

8,8  
(nota exat.)  
HBM

RODOLFO AMADEU SCOLARI

## CONTROLE DE ORDENS DE CHÃO DE FÁBRICA EM UMA GRÁFICA

Trabalho apresentado à Escola  
Politécnica da Universidade de São  
Paulo para obter Graduação em  
Engenharia Mecatrônica

São Paulo  
2002

**RODOLFO AMADEU SCOLARI**

**CONTROLE DE ORDENS DE CHÃO DE FÁBRICA EM  
UMA GRÁFICA**

Trabalho apresentado à Escola  
Politécnica da Universidade de São  
Paulo para obter Graduação em  
Engenharia Mecatrônica

Área de concentração:  
Engenharia Mecatrônica

Orientador:  
Prof. Doutor Marcos R. P. Barreto

São Paulo  
2002

A meus pais, pelo incentivo e  
inspiração.

## **AGRADECIMENTOS**

Ao Professor Dr. MARCOS R. P. BARRETO, mais que um orientador, um parceiro, pela orientação no trabalho, por sempre estar disposto a ajudar, pela cordialidade e compreensão.

À família e amigos pessoais, pelo apoio e incentivo nos momentos difíceis dessa etapa de minha vida, os quais não foram poucos,.

Finalmente, aos amigos da classe que de uma forma ou de outra me ajudaram a concluir a faculdade. Agradecimentos especiais ao Rodrigo (Kiko) cuja ajuda foi fundamental na conclusão desse trabalho, ao Fernando pelo apoio e ao Cássio por sempre conseguir resolver meus problemas com matrículas e afins.

## RESUMO

Este trabalho trata do projeto e implementação de um sistema de controle de ordens de chão de fábrica em uma gráfica. Apesar do enfoque, o sistema em questão possui flexibilidade para ser usado por qualquer empresa que trabalhe com produção sob encomenda, como é o caso da gráfica.

As funções principais são: fazer a alocação de recursos e a programação da produção. Vale salientar que a programação da produção nesse caso não é automática, o que faz do sistema implementado uma útil ferramenta de apoio para o engenheiro responsável por essa difícil tarefa.

Para a modelagem foi estudada e utilizada a UML, linguagem bastante utilizada para a modelagem de sistemas orientados a objetos. Para a implementação foram utilizadas a Plataforma J2EE e a linguagem de programação Java, mais especificamente os *servlets*.



## ÍNDICE

1. Introdução.....	4
2. Sistemas de Informação.....	6
3. Manufacturing Execution Systems (MES).....	8
3.1. Definição.....	8
3.2. Fluxos de informações entre sistemas .....	9
3.2.1. Informações do MES para outros sistemas .....	9
3.2.2. Informações de outros sistemas para o MES.....	10
3.3. Funções do MES.....	10
4. Plataforma J2EE.....	13
4.1. Aplicativos de multicamadas distribuídas .....	13
4.2. Componentes do aplicativo J2EE.....	14
4.2.1. Componentes de Cliente .....	15
4.2.1.1. <i>Aplicativos de cliente</i> .....	15
4.2.1.2. <i>Web Browsers</i> .....	16
4.2.1.3. <i>Applets</i> .....	16
4.2.1.4. <i>Arquitetura de componentes JavaBeans</i> .....	16
4.2.1.5. <i>Comunicações com o servidor J2EE</i> .....	16
4.2.2. Componentes Web.....	17
4.2.3. Componentes de Negócio .....	18
4.3. Camada do Sistema de Informação da Empresa.....	19
5. Linguagem de Modelagem Unificada (UML).....	20
5.1. Objetivos da UML.....	20
5.2. Uso da UML.....	21
5.3. Fases do Desenvolvimento de um Sistema em UML.....	22
5.3.1. Análise de Requisitos.....	23
5.3.2. Análise .....	23
5.3.3. Design (Projeto).....	24
5.3.4. Programação.....	24
5.3.5. Testes .....	25
5.4. A Notação da Linguagem de Modelagem Unificada – UML.....	25
5.4.1. Diagramas .....	26
5.4.1.1. <i>Diagrama Use-Case</i> .....	26
5.4.1.2. <i>Diagrama de Classes</i> .....	28
5.4.1.3. <i>Diagrama de Sequência</i> .....	29
6. Sistema de Controle de Ordens de Chão de Fábrica.....	31
6.1. Funções a serem implementadas.....	31
6.1.1. Alocação de Recursos.....	31
6.1.2. Cronograma de Operações.....	32
6.2. Requisitos do Projeto.....	32
6.2.1. Descrição geral.....	32
6.2.1.1. <i>Projeto do banco de dados</i> .....	32
6.2.1.2. <i>Desenvolvimento dos aplicativos</i> .....	32
6.2.1.3. <i>Design das interfaces do sistema com o usuário</i> .....	33



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

6.2.2. Requisitos Não-Funcionais .....	33
6.2.3.1. <i>Ambiente de execução</i> .....	33
6.2.3.2. <i>Interfaces com outros sistemas</i> .....	33
6.2.3.3. <i>Desempenho</i> .....	33
6.2.3.4. <i>Escalabilidade</i> .....	34
6.2.3.5. <i>Segurança e Privacidade (security e privacy)</i> .....	34
7. Especificação UML .....	35
7.1. Diagramas de use-case .....	36
7.1.1. Pedido .....	36
7.1.1.1. <i>Inserir Pedido</i> .....	36
7.1.1.2. <i>Alterar Pedido</i> .....	36
7.1.1.3. <i>Alterar estado</i> .....	37
7.1.1.4. <i>Consultar pedido ad-hoc</i> .....	37
7.1.1.5. <i>Obter quanto em dinheiro o cliente possui em pedidos em aberto</i> ..	37
7.1.1.6. <i>Obter relatório de produção de um pedido</i> .....	37
7.1.2. Operação .....	38
7.1.2.1. <i>Inserir Operação</i> .....	38
7.1.2.2. <i>Alterar Operação</i> .....	38
7.1.2.3. <i>Excluir Operação</i> .....	38
7.1.3. Máquina/ Monitoração de máquina .....	39
7.1.3.1. <i>Inserir Máquina</i> .....	39
7.1.3.2. <i>Alterar Máquina</i> .....	39
7.1.3.3. <i>Excluir Máquina</i> .....	39
7.1.3.4. <i>Consultar qual serviço está na máquina</i> .....	39
7.1.3.5. <i>Consultar quais serviços estão na fila</i> .....	40
7.1.4. Cronograma de Produção .....	41
7.1.4.1. <i>Obter uso dos recursos</i> .....	41
7.1.4.2. <i>Verificar consistência do programa</i> .....	41
7.1.4.3. <i>Consultar a programação de um pedido</i> .....	41
7.1.4.4. <i>Fazer a programação de um pedido</i> .....	42
7.1.4.5. <i>Obter Pedidos a programar</i> .....	42
7.1.5. Cliente .....	43
7.1.5.1. <i>Inserir Cliente</i> .....	43
7.1.5.2. <i>Alterar Cliente</i> .....	43
7.1.5.3. <i>Excluir Cliente</i> .....	43
7.1.6. Controle de Acesso .....	44
7.1.6.1. <i>Inserir Usuário</i> .....	44
7.1.6.2. <i>Alterar Usuário</i> .....	44
7.1.6.3. <i>Excluir Usuário</i> .....	44
7.1.7. Apontamento .....	45
7.1.7.1. <i>Inserir Apontamento</i> .....	45
7.1.7.2. <i>Alterar Apontamento</i> .....	45
7.2. Projeto da Interface .....	46
7.2.1. Telas de Pedido .....	47
7.2.2. Telas de Operação .....	50
7.2.3. Telas de Máquina/Monitoração .....	51
7.2.4. Telas de Cronograma de Produção .....	53
7.2.5. Telas de Cliente .....	55



**PMC581-Projeto Mecânico II**  
**Mecatrônica**  
Escola Politécnica da Universidade de São Paulo

7.2.6. Telas de Usuário .....	57
7.2.7. Telas de Apontamento .....	58
7.3. Diagrama de Classes.....	60
8. Referências Bibliográficas.....	61
Apêndice – Cronograma de Atividades .....	62





## 1. Introdução

Este trabalho tem como objetivos a modelagem e a implementação de um sistema de controle de ordens de chão de fábrica em uma gráfica que trabalha sob encomenda, ou seja, só produz o necessário para atender os pedidos confirmados pelos clientes. A gráfica em questão também não trabalha com estoque de matéria-prima, uma vez que, teoricamente, nunca se sabe qual tipo de pedido entrará na fábrica.

Atualmente, existem no mercado diversos tipos de *softwares* que possuem como objetivo a automação de processos e fluxos de informação dentro de uma empresa. Dentre eles, existe o *Manufacturing Execution Systems* (MES). Como será visto adiante, o controle de ordens de chão de fábrica é um caso particular do MES na medida em que nem todas suas funções são implementadas.

O sistema ao qual o trabalho se refere tem como funções principais fazer a programação da produção e a alocação dos recursos de modo que o usuário tenha uma melhor visão do que realmente está acontecendo ou irá acontecer na fábrica.

Este sistema poderia ser projetado de modo a ser útil em qualquer fábrica de qualquer ramo de atividade. Porém, preferiu-se personalizar o sistema no intuito de uma possível utilização imediata ao término de sua implementação.

Como se trata de um *software*, há a necessidade de uma ferramenta de programação. Essa ferramenta será o J2EE (Java 2 *Enterprise Edition*) onde, logicamente será usada a linguagem de programação Java. A linguagem de programação Java é uma linguagem orientada a objetos; sendo assim, utilizar-se-á neste trabalho a UML (*Unified Modeling Language*) para se fazer a modelagem do sistema em questão. Não será feito, entretanto, uma especificação UML completa, com todos os seus



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

diagramas. Serão feitos somente alguns diagramas e o funcionamento não será explicado através de especificações UML's.



## 2. Sistemas de Informação

Hoje em dia existem diversos tipos de sistemas de informação usados por empresas. Cada uma dessas categorias de sistemas possui várias funções e tipos de produtos. As principais categorias de *softwares* usados em empresas de manufatura hoje em dia são:

- **Enterprise Resources Planning (ERP):** que consiste daqueles sistemas que fornecem gerenciamento de finanças e pedidos, planejamento da produção e material e funções correlatas.
- **Supply Chain Management (SCM):** que possui funções de previsão, distribuição e logística, gerenciamento de transporte, comércio eletrônico e sistemas de planejamento avançados.
- **Sales and Service Management (SSM):** que compreende *software* para automação de venda, serviço de cotação, retorno de produto, etc.
- **Product and Process Engineering (P&PE):** que inclui projeto e fabricação auxiliado por computador (CAD/CAM), modelagem de processos e gerenciamento de dados do produto (PDM).
- **Controles:** que são geralmente sistemas híbridos de *hardware* e *software* como sistemas de controle distribuído (DCS), controladores lógicos programáveis (PLC), controle numérico distribuído (DNC), sistemas de aquisição de dados (SCADA) e outros controles de processos computadorizados projetados para controlar de que forma o produto esta sendo fabricado.



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

E, logicamente o MES, que será explicado com um pouco mais de detalhes a seguir.



### 3. Manufacturing Execution Systems (MES)

O MES consiste, sucintamente, de um sistema de informação que fornece informações para se saber quais operações devem ser executadas para se atingir metas de negócio.

#### 3.1. Definição

Existe mais de uma definição para o sistema MES. Vai-se aqui transcrever duas delas a fim de se dar uma idéia mais completa do que um sistema deve ser capaz de fazer para ser considerado um sistema MES.

Segundo a *Manufacturing Execution Systems Association International* “um sistema de execução de fabricação entrega informação que permite a otimização das atividades desde o lançamento da ordem até o produto acabado. Usando dados atuais e acurados, MES guia, inicia, responde e noticia as atividades do chão de fábrica assim que elas ocorrem. O resultado da resposta rápida para mudança de condições somada ao foco na redução de atividades que não geram valor agregado, controla efetivamente as operações e processos no chão de fábrica. MES fornece informação sobre atividades de produção para o ERP e o SCM via comunicações bi laterais.”

Também há outra definição: “Sistemas de execução de fabricação são sistemas de informação que residem no chão de fábrica entre os sistemas de planejamento dos escritórios e os controles industriais diretos no processo.”



### **3.2. Fluxos de informações entre sistemas**

Uma empresa geralmente necessita de funções de cada um dos tipos de sistemas, e a integração entre MES e os outros cinco principais tipos de sistemas é a chave para o ganho de grandes benefícios em todos os sistemas de informação presentes na empresa. Um sistema MES se relaciona com todas as outras categorias de sistemas de informações citadas anteriormente, havendo assim um fluxo de informações entre eles.

#### **3.2.1. Informações do MES para outros sistemas**

O sistema MES alimenta cada um dos principais tipos de sistemas de informação da seguinte forma:

- ERP recorre ao MES por dados reais de custos, tempo de ciclo e outros dados de performance da produção;
- SCM recebe dados sobre o real status da ordem de produção e capacidade de produção;
- SSM depende das informações do MES na medida em que o sucesso na cotação e entrega depende do que está acontecendo com os recursos num dado momento;
- P&PE depende da capacidade de fabricação do produto e da qualidade medida pelo MES;
- Controles podem receber receitas e instruções do MES.



### **3.2.2. Informações de outros sistemas para o MES**

O sistema MES também recebe dados de outros sistemas de informação.

Como se pode ver nos exemplos a seguir:

- ERP: seus planos alimentam o despacho de trabalho do MÊS;
- SCM: seus planos mestres e cronogramas controlam os tempos de atividades na planta;
- SSM: suas configurações e cotações fornecem uma linha base da informação da ordem para fabricação;
- P&PE: controlam instruções de trabalho, receitas e parâmetros operacionais;
- Controles: seus dados são usados para medir a performance real e as condições de operação.

### **3.3. Funções do MES**

Assim como todas as categorias de sistemas de informação, o MES não possui apenas uma função. Na realidade o MES possui 11 (onze) funções que são responsáveis por toda a utilidade do mesmo. São elas:

- **Cronograma de Operações** – seqüenciamento e duração das atividades de modo a otimizar a performance da planta baseado na capacidade finita dos recursos;



- **Alocação de Recursos** – determinação do que pessoas, máquinas, ferramentas e materiais devem fazer. E rastreamento do que eles atualmente estão fazendo ou fizeram;
- **Despacho de unidades de produção** – dá o comando para enviar materiais ou ordens para certas partes da fábrica para começar um processo ou passo;
- **Controle de Documentos** – gerenciamento e distribuição de informações de produtos, processos, projetos ou ordens, assim como agrupamento de relatórios de certificação de trabalho e condições;
- **Rastreamento de produtos e Genealogia** – monitoramento do progresso das unidades ou lotes de saída para criar uma história completa do produto;
- **Análise de Performance** – comparação dos resultados medidos na fábrica em relação a metas determinadas pela corporação, clientes ou entidades de regulação;
- **Gerenciamento de Trabalho** – rastreamento e direcionamento do pessoal de operação durante uma troca baseada em qualificações, trabalho padrão e necessidades de negócio;
- **Gerenciamento de Manutenção** – planejamento e execução de atividades apropriadas para manter equipamentos e outros recursos trabalhando a fim de atingirem suas metas;
- **Gerenciamento de Processo** – direcionamento do fluxo de trabalho na fábrica baseado nas atividades de produção planejadas e reais.





PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

- **Gerenciamento da Qualidade** – registro, rastreamento e análise de características de produtos e processos em relação ao plano de engenharia;
- **Aquisição de Dados** – monitoramento, agrupamento e organização de dados sobre processos, materiais e operações de pessoas, máquinas ou controles.



## 4. Plataforma J2EE

Hoje em dia, devido à rápida inovação e à demanda do mundo de *e-commerce* e tecnologia da informação, os aplicativos empresariais têm sido projetados, construídos e produzidos com menos dinheiro, mais rápido e com menos recursos do que era antigamente.

A fim de reduzir custos nos projetos e desenvolvimento de aplicativos empresariais, a tecnologia da plataforma *Java 2 Enterprise Edition* (J2EE™) fornece uma abordagem baseada em componente para o projeto, desenvolvimento e montagem de aplicativos para empresa. A plataforma J2EE fornece um modelo de aplicativo baseado em multicamadas distribuídas, a capacidade de reutilização de componentes, um modelo de segurança unificado e controle de transação flexível. Além de entregar soluções inovadoras ao mercado mais rápido que antes, suas soluções não são atreladas a produtos e API's de um fornecedor específico, pois o J2EE é independente de plataforma.

### 4.1. Aplicativos de multicamadas distribuídas

Como já foi dito, a plataforma J2EE fornece um modelo de aplicativo baseado em multicamadas distribuídas. Isso significa que a lógica do aplicativo é dividida segundo a função dos componentes, e os vários componentes são instalados em máquinas diferentes dependendo de em que camada esse componente pertence no ambiente de multicamadas do J2EE. A figura 1 mostra dois aplicativos J2EE baseados em multicamadas, mostrando em que máquina roda os componentes de cada camada.

- Os componentes da camada do cliente (*Cliente tier*) rodam na máquina do cliente;



- Os componentes da camada da web (*Web tier*) rodam no servidor J2EE;
- Os componentes da camada de negócio (*Business tier*) rodam no servidor J2EE;
- O *software* da camada do sistema de informação da empresa (EIS) roda no servidor EIS.

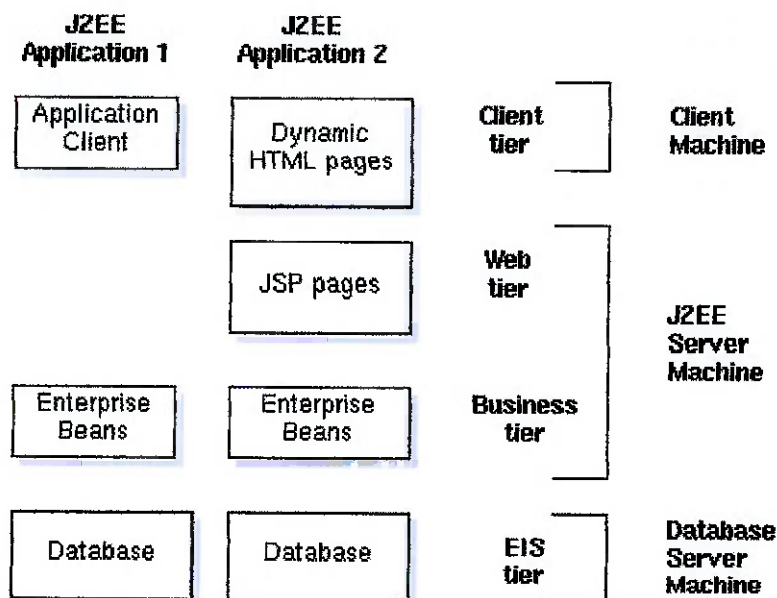


Figura 1: Aplicativos baseados em multicamadas

Embora um aplicativo J2EE possa consistir de três ou quatro camadas como mostrada na figura 1, esses aplicativos são geralmente considerados de três camadas porque eles estão distribuídos em três diferentes locais: a máquina do cliente, a máquina servidor J2EE e a máquina da base de dados.

#### 4.2. Componentes do aplicativo J2EE

Os aplicativos J2EE são compostos por componentes e um componente J2EE é uma unidade de software funcional que são montados num aplicativo J2EE com suas referidas classes, arquivos e comunicações com outros componentes. A especificação J2EE define os seguintes componentes:



- Aplicativos de clientes e *applets* são componentes de cliente;
- Componentes de tecnologia *Java Servlet* e *JavaServer Pages* (JSP) são componentes web;
- Componentes *Enterprise JavaBeans* (EJB) são componentes de negócio.

Os componentes J2EE são escritos em linguagem de programação Java e são compilados da mesma maneira que qualquer programa de linguagem Java. A diferença é que quando se trabalha com a plataforma J2EE, os componentes J2EE são montados em um aplicativo J2EE, são verificados se estão bem formados e em concordância com a especificação J2EE e implementados para produção onde eles rodam e são gerenciados pelo servidor J2EE.

#### 4.2.1. Componentes de Cliente

Um aplicativo J2EE pode ser baseado em web ou não baseado em web. Um aplicativo de cliente é executado na máquina do cliente, em um aplicativo J2EE não baseado em web. Um *web browser* carrega páginas web e *applets* para a máquina do cliente em um aplicativo baseado em web.

##### 4.2.1.1. Aplicativos de cliente

Um aplicativo de cliente roda na máquina do cliente e fornece um meio para os usuários manipularem tarefas. Ele geralmente possui uma interface gráfica com o usuário criada a partir do conjunto de classes *Swing* ou *Abstract Window Toolkit* (AWT), mas uma interface em linha de comando também é possível..

Aplicativos de cliente acessam diretamente *enterprise beans* que rodam na camada de negócio. Entretanto, se um aplicativo de cliente requerer, ele



pode abrir uma conexão http para estabilizar comunicação com um *servlet* que roda na camada web.

#### 4.2.1.2. Web Browsers

O *web browser* do usuário carrega páginas da web em *Hypertext Markup Language* (HTML) estática ou dinâmica, *Wireless Markup Language* (WML) ou *Extensible Markup Language* (XML) da camada web. Páginas web dinâmicas são geradas por *servlets* ou páginas JSP que rodam na camada web.

#### 4.2.1.3. Applets

*Applet* é um pequeno aplicativo de cliente escrito em linguagem de programação Java que é executado no Java VM instalado no *web browser*. Uma página da web carregada da camada web pode conter um *applet*.

#### 4.2.1.4. Arquitetura de componentes JavaBeans

A camada de cliente também pode incluir um componente baseado na arquitetura de componentes *JavaBeans* para gerenciar o fluxo de dados entre um aplicativo de cliente ou um *applet* e os componentes que rodam no servidor J2EE. Vale notar que os componentes *JavaBeans* não são considerados componentes pela especificação J2EE.

Componentes *JavaBeans* escritos para plataforma J2EE têm variáveis de instância e pegam e alteram métodos para acessar os dados nas variáveis de instância. Esses componentes usados desta maneira são simples de projetar e implementar.

#### 4.2.1.5. Comunicações com o servidor J2EE



Afigura 2 mostra os vários elementos que podem compõe a camada de cliente. O cliente pode se comunicar diretamente com a camada e negócio que roda no servidor J2EE ou, como no caso de um cliente rodando em um *browser*, através de páginas JSP ou *servlets* que rodam na camada web.

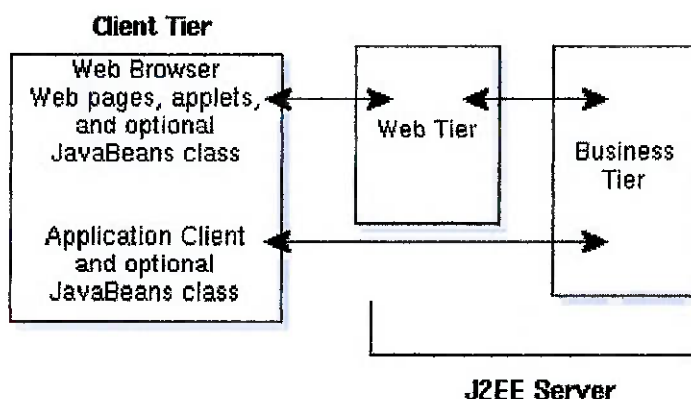


Figura 2: Comunicação com o servidor

#### 4.2.2. Componentes Web

Componentes web podem ser páginas JSP ou *servlets*. *Servlets* são classes em linguagem de programação Java que processam requisições e constroem respostas dinamicamente. Páginas JSP são documentos baseados em texto que contêm fragmentos estáticos de código em linguagem de programação Java para gerarem conteúdos dinâmicos. Quando se carrega uma página JSP, um *servlet* de segundo plano executa os fragmentos de código e retorna uma resposta.

Como na camada de cliente e mostrado na figura 3, a camada web pode conter um objeto *JavaBeans* para gerenciar a entrada do usuário e enviar essa entrada para o *enterprise beans* que roda na camada de negócio para processamento.

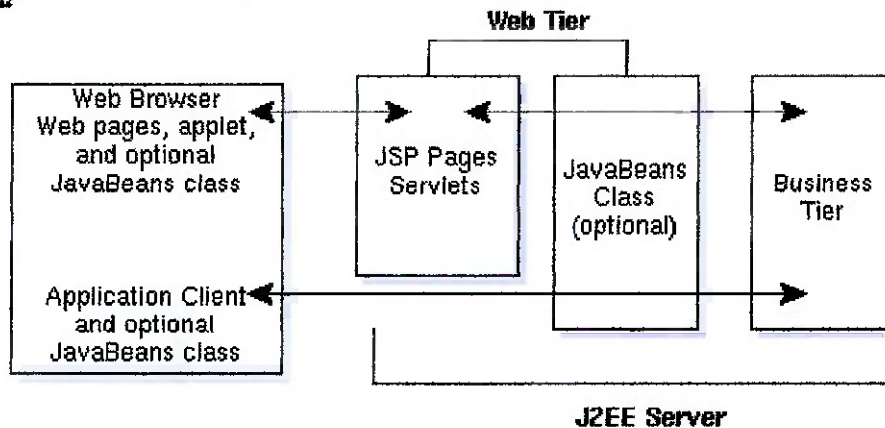


Figura 3: Camada web e aplicativo J2EE

#### 4.2.3. Componentes de Negócio

Código de negócio, que é a lógica que resolve e encontra as necessidades de um domínio particular de negócio como, por exemplo, negócio bancário, vendas ou finanças é manipulado pelo *enterprise beans* que roda na camada de negócio. A figura 4 mostra como um *enterprise beans* recebe dados de programas clientes, os processa se necessário, e os envia para a camada do sistema de informação da empresa (*enterprise information system (EIS)*) para armazenamento. Um *enterprise bean* também recupera dado do armazenamento, o processa se necessário, e o manda de volta para o programa cliente.

Há três tipos de *enterprise beans*: o *session beans*, o *entity beans* e *message-driven beans*. Uma *session bean* representa a troca de dados (comunicação) transitória como o cliente. Quando o cliente termina a execução, a *session bean* e seus dados se perdem. Em contraste, uma *entity bean* representa o armazenamento persistente de dados em uma linha de uma tabela de base de dados. Se o cliente finaliza ou o servidor é desligado, os dados da *entity bean* são salvos.

Uma *message-driven bean* combina características de *session bean* com o ouvinte de mensagem *Java Message Service (JMS)*, permitindo um componente de negócio receber mensagens JMS assincronamente.

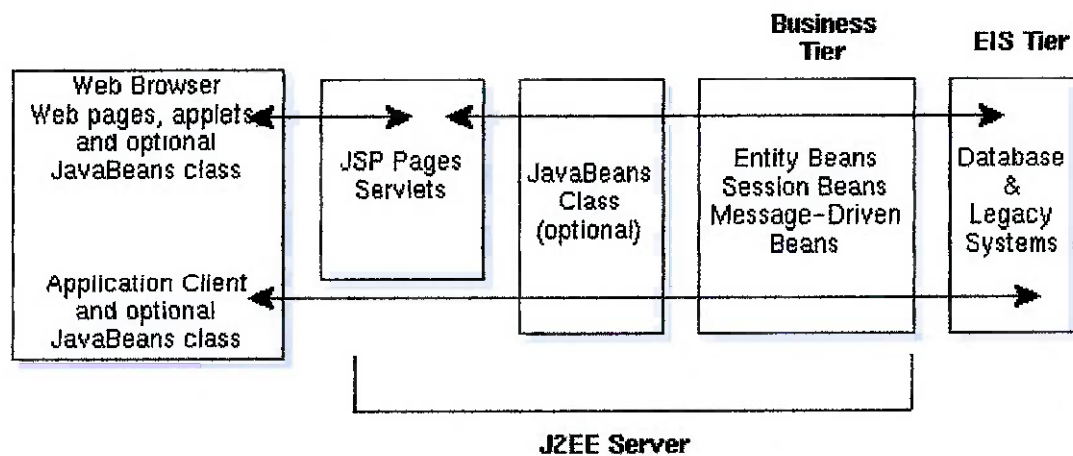


Figura 4: Camadas de negócio e EIS

### 4.3. Camada do Sistema de Informação da Empresa

A camada do sistema de informação da empresa (EIS) manipula o software do sistema de informação da empresa e inclui sistemas de infraestrutura da empresa como o ERP, sistemas de base de dados e outros sistemas de informação legais.





## 5. Linguagem de Modelagem Unificada (UML)

O grande problema do desenvolvimento de novos sistemas utilizando a orientação a objetos nas fases de análise de requisitos, análise de sistemas e design é que não existe uma notação padronizada e realmente eficaz que abranja qualquer tipo de aplicação que se deseje. Cada simbologia existente possui seus próprios conceitos, gráficos e terminologias, resultando numa grande confusão, especialmente para aqueles que querem utilizar a orientação a objetos não só sabendo para que lado aponta a seta de um relacionamento, mas sabendo criar modelos de qualidade para ajudá-los a construir e manter sistemas cada vez mais eficazes.

Quando a *Unified Modeling Language* (UML) foi lançada, muitos desenvolvedores da área da orientação a objetos ficaram entusiasmados já que essa padronização proposta pela UML era o tipo de força que eles sempre esperaram.

A UML é muito mais que a padronização de uma notação. É também o desenvolvimento de novos conceitos não normalmente usados. Por isso e muitas outras razões, o bom entendimento da UML não é apenas aprender a simbologia e o seu significado, mas também significa aprender a modelar orientado a objetos no estado da arte.

### 5.1. Objetivos da UML

Os objetivos da UML são:

- A modelagem de sistemas (não apenas de software) usando os conceitos da orientação a objetos;
- Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis;
- Criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.



A UML está destinada a ser dominante, a linguagem de modelagem comum a ser usada nas indústrias. Ela está totalmente baseada em conceitos e padrões extensivamente testados, provenientes das metodologias existentes anteriormente, e também é muito bem documentada com toda a especificação da semântica da linguagem representada em meta-modelos

## **5.2. Uso da UML**

A UML é usada no desenvolvimento dos mais diversos tipos de sistemas. Ela abrange sempre qualquer característica de um sistema em um de seus diagramas e é também aplicada em diferentes fases do desenvolvimento de um sistema, desde a especificação da análise de requisitos até a finalização com a fase de testes.

O objetivo da UML é descrever qualquer tipo de sistema, em termos de diagramas orientado a objetos. Naturalmente, o uso mais comum é para criar modelos de sistemas de software, mas a UML também é usada para representar sistemas mecânicos sem nenhum software. Esses são alguns tipos diferentes de sistemas com suas características mais comuns:

**Sistemas de Informação:** armazenar, pesquisar, editar e mostrar informações para os usuários. Manter grandes quantidades de dados com relacionamentos complexos, que são guardados em bancos de dados relacionais ou orientados a objetos.

**Sistemas Técnicos:** manter e controlar equipamentos técnicos como de telecomunicações, equipamentos militares ou processos industriais. Eles devem possuir interfaces especiais do equipamento e menos programação de software de que os sistemas de informação. Sistemas Técnicos são geralmente sistemas real-time.



**Sistemas Real-time Integrados:** executados em simples peças de hardware integrados a telefones celulares, carros, alarmes etc. Estes sistemas implementam programação de baixo nível e requerem suporte real-time.

**Sistemas Distribuídos:** distribuídos em máquinas onde os dados são transferidos facilmente de uma máquina para outra. Eles requerem mecanismos de comunicação sincronizados para garantir a integridade dos dados e geralmente são construídos em mecanismos de objetos como CORBA, COM/DCOM ou Java Beans/RMI.

**Sistemas de Software:** definem uma infra-estrutura técnica que outros softwares utilizam. Sistemas Operacionais, bancos de dados, e ações de usuários que executam ações de baixo nível no hardware, ao mesmo tempo em que disponibilizam interfaces genéricas de uso de outros softwares.

**Sistemas de Negócios:** descreve os objetivos, especificações (pessoas, computadores etc.), as regras (leis, estratégias de negócios etc.), e o atual trabalho desempenhado nos processos do negócio.

É importante perceber que a maioria dos sistemas não possui apenas uma destas características acima relacionadas, mas várias delas ao mesmo tempo. Sistemas de informações de hoje, por exemplo, podem ter tanto características distribuídas como real-time. E a UML suporta modelagens de todos estes tipos de sistemas.

### **5.3. Fases do Desenvolvimento de um Sistema em UML**

Existem cinco fases no desenvolvimento de sistemas de software: análise de requisitos, análise, design (projeto), programação e testes. Estas cinco fases não devem ser executadas na ordem descrita acima, mas



concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente de forma que o resultado global gere um produto de alta qualidade e performance. A seguir falaremos sobre cada fase do desenvolvimento de um sistema em UML:

### 5.3.1. Análise de Requisitos

Esta fase captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas "use-cases". Através do desenvolvimento de "use-case", as entidades externas ao sistema (em UML chamados de "atores externos") que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem, funções estas chamadas de "use-cases". Os atores externos e os *use-cases* são modelados com relacionamentos que possuem comunicação associativa entre eles ou são desmembrados em hierarquia. Cada *use-case* modelado é descrito através de um texto, e este especifica os requerimentos do ator externo que utilizará este *use-case*. O diagrama de *use-cases* mostrará o que os atores externos, ou seja, os usuários do futuro sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade sem importar como esta será implementada. A análise de requisitos também pode ser desenvolvida baseada em processos de negócios, e não apenas para sistemas de *software*.

### 5.3.2. Análise

A fase de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de Classe. As colaborações entre



classes também são mostradas neste diagrama para desenvolver os *use-cases* modelados anteriormente, estas colaborações são criadas através de modelos dinâmicos em UML. Na análise, só serão modeladas classes que pertençam ao domínio principal do problema do *software*, ou seja, classes técnicas que gerenciem banco de dados, interface, comunicação, concorrência e outros não estarão presentes neste diagrama.

### 5.3.3. Design (Projeto)

Na fase de *design*, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros. As classes do domínio do problema modeladas na fase de análise são mescladas nessa nova infra-estrutura técnica tornando possível alterar tanto o domínio do problema quanto à infra-estrutura. O *design* resulta no detalhamento das especificações para a fase de programação do sistema.

### 5.3.4. Programação

Na fase de programação, as classes provenientes do design são convertidas para o código da linguagem orientada a objetos escolhida (a utilização de linguagens procedurais é extremamente não recomendada). Dependendo da capacidade da linguagem usada, essa conversão pode ser uma tarefa fácil ou muito complicada. No momento da criação de modelos de análise e *design* em UML, é melhor evitar traduzi-los mentalmente em código. Nas fases anteriores, os modelos criados são o significado do entendimento e da estrutura do sistema, então, no momento da geração do código onde o analista conclua antecipadamente sobre modificações em seu conteúdo, seus modelos não estarão mais demonstrando o real perfil do sistema. A



programação é uma fase separada e distinta, onde os modelos criados são convertidos em código.

#### **5.3.5. Testes**

Um sistema normalmente é rodado em testes de unidade, integração, e aceitação. Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador. Os testes de integração são aplicados já usando as classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos. Os testes de aceitação observam o sistema como uma "caixa preta" e verificam se o sistema está funcionando como o especificado nos primeiros diagramas de *use-cases*.

O sistema será testado pelo usuário final e verificará se os resultados mostrados estão realmente de acordo com as intenções do usuário final.

#### **5.4. A Notação da Linguagem de Modelagem Unificada – UML**

Tendo em mente as cinco fases do desenvolvimento de *softwares*, as fases de análise de requisitos, análise e *design* utilizam-se, em seu desenvolvimento, cinco tipos de visões, nove tipos de diagramas e vários modelos de elementos que serão utilizados na criação dos diagramas e mecanismos gerais. Todos juntos modelam tanto a funcionalidade estática quanto dinâmica de um sistema.

Como mencionado no escopo deste trabalho, não foram utilizados todos os componentes especificados pela UML. Portanto, a fim de explicar apenas o que foi utilizado, só serão abordados os componentes que acrescentam algo ao entendimento do referido trabalho. São eles: diagramas de *use-cases*, diagramas de classes, e relacionamentos entre classes.



### 5.4.1. Diagramas

Os diagramas utilizados pela UML são compostos de nove tipos: diagrama de *use case*, de classes, de objeto, de estado, de seqüência, de colaboração, de atividade, de componente e o de execução.

Todos os sistemas possuem uma estrutura estática e um comportamento dinâmico. A UML suporta modelos estáticos (estrutura estática), dinâmicos (comportamento dinâmico) e funcional. A Modelagem estática é suportada pelo diagrama de classes e de objetos, que consiste nas classes e seus relacionamentos. Os relacionamentos podem ser de associações, herança (generalização), dependência ou refinamentos. As modelagens dinâmicas são suportadas pelos diagramas de estado, seqüência, colaboração e atividade. E a modelagem funcional é suportada pelos diagramas de componente e execução. Abordar-se-á, apenas os diagramas utilizados no projeto:

#### 5.4.1.1. Diagrama Use-Case

A modelagem de um diagrama use-case é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. Eles são escritos em termos de atores externos, use-cases e o sistema modelado. Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado. Os atores iniciam a comunicação com o sistema através dos use-cases, onde o use-case representa uma seqüência de ações executadas pelo sistema e recebe do ator que lhe utiliza dados tangíveis de um tipo ou formato já conhecido, e o valor de resposta da execução de um use-case (conteúdo) também já é de um tipo conhecido, tudo isso é definido juntamente com o use-case através de texto de documentação.

Atores e *use-cases* são classes. Um ator é conectado a um ou mais use-cases através de associações, e tanto atores quanto use-cases podem



possuir relacionamentos de generalização que definem um comportamento comum de herança em superclasses especializadas em subclasses.

O uso de use-cases em colaborações é muito importante, onde estas são a descrição de um contexto mostrando classes/objetos, seus relacionamentos e sua interação exemplificando como as classes/objetos interagem para executar uma atividade específica no sistema. Uma colaboração é descrita por diagramas de atividades e um diagrama de colaboração.

Quando um *use-case* é implementado, a responsabilidade de cada passo da execução deve ser associada às classes que participam da colaboração, tipicamente especificando as operações necessárias dentro destas classes juntamente com a definição de como elas irão interagir. Um cenário é uma instância de um *use-case*, ou de uma colaboração, mostrando o caminho específico de cada ação. Por isso, o cenário é um importante exemplo de um *use-case* ou de uma colaboração. Quando visto ao nível de um *use-case*, apenas a interação entre o ator externo e o *use-case* é vista, mas já observando ao nível de uma colaboração, toda as interações e passos da execução que implementam o sistema serão descritos e especificados.

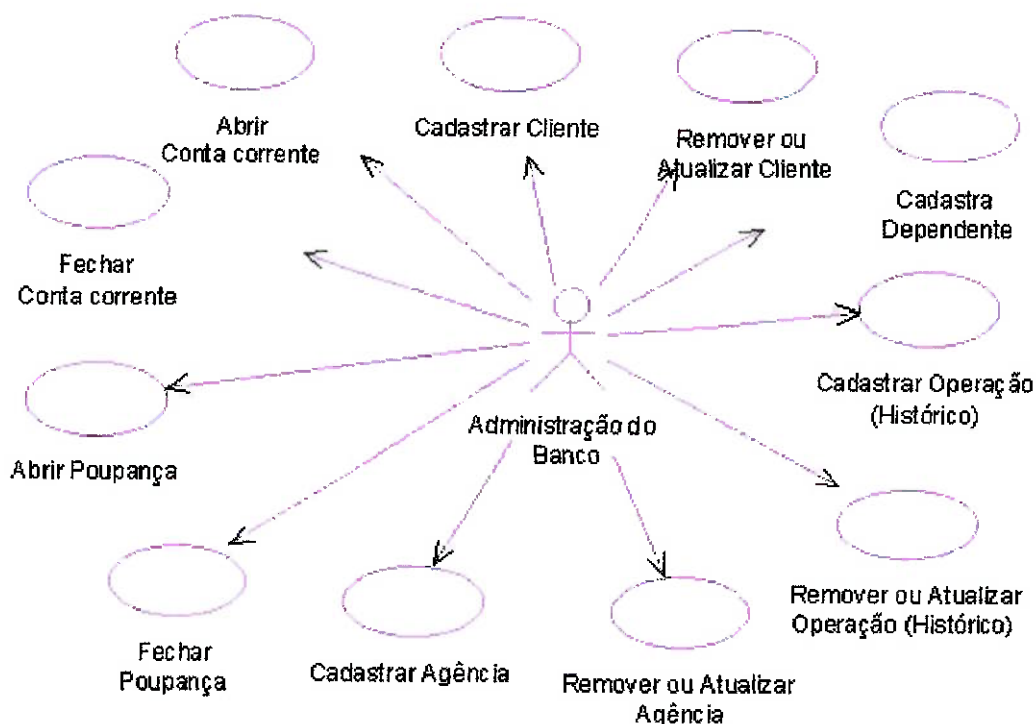


Figura 5: Diagrama de use-case

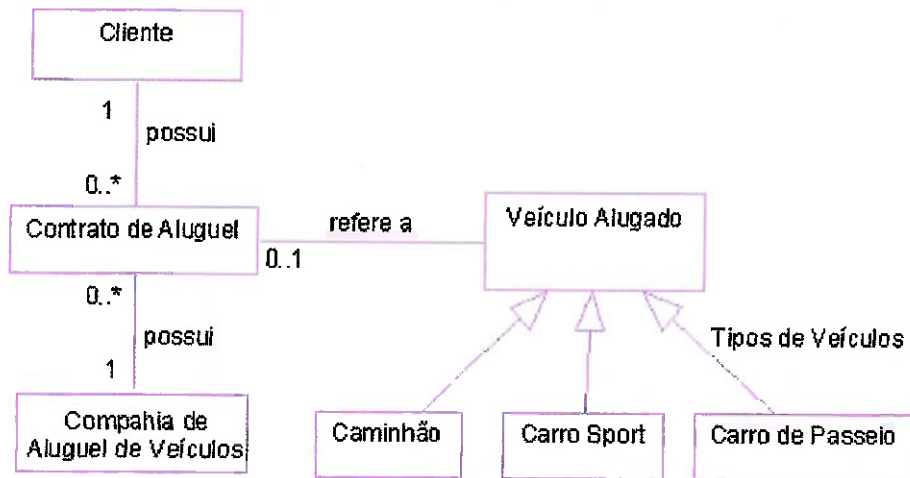




O diagrama de *use-cases* acima demonstra as funções de um ator externo de um sistema de controle bancário de um banco fictício. O diagrama especifica que funções o administrador do banco poderá desempenhar. Pode-se perceber que não existe nenhuma preocupação com a implementação de cada uma destas funções, já que este diagrama apenas se resume a determinar que funções deverão ser suportadas pelo sistema modelado.

#### 5.4.1.2. Diagrama de Classes

O diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam as "coisas" que são gerenciadas pela aplicação modelada. Classes podem se relacionar com outras através de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares). Todos estes relacionamentos são mostrados no diagrama de classes juntamente com as suas estruturas internas, que são os atributos e operações. O diagrama de classes é considerado estático já que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema. Um sistema normalmente possui alguns diagramas de classes, já que não são todas as classes que estão inseridas em um único diagrama e uma certa classe pode participar de vários diagramas de classes.



**Figura 6: Diagrama de classe**

Uma classe num diagrama pode ser diretamente implementada utilizando-se uma linguagem de programação orientada a objetos que tenha suporte direto para construção de classes. Para criar um diagrama de classes, as classes têm que estar identificadas, descritas e relacionadas entre si.

#### 5.4.1.3. Diagrama de Seqüência

Um diagrama de seqüência mostra a colaboração dinâmica entre os vários objetos de um sistema. O mais importante aspecto deste diagrama é que a partir dele percebe-se a seqüência de mensagens enviadas entre os objetos. Ele mostra a interação entre os objetos, alguma coisa que acontecerá em um ponto específico da execução do sistema. O diagrama de seqüência consiste em um número de objetos mostrado em linhas verticais. O decorrer do tempo é visualizado observando-se o diagrama no sentido vertical de cima para baixo. As mensagens enviadas por cada objeto são simbolizadas por setas entre os objetos que se relacionam.

Diagramas de seqüência possuem dois eixos: o eixo vertical, que mostra o tempo e o eixo horizontal, que mostra os objetos envolvidos na seqüência de uma certa atividade. Eles também mostram as interações para um cenário específico de uma certa atividade do sistema.



No eixo horizontal estão os objetos envolvidos na seqüência. Cada um é representado por um retângulo de objeto (similar ao diagrama de objetos) e uma linha vertical pontilhada chamada de linha de vida do objeto, indicando a execução do objeto durante a seqüência, como exemplo citam-se: mensagens recebidas ou enviadas e ativação de objetos. A comunicação entre os objetos é representada como linha com setas horizontais simbolizando as mensagens entre as linhas de vida dos objetos. A seta especifica se a mensagem é síncrona, assíncrona ou simples. As mensagens podem possuir também números seqüenciais, eles são utilizados para tornar mais explícito as seqüências no diagrama.

Em alguns sistemas, objetos rodam concorrentemente, cada um com sua linha de execução (thread). Se o sistema usa linhas concorrentes de controle, isto é mostrado como ativação, mensagens assíncronas, ou objetos assíncronos.

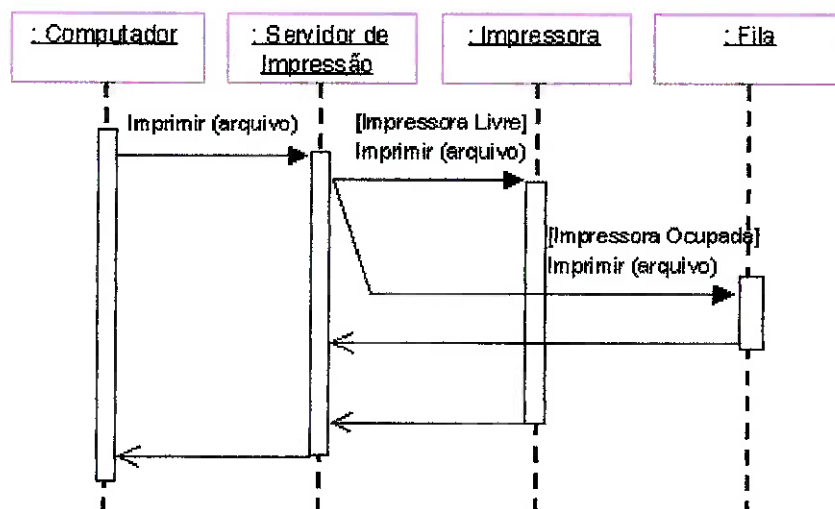


Figura 7: Diagrama de Seqüência

Os diagramas de seqüência podem mostrar objetos que são criados ou destruídos como parte do cenário documentado pelo diagrama. Um objeto pode criar outros objetos através de mensagens. A mensagem que cria ou destrói um objeto é geralmente síncrona, representada por uma seta sólida.



## **6. Sistema de Controle de Ordens de Chão de Fábrica**

É possível notar que um sistema MES possui uma funcionalidade muito ampla, suas onze funções fornecem informações para fazer funcionar quase qualquer tipo de fábrica. Sendo assim, implementar um sistema MES completo é uma tarefa árdua e grande demais para ser executada por apenas uma pessoa nesse curto período de tempo. Portanto, este trabalho constará do projeto e implementação de apenas duas das onze funções descritas anteriormente, visto que essa implementação é suficiente para dar a funcionalidade que um sistema de ordens de chão de fábrica necessita. A seguir se segue uma descrição um pouco mais detalhada das duas funções a serem implementadas.

### **6.1. Funções a serem implementadas**

#### **6.1.1. Alocação de Recursos**

Essa função gerencia recursos, que no caso do projeto em questão, diz respeito somente às máquinas da empresa, que precisam estar disponíveis na ordem para a operação começar a ser executada. Ela fornece a história detalhada dos recursos e garante que o equipamento está corretamente configurado para o processo e fornece o status em tempo real. O gerenciamento desses recursos inclui reserva e despacho para atingir os objetivos do cronograma de operações.



### **6.1.2. Cronograma de Operações**

Essa função fornece um sequenciamento baseado em prioridades, atributos, características e/ou receitas associadas às unidades de produção numa operação tal como, seqüência de cores ou outras características que, quando programadas na seqüência correta, minimiza o *set up*. Ele é limitado, porém é capaz de calcular em detalhe tempos exatos ou carregamento.

## **6.2. Requisitos do Projeto**

### **6.2.1. Descrição geral**

Tendo em vista a idéia geral do que significa cada uma das funções principais do sistema em questão, é possível traçar quais são os requisitos do projeto a fim de se chegar ao objetivo estipulado, que é o controle de ordens de chão de fábrica.

#### *6.2.1.1. Projeto do banco de dados*

Durante a execução do sistema, todas as informações necessárias estarão armazenadas em banco de dados *MS Access* que deverá ser a fonte de informações utilizada pelo sistema.

#### *6.2.1.2. Desenvolvimento dos aplicativos*

O sistema será escrito na linguagem Java, utilizando-se para isso a plataforma J2EE, já discutida em itens anteriores. Optou-se por esta linguagem e esta plataforma por ser esta bastante dedicada ao projeto e



implementação de sistemas empresariais, como é o caso do trabalho em questão.

#### *6.2.1.3. Design das interfaces do sistema com o usuário*

Existirão telas de interface para o usuário poder entrar com todos os dados necessários ao funcionamento do programa e também telas de saída das informações consultadas pelo usuário.

Elas serão escritas em HTML, utilizando-se para isso a tecnologia Java Servlet, devido a este sistema poder ser operado via rede.

### **6.2.2. Requisitos Não-Funcionais**

#### *6.2.3.1. Ambiente de execução*

A máquina utilizada para execução do projeto será um PC *Pentium* III (366 MHz) de 64 Mbyte de memória RAM.

O sistema operacional será o MS *Windows* 98. O próprio processador representa o cliente, o servidor J2EE e o servidor EIS, embora para a apresentação possa ser montado um conjunto de processadores para demonstração da tecnologia da plataforma J2EE.

#### *6.2.3.2. Interfaces com outros sistemas*

Não é previsto nenhum tipo de interface com outros sistemas.

#### *6.2.3.3. Desempenho*

O desempenho do sistema vem do poder de processamento da máquina utilizada, pois que não há nenhum requisito de projeto para esta característica.



#### 6.2.3.4. Escalabilidade

Este projeto constitui apenas da implementação de duas das funções de um Sistema MES. Seria possível adicionar mais características funcionais ao programa, porém a alteração de política de estoques e política de produção (sob encomenda) é inviável pois alteram os conceitos básicos de projeto.

#### 6.2.3.5. Segurança e Privacidade (*security e privacy*)

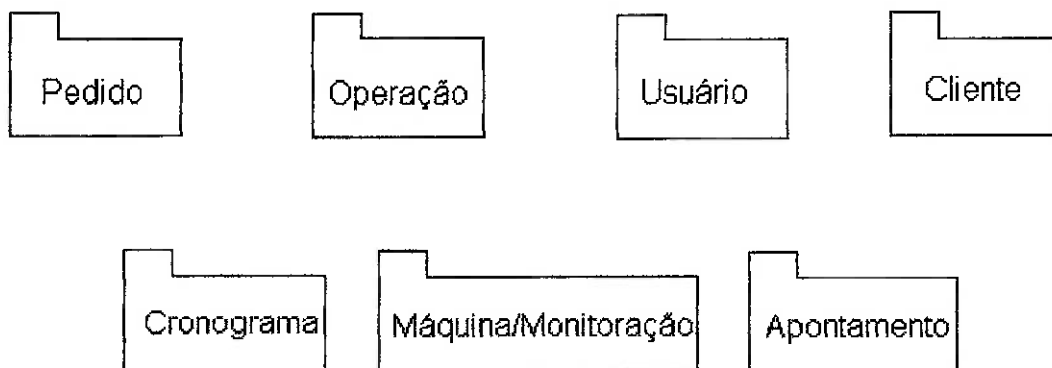
Existirá no sistema um controle de acesso de usuários simplificado, em que todos os usuários cadastrados terão acesso a todo o sistema. Não haverá, portanto, seções do sistema em que apenas alguns usuários terão acesso.



## 7. Especificação UML

Como já foi dito, não se tem como objetivo neste projeto, fazer uma especificação UML completa do sistema. Apenas foram feitos os diagramas de use-cases e de classes, com os quais é possível se ter o entendimento da funcionalidade do sistema, com intuito de restar para a próxima etapa do projeto apenas a implementação do mesmo em um estágio avançado.

Devido à complexidade do sistema, este foi dividido em pacotes, que são:

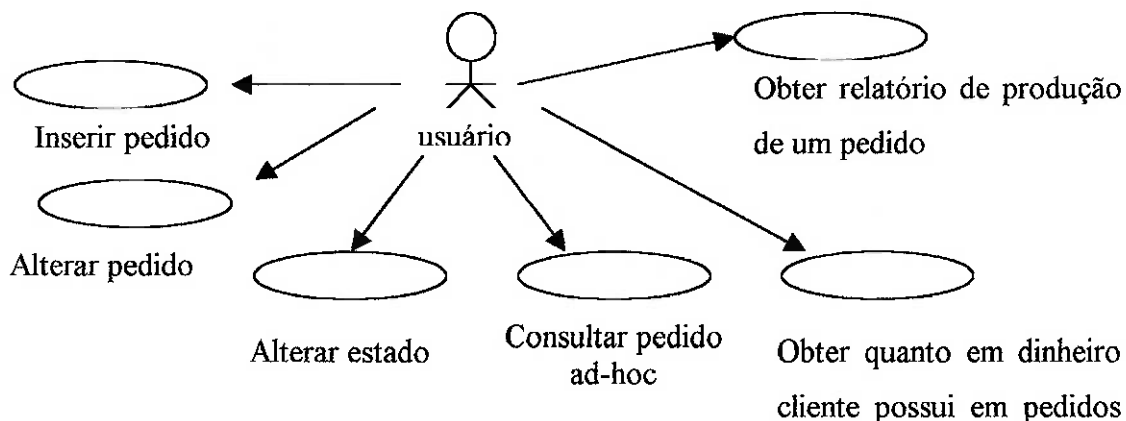






## 7.1. Diagramas de use-case

### 7.1.1. Pedido



#### 7.1.1.1. Inserir Pedido

Insere um pedido

##### Atributos:

Numero do pedido: gerado automaticamente  
Código de cliente: selecionado  
Estado  
Data de entrega  
Local de entrega  
Preço de venda  
Descrição de cada serviço  
Quantidade de cada serviço  
Operações de cada serviço

##### Regras de negócio:

Não será possível inserir um pedido com preço de venda menor que R\$ 100.00

Todos os campos devem ser preenchidos com pelo menos um serviço, com pelo menos uma operação.

#### 7.1.1.2. Alterar Pedido

Altera atributos de um pedido cadastrado

##### Atributos:

Os mesmos de "inserir pedido"



Regras de negócio:

Um pedido não pode ser alterado depois de ser programado ou entrar em execução, ou seja, só pode ser alterado se seu estado for "solicitado".

*7.1.1.3. Alterar estado*

Altera estado de um pedido

Atributos:

Número do pedido

Estado, que pode ser: solicitado, programado, em execução, pronto, entregue e cancelado

Regras de negócio:

O usuário só pode cancelar e dar baixa. As outras mudanças são automáticas

Não pode cancelar pedido entregue nem dar baixa em pedido cancelado

*7.1.1.4. Consultar pedido ad-hoc*

Permite ao usuário visualizar dados sobre o pedido. Permite ainda a utilização de filtros programados pelo usuário, por exemplo, por estado.

*7.1.1.5. Obter quanto em dinheiro o cliente possui em pedidos em aberto*

É considerado pedido em aberto todos os não entregues nem cancelados. Mostra o detalhamento dos pedidos do cliente e mostra o total a creditar.

Atributos:

Código do cliente

*7.1.1.6. Obter relatório de produção de um pedido*

Mostra o detalhamento de um pedido com seus apontamentos, se estes existirem, e as datas programadas.

Atributos:

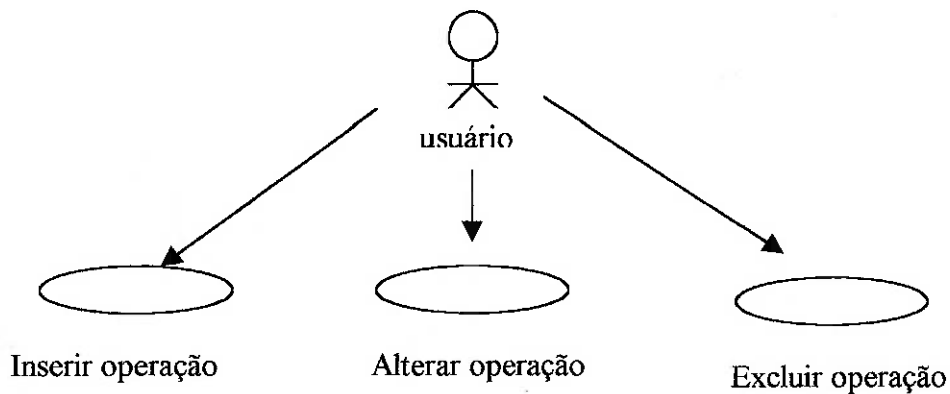
Código do pedido

Regras de negócio:

Caso o pedido não esteja nem ao menos programado, o usuário será informado disso e não será gerado relatório



## 7.1.2. Operação



### 7.1.2.1. Inserir Operação

Insere um *template* de operação

Atributos:

Código da operação: gerado automaticamente  
Conjunto de recursos: selecionado  
Descrição da operação  
Tempo por mil unidades

Regras de negócio:

Todos os campos devem ser preenchidos com pelo menos um recurso.

### 7.1.2.2. Alterar Operação

Altera atributos de um *template* de operação cadastrado

Atributos:

Os mesmos de "inserir operação"

### 7.1.2.3. Excluir Operação

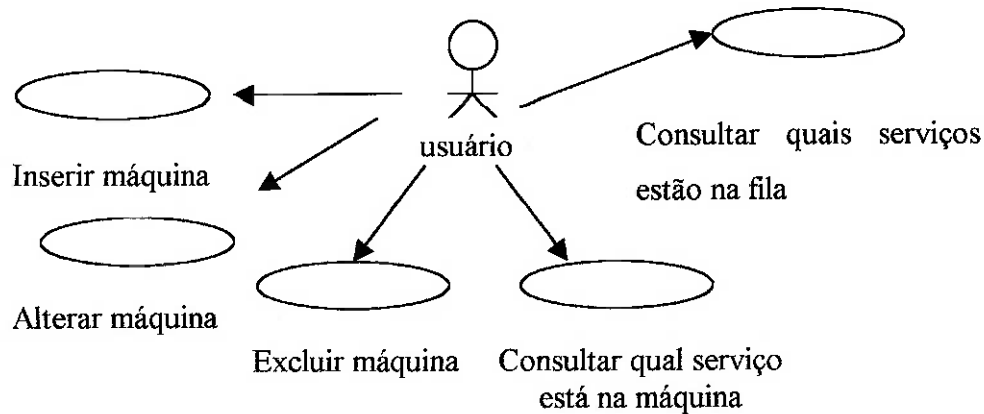
Exclui uma operação cadastrada

Atributos:

Código da operação



### 7.1.3. Máquina/ Monitoração de máquina



#### 7.1.3.1. Inserir Máquina

Insere uma máquina

Atributos:

Código da máquina: gerado automaticamente  
Descrição da máquina  
Tempo de setup

Regras de negócio:

Todos os campos devem ser preenchidos.

#### 7.1.3.2. Alterar Máquina

Altera atributos de uma máquina cadastrada

Atributos:

Os mesmos de "inserir máquina"

#### 7.1.3.3. Excluir Máquina

Exclui uma máquina cadastrada

Atributos:

Código da máquina

#### 7.1.3.4. Consultar qual serviço está na máquina

Mostra qual serviço de qual pedido está na máquina no momento

Atributos:



Código da máquina

Regras de negócio:

Podem ser feitas quantas consultas quiser.

*7.1.3.5. Consultar quais serviços estão na fila*

Mostra qual a ordem dos serviços a serem executados na máquina no momento da consulta

Atributos:

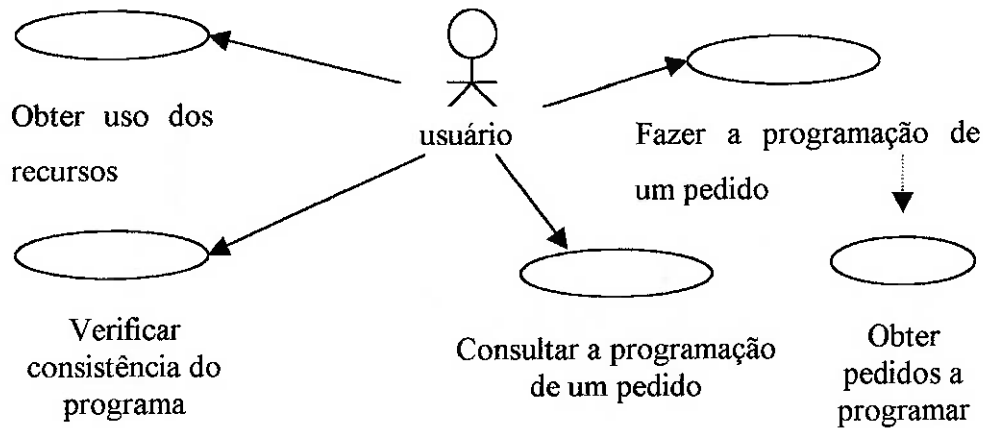
Código da máquina

Regras de negócio:

Podem ser feitas quantas consultas quiser.



#### 7.1.4. Cronograma de Produção



##### 7.1.4.1. Obter uso dos recursos

Mostra a quantos por cento da capacidade de uma máquina está sendo utilizada pelos pedidos programados.

Atributos:

Código da máquina  
Capacidade da máquina [h/dia], por exemplo, 8 h

##### 7.1.4.2. Verificar consistência do programa

Verifica se as datas escolhidas no momento da programação feita pelo usuário estão corretas no seguinte sentido:

Verifica se a data de início para uma determinada operação não é anterior à data atual do sistema

Verifica se duas operações subsequentes não estão sobrepostas

Atributos:

Data de início prevista de uma operação  
Data de término prevista de uma operação  
Data atual do sistema

##### 7.1.4.3. Consultar a programação de um pedido

Mostra a data de início prevista e a data de término previsto de um serviço de um determinado pedido.

Atributos:

Código do pedido



#### *7.1.4.4. Fazer a programação de um pedido*

A programação é manual. Sendo assim as datas são escolhidas pelo usuário, que deve executar o seguinte processo para cada operação:

- 1) escolher pedido
- 2) escolher serviço
- 3) escolher operação

Então o sistema mostra o cronograma atual dos recursos capazes de executar a operação

- 4) escolhe a data

O sistema mostra o cronograma atualizado

#### Atributos:

Código do pedido

#### Regras de negócio:

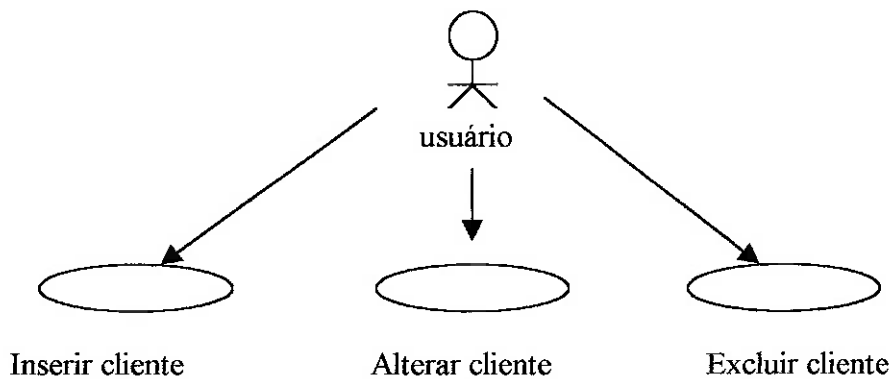
Todas as operações ainda não apontadas podem ser programadas e reprogramadas quantas vezes o usuário achar necessário.

#### *7.1.4.5. Obter Pedidos a programar*

Mostra a lista dos pedidos solicitados que ainda não foram programados. Esta função serve de apoio ao usuário no momento da programação da produção



### 7.1.5. Cliente



#### 7.1.5.1. Inserir Cliente

Cadastra um cliente

Atributos:

Código do cliente: gerado automaticamente  
Nome do cliente  
Nome do contato  
Telefone  
Fax  
Endereço principal

Regras de negócio:

Todos os campos devem ser preenchidos. Sendo que fax é de preenchimento opcional.

#### 7.1.5.2. Alterar Cliente

Altera atributos de um cliente cadastrado

Atributos:

Os mesmos de "inserir cliente"

#### 7.1.5.3. Excluir Cliente

Exclui um cliente cadastrado

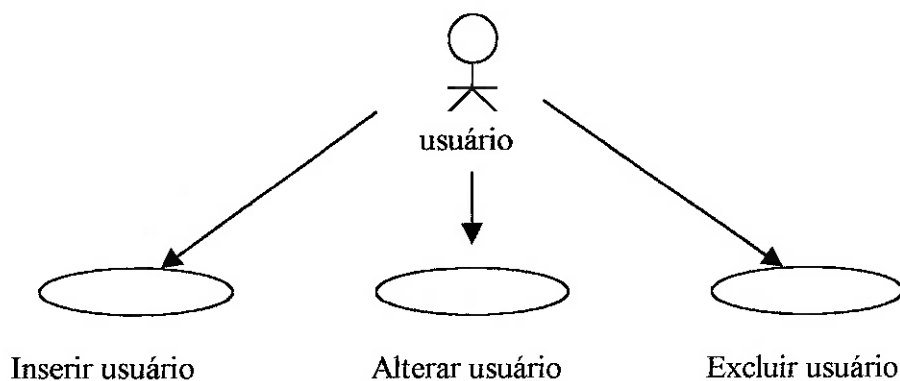
Atributos:

Código do cliente





### 7.1.6. Controle de Acesso



#### 7.1.6.1. Inserir Usuário

Insere um usuário

Atributos:

Login  
Senha  
Nome completo  
Setor da empresa em que trabalha

Regras de negócio:

Todos os campos devem ser preenchidos.  
Não pode haver dois usuários com o mesmo *login*

#### 7.1.6.2. Alterar Usuário

Altera atributos de um usuário cadastrado

Atributos:

Os mesmos de "inserir usuário"

#### 7.1.2.3. Excluir Usuário

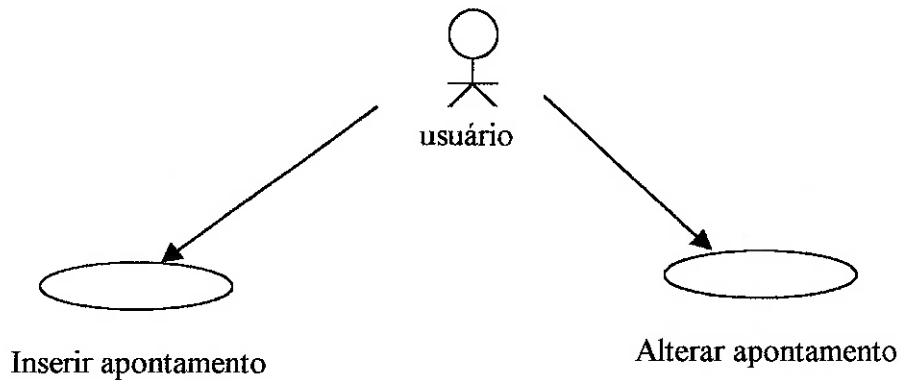
Exclui um usuário cadastrado

Atributos:

Login do usuário



### 7.1.7. Apontamento



#### 7.1.7.1. Inserir Apontamento

Insere um apontamento

Atributos:

Operador  
Número do pedido para o qual será feito o apontamento  
Serviço para o qual será feito o apontamento  
Operação para a qual será feito o apontamento  
Data de início efetiva  
Data de término efetiva  
Quantidade útil produzida  
Quantidade de perda gerada  
Observação

Regras de negócio:

Não pode ser inserido mais de um apontamento para a mesma operação de um mesmo serviço

#### 7.1.7.2. Alterar Apontamento

Altera atributos de um apontamento

Atributos:

Os mesmos de "inserir apontamento"



Geralmente, para cada *use case* existe pelo menos uma tela. Nesse sentido, a partir dos *use cases* determinados anteriormente, foi feito o esboço das telas de interface do sistema com o usuário. Estas telas são de grande auxílio tanto no entendimento do sistema em si como também na elaboração dos diagramas de sequência.

The top screenshot shows a web browser window titled "Control de Ordens - Login [Início] - Microsoft Internet Explorer". The address bar shows "http://localhost:8080/samples/login.htm". The page content includes a login form with fields for "Login:" and "Senha:" and a button labeled "Enviar consulta".

The bottom screenshot shows a web browser window titled "Control de Ordens Menu Principal - Microsoft Internet Explorer". The address bar shows "http://localhost:8080/samples/menuet/login". The page content includes a "Menu Principal" section with a list of links: "Inicio", "Inscrição", "Alteração de Senha", "Consultas de Produtos", "Produtos", "Clientes", and "Aplicativos".

46



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

### 7.2.1. Telas de Pedido

A tela a que dá acesso o menu de pedido é:

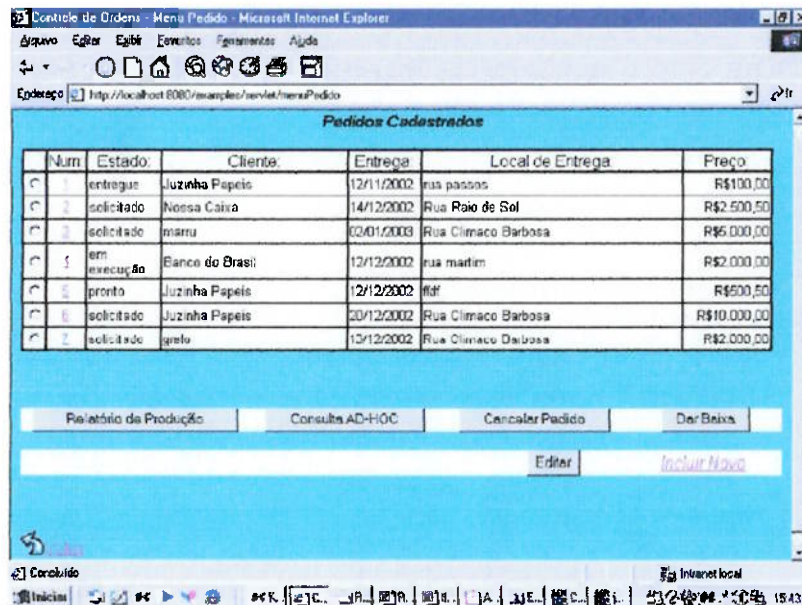


Figura 9: Tela geral de pedidos

A partir desta tela o usuário, selecionando um pedido listado, pode obter seu relatório de produção, cancela-lo, altera-lo ou visualizar seus atributos. Ainda é possível incluir um novo pedido ou fazer uma consulta ad-hoc. O botão "Relatório de Produção do Pedido" dá acesso à seguinte tela:

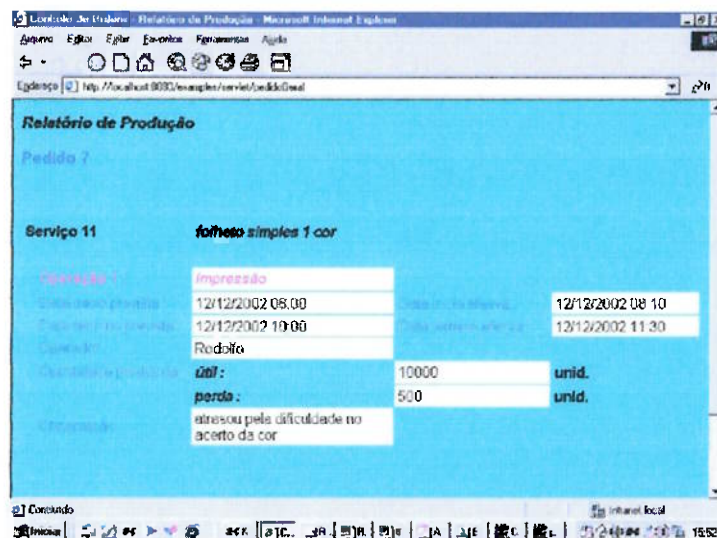


Figura 10: Tela de relatório de produção de um pedido



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

O botão "Consultar Pedido ad-hoc" leva a:

Figura 11: Tela de consulta ad-hoc

O botão "Cancelar pedido" dá acesso a:

Figura 12: Tela de aviso de cancelar pedido, em caso afirmativo





PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

O botão “Incluir Novo” leva a:

Figura 13: Tela geral de inserir pedido

O botão “Inserir novo serviço” conduz à seguinte tela:

Figura 14: Tela de inserir serviço no pedido

Os botões “Editar” levam a telas iguais às telas de inserir.



## 7.2.2. Telas de Operação

O menu de operação leva à seguinte tela:

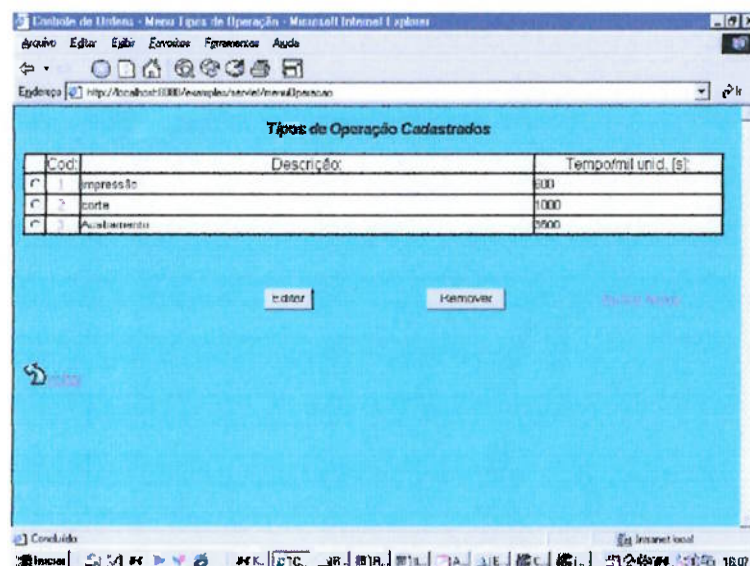


Figura 15: Tela geral de operação

Esta tela, que lista todos os *templates* de operação cadastrados, possui o botão "Inserir Nova" que leva à tela a seguir, que é a mesma de alterar:

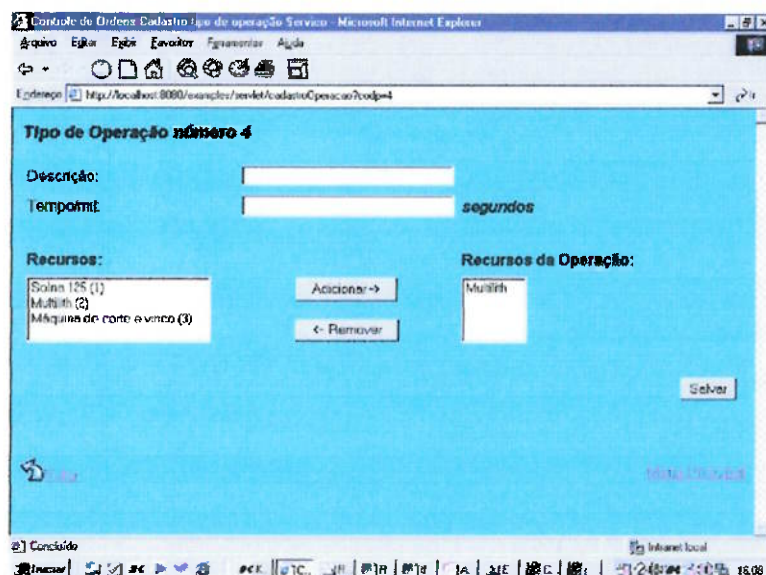


Figura 16: Tela de inserir operação



### 7.2.3. Telas de Máquina/Monitoração

O menu de máquina/monitoração leva à seguinte tela:

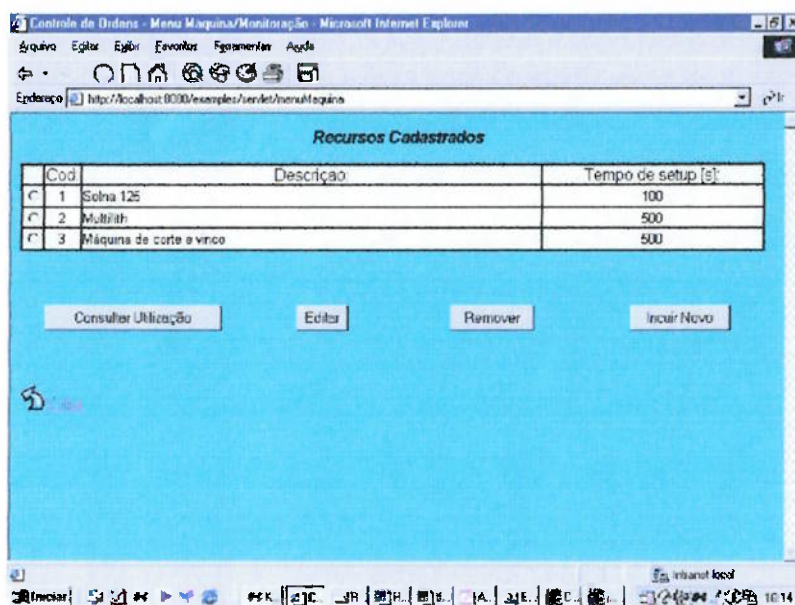


Figura 17: Tela geral de máquina/monitoração

O botão “Inserir Nova” abre a seguinte tela:

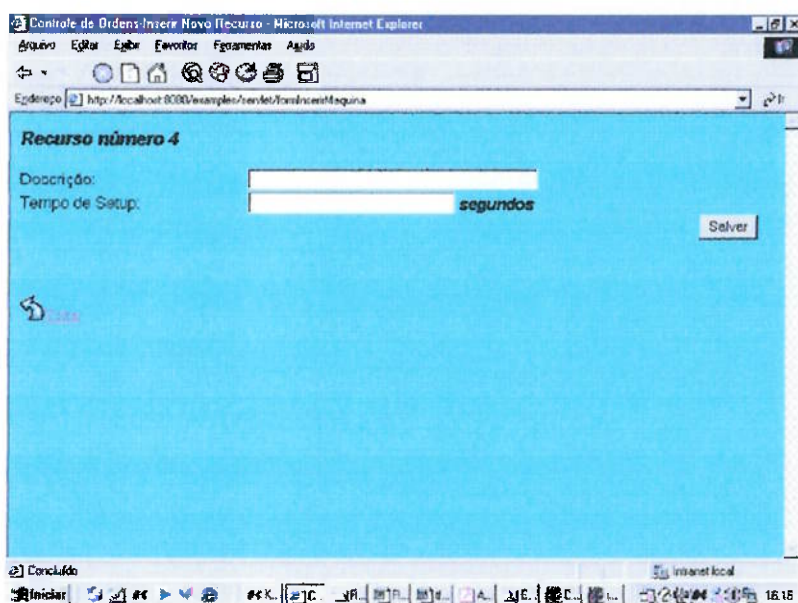


Figura 18: Tela de inserir máquina

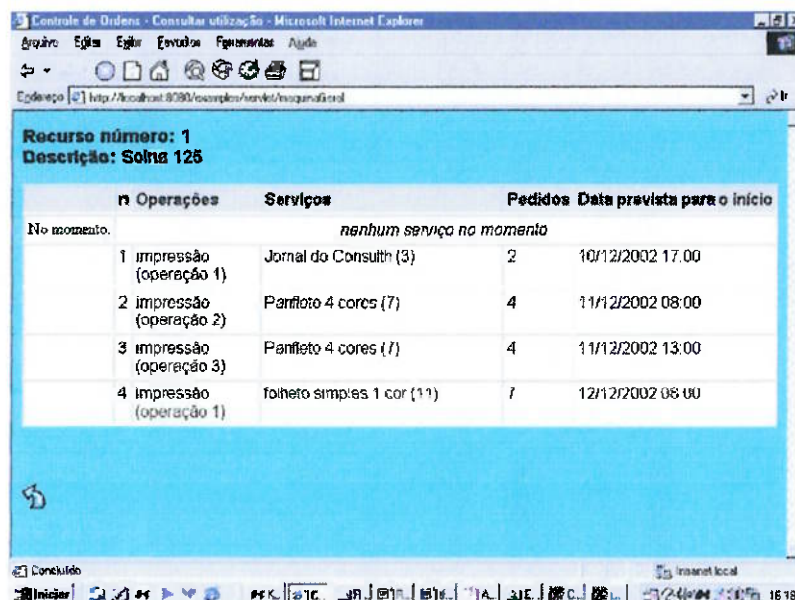
O botão “Editar” leva a uma tela semelhante.





**PMC581-Projeto Mecânico II**  
**Mecatrônica**  
**Escola Politécnica da Universidade de São Paulo**

O botão “Consultar utilização da Máquina”, quando há uma máquina selecionada, leva à seguinte tela:



The screenshot shows a web browser window titled "Controle de Ordens - Consultar utilização - Microsoft Internet Explorer". The address bar shows the URL "http://localhost:8080/exemplos/ordem/maquinas.html". The page content displays the following information:

**Recurso número: 1**  
**Descrição: Solna 125**

	n Operações	Serviços	Pedidos	Data prevista para o início
No momento.		<i>nenhum serviço no momento</i>		
	1 impressão (operação 1)	Jornal do Consultor (3)	2	10/12/2002 17:00
	2 impressão (operação 2)	Panfleto 4 cores (7)	4	11/12/2002 08:00
	3 impressão (operação 3)	Panfleto 4 cores (7)	4	11/12/2002 13:00
	4 impressão (operação 1)	folheto simples 1 cor (11)	7	12/12/2002 08:00

**Figura 19: Tela de consulta à utilização do recurso**



#### 7.2.4. Telas de Cronograma de Produção

O menu de cronograma de produção leva a seqüência de telas onde se definem o pedido, o serviço e a operação que se deseja programar.

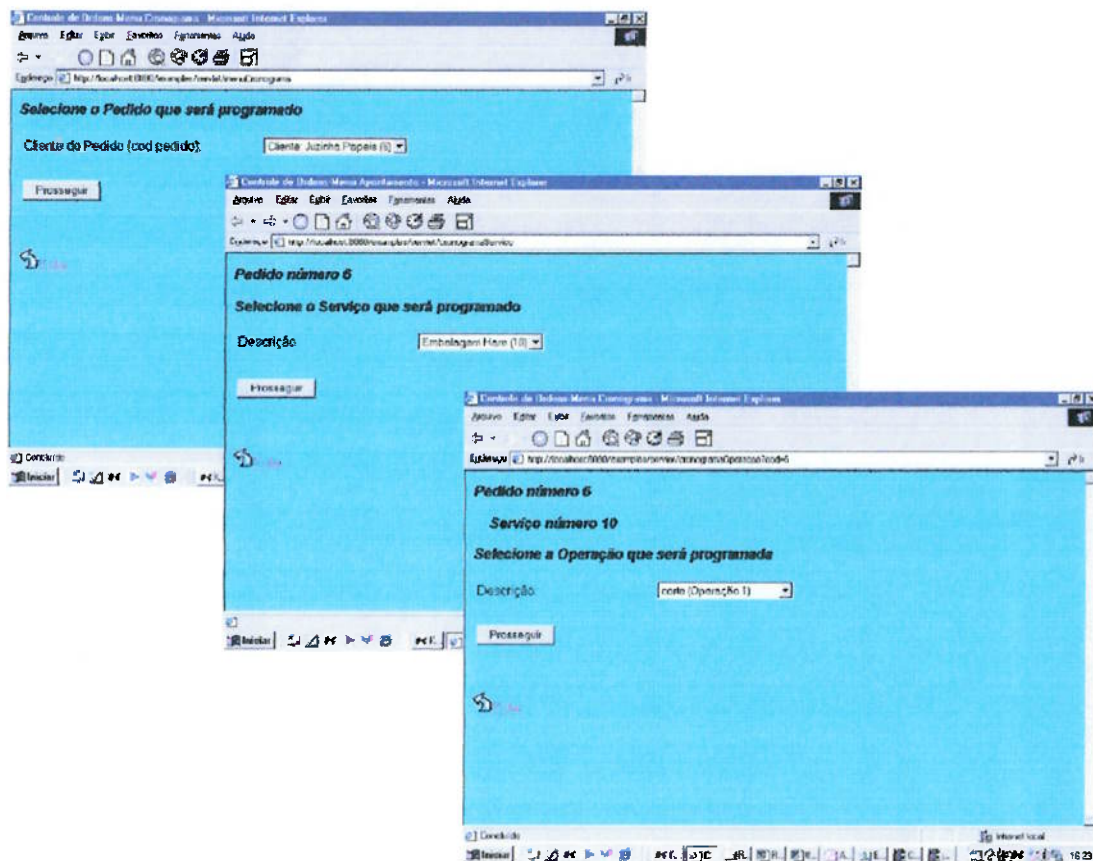


Figura 20: Telas de seleção de pedido, serviço e operação a programar



**PMC581-Projeto Mecânico II**  
**Mecatrônica**  
Escola Politécnica da Universidade de São Paulo

O botão "Prosseguir" da tela acima abre:

**Figura 21: Tela de programação de uma operação**

Nesta tela o usuário seleciona uma data de início através de um dos *radio buttons* e clica em "Programar". Esse procedimento abre a tela:

**Figura 22: Tela de cronograma atualizado**



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

### 7.2.5. Telas de Cliente

O menu de cliente leva à seguinte tela:

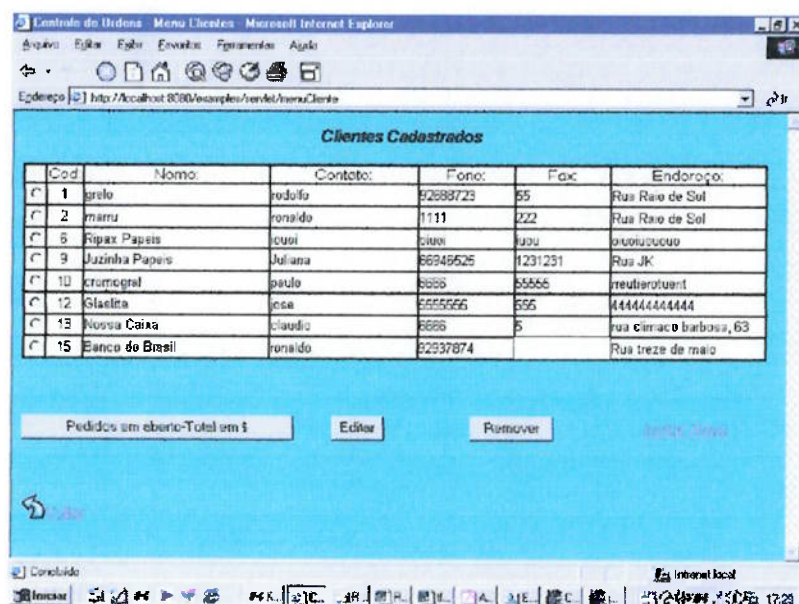


Figura 23: Tela geral de cliente

O botão "Inserir Novo" abre a tela:

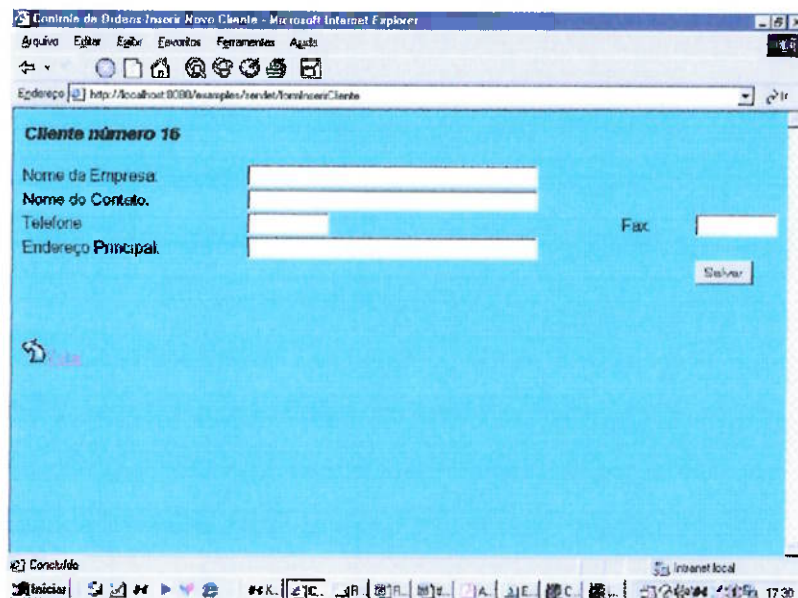


Figura 24: Tela de inserir cliente

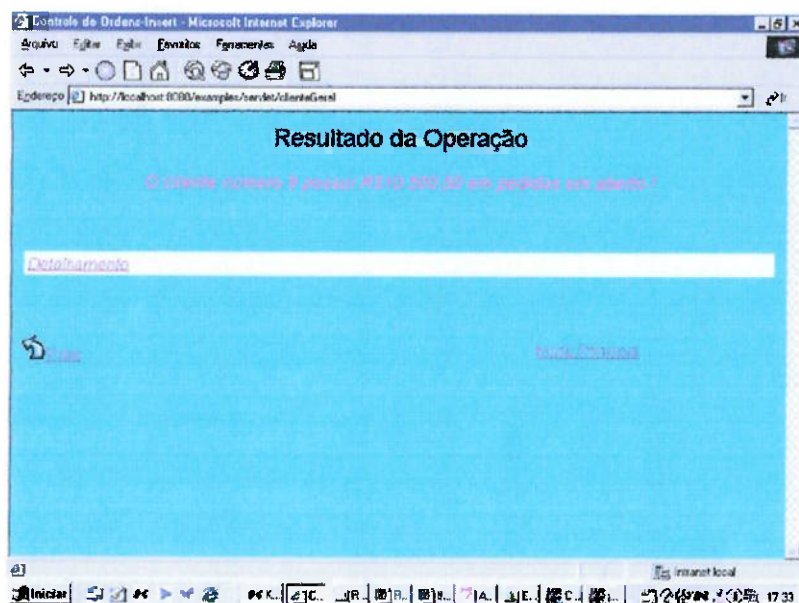
O botão "Editar" leva à tela semelhante.





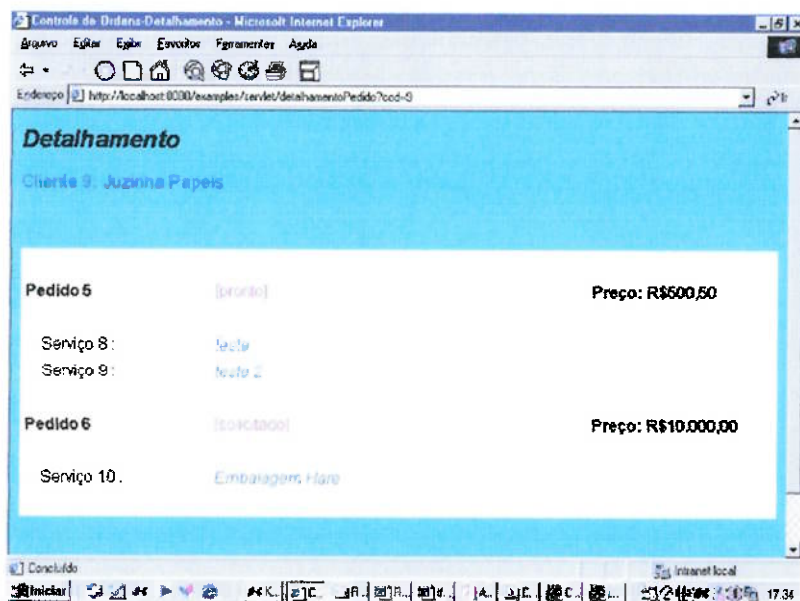
**PMC581-Projeto Mecânico II**  
**Mecatrônica**  
Escola Politécnica da Universidade de São Paulo

O botão “Total em \$\$\$ pedidos em aberto” é referente ao use case “obter quanto em dinheiro o cliente possui em pedidos em aberto”. Foi escolhido deixa-lo nesta tela para facilitar a compreensão do usuário. Esse botão abre a tela:



**Figura 25: Tela de total em dinheiro de pedidos em aberto**

O *hyperlink* “Detalhamento” abre a tela:



**Figura 26: Tela de detalhamento dos preços dos pedidos em aberto**



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

## 7.2.6. Telas de Usuário

O menu de usuário leva à seguinte tela:

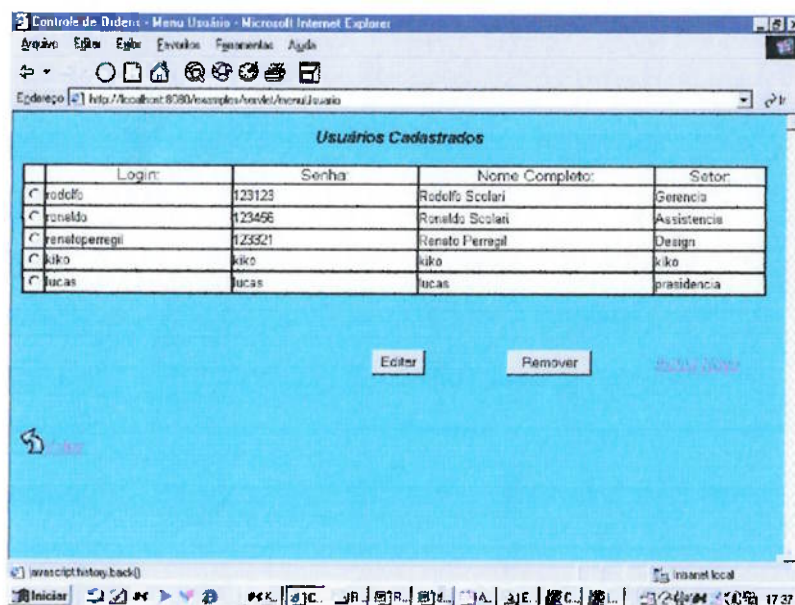


Figura 27: Tela geral de usuário

O botão "Inserir Novo" abre a tela:

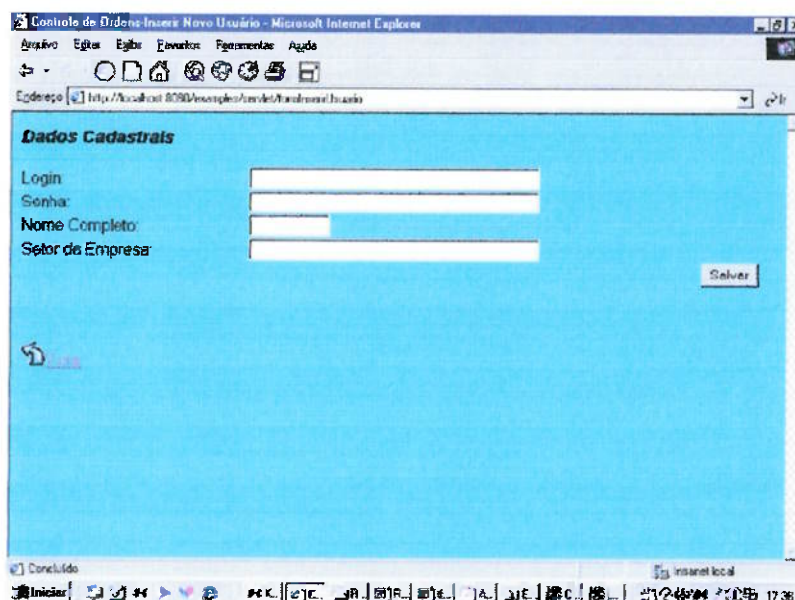


Figura 28: Tela de inserir usuário

O botão "Editar" abre tela semelhante à que está acima.



### 7.2.7. Telas de Apontamento

O menu de apontamento leva à sequência de telas abaixo:

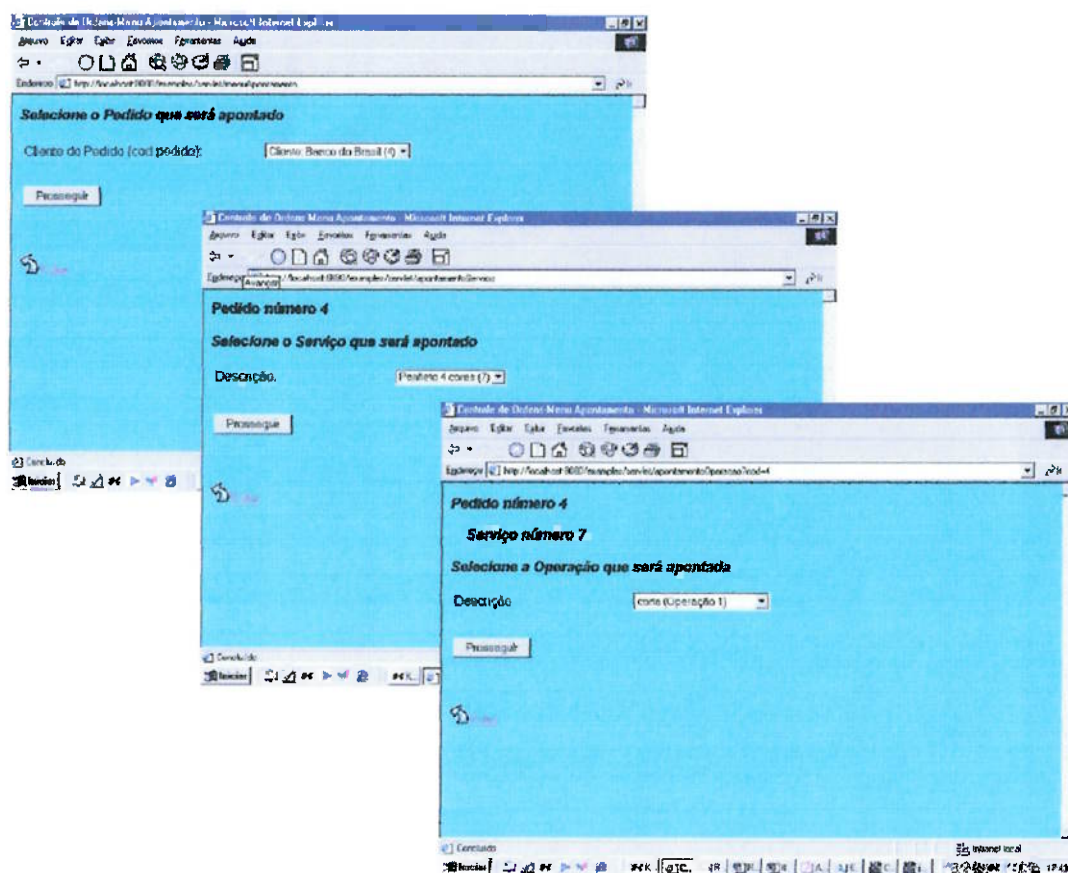


Figura 29: Telas de seleção da operação para a qual será feito o apontamento



PMC581-Projeto Mecânico II  
Mecatrônica  
Escola Politécnica da Universidade de São Paulo

O botão “Prosseguir” da ultima tela abre a seguinte tela, onde é feito o apontamento da operação selecionada pelo procedimento demonstrado acima.

Figura 31: Tela de apontamento de uma operação

Uma vez que o *checkbox* com a inscrição “fim da operação” é assinalado, o sistema não permite mais que o apontamento seja alterado.





### 7.3. Diagrama de Classes

Segue abaixo o diagrama de classes do sistema:

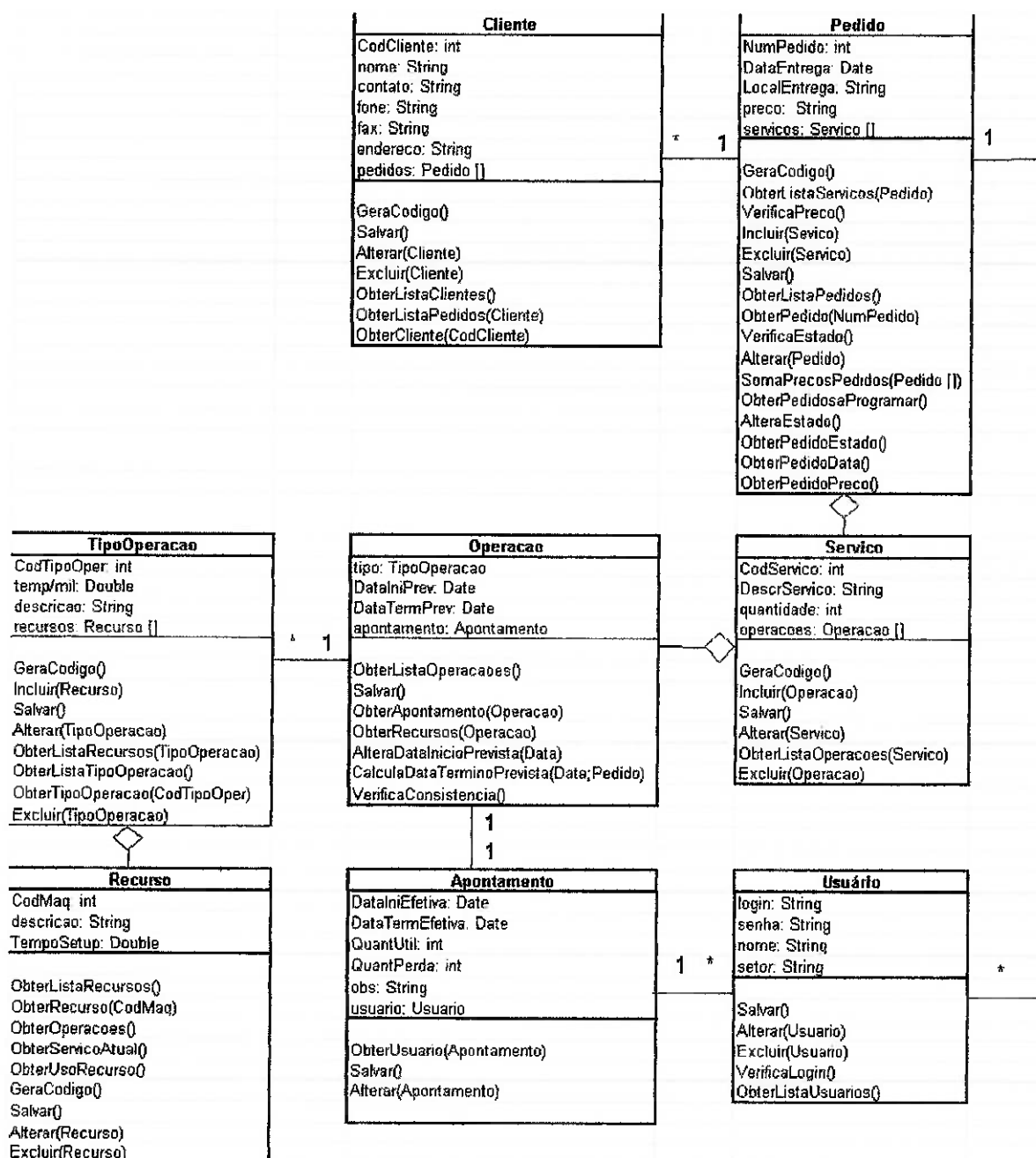


Figura 31: Diagrama de Classes do sistema



## 8. Referências Bibliográficas

1. *MES Explained: A High Level Vision*, MESA White Paper Number 6
2. *Controls Definition & MES to Controls Data Flow Possibilities*, MESA White Paper Nb.
3. Lemay, Laura; Cadenhead, Rogers. *Aprenda em 21 dias Java 2*. Rio de Janeiro. Campus, 1999.
4. Pawlan, Mônica. *Introduction to the J2EE Platform*. 2001.
5. Barros, Pablo. *UML – Tutorial*.



## Apêndice – Cronograma de Atividades

Ano: 2002

	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro
Java								
UML								
Requisitos								
Projeto								
Implementação								