ao conserto, ele seleciona a opção de cadastrar orçamento no sistema, após sua autenticação, e cadastra as peças e serviços.	
<u>Seqüência de Eventos</u>	
Ações do Ator	Respostas do Sistema
1 – Este caso inicia quando o técnico identifica a(s) peça(s) danificada(s) e os serviços de mão de obra necessário ao conserto do equipamento.	
2 – O Técnico aciona o modulo de cadastro de orçamento e seleciona qual a ordem de serviço irá preencher o orçamento	3 – O sistema exibe um formulário solicitando quais as peças e mão de obra do orçamento.
4 – O técnico fornece as peças com os preços correspondentes para cada uma, bem como o valor da mão de obra e a descrição dos procedimentos realizado na execução do serviço.	5 – O sistema grava as informações no banco de dados e alterar o status da ordem de serviço, bem como gera um arquivo que será enviado ao cliente para aprovação do orçamento.
<u>Seqüências Alternativas de Eventos:</u>	
Passo 5 → As peças não estão ainda disponível na assistência. O sistema altera o status para <i>OS Pendente</i> , até as peças serem entregue pelo fornecedor.	

3.7.5.2. Modelo de domínio

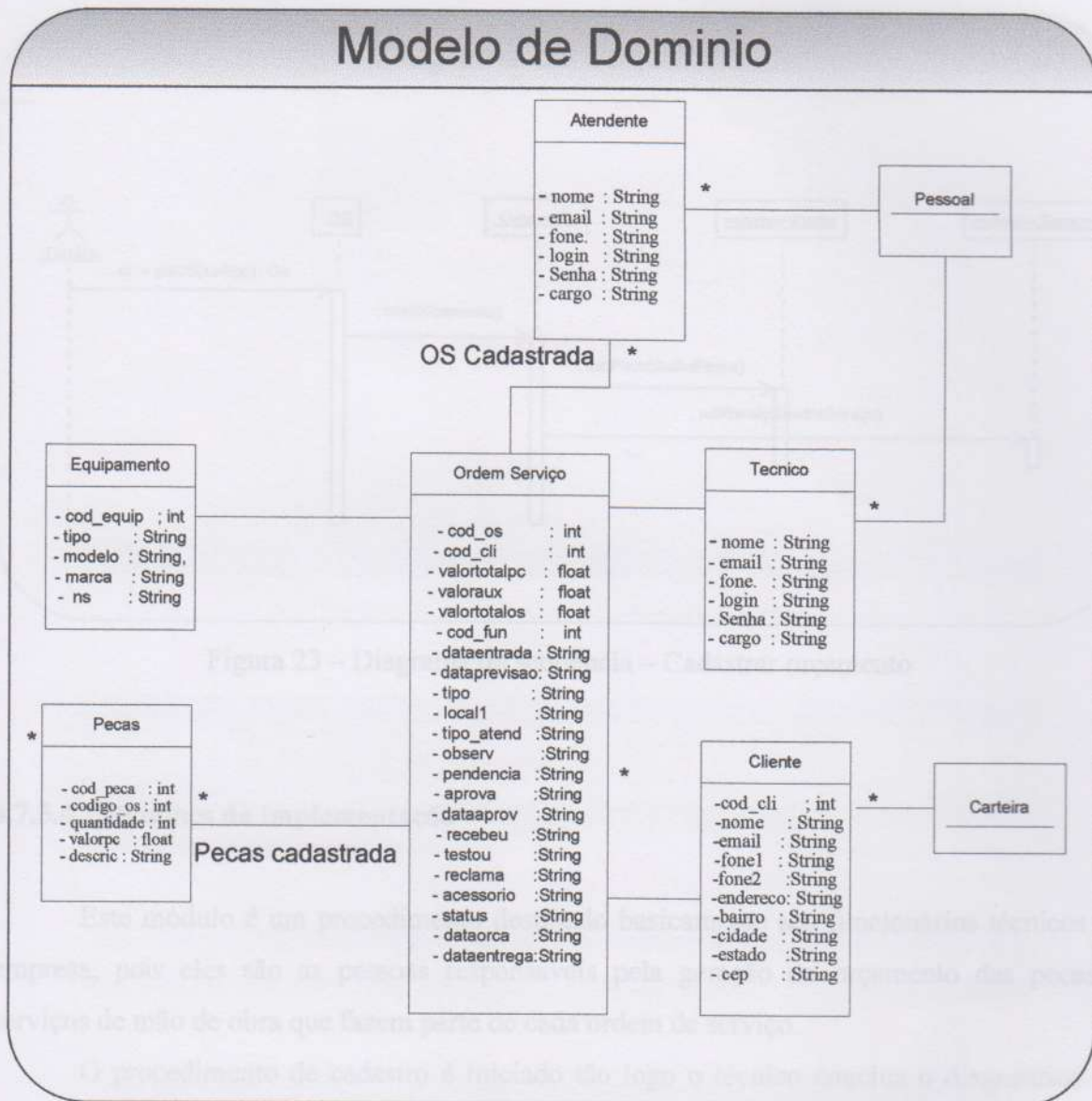


Figura 22 – Diagrama modelo domínio – cadastro orçamento

3.7.5.3. Diagrama de Sequência

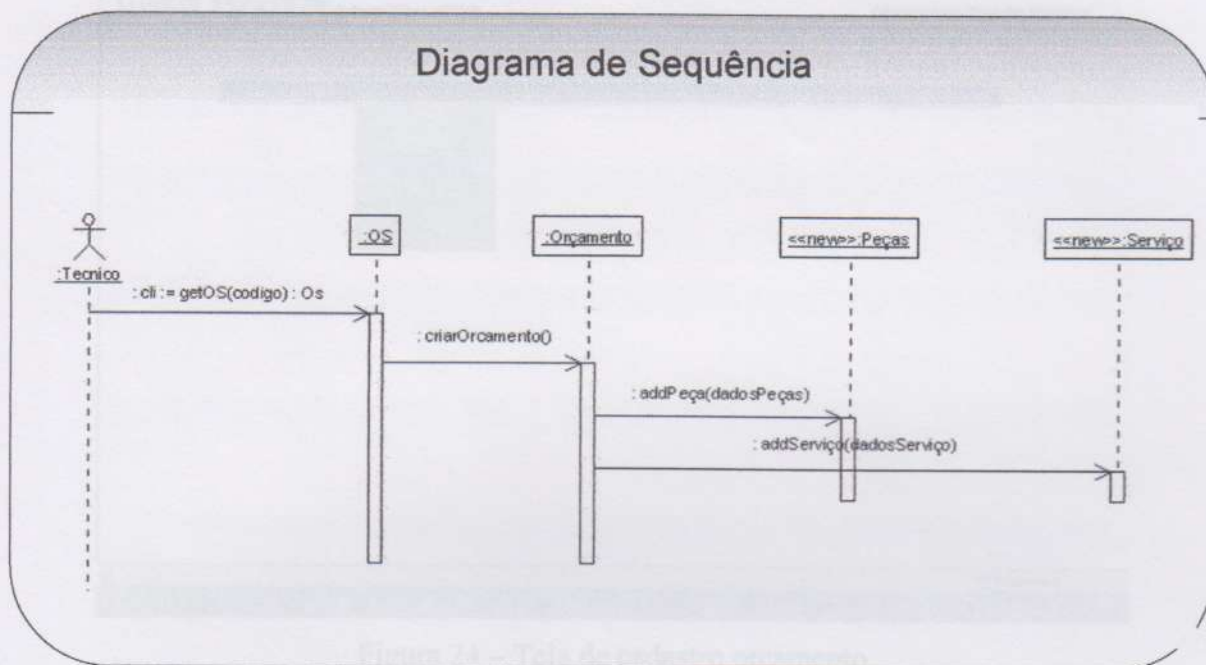


Figura 23 – Diagrama de sequência – Cadastrar orçamento

3.7.5.4. Detalhes da implementação

Este módulo é um procedimento destinado basicamente aos funcionários técnicos da empresa, pois eles são as pessoas responsáveis pela geração do orçamento das peças e serviços de mão de obra que fazem parte de cada ordem de serviço.

O procedimento de cadastro é iniciado tão logo o técnico conclua o diagnóstico do problema apresentado no equipamento em reparo.

O técnico após ser autenticado no sistema, seleciona o módulo de consulta de ordem de serviço, figura 24 e seleciona a OS a qual será gerado o orçamento, figura 25, em seguida é realizados o cadastro da relação de peças e mão de obra necessária ao conserto com seus respectivos valores, em um formulário, figura 26, que ao concluir o cadastro, o sistema atualiza os valores da ordem de serviço e alterar o status da ordem de serviço de *em serviço* para *orçamento*.

Figura 25 – Tela Seleção de OS disponível para orçamento

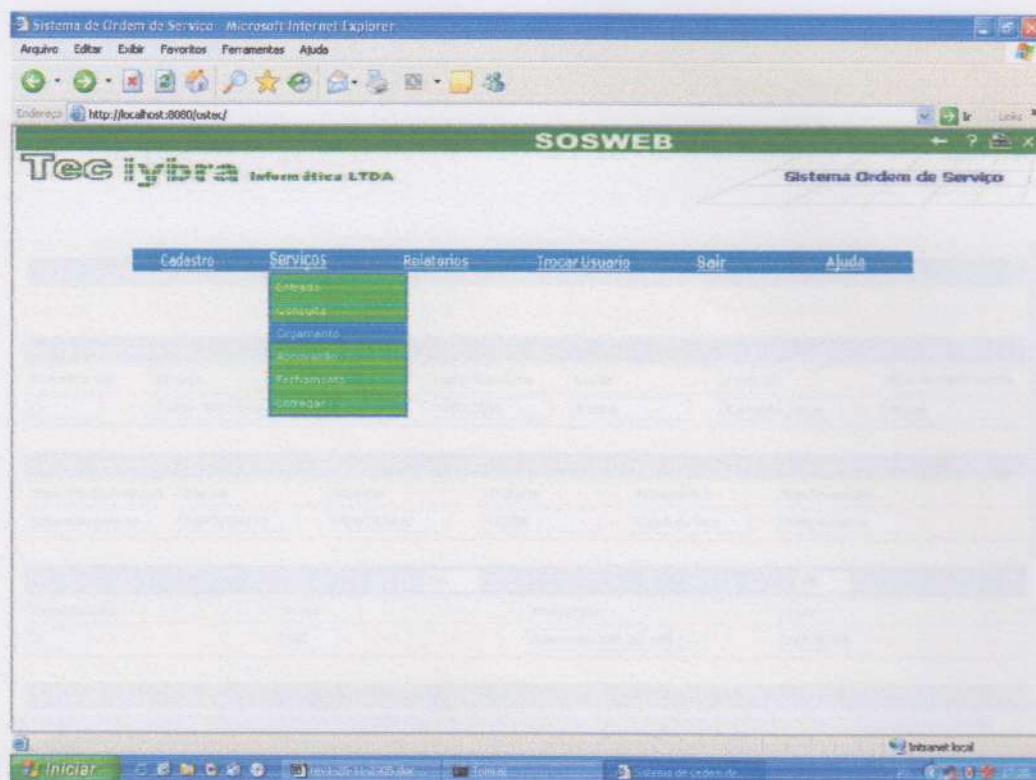


Figura 24 – Tela de cadastro orçamento

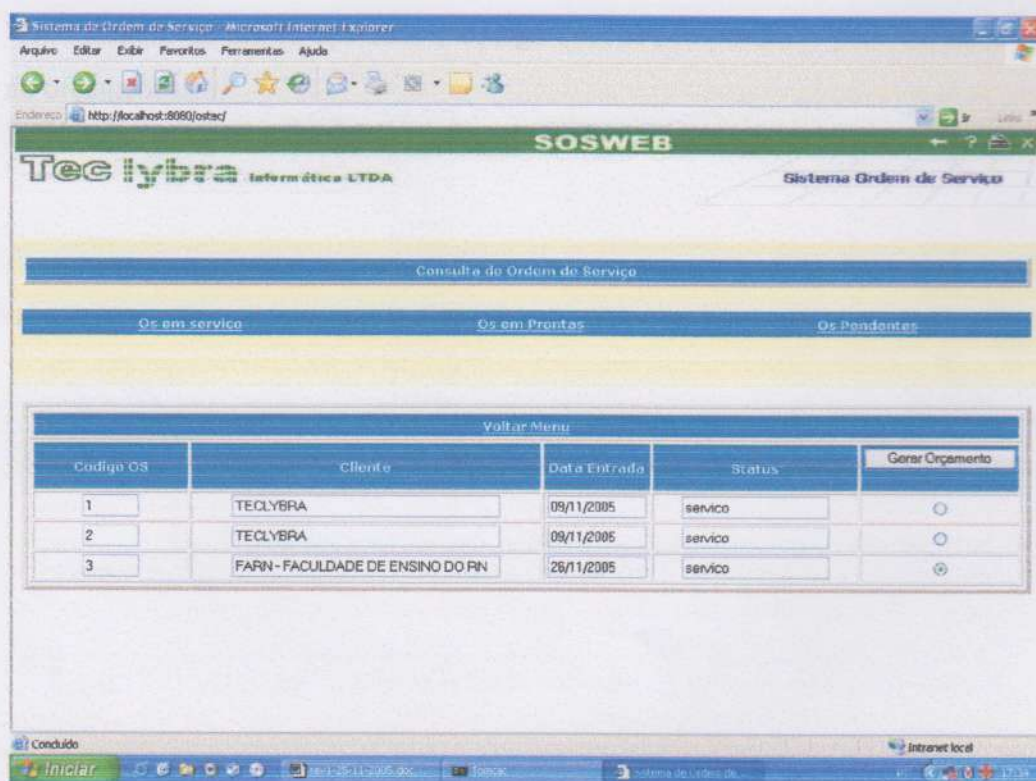


Figura 25 – Tela Seleção de OS disponível para orçamento

Sistema de Ordem de Serviço - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço: http://localhost:8080/sosweb/

SOSWEB

Tec lybra Informática LTDA Sistema Ordem de Serviço

Orçamento

Dados Ordem de Serviço

Numero OS	Cliente	Data Prevista	Local	TIPO OS	Tipo Atendimento
3	FARN - FACULDADE DE ENSINO DO RN	17/12/2005	Interna	Conserto_Pago	Avulso

Dados Equipamentos

Tipo Equipamento	Marca	Modelo	Nº Serie	Acessórios	Reclamação
Microcomputador	IMPORTADO	PENTIUM IV	123456	Caixa de Som	Não inicializa

Peças e Serviços

Valor total peças: 195 Valor total OS: 245

Quantidade	Valor	Descrição	Data
1	195	Memoria DDR 256 MB	26/11/2005

Cadastrar peças

Concluido

Iniciar

Figura 26 – Tela formulário cadastro orçamento

Na listagem do anexo 5, temos o código em JSP, responsável pelo cadastramento das peças no orçamento.

Figura 27: Diagrama entidade-relacionamento de base de dados do sistema.

3.7.6. Modelo de dados

O SGBD escolhido para o desenvolvimento deste sistema é o PostgreSQL, desenvolvido por *Global Development Group*, por ser um SGBD que permite ser utilizado na maioria dos sistemas operacionais existente atualmente e possuir o conceito de Software livre, o que permite redução de custo para a empresa em relação à aquisição de licenciamento. O Postgres é um SGBD com recurso de acesso ao banco relacional e permite grande facilidade de criação e manutenção do banco. Na figura 27 temos as tabelas do banco com o seus relacionamentos.

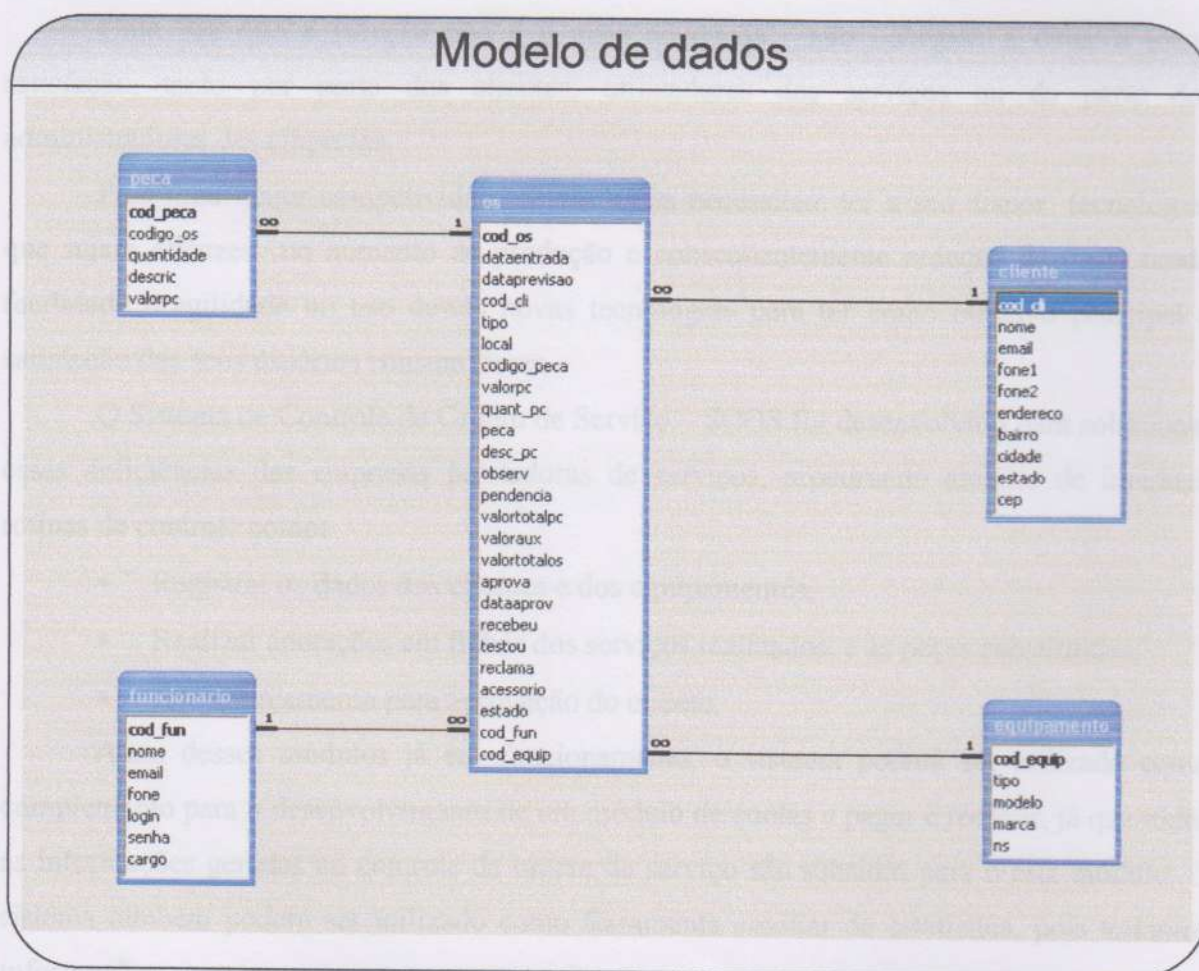


Figura 27. Diagrama entidade-relacionamento da base de dados do sistema.

4. CONCLUSÃO

Apesar da maioria das empresas prestadoras de serviços utilizarem a informática como elemento facilitador dos seus serviços, tentando com o uso, uma menor agilidade em busca da satisfação plena do consumidor ou usuários final dos seus serviços, muitas empresas não fazem por razões diversas.

A grande maioria das empresas ainda faz uso de tecnologia ultrapassada ou proprietária, exigindo um alto investimento, seja em manutenção da solução adotada ou em consumíveis necessários ao uso desta tecnologia no intuito de obter uma utilização razoável, porém não adequada.

Uma vez que a solução não é a mais adequada, não chegam a obter a plena satisfação, tanto por parte dos clientes, utilizadores dos serviços ou da parte dos administradores das empresas.

Para uma maior competitividade as empresas necessitam ter a seu dispor tecnologias que sejam eficazes no aumento da produção e conseqüentemente redução de custo tendo facilidade e agilidade no uso destas novas tecnologias para ter como objetivo principal a satisfação dos seus usuários consumidores.

O Sistema de Controle de Ordem de Serviço – SCOS foi desenvolvido para solucionar essas deficiências das empresas prestadoras de serviços, procurando atender de imediato rotinas de controle como:

- Registrar os dados dos clientes e dos equipamentos;
- Realizar anotações em fichas dos serviços realizados e as peças substituídas;
- Emitir orçamento para aprovação do cliente.

Além desses módulos já em funcionamento, o sistema poderá ser utilizado como complemento para o desenvolvimento de um módulo de contas a pagar e receber, já que todas as informações geradas no controle de ordem de serviço são subsídio para o este módulo. O sistema também poderá ser utilizado como ferramenta auxiliar de estatística, pois teríamos informações, que responderiam perguntas tais como:

Qual o cliente com maior número de entrada de ordem de serviços na empresa? É o mesmo equipamento? Se for, é viável a manutenção desse equipamento?

REFERÊNCIAS

- 1 - BOOCH, G; Rumbaugh, J; Jacobson, I. **UML, guia do usuário**, Rio de Janeiro, Campus, 2003.
- 2 - CACHO, Ricardo Luiz Azevedo. **Biblioteca digital: Analise orientada a objeto implementação centrada no usuário**. Natal: FARN, 2003, 43p. Monografia (Graduação em Bacharelado em Sistema de Informação). – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, Natal, 2003.
- 3 - CONNALEN, Jim. **Desenvolvimento de aplicação web com UML**, Rio de Janeiro, Campus, 2003.
- 4 - DATE, C, J. **Introdução a Sistema de Banco de Dados**, Rio de Janeiro, Campus, 2000.
- 5 - DEMARCO, Tom. **Analise estruturada e especificação de Sistema**. Rio de Janeiro, Campus. 1989
- 6 - DU BOIS, André Rauber. **Programação Orientada a Objetos**, 2005. Disponível em: <<http://atlas.ucpel.tche.br/~dubois/poo/POO.pdf>>. Acesso em 14/4/2005 10:33:46
- 7 - FIGUEIREDO, Marcelo Luiz de. **Analise orientada a objeto de um sistema acadêmico na web usando UML**. Natal: FARN, 2002, 52p. Monografia (Graduação em Bacharelado em Sistema de Informação). – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, Natal, 2003.
- 8 - FURLAN, Jose Davi. **Modelagem de Objetos através da UML**, São Paulo. Makron Books. 1998.
- 9 - HOLZSCHLAG, M. E.; KAISER, S., E. **O que são as web standards e por que devo usá-las**. 2002. Disponível em: <<http://www.maujor.com/w3ctuto/faqwsp.html#p211>>. Acesso em 12/4/2005 16:23:04
- 10 - LARMAN, Craig. **Utilizando UML e padrões**. Porto Alegre, Bookman, 2000.

- 11 – MATOS, A. V. **UML: Prático e descomplicado**. São Paulo: Erica, 2002.
- 12 - NIEMEYER, Patrick; Knudsen, Jonatahn. **Aprendendo Java**, Rio de Janeiro, Campus, 2000.
- 13 – MORISSEAU-LEROY, Nirva; Solomom, M. K; Basu, J; **ORACLE8i: Programação de componentes Java com EJB, CORB e JSP**. Rio de Janeiro, Campus, 2001.
- 14 - RAMALHO, Jose Antonio. **Iniciando em HTML**, São Paulo, Makron Books, 1996.
- 15 - RÜCKER, Bernardo. **Aspectos Jurídicos do domínio de Internet**, 2004. Disponível em: < <http://www.ampaco.com.br/art1.1.htm>>. Acesso em 3/5/2005.
- 16 – ROCHA NETO, Aluisio Ferreira da. **Banco de dados I**, Apostilha da disciplina de banco de dados do curso BSI da FARN, 2001
- ANEXOS
- 17 – SILVA, Mauricio da. **Apostilha on-line**, Disponível em: <<http://aprenda.html.vilabol.uol.com.br/Conceito.htm#Inicio>>. Acesso em 15/06/2005
- 18 – TAFNER, Malcon Anderson; Correia, Carlos Henrique. **Análise orientada a objetos**. Florianópolis, Bookstore Livraria Ltda. 2001.
- 19 – TUTORIAL POSTGRESSQL. **TUTORIAL POSTGRESSQL 7.3.4**. Disponível em: < <http://www.postgresql.org.br/tutorial2/tutorial-preface.html#INTRO-WHATIS>>. Acesso em 29/11/2005.
- 20 – TUTORIAL TOMCAT. **TUTORIAL TOMCAT 5.5**. Disponível em: <<http://www.mhavila.com.br/topicos/java/tomcat.html>>. Acesso em 29/11/2005.

Anexo 1 – Código fonte em JSP – cadastro de funcionário

```

1      <%@ page import="javax.servlet.http.*, javax.swing.*" %>
2      <html>
3      <head> <title>Cadastro Funcionario </title> </head>
4      <body>
5          <%
6              String tipo1=request.getParameter("nome");
7              String tipo2=request.getParameter("email");
8              String tipo3=request.getParameter("login");
9              String tipo4=request.getParameter("senha");
10             if ((tipo1==null || tipo1.equals("")) || tipo2==null || tipo2.equals("")) ||
11                 tipo3==null || tipo3.equals("")) || tipo4==null || tipo4.equals("")) { %>
12                 <jsp:forward page="cadastro.jsp?coderro=1">
13                 <jsp:param name="erro" value="1" invalid"/>
14             </jsp:forward>
15             <%>
16             Funcionario func;
17             funcionario.setNome(request.getParameter("nome"));
18             funcionario.setEmail(request.getParameter("email"));
19             funcionario.setPass(request.getParameter("pass"));
20             funcionario.setCargo(request.getParameter("cargo"));
21             funcionario.setLogin(request.getParameter("login"));
22             funcionario.setSenha(request.getParameter("senha"));
23             Funcionario fDB = new FuncionarioDB();
24             funcionarioCadastra(funcionario);
25             Funcionario fun = FuncionarioDB.busca(funcionario);
26             %>
27             <jsp:forward page="senha.htm"/>
28         </body>
29     </html>

```

ANEXOS

Anexo 1 – Código fonte em JSP – cadastro de funcionário

```

1      <%@ page import="javax.servlet.http.*, javax.swing.*, tec.os.*" %>
2      <html>
3      <head><title>Cadastro Funcionario </title></head>
4      <body>
5      public <%      String tipo1=request.getParameter("nome");
6      // De      String tipo2=request.getParameter("email");
7      private      String tipo3=request.getParameter("login");
8      // De      String tipo4=request.getParameter("senha");
9      // De      if (tipo1==null || tipo1.equals("") || tipo2==null || tipo2.equals("") ||
10      tipo3==null || tipo3.equals("") || tipo4==null || tipo4.equals("")) { %>
11      public <jsp:forward page="cadastro.jsp?coderro=1">
12      // De      <jsp:param name="erro" value="p_invalid" />
13      public </jsp:forward>
14      // De      <%}
15      this      Funcionario funcionario = new Funcionario();
16      this      funcionario.setNome(request.getParameter("nome"));
17      // De      funcionario.setEmail(request.getParameter("email"));
18      // De      funcionario.setFone(request.getParameter("fone"));
19      // De      funcionario.setCargo(request.getParameter("cargo"));
20      // De      funcionario.setLogin(request.getParameter("login"));
21      // De      funcionario.setSenha(request.getParameter("senha"));
22      this      FuncionarioDB funcionarioDB = new FuncionarioDB();
23      // De      funcionarioDB.salva(funcionario);
24      // De      Funcionario fun = FuncionarioDB.trasFun(tipo3);
25      // De      %>
26      // De      <jsp:forward page="senha.htm" />
27      // De      </body>
28      // De      </html>
29      // De      Funcionario fun = FuncionarioDB.getByPrimaryKey(cod_fun);
30      // De      this.setCod_fun(cod_fun);
31      // De      this.setNome(funcionario.getNome());

```

Anexo 2 - Código Java da classe Funcionário com seus atributos e métodos.

```
1 package tec.os;
2 /*Created on 18/08/2005
3  *@author Saulo Cantalice
4  */
5 public class Funcionario {
6     /** Definições dos atributos linhas 7 e 8 */
7     private int cod_fun;
8     private String nome,email,fone,login, senha, cargo;
9     /** construtores - linhas 11 a 34 */
10
11     public Funcionario() {
12     }
13     public Funcionario(String login, String senha, String email){
14         this.setLogin(login);
15         this.setSenha(senha);
16         this.setEmail(email);
17     }
18     public Funcionario(int cod_fun,String nome,String email,String fone,String
19 login,String senha,String cargo){
20         this.setCod_fun(cod_fun);
21         this.setNome(nome);
22         this.setEmail(email);
23         this.setFone(fone);
24         this.setLogin(login);
25         this.setSenha(senha);
26         this.setCargo(cargo);
27     }
28     public Funcionario (int cod_fun1){
29         Funcionario funcionario = FuncionarioDB.getByPrimaryKey(cod_fun);
30         this.setCod_fun(cod_fun);
31         this.setNome(funcionario.getNome());
```

```

31  this.setCargo(funcionario.getCargo());
32  this.setEmail(funcionario.getEmail());
33  System.out.println("Funcionario" + funcionario.getCod_fun() );
34  }
35  @param email The email to set.
36  /** definições dos métodos - linhas 38 a 121
37  public void setEmail(String email) {
38  @return Returns the cargo.
39  */
40  public String getCargo() {
41  return cargo; the cargo.
42  }
43  /** String getCod_fun() {
44  @param cargo The cargo to set.
45  */
46  public void setCargo(String cargo) {
47  this.cargo = cargo; the cargo to set.
48  }
49  /** void setCod_fun(String cod_fun) {
50  @return Returns the cod_fun.
51  */
52  public int getCod_fun() {
53  return cod_fun; the cod_fun.
54  }
55  /** String getLogin() {
56  @param cod_fun The cod_fun to set.
57  */
58  public void setCod_fun(int cod_fun) {
59  this.cod_fun = cod_fun; the cod_fun to set.
60  }
61  /** void setLogin(String login) {
62  @return Returns the email.
63  */

```

```
64  public String getEmail() {
65      return email;
66  }
67  /**
68   * @param email The email to set.
69   */
70  public void setEmail(String email) {
71      this.email = email;
72  }
73  /**
74   * @return Returns the fone.
75   */
76  public String getFone() {
77      return fone;
78  }
79  /**
80   * @param fone The fone to set.
81   */
82  public void setFone(String fone) {
83      this.fone = fone;
84  }
85  /**
86   * @return Returns the login.
87   */
88  public String getLogin() {
89      return login;
90  }
91  /**
92   * @param login The login to set.
93   */
94  public void setLogin(String login) {
95      this.login = login;
96  }
```

```

97  /**
98   * @return Returns the nome.
99   */
100 public String getNome() {
101     return nome;
102 }
103 /**
104 * @param nome The nome to set.
105 */
106 public void setNome(String nome) {
107     this.nome = nome;
108 }
109 /**
110 * @return Returns the senha.
111 */
112 public String getSenha() {
113     return senha;
114 }
115 /**
116 * @param senha The senha to set.
117 */
118 public void setSenha(String senha) {
119     this.senha = senha;
120 }
121 }

```

Anexo 3 – Código Java da classe FuncionarioDB com os seus atributos e métodos.

```

1  /*Created on 18/08/2005
2  *
3  */
4  package tec.os;
5  import java.sql.*;
6  import java.util.Vector;
7  /**@author Saulo Cantalice
8  *
9  */
10 public class FuncionarioDB {
11     public FuncionarioDB() {}
12     /** Método que realiza gravação de registro no banco de dados linhas 13 a 29 */
13     public void salva (Funcionario funcionario){
14         try {
15             Connection con = Banco.getConexao();
16             Statement stmt = con.createStatement();
17             String query = "INSERT INTO funcionario(nome,email,fone,login,senha,cargo)" +
18                 " VALUES ('"+ funcionario.getNome() + "','" + funcionario.getEmail() + "','" +
19                 funcionario.getFone() + "','" + funcionario.getLogin() + "','" +
20                 funcionario.getSenha() + "','" + funcionario.getCargo() + "' ) ";
21             System.out.println(query);
22             stmt.executeUpdate(query);
23             stmt.close();
24             con.commit();
25             con.close();
26         } catch (SQLException e){
27             System.out.println("Problemas ao abrir a conexao com o BD");
28         }
29     }
30     /** Método que realiza atualização no banco de dados - linhas 31 a 48 */

```

```

31 public void atualiza (Funcionario funcionario1){
32     try {
33         int n = funcionario1.getCod_fun();
34         Connection con = Banco.getConexao();
35         Statement stmt = con.createStatement();
36         String query = "UPDATE funcionario SET nome = '"+funcionario1.getNome()+
37             "',email='"+funcionario1.getEmail()+"',fone='"+funcionario1.getFone()+
38             "',login='"+funcionario1.getLogin()+"',senha='"+funcionario1.getSenha()+"',carg
39             o='"+funcionario1.getCargo()+
40             "' WHERE cod_fun='"+funcionario1.getCod_fun()+"'";
41         System.out.println(query);
42         stmt.executeUpdate(query);
43         stmt.close();
44         con.commit();
45         con.close();
46     } catch (SQLException e){
47         System.out.println("Problemas ao abrir a conexao com o BDff");
48     }
49 }
50 /** Método que recupera todos os registros no banco de dados - linhas 50 a 69 */
51 public static Vector getAll(){
52     Vector listaFuncionarios= new Vector();
53     Funcionario funcionario = null;
54     try {
55         Connection con = Banco.getConexao();
56         Statement stmt = con.createStatement();
57         String query = "SELECT * FROM funcionario" ;
58         ResultSet resp = stmt.executeQuery(query);
59         while (resp.next() ){
60             funcionario = new Funcionario(resp.getInt(1),resp.getString(2),resp.getString(3),
61             resp.getString(4),resp.getString(5),resp.getString(6),resp.getString(7));
62             listaFuncionarios.add(funcionario);
63         }

```

```

63  stmt.close();
64  con.close();
65  } catch (SQLException e){
66      System.out.println("Problemas ao abrir a conexao com o BD");
67  }
68  return listaFuncionarios;
69  }
70  /** Recupera um funcionário através do campo código de usuário linhas 71 a 92
71  */
72  public static Funcionario getByPrimarykey(int cod_fun){
73      Funcionario funcionario = null;
74      try {
75          Connection con = Banco.getConexao();
76          Statement stmt = con.createStatement();
77          String query = "SELECT * FROM funcionario " +
78              " WHERE cod_fun = " + cod_fun ;
79          ResultSet resp = stmt.executeQuery(query);
80          while (resp.next() ){
81              funcionario = new Funcionario(resp.getInt(1),resp.getString(2),resp.getString(3),
82              resp.getString(4),resp.getString(5),resp.getString(6),resp.getString(7));
83              System.out.println("FuncionarioDB" + resp.getInt(1));
84          }
85
86      stmt.close();
87      con.close();
88      } catch (SQLException e){
89          System.out.println("Problemas ao abrir a conexao com o BD");
90      }
91      return funcionario;
92  }
93  /** Recupera o registro de um funcionário, passando como parametro o login do
94  funcionario - linhas 94 a 114 */

```

```

94 public static Funcionario trasFun(String login){
95     Funcionario funcionario=null;
96     Connection con=null;
97     con = Banco.getConexao();
98     String consulta="SELECT * FROM funcionario WHERE login=?";
99     if (con!=null){
100         try {
101             PreparedStatement ps=con.prepareStatement(consulta);
102             ps.setString(1,login);
103             ResultSet rs =ps.executeQuery();
104             while (rs.next() ){
105                 funcionario = new Funcionario(rs.getInt(1),rs.getString(2),rs.getString(3),
106                 rs.getString(4), rs.getString(5), rs.getString(6), rs.getString(7));
107             }
108             con.close();
109         }catch (SQLException SQLex){
110             SQLex.printStackTrace();
111         }
112     }
113     return funcionario;
114 }
115 }

```

Anexo 4 – Código fonte em JSP – Cadastrar OS

```

1      <%@page import="javax.servlet.http.*, javax.swing.*,
      java.util.Date, java.text.SimpleDateFormat, tec.os.*" %>
2      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
3      <html>
4      <head>
5      <title>Cadastro Ordem de Servico</title>
6      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
7      </head>
8      <body>
9      <%
10     String datah=OsDB.dataOS();
11     Os os = new Os();
12     String login = (String) session.getAttribute("usrinic");
13     Funcionario func = FuncionarioDB.trasFun(login);
14     int codigo_func=func.getCod_fun();
15     String dia1 = request.getParameter("dia");
16     String mes1 = request.getParameter("mes");
17     String ano1 = request.getParameter("ano");
18     String dataprev="", aux="/";
19     dataprev+=dia1+aux+mes1+aux+ano1;
20     os.setDataentrada(datah);
21     os.setDataprevisao(dataprev);
22     os.setCod_cli(Integer.parseInt(request.getParameter("cli")));
23     os.setTipo(request.getParameter("tipo"));
24     os.setLocal1(request.getParameter("local2"));
25     os.setTipo_atend(request.getParameter("tipoant"));
26     os.setObserv(request.getParameter("observ"));
27     os.setRecebeu(request.getParameter("recebeu"));
28     os.setTestou(request.getParameter("testou"));
29     os.setReclama(request.getParameter("reclama"));

```

```

30 os.setAcessorio(request.getParameter("acesso"));
31 os.setStatus("servico");
32 os.setCod_func(codigo_func);
33 os.setTipo_equip(request.getParameter("tipo_equip"));
34 os.setModelo(request.getParameter("modelo"));
35 os.setMarca(request.getParameter("marca"));
36 os.setNs(request.getParameter("ns"));
37 OsDB osDB = new OsDB();
38 osDB.salva(os);
39 %>
40 <jsp:forward page="menu_principal.html" />
41 </body>
42 </html>

```

```

10 int numeroOs = (Integer.parseInt(request.getParameter("numero")));
11 OsDB osDB = new OsDB();
12 int status = (Integer.parseInt(request.getParameter("valor")));
13 int statusOs = osDB.getStatus(numeroOs);
14 String statusOs = statusOs;
15 String statusOs = statusOs;
16 String tipo1 = request.getParameter("tipo1");
17 String tipo2 = request.getParameter("tipo2");
18 String tipo3 = request.getParameter("tipo3");
19 if (tipo1 == null || tipo1.equals("") || tipo2 == null || tipo2.equals("") || tipo3 == null ||
    tipo3.equals("")) { %>
20 <!--
21 -->
22 OsDB osDB = new OsDB();
23 osDB.atualizaValor(statusOs);
24 Pessoa pessoa = new Pessoa();
25 pessoa.setCodigoOs(Integer.parseInt(request.getParameter("numero")));
26 pessoa.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
27 pessoa.setValor(Integer.parseInt(request.getParameter("valor")));
28 pessoa.setDescricao(request.getParameter("descricao"));

```

Anexo 5 – Código fonte em JSP – Cadastramento de peças

```
1      <%@pageimport="javax.servlet.http.*,javax.swing.*,java.util.Date,  
java.text.SimpleDateFormat,tec.os.*" %>  
2      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
3      <html>  
4      <head>  
5      <title>Cadastro pecas</title>  
6      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">  
7      </head>  
8      <body>  
9      <%  
10     int numeroOs =(Integer.parseInt(request.getParameter("numos")));  
11     Os osa = OsDB.consult(numeroOs);  
12     int soma1 =(Integer.parseInt(request.getParameter("valor")));  
13     int somat=osa.getValortotalpc();  
14     somat+=soma1;  
15     osa.setValortotalpc(somat);  
16     String tipo1=request.getParameter("matAux");  
17     String tipo2=request.getParameter("maoobra");  
18     String tipo3=request.getParameter("observa2");  
19     if (tipo1==null || tipo1.equals("") || tipo2==null || tipo2.equals("") || tipo3==null |  
| tipo3.equals("") ) { %>  
20     <teste aqui>  
21     <%}  
22     OsDB osdx=new OsDB();  
23     osdx.atualizaValorpc(osa);  
24     Pecas pecas = new Pecas();  
25     pecas.setCodigo_os(Integer.parseInt(request.getParameter("numos")));  
26     pecas.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));  
27     pecas.setValorpc(Integer.parseInt(request.getParameter("valor")));  
28     pecas.setDescric(request.getParameter("descri4"));
```

```
29  PecasDB pecasDB = new PecasDB();
30  pecasDB.salva(pecas);
31  %>
32  <jsp:forward page="menu_principal.html" />
33  </body>
34  </html>
```