9,7 (nove e dois)

**Escola Politécnica da Universidade de São Paulo**

**Departamento de Engenharia Mecânica**

**Automação e Sistemas**

# Projeto Mecânico II - Desenvolvimento de um MRP Didático

**Aluno: Fernando Machado de Figueiredo** **N.USP: 2367820**

**Professor Orientador: Dr. Marcos R. P. Barreto**
**Professor Coordenador: Dr. Lucas Antônio Moscato**

A Jujuba, pelo incentivo,
companhia, amor e carinho.

# AGRADECIMENTOS

Ao Professor Dr. MARCOS R. P. BARRETO por sua eficaz orientação no trabalho, colaboração e prontidão em todos os momentos em que lhe foi solicitado, os quais não foram poucos durante toda a execução do mesmo.

À família, cujo apoio foi fundamental nos momentos árduos pelos quais fui obrigado a passar e onde o aprendizado pode de fato ser dado.

Ao colega da turma ingressante em 1997, Ricardo Kenji, por sua ajuda importantíssima ao *design* das páginas de apresentação ao usuário.

Finalmente, a minha querida Juliana que me acompanhou em todos os finais de semana e feriados (infelizmente) utilizados para desenvolvimento do sistema.

# ÍNDICE

*ÍNDICE* ..... *i*
*LISTA DE FIGURAS* ..... *i*
*RESUMO* ..... *iii*
*ABSTRACT* ..... *iv*

*1 Introdução* ..... *1*
*2 Escopo do projeto* ..... *2*
*3 MRP – Planejamento das Necessidades de Materiais* ..... *3*
**3.1 Metas, Objetivos e Funções dos MRP** ..... 4
**3.2 Itens de baixo valor e o MRP** ..... 11
**3.3 MRP ao MRP II** ..... 12
**3.4 Aplicação do Conceito MRP** ..... 14
**3.4.1 Registro de Inventário** ..... 15
**3.4.2 Cálculo da demanda dependente – a árvore do produto** ..... 16
**3.4.3 Sumário da Lógica** ..... 20
*4 Tecnologia Servlet* ..... *22*
**4.1 Arquitetura do Sistema** ..... 24
**4.2 Outros tipos de utilização dos servlets** ..... 25
*5 Requisitos de Projeto* ..... *26*
**5.1 Introdução** ..... 28
**5.1.1 Nome do Documento** ..... 28
**5.1.2 Marcas** ..... 28
**5.1.3 Escopo** ..... 28
**5.1.4 Conceitos, terminologia e abreviaturas** ..... 29
**5.2 Descrição Geral** ..... 30
**5.2.1 Perspectiva Histórica** ..... 30
**5.2.2 Objetivos** ..... 31

**5.3 Requisitos Funcionais** ..... 33
5.3.1 **Cadastro** ..... 33
5.3.2 **Políticas de Estoque** ..... 34
5.3.3 **Entrada das necessidades** ..... 34
5.3.4 **Cálculo MRP** ..... 35
5.3.5 **Consulta Online** ..... 36

**5.4 Requisitos Não-Funcionais** ..... 37
5.4.1 **Ambiente de execução** ..... 37
5.4.2 **Interfaces com outros sistemas** ..... 37
5.4.3 **Desempenho** ..... 38
5.4.4 **Escalabilidade** ..... 38
5.4.5 **Segurança e Privacidade (security e privacy)** ..... 39

**5.5 Perspectiva Gerencial** ..... 40

**5.6 Controle de Revisão** ..... 41

**5.7 Histórico de Alterações** ..... 41

***6 Modelagem ER do Banco de dados e Tabelas Access*** ..... *42*

**6.1 Estrutura das tabelas de dados** ..... 43

***7 Desempenho do Sistema*** ..... *45*

**7.1 Barra de Opções** ..... 45

**7.2 Cadastro de Produtos** ..... 46
7.2.1 Funcionamento do Servlet ..... 47

**7.3 Entrada das Necessidades** ..... 49
7.3.1 Funcioamento do servlet ..... 50

**7.4 Cadastro do Bill of Material** ..... 51
7.4.1 Funcionamento do servlet ..... 53

**7.5 MRP** ..... 54

**7.6 Consulta ao Registro de Inventário** ..... 56
7.6.1 Funcionamento do Servlet ..... 58
7.6.2 Teste de Consistência ..... 60

**7.7 Exibição do Produto** ..... 61
7.7.1 Funcionamento do Servlet ..... 62
7.7.2 Teste de Consistência ..... 67

**7.8 Listagem de Ordens de Obtenção de Produtos** ..... 68
7.8.1 Funcionamento do Servlet ..... 68

**7.9 Exclusão de Produtos** ..... 69
7.9.1 Funcionamento do Servlet ..... 70
7.9.2 Teste de Consistência ..... 71

***8 Conclusões e Recomendações*** ..... *72*

**8.1 Conclusões** ..... 72

**8.2 Recomendações e Sugestões** ..... 73

***9 Bibliografia*** ..... *74*

***10 Anexo – Listagens*** ..... *75*

**10.1 Listagens HTML** ..... 75

10.1.1 Inicio.htm ..... 75
10.1.2 index.htm ..... 75
10.1.3 Barra2.htm ..... 76
10.1.4 Cadastro.htm ..... 77
10.1.5 Entrada.htm ..... 79
10.1.6 atualização.htm ..... 80
10.1.7 Exibirproduto.htm ..... 81
10.1.8 Inventário2.htm ..... 82

**10.2 Listagens Java** ..... 83

10.2.1 Cadastroprodutoservlet ..... 83
10.2.2 Cadastrobomservlet ..... 85
10.2.3 Insertneedservlet ..... 87
10.2.4 removeservlet ..... 89
10.2.5 exibeprodutoservlet ..... 93
10.2.6 ListNecessidadeservlet ..... 96
10.2.7 Mrpservlet ..... 105
10.2.8 ListOrdensServlet ..... 112

# LISTA DE FIGURAS

Figura 3.1- Principios do MRP ..... 14

Figura 3.2 – Registro de Inventário ..... 15

Figura 3.3 – Árvore do Produto ..... 17

Figura 3.4 – Exemplificação da Lógica ..... 19

Figura 3.5 – Lista de um Único Nível ..... 20

Figura 4.1 - Arquitetura simplificada de ambiente cliente servidor ..... 23

Figura 4.2 - Esquema da arquitetura do sistema ..... 24

Figura 6.1 - Modelagem Entidade-Relacionamento do Banco de Dados ..... 42

Figura 7.1 Barra de Opções ..... 46

Figura 7.2 Página de Cadastro de Produtos ..... 47

Figura 7.3 – Saída da Inserção de Produtos ..... 48

Figura 7.5 - Página da Entrada das Necessidades ..... 49

Figura 7.6 – Saída da Inserção de Necessidades ..... 50

Figura 7.8 Cadastro do Bill of Material ..... 52

Figura 7.9 – Saída da Inserção de Estrutura do Produto ..... 53

Figura 7.10 - Tabela de Bill of Material ..... 54

Figura 7.12 – Página de Entrada para Consulta ao Registro de Inventário ..... 57

Figura 7.13 – Demanda diária de material (para Outubro de 2001) ..... 58

Figura 7.15 – Tabela de dados que serve a demonstração neste item ..... 60

Figura 7.16 – Erro devido à má digitação do usuário ..... 60

Figura 7.18 – Saída da exibição do item 10 ..... 63

Figura 7.19 – Saída da exibição do item 40 ..... 64

Figura 7.20 – Saída da exibição do item 60 ..... 65

Figura 7.23 – Saída para má digitação do usuário ..... 67

Figura 7.24 – Listagem de Ordens de Obtenção de Produtos..... 69

Figura 7.25 – Página de entrada de Exclusão de Produtos..... 70

Figura 7.26 – Saída da Exclusão de Produtos ..... 71



# RESUMO

Este trabalho foca o aprendizado do conceito de *Material Requirement Planning* (MRP), um conceito de três décadas de idade cuja aplicação é amplamente difundida ao longo das indústrias atuais e cuja performance ainda não foi ultrapassada por nenhum modo de gestão existente.

O projeto consiste no estudo e apresentação do conceito, passando pelo design da solução através da modelagem de banco de dados relacional, chegando até a implementação do sistema.

O diferencial deste trabalho para um de gestão de produção pura é a implementação da lógica na linguagem de programação orientada a objetos JAVA, utilizando para tanto, os servlets, também muito empregado comercialmente.

Desta maneira é possível criar um ambiente cliente – servidor, comunicado via web de forma a permitir o usuário realizar transações no banco de dados por meio de interfaces HTML, padrão para a *browsers* de Internet.

# ABSTRACT

This work focuses the learning of the Material Requirement Planning (MRP) concept, which is three decades old and whose application is widely used among the nowadays industries and also whose performance has not been beaten by any other existing management method.

The project consists of the study and presentation of the concept, passing by the solution design relied on the relational database modeling, ending at the system's implementation.

What differs this work from one of pure production management is the way the logic is implemented based on the JAVA object oriented language, making use therefore of servlets, also very commercially utilized.

All of this makes possible to create a client-server environment, communicated by web so that it's allowed to the user to make database transactions within HTML interfaces, the standard browser language for Internet.

# 1 Introdução

Os sistemas de planejamento de recursos da empresa (**ERP**), revolucionaram a administração da maioria das companhias de médio a grande porte, no fim dos anos 80 e começo dos anos 90, viabilizando a integração de suas atividades principais. Estes sistemas permitem que setores como o financeiro e contábil integrem-se com os de vendas, compras, produção e recursos humanos automatizando tarefas operacionais, estruturando a empresa e eliminando redundâncias, excesso de papelada e a eventual discordância entre as áreas.

O presente trabalho baseia-se em um projeto de um dos módulos essenciais do ERP, o planejamento da produção, que se fundamenta em outro conceito bem difundido, o **MRP** - planejamento das necessidades de materiais - que em um nível acima veio a ser chamado de **MRP II** - planejamento dos recursos da manufatura.

Para o projeto será utilizada a tecnologia *servlet,* através da linguagem orientada a objetos JAVA, integrada a banco de dados MS *Access* por meio de comandos SQL (*Structured Query Language*). A linguagem HTML será utilizada para interface amigável do sistema com o usuário. Com isso o sistema MRP é disponibilizado via rede (como a *internet*) para qualquer tipo de acesso remoto de um cliente num ambiente cliente servidor.

# 2 Escopo do projeto

O projeto visa o aprendizado, construção e documentação de um sistema de planejamento da produção baseado em MRP isto é, um sistema que processe informações de demanda de itens, para uma empresa hipotética, geradas por pedidos de clientes ou por previsão de mercado e que produza, junto a informações de quantidade de inventário, ordens de compra ou de produção dos mesmos itens ou de seus componentes, necessários à satisfação desta demanda.

As entradas do sistema são feitas manualmente, representando os pedidos tirados pelo setor de vendas, e as saídas, ordens de compra ou produção de itens, são baseadas nos dados cadastrais de nível ideal de estoque, árvore do produto, *lead time* de produção ou fornecimento, configuração dos centros produtivos, etc. aliados a níveis reais (disponíveis) de estoque atualizados periodicamente pelo próprio sistema.

São disponibilizadas ao usuário, operações de ativação e desativação de produtos, alterações de cadastro e listagens de níveis atuais de estoque.

# 3 MRP – Planejamento das Necessidades de Materiais

Primeiramente, original dos anos 60, o MRP significava Planejamento das Necessidades de Materiais (*Material Requirement Planning*), conhecido atualmente como MRP I. Depois, de uma forma mais abrangente, passou a ser chamado de Planejamento dos Recursos da *Manufatura (Manufacturing Resource Planning*), ou MRP II.

O MRP I permite que as empresas calculem quantos materiais de determinado tipo são necessários e em que momento. Para fazer isso, ele utiliza os pedidos em carteira, assim como uma previsão para os pedidos que a empresa acha que irá receber. O MRP verifica, então, todos os ingredientes ou componentes que são necessários para completar esses pedidos, garantindo que sejam providenciados a tempo.

Com o MRP II os sistemas operacional e financeiro tornam-se um único sistema, ou seja, é possível que as empresas avaliem as implicações da futura demanda nas áreas financeiras e de engenharia, além da análise das implicações quanto à necessidade de materiais. O planejamento e controle além de materiais atingem dinheiro, pessoas e equipamentos. Torna-se um sistema global para a empresa **[C.Fullmann, 1989]**.

## 3.1 Metas, Objetivos e Funções dos MRP

Veja a tabela 2.1 que contém metas e objetivos do MRP bem como suas funções e atividades.

| METAS E OBJETIVOS DO MRP | FUNÇÕES E ATIVIDADES DO MRP |
|---|---|
| • Rotatividade de Estoque | • Previsão e Entrada dos Pedidos |
| • Atendimento ao Cliente | • Plano Mestre da Produção |
| • Produtividade da Mão de Obra | • Plano Geral da Produção |
| • Utilização da Capacidade | • Liberação das Ordens (Colocação, Aumento, Redução, Cancelamento) |
| • Custo do Material | • Seguimento (Compras, Recebimento e Controle da Produção) |
| • Custo do Transporte | • Planejamento da Capacidade |
| • Custo do Sistema | • Manutenção dos Registros |
| | • Coordenação |

Tabela 2.1

### 3.1.1. Rotatividade de Estoque

Uma meta muito importante do MRP é o baixo nível dos estoques, principalmente quando as taxas de juros do mercado são altas. Um termo muito utilizado na medição de estoques é: **Giro de inventário**, ou **rotatividade de estoque.** Normalmente são medidos através de dias, semanas ou meses

de suprimento. Isto é, se o estoque estiver balanceado, quanto tempo ele durará até acabarem-se as peças? Dividindo este tempo, pelo período considerado, obtém-se o giro (rotatividade) do estoque (ou seja, um estoque de quinze dias "gira" duas vezes no período de um mês).

Não obstante todas as medidas de minimização do estoque, ele pode ser considerado como um "pulmão" para proteger as empresas dos acontecimentos imprevisíveis (ou indesejáveis), como perdas por rejeição, quebra de equipamentos, fornecedores não confiáveis, etc.

Por outro lado o inventário custa muito caro, e a seguir listam-se os motivos:

- Custo de capital: cada real investido no inventário deixa de ser investido em outro lugar, que proporcione juros (custo de oportunidade).
- Custos de estocagem e movimentação: quanto maior o estoque, maiores as capacidades de armazenagem e movimentação necessárias.
- Tempo de vida, obsolescência: produtos que sofram freqüentes alterações de engenharia em suas listas de material; itens com demandas variáveis ou imprevisíveis, o cliente, exigindo alterações, peças antigas ficam no estoque; produtos perecíveis que perdem o valor após "vencidos".

### 3.1.2. Atendimento ao Cliente

Embora medir os custos de manter estoques seja um problema, um problema muito maior é medir **Falta de Estoque.** Ao comprarem de um fornecedor não confiável, as pessoas começam a pensar em fontes alternativas, evidenciando-se a importância do atendimento ao cliente neste cenário de competitividade acirrada em que vive-se atualmente.

Há ainda a outra parte do atendimento, que é o atendimento de suprimentos à fábrica. Se os itens comprados não estiverem disponíveis para atender o programa de produção, serão necessárias reprogramações, que podem causar, num primeiro momento, ociosidade e num segundo, "horas extras". Em ambos os casos a falta de material é altamente onerosa.

Fala-se aqui de capacidade de cumprir prazos prometidos para entrega, bem como de sua otimização.

### 3.1.3. Produtividade da Mão-de-Obra

A Produtividade pode ser afetada por adversidades como falta de materiais (referente tanto aos itens comprados como aos fabricados), tempo de preparação (redução do tempo de preparação aumenta a produtividade), quebra de máquina, hora extra e variação na equipe de trabalho.

Convém colocar que lotes de produção maiores requerem menor número de preparações, levando à utilização de mais recursos diminuindo a improdutividade.

A manutenção preventiva ou a preditiva também deve ser levada em consideração para evitar a quebra de equipamentos e os conseqüentes picos e vales gerados na produção.

Por último, a utilização da capacidade de instalação bem como seu dimensionamento devem ser corretos frente a necessidade de atendimento ao cliente e ainda aos investimentos dos acionistas que esperam retorno.

### 3.1.4. Custos de Materiais

Estes são afetados por algumas decisões tomadas pela Administração de Materiais, tais como: quando, quanto e como comprar e quando negociar pedidos abertos.

Conseguir entregas no prazo com baixo nível de estoque propõe aumento da freqüência de entregas, entretanto, existe uma outra variável, **o custo do transporte** que pode ser bastante significativo no custo total de materiais o que cria a necessidade de otimização das entregas.

Agregados ao custo de materiais, existem ainda os **sistemas de custo** formais (salários, computadores, etc.) e os informais (tempo dos compradores e

seguidores perdido na verificação da crise do momento e reagir a ela, assim como o do pessoal da fábrica) mostrando-se mais temas que o MRP deve considerar.

Resumindo, o que deve ser feito pela Administração de Materiais é melhorar continuamente as metas e objetivos mostrados. O desafio, portanto é atender o cliente da melhor forma, com o menor investimento em estoque.
A seguir são analisados então, quais fatores influenciam estas metas, isto é, quais são as funções e atividades da Administração de Materiais que devem ser vistas pelo MRP.

### 3.1.5. Previsão de Vendas

A Previsão de Vendas depende do tipo de produção, se é sob encomenda ou para estoque sendo que ambos os tipos são considerados, caso houverem. Visa-se **antecipar à necessidade do cliente** e fornecer os parâmetros o mais seguro possível para o sistema. Produz como resultado, dados a serem seguidos pelo plano da demanda independente (assunto a ser discutido a seguir).
Prever futuras tendências é uma atividade difícil, e há quem diga que o trabalho da Administração de Materiais é replanejar e não simplesmente planejar.

### 3.1.6. Plano Mestre de Produção

O Plano Mestre de Produção é o plano de quanto vai ser produzido esta semana, na semana seguinte, na outra, etc. Enfim o que será entregue aos clientes. É a fase mais importante do planejamento e controle de uma empresa, constituindo-se na principal entrada para o planejamento das necessidades de materiais dependendo de informações como estoque disponível e carteira de pedidos (ou pedidos em aberto).

Note que nem sempre o planejado corresponde à necessidade das vendas; havendo sazonalidade, por exemplo, pode ser decidido produzir mais por um certo período para atender as necessidades do pico de vendas. Ou em outro caso, por causa de alto custo de preparação pode ser decidido produzir em lotes maiores. Portanto, a previsão de vendas e o plano de produção para atender esta previsão podem se diferenciar muito ao contrário do que se induz.

No Plano Mestre está ainda implícito o fluxo de caixa da empresa já que, projetados a quantidade de mão de obra e inventário necessários para o futuro, podem ser identificados pela área financeira eventuais problemas relacionados ao capital investido ou recebido.

Veja que na preparação do plano deve haver participação de todos os envolvidos, para, principalmente, verificar onde ele é **cabível** ou não.

### 3.1.7. Liberação de Ordens

Esta atividade ocorre a todo instante, em todo caso de colocar uma nova ordem (de compra ou produção), ou de alterar uma ordem colocada, ou ainda de quando cancelar, etc. QUANDO e QUANTO são decisões tomadas para esta atividade também a todo instante, independente da quantidade de itens envolvidos (8.000. 50.000 ou 300). Consideram-se então todos os itens comprados ou produzidos no processo, peças, submontagens, enfim, todos, **exceto para o produto final**, cujas decisões são tomadas no Plano Mestre. A Liberação de Ordens, obviamente atua ligada ao Plano Mestre.

### 3.1.8. Seguimento ("follow-up")

Existem dois tipos de seguimento: ambos consideram as ordens já liberadas para compra ou produção. Chamam-se então de seguimento de compras e de controle de produção. Se por exemplo for constatado que determinado fornecedor não entregará no prazo, pode ser necessária uma reprogramação das atividades envolvidas para evitar desequilíbrios na companhia.

### 3.1.9. Planejamento da Capacidade

Outra atividade da Administração de Materiais. Nela constata-se se existem altos e baixos ou ainda sobrecargas de capacidade podendo-se tomar as medidas necessárias com antecedência. Por exemplo, se serão necessárias horas ou homens extras, turnos adicionais, ou ainda, no caso de um pedido novo para entrega urgente, é possível verificar se ele pode ser atendido sem afetar os já existentes. Melhor do que receber simplesmente os pedidos "urgentes" é administrar a capacidade revelando-se esta como uma atividade muito importante do MRP.

### 3.1.10. Manutenção de Registros

O programa mestre de produção é constituído de registros com escala de tempo que contém para cada produto final, as informações da demanda e estoque disponível atual. Usando esta informação, o estoque disponível é projetado à frente no tempo. Quando não há estoque suficiente para satisfazer a demanda futura, quantidades de pedido são inseridas na linha do programa mestre.

É de suma importância a acuracidade dos registros, caso contrário seria impossível o funcionamento do MRP. Uma acuracidade aceitável no controle de estoques está em torno de 95% [**Fullmann, 1989**].

Este índice pode ser conseguido através de contagens cíclicas (ou inventário rotativo) e de verificação da lógica das transações. Da mesma forma é necessário certeza nas Listas de Materiais, para que não ocorram erros no momento de explosão do produto final (decomposição do mesmo nos subcomponentes).

## *3.2 Itens de baixo valor e o MRP*

Todos os itens de volume – como porcas e parafusos, óleo combustível, produtos de limpeza, itens que não podem ser facilmente calculados quanto ao consumo final – abrem a questão: deveriam fazer parte do MRP ?

Estes tipos de produto gerariam listas de materiais extremamente complexas, problemas com determinação de tipos de processos e itens onde são empregados, etc.

Sugere-se então planejamento com ponto de reposição para estes itens. Como devem possuir baixo valor, estima-se uma quantidade de estoque mínima (considerando ou não estoque de segurança) e um momento de compra que leva em conta os tempos de reposição ou fornecimento. Assim trabalha-se sempre com estoque e uma vez ultrapassado o ponto de reposição, basta ser feito o pedido.

## *3.3 MRP ao MRP II*

Quando o MRP é realçado com capacidade para o planejamento de prioridades e de capacidade, às vezes é chamado de "MRP em ciclo fechado" [**Correa , 1990**]. Conhecer as necessidades de materiais e manter as prioridades atualizadas não é suficiente, se não existir capacidade disponível. O termo "ciclo fechado" significa que todos os elementos faltantes (planejamento de prioridades e planejamento de capacidade) são completados e existem "circuitos" de retorno quando for encontrado um problema no plano de execução.

Um passo final faz a transição do MRP em ciclo fechado para o MRP II. A interligação deste ao sistema financeiro, verificando o custo das liberações planejadas de pedidos, projetadas nos registros de inventário. Lembrando que estes registros representam o que realmente será produzido e comprado, é possível projetar a partir deles as futuras vendas, custos, compras e inventários.

O MRP evoluiu a partir deste tipo de pensamento, tornando-se um sistema de toda a empresa, envolvendo vários aspectos – manufatura, finanças, *marketing*, engenharia, compras, distribuição e processamento de dados. Com o MRP II, os sistemas de operação e financeiro são convertidos em um único, utilizando as mesmas transações e os mesmos números.

O MRP II pode também ser usado como um simulador, permitindo fazer perguntas a respeito da necessidade de mão de obra, necessidades de equipamentos, disponibilidade de peças, inventários, prazos de entrega e acúmulo de pedidos.

O usuário compara, então, o novo material e as necessidades de capacidade com as mais recentes operações do MRP/CRP (Planejamento das Necessidades de Capacidade), para ter acesso ao impacto de modificação.

A diferença real com o MRP II é a forma como a gerência usa o sistema [**Dias, 1988**]. Apenas quando o sistema se difundir pela empresa, não sendo apenas um sistema de operação para controle da produção e inventário, é que realmente será um MRP II.

## 3.4 Aplicação do Conceito MRP

Falou-se até o presente momento de conceitos e objetivos do MRP. Mostra-se a seguir como o MRP funciona, quais os seus princípios. Veja a figura 3.1 que resume as entradas e saídas deste sistema.

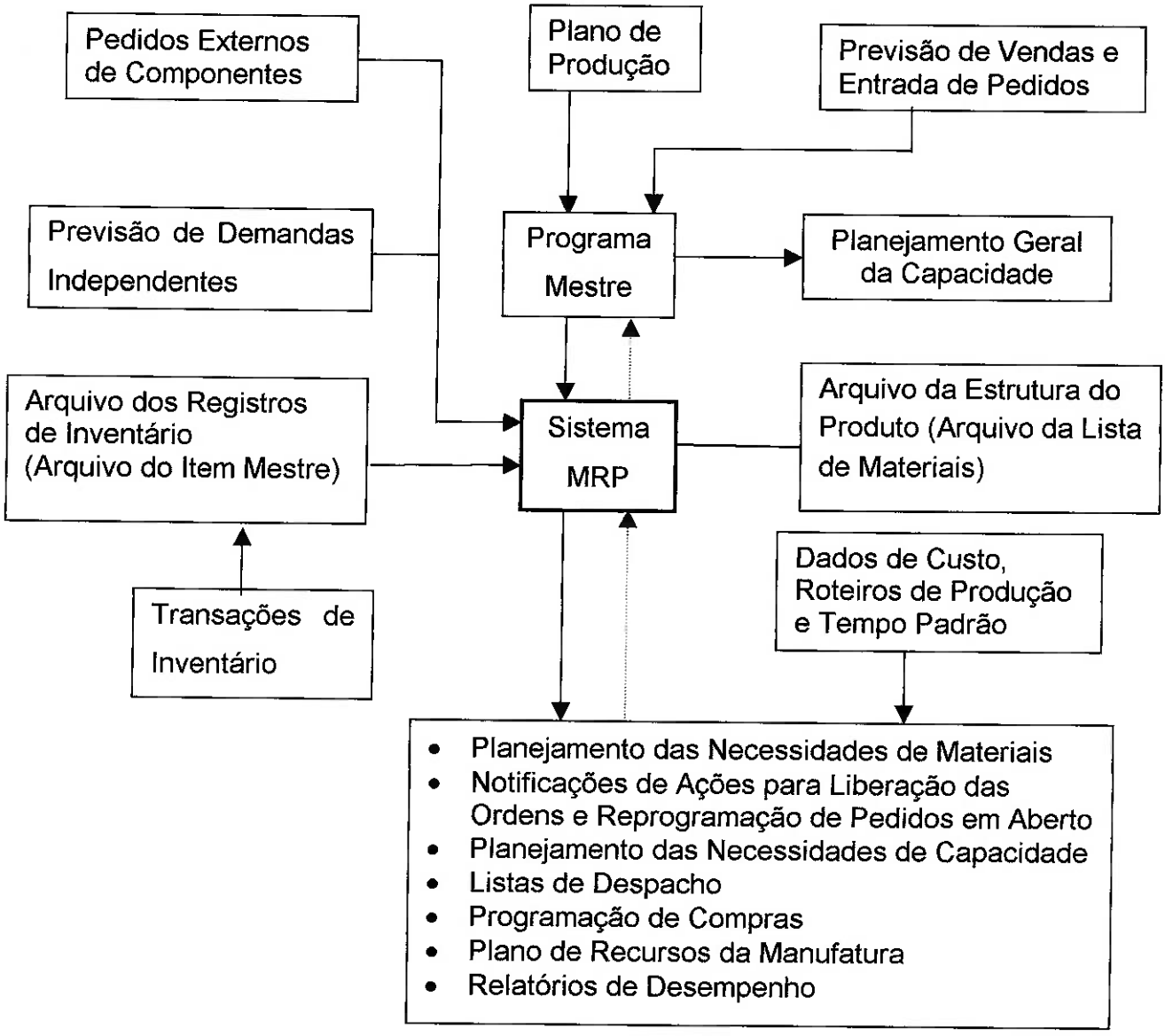



*Figura 3.1- Principios do MRP*

No MRP um produto final é o "pai" do subconjunto de itens que o compõe. O "pai" ocupa um nível zero de planejamento enquanto que os componentes ocupam do nível um até subníveis dependendo do plano de cada empresa. Esta relação parece trivial, mas possui uma decorrência fundamental que atinge previsão de demanda. As incertezas do mercado refletem somente na quantidade de produtos finais a ser produzida, chamada de **demanda independente.** Porém determinado este número, a necessidade dos componentes automaticamente também está, contando com os registros de listas de materiais, e neste caso tem-se a **demanda dependente**.

### 3.4.1 Registro de Inventário

| Período N.º | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|
| Necessidades Brutas | | 36 | 10 | | | | 30 | 5 |
| Recebimentos Programados | | 40 | | | | | | |
| Disponível | 20 | 24 | 14 | 14 | 14 | 14 | -16 | -21 |
| Necessidades Líquidas | | | | | | | 26 | 5 |
| Liberação Planejada de Pedidos | | | | 26 | | | | |

Item: T701800 - B — *Lead Time* = 3 semanas

Descrição: Motor eixo Y — Estoque Mínimo = 10 unidades

*Figura 3.2 – Registro de Inventário*

Por meio de banco de dados, existem registros de inventário do tipo mostrado acima para cada peça que apresente necessidade cadastrada.

O método considera estoque disponível, as necessidades brutas provenientes das entradas (necessidades independentes), recebimentos programados e estoque mínimo da peça, fazendo a contabilidade destes dados e produzindo as necessidades líquidas da peça, isto é, o valor efetivamente necessário a ser produzido ou comprado do item.

Note que as necessidades líquidas são geradas com a antecedência necessária para o item ser comprado ou produzido, dado o *lead time*, também em banco de dados.

### 3.4.2 Cálculo da demanda dependente – a árvore do produto

Veja a Figura 3.2 que mostra uma árvore do produto (BOM - *Bill of Materials*) para um "item A" hipotético. O item A é formado por **B, C** e **D**. **C** é comprado, **B** e **D** são manufaturados.

As necessidades de um item são calculadas a partir de seu **superior imediato** num processo de explosão nível a nível.

O algoritmo faz uso de uma **Lista de Material Identada** onde a configuração de dependência entre os itens coexiste com a quantidade dos componentes integrantes do item "pai".

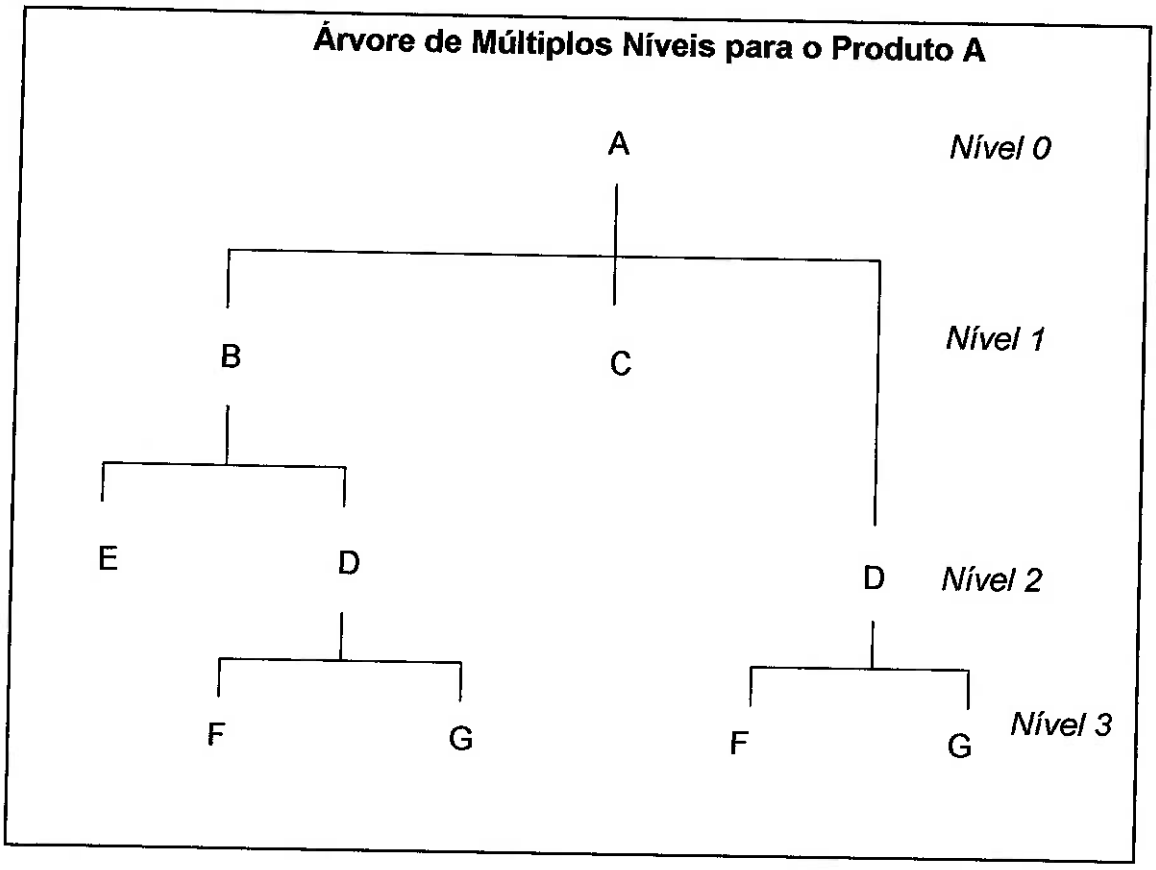



*Figura 3.3 – Árvore do Produto*

É possível identificar, portanto, para qualquer item quais são os subseqüentes e em que quantidades se apresentam.

A varredura da árvore bem como sua estrutura de dados faz uso do LLC (*Low Level Code*) ou Listas de um Único Nível. É um método de explosão que permite a contagem precisa e rápida dos componentes numa gama de produtos.

O interessante a ser mencionado neste item é que esta estrutura é também armazenada em banco de dados funcionando como elemento de ligação entre os n registros de inventário armazenados.

Na figura 3.2 foi possível observar como as necessidades líquidas de um item são determinadas. Observe que é exatamente esta necessidade que é passada ao registro dos componentes (filhos) do item em questão, na proporção registrada na estrutura do produto, para a concepção do registro dos filhos e assim por diante. A entidade utilizada para receber este valor é a **necessidade dependente**.

Este algoritmo pode ser representado por uma equação bem simples:

$$nec\_dep\_filho = nec\_liq\_pai * qtde \quad \text{..... Equação 3.1}$$

$$data\_filho = data\_pai - lead\_time \quad \text{..... Equação 3.2}$$

Onde:

Nec_dep = Necessidade dependente;

Neq_liq = Necessidade Líquida;

Data = data em que a necessidade é inserida em banco de dados.

Veja as figuras 3.4 e 3.5 para **exemplo** desta ligação.

ITEM X  Lead Time = 1  Estoque Mínimo = 0

| Data | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|
| Necessidades Brutas | 10 | 15 | | | 10 | | 20 |
| Reposição | | | | | | | |
| Recebimentos Programados | | 10 | | | | | |
| Estoque Disponível \|10 | 0 | -5 | -5 | -5 | -15 | -15 | -35 |
| Necessidades Líquidas | 5 | | | 10 | | 20 | |

ITEM Y  Lead Time = 2  Estoque Mínimo = 0

| Data | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|
| Necessidades Brutas | 10 | | | 20 | | 40 | |
| Reposição | | | | | | | |
| Recebimentos Programados | | | | | 50 | | |
| Estoque Disponível \|23 | 13 | 13 | 13 | -7 | 43 | 3 | 3 |
| Necessidades Líquidas | | 7 | | | | | |

*Figura 3.4 – Exemplificação da Lógica*

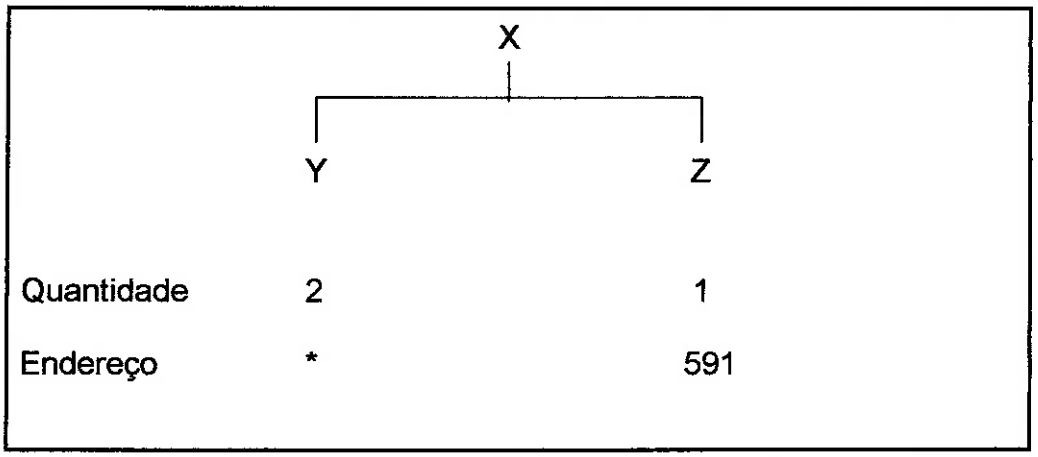



*Figura 3.5 – Lista de um Único Nível*

Note que em termos computacionais a árvore do produto resume-se a listas de um único nível que apresentam o item pai, os itens filhos, as proporções em que estes constituem o pai e o endereço da lista de um único nível dos componentes-filhos.

### 3.4.3 Sumário da Lógica

O MRP reúne as três entradas básicas – Programa Mestre, Registros de Inventário e Lista de Materiais. Verifica-se desta maneira, quantidade disponível em estoque, mais em pedidos feitos previamente. No registro de inventário há ainda tamanho mínimo de lote de produção. Este está relacionado com o menor número de peças a serem fabricadas de modo a viabilizar a utilização da máquina.

Calcula-se então **quanto** deve ser produzido, em quantidades de faltas atribuíveis a cada período considerado, apurando as necessidades líquidas ou o saldo de inventário considerando para tanto os pedidos e previsões estipulados por necessidades independentes (para produtos acabados) e dependentes para os componentes, o estoque disponível e o estoque mínimo exigido pela administração.

Para completar, o MRP determina **quando** produzir baseado em datas de entradas de pedido, prazos estipulados, *lead times* de processos e datas de fechamentos de ordens em aberto (note que em os *lead times* e as ordens de produção em aberto são analisados para cada componente do produto final e para ele inclusive) e ainda tempos processo e de entrega de fornecedores para os produtos de terceiros (**tempo de reposição**).

Importantíssimo ressaltar que este trabalho apega-se a apenas uma parte do conceito de MRP que é extremamente complexo e amplo.

# 4 Tecnologia *Servlet*

*Servlets* são programas que estendem a funcionalidade básica de um servidor sob requisições e respostas dos clientes do sistema. São executados nos próprios servidores, (chamados neste caso servidores habilitados a Java) provendo, portanto, documentos dinâmicos que são mais fáceis de escrever e de processar.

Um *servlet*, por exemplo, pode ser responsável por coletar dados da entrada de uma ordem genérica, via interface HTML e por aplicar o processamento necessário à atualização de dados da operação em questão de uma empresa hipotética.

O processo de escrita dos *servlets* (através de um API - *Java Servlet*), não assume nenhum atributo do ambiente ou do protocolo do servidor, eliminando eventuais restrições ou incompatibilidade entre tipos de *servlets* e servidores. Por esta razão estes programas tornaram-se muito freqüentes dentre servidores HTTP, e desta maneira, a maioria dos servidores atuais para internet suportam a tecnologia *Servlet*. Vide figura 4.1

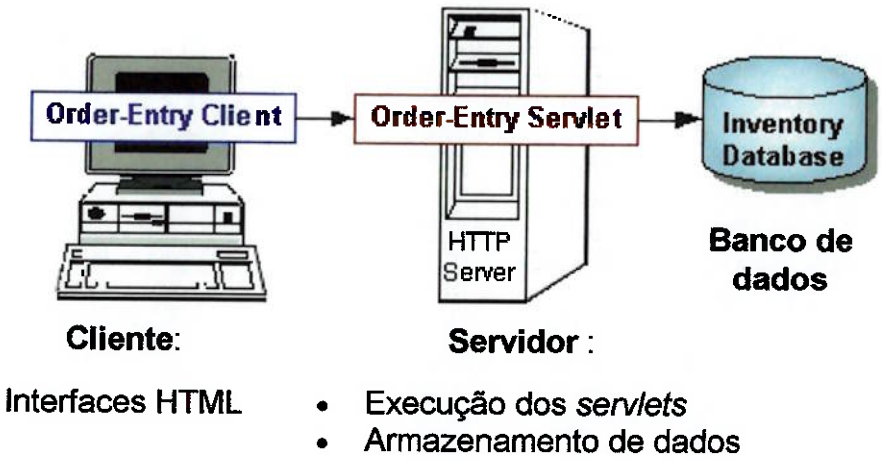



*Figura 4.1 - Arquitetura simplificada de ambiente cliente servidor*

Focando o presente trabalho, nota-se que esta tecnologia atende em sua definição, o objetivo proposto, à medida que permite simular transações que envolvam entrada de dados provenientes do cliente, via interfaces HTML, processamento da lógica empregada (MRP) baseado em banco de dados, com atualizações e saídas gráficas através das mesmas interfaces.

Embora neste caso, **cliente e servidor estejam na mesma máquina**, faz-se uso da tecnologia da mesma maneira que num acesso remoto via rede. Evidencia-se a possibilidade da conexão remota entre o cliente e o servidor através de redes de transmissão de dados, bastando para isto uma configuração rápida, o endereço **IP** (*Internet Protocol*) das máquinas a serem ligadas e cabos de rede.

## 4.1 Arquitetura do Sistema

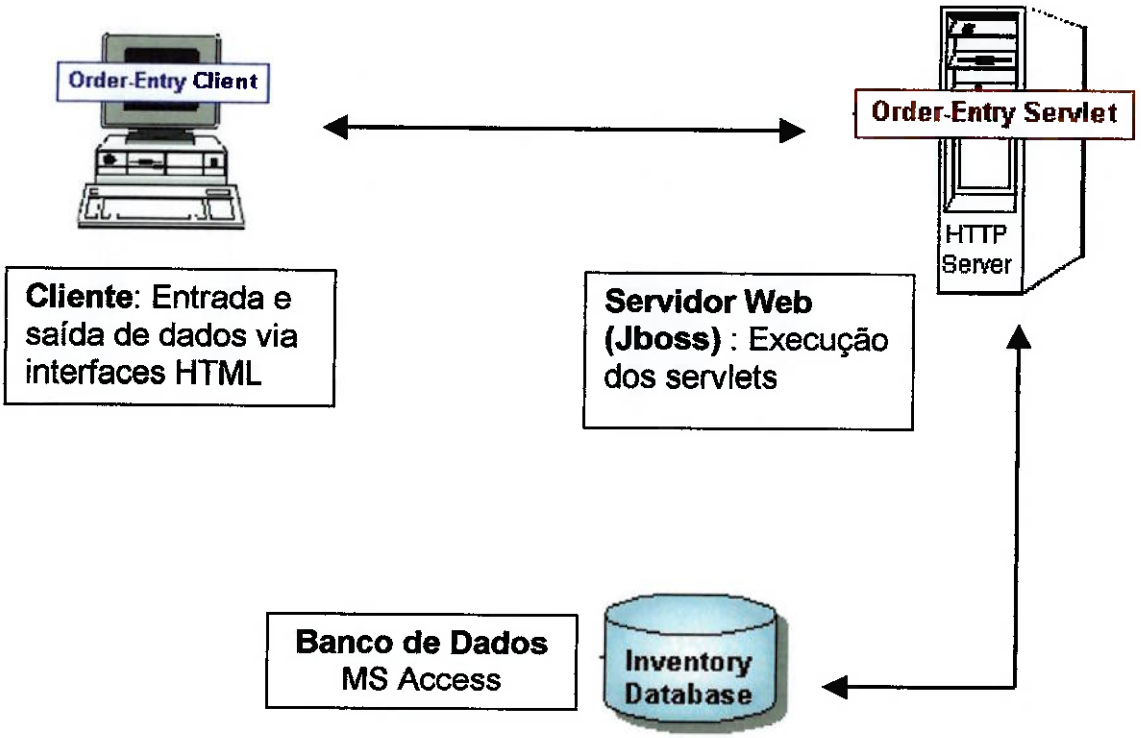



*Figura 4.2 - Esquema da arquitetura do sistema*

O funcionamento é simples:

- O cliente produz requisições ao servidor via comandos coletados via HTML;
- O servidor as processa junto a informações do banco de dados utilizando-se de comandos SQL, e dos relacionamentos construídos no próprio MS Access, para acesso ao banco de dados durante a execução dos programas *servlets* que são redigidos em Java;
- O servidor atualiza o banco de dados;

- O servidor envia dados ao cliente;
- O cliente lê as saídas do sistema através de novas interfaces HTML.

Note que geralmente, o servidor tem em sua administração um número n de clientes produzindo um fluxo de informações a ser gerenciado, cada um em uma máquina (*hardware*) diferente. Neste caso, porém **as duas entidades estão no mesmo espaço físico**, embora o servidor esteja funcionando **normalmente** e os acessos sejam feitos **via *web***.

Repare também que as interfaces HTML condicionam o usuário, com os campos apresentados na tela, a introduzirem os dados necessários para execução das tarefas desejadas.

Por fim a arquitetura mostra que este sistema difere muito de um simples programa redigido de forma integral no Access, pois o funcionamento é disponibilizado via rede para acesso remoto de vários clientes diferentes.

## *4.2 Outros tipos de utilização dos servlets*

- Um servlet pode atender múltiplas requisições e/ou sincronizá-las permitindo colaboração entre pessoas, por exemplo, suportando sistemas como conferências *on-line*.
- Servlets podem encaminhar requisições a outros conjuntos de servidores e servlets que tenham os mesmos tipos de tarefas. Desta maneira, podem ser

empregados em balanceamento de discos, configurações de redundância, partição de serviço, etc. onde os vários servidores.

# 5 Requisitos de Projeto

Este item utiliza-se de um *template* em formato profissional de descrição de sistemas, com características pertinentes a uma caracterização de sistemas de um desenvolvimento real.

Existe controle de revisões e versões do documento, e gerenciamento de cronogramas.

Nele são descritos objetivos funcionais do projeto e requisitos a serem satisfeitos.

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

| | |
|---|---|
| **Identificação**: | <MRP Didático>.ERS.xxx |
| **Título**: | Especificação de Requisitos de Software |
| **Tipo**: | ERS: Especificação de Requisitos de Software |
| **Categoria**: | Classificado/Preliminar |
| **Autor**: | <Fernando Machado de Figueiredo> |
| **Versão**: | 1 |
| **Data**: | 04/06/2001 |

## *5.1 Introdução*

### 5.1.1 Nome do Documento

Projeto de componentes de software para o Desenvolvimento de um MRP Didático

### 5.1.2 Marcas

Várias marcas e nomes de produtos podem estar citados neste documento e que pertencem a terceiros. Dado que este documento pretende ter circulação apenas interna, não será feita uma discriminação caso a caso.

### 5.1.3 Escopo

Este documento formaliza a especificação de requisitos do projeto, descrevendo suas características básicas, objetivo, funções principais, cronograma de desenvolvimento e implementação. O trabalho está em fase de

projeto básico e desta maneira é passível de ser levemente ajustado; por isso destaca-se neste documento um histórico das versões e alterações que eventualmente surgirão.

### 5.1.4 Conceitos, terminologia e abreviaturas

| Termo | Significado |
|---|---|
| MRP | Planejamento das Necessidades de Materiais |
| HTML | *Hyper Text Markup Language* |
| SQL | *Structured Query Language* |
| *Jboss* | Servidor *web* distribuído gratuitamente pela Sun. |
| | |

## 5.2 Descrição Geral

### 5.2.1 Perspectiva Histórica

A década de 80 foi aquela em que **Produtividade e Qualidade** tornaram-se palavras chaves nas indústrias de manufatura mundiais. A administração de materiais foi um dos agentes de ambas e em grande maioria com a utilização do MRP.

Nesta época já existiam sistemas de computação capazes de colocar o conceito em prática. A evolução desde então até os dias de hoje, aumentou drasticamente a velocidade de processamento dos grandes computadores bem como a capacidade de manipulação e armazenamento de dados, contudo não modificou em sua essência este famoso conceito estudado na prática no presente trabalho.

O banco de dados utilizado também não traz novidades sendo que o conceito mais recente envolvido é o de entidade-relacionamento em banco de dados relacional, já conhecido na mesma década de 80.

A tecnologia *servlet* utilizada é a mais nova, esta que permite que programas sejam acessados via rede por meio de um servidor, porém já não é novidade no mercado atual.

O Desenvolvimento de um MRP Didático traz a oportunidade do aluno aprender estas importantes ferramentas e documentar de forma a outros poderem também aprender.

### 5.2.2 Objetivos

O objetivo deste projeto é único - cálculo de necessidades no tempo de produtos e seus componentes, baseado no conceito MRP. Porém, para efeito de apresentação o mesmo é dividido em partes:

#### *5.2.2.1 Projeto e design de 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 servidor *web* para os processamentos necessários em cada solicitação do cliente.

O projeto de banco de dados envolve muito da inteligência do sistema e desta forma existe um item totalmente dedicado à explanação das tabelas utilizadas e do modelo relacional construído.

*5.2.2.2 Desenvolvimento dos aplicativos*

O sistema será escrito na linguagem Java pois visa-se a operação do mesmo via rede, como a *internet* por exemplo, e desta forma optou-se por esta linguagem que é amplamente utilizada atualmente.

Como é sabido, a linguagem Java é orientada a objetos, outro conceito que é permitido ser estudado por este trabalho.

*5.2.2.3 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 em cada transação determinada no próximo item.

Elas serão escritas em HTML devido a este sistema poder ser operado via rede, como dito anteriormente. Note que as mesmas permitem a correta interação do cliente com o aplicativo pois encaminham os *inputs* a serem fornecidos.

## 5.3 Requisitos Funcionais

### 5.3.1 Cadastro

Os itens fictícios que servirão de objeto de trabalho ao sistema estarão presentes em cadastro onde estarão armazenadas suas características básicas como descrição e unidade, e três características funcionais: código do produto, estoque de segurança e categoria (comprado ou produzido).

Numa outra tabela, será armazenada a estrutura de cada item, isto é, a dependência de cada item de possíveis componentes (filhos), seguida das quantidades que cada filho aparece, do tempo necessário (*lead time*) de produção do item pai a partir de seus filhos, e seu código de baixo nível (*low level code*).

É importante notar que *lead time* é dado de **operação**, mas como não se trabalha com operações neste trabalho, ajustou-se o banco assumindo como hipótese que o roteiro de fabricação de um determinado item é constante.

- O sistema deverá permitir ao operador: cadastrar novos produtos, eliminar aqueles que não serão mais utilizados e listar os existentes.
- O sistema poderá, conforme a necessidade, permitir a seleção de materiais a partir de um código, da primeira letra da descrição, ou outro atributo que facilite a operação do usuário.

### 5.3.2 Políticas de Estoque

As políticas de estoque refletem imposições externas ao comportamento do gerenciamento de estoque como, por exemplo: segurança projetada pelo administrador de materiais através do estoque de segurança (ou estoque mínimo); exigência de lote mínimo de produção ou de compra devido a fornecedores ou a processos de fabricação para obtenção de um determinado item; e ainda prioridade de produção se o determinado item passa por gargalos; e outros.

- O sistema deverá trabalhar com estoque de segurança de maneira que este servirá de critério de disparo de necessidade líquida dos itens que tiverem o estoque do determinado período abaixo do de segurança.
- O sistema não deverá trabalhar com as outras políticas como definido entre o aluno e o professor orientador do projeto.

### 5.3.3 Entrada das necessidades

A entrada da demanda é fundamental para o funcionamento de um sistema MRP que a utiliza para atribuir necessidades líquidas aos itens pai e seus respectivos componentes em um horizonte de tempo pré-determinado, baseando-se para isto em quantidades presentes em estoque e existência de recebimento programado para o período considerado, estoque de segurança

para o item, *lead time* de produção (se item produzido) ou de compra (se item recebido), e da estrutura do produto.

- O sistema deverá permitir a entrada de necessidades pelo usuário sendo que esta deve ser determinada por data da entrega acordada, o produto a que atribui a necessidade e evidentemente em que quantidade.

### 5.3.4 Cálculo MRP

Como mencionado anteriormente (item 3.3) o cálculo MRP é dado em função das necessidades existentes em banco de dados, bem como dos outros atributos necessários.

Existe, porém duas maneiras de se executar o cálculo das necessidades: de forma **regenerativa** ou absoluta e de forma **incremental** ou de mudança líquida.

O cálculo regenerativo reproduz todas as saídas do sistema refazendo todos os cálculos a partir dos dados atuais existentes em banco. O cálculo incremental por sua vez calcula as necessidades somente daqueles itens que tiveram algum atributo modificado desde a última atualização do sistema.

- O sistema deverá executar o cálculo das necessidades de maneira regenerativa e poderá fazê-lo de maneira incremental e neste caso deverá discriminar, portanto as duas opções em sua interface com o usuário.

### 5.3.5 Consulta Online

A consulta é dita *online* lembrando que o sistema é construído para servir usuários em rede a partir da tecnologia *servlet.*

Poderão ser realizadas consultas ao banco de dados em campos variados e dependendo do caso será possível uma certa seleção de dados interessantes.

- O sistema deverá permitir consulta ao registro de inventário e eventualmente poderá separá-lo em conjunto de dados combinados com os de cadastro como descrição do item e estoque assumido no período, por exemplo.
- O sistema deverá permitir consulta das necessidades líquidas geradas pelo MRP registradas em todo período para todos os itens e eventualmente selecioná-las por itens ou por período (ou data) desejado, ou ambos.
- O sistema poderá permitir a consulta a apenas necessidades independentes, isto é, aquelas registradas a partir de entradas no sistema.
- O sistema deverá permitir consulta à árvore (estrutura) do produto a partir da entrada de um item pai produzindo como saída os filhos diretos. Poderá conter *hyperlinks* para aqueles componentes que por sua vez possuírem seus próprios filhos para a visualização da estrutura do mesmo.

## 5.4 Requisitos Não-Funcionais

### 5.4.1 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 *2000*, que contem o MS *Internet Explorer* como *browser* para utilização do servidor instalado e da visualização das páginas HTML.

O próprio processador representa o cliente e o servidor, embora para a apresentação possa ser montado um conjunto de processadores para demonstração da tecnologia *servlet*.

O servidor utilizado é o **JBOSS**, proveniente do site da *Sun*, de forma gratuita, instalado na máquina pelo professor em conjunto com o aluno.

### 5.4.2 Interfaces com outros sistemas

Este sistema roda via rede e por isso hipoteticamente, pode ser acessado de outro PC que esteja conectado ao servidor, simbolizando um cliente qualquer.

Outros sistemas poderiam acessar o banco de dados deste através de comandos SQL, ou de saídas que atingissem as páginas HTML desenvolvidas

neste trabalho. Funcionalmente porém, não é previsto nenhum tipo de interface com outros sistemas.

### 5.4.3 Desempenho

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

### 5.4.4 Escalabilidade

Este projeto constitui apenas um módulo da técnica MRP. É possível adicionar mais características funcionais ao programa, desde mais comandos de consulta online até outros módulos como, estratégias de estoque que trabalhariam com ordens de cliente passando-as a necessidades de acordo com a estratégia da operação e do item, ou ordens de produção e de compra a serem geradas e enviadas por EDI ao fornecedor, não sendo impossível extrapolar-se a um nível de ERP, embora isto fosse inviável com tantos *players* no mercado.

Aumentando o número de usuários e da abrangência do sistema seria necessário aumentar a capacidade computacional, com CPUs dedicadas, mainframes ou servidores, (ou os dois), numa arquitetura que por ser centralizada teria um limite de capacidade operacional, com o limite tecnológico para o desempenho de grandes *hardwares*.

### 5.4.5 Segurança e Privacidade (security e privacy)

Não faz parte do escopo controle de acesso de usuários ou qualquer outro tipo de segurança transacional, não sendo difícil porém introduzí-la num nível simplificado.

## 5.5 Perspectiva Gerencial

*Neste item, deve-se dar uma visão do cronograma de implementação das funções.*

| | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.Preparação | | | | | | | | | | |
| 1.1.Java | | | | | | | | | | |
| 1.2.HTML | | | | | | | | | | |
| 1.3.MRP | | | | | | | | | | |
| 1.4.SQL | | | | | | | | | | |
| 2.Especificação | | | | | | | | | | |
| 3.Projeto básico | | | | | | | | | | |
| 3.1.Interfaces | | | | | | | | | | |
| 3.2.Tabelas | | | | | | | | | | |
| 4.Implementação | | | | | | | | | | |
| 4.1.Estoque | | | | | | | | | | |
| 4.2.Demanda | | | | | | | | | | |
| 4.3.BOM | | | | | | | | | | |
| 4.4.MRP | | | | | | | | | | |

*Tabela 5.1 - Cronograma de construção do projeto*

## 5.6 Controle de Revisão

| | |
|---|---|
| **Autor:** | **Fernando Machado de Figueiredo** |
| **Revisão:** | |
| **Versão:** | **1.0** |
| **Documento:** | |
| **Revisores:** | |
| **Data:** | **02/07/01** |
| **Aprovação:** | |
| **Responsável :** | |
| **Data:** | |

## 5.7 Histórico de Alterações

$Log:$

# 6 Modelagem ER do Banco de dados e Tabelas Access

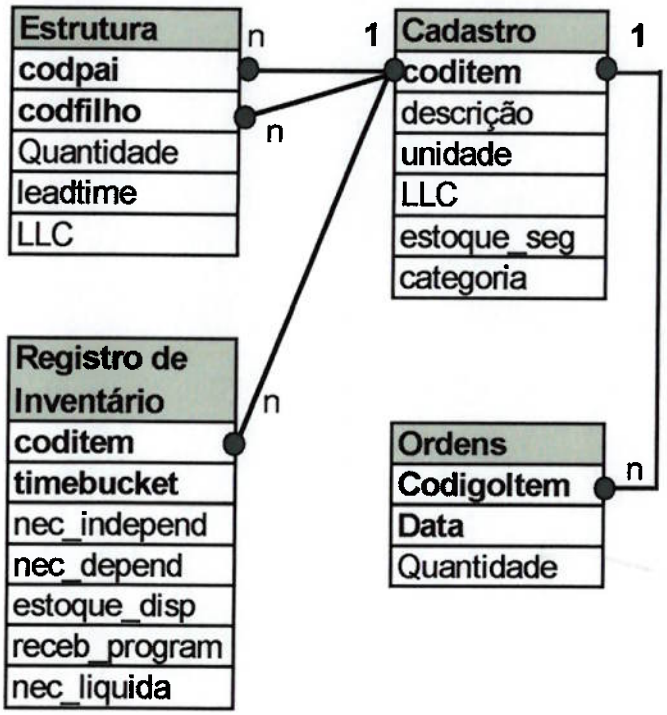



*Figura 6.1 - Modelagem Entidade-Relacionamento do Banco de Dados*

A figura acima mostra os relacionamentos das tabelas de dados criadas através do MS Access.

A explanação das tabelas e seus principais tipos de dados vêm a seguir:

## 6.1 Estrutura das tabelas de dados

As tabelas foram construídas utilizando-se dos campos mostrados na figura 7.1. Os campos em negrito são as chaves das tabelas.

O banco é constituído de três tabelas funcionais e duas suporte, como segue:

- **Cadastro**: Onde se encontram todos os itens existentes, suas características básicas, quantidade do estoque mínimo (ou de segurança). Há a informação do **LLC** (*Low Level Code*), isto é, o mais baixo nível que o filho aparece em todo e qualquer produto existente na estrutura. Esta informação é importante para evitar tarefas redundantes por parte do processador e para viabilizar o cálculo em caso de existência de lote mínimo de produção.

  Há ainda a informação de categoria do item, isto é, a caracterização em comprado, produzido, ou produto final. Não é uma característica funcional.
- **Registro de Inventário**: Utiliza a chave concatenada, [código item + data pretendida para satisfação da necessidade]. Relaciona as necessidades existentes, diferenciando as independentes (provindas das entradas manuais) das dependentes (calculadas segundo a árvore do produto). Contém o estoque disponível, recebimento programado e necessidade líquida de cada item existente em seu respectivo período, obviamente.
- **Estrutura do item**: Registra para cada item pai, os seus respectivos filhos, nas quantidades em que aparecem e o *lead time* de obtenção dos mesmos. O código LLC se repete devido ao mesmo pertencer de fato à tabela

estrutura. Devido a simplificação adotada neste trabalho, houve a obrigação de colocá-lo na tabela de cadastro para permitir a elaboração de *queries* ao banco de dados.

Um item pai pode apresentar vários filhos; como cada linha da tabela apresente um item pai e um item filho, com as características comentadas acima, pela s normas de bancos de dados estruturados, exige-se a chave concatenada [cod/. do pai + cod/. do filho].

- **Ordens:** Tabela importantíssima utilizada para determinação de **quando** deve ser adquirido, (produzido ou comprado), um item com necessidade liquida existente na tabela de Registro de Inventário. O cálculo faz uso de um **período crítico** para liberação destas ordens, que nada mais é do que uma simplificação da análise de processos de fabricação ou compra de itens. Este tempo crítico é dado de um item pai e vem do maior lead time de um conjunto de filhos de um pai comum, o que significa que se considerou montagem do item pai, a partir de seus filhos, com **processos em paralelo**. Exemplificando: um carro é feito de motor e chassi. O lead time carro-motor é 3 dias e o carro-chassi é 5 dias. O tempo de fabricação de um carro é, portanto 5 dias.

# 7 Desempenho do Sistema

O sistema, conforme mencionado anteriormente, executa as ordens do usuário cujo meio é a interface, escrita em linguagem HTML, padrão atual utilizado para desenvolvimento de páginas *web*.

A estrutura do sistema é passada às interfaces que condicionam o usuário às entradas corretas para o bom funcionamento.

Abaixo são mostradas as mesmas, uma por uma, para melhor compreensão das finalidades do sistema descritas no item 6.3.

Em seguida é possível observar o funcionamento dos *servlets* associados.

## *7.1 Barra de Opções*

Esta é a página inicial que permite o usuário escolher que funcionalidade, entre as possíveis, ele deseja que o sistema execute. Veja a figura 7.1.

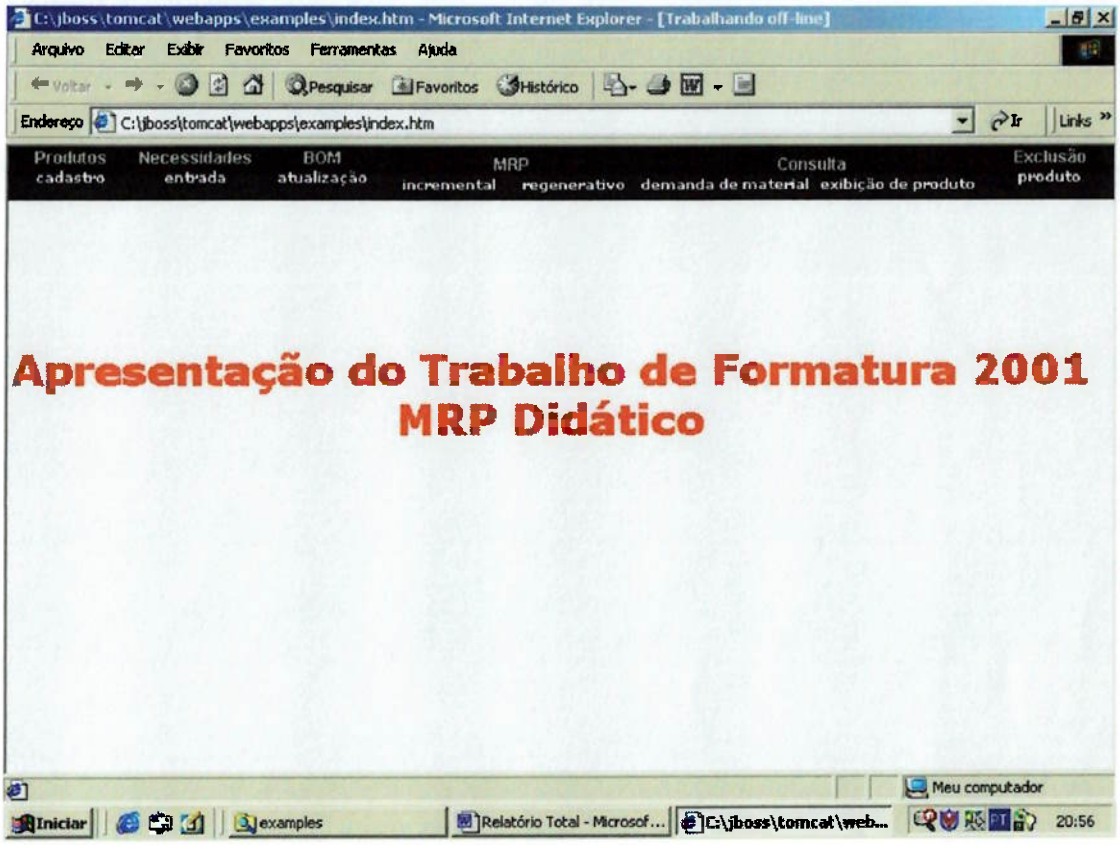



*Figura 7.1 Barra de Opções*

Escolhendo um dos *links*, o usuário é levado a uma segunda tela de acordo com a opção desejada. Veja os itens subseqüentes.

## *7.2 Cadastro de Produtos*

Uma vez que o link cadastro, abaixo de Produtos, seja clicado, o navegador leva o usuário a tela onde deve ser feita a entrada das características básicas do produto. Veja a figura 7.2.

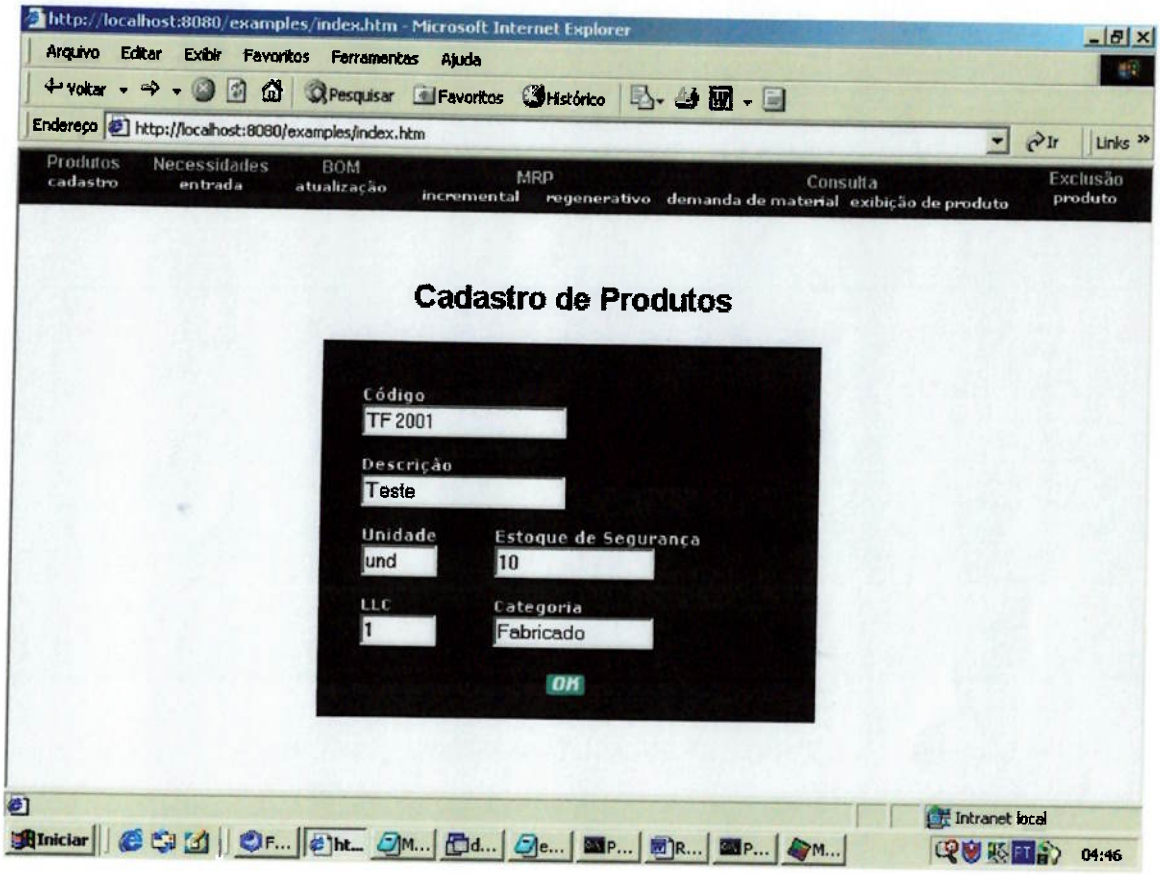



*Figura 7.2 Página de Cadastro de Produtos*

## 7.2.1 FUNCIONAMENTO DO SERVLET

Abaixo é mostrada a saída do sistema ao usuário, da inserção-exemplo de cadastro com os dados da figura 10.2. Veja figura 10.3.

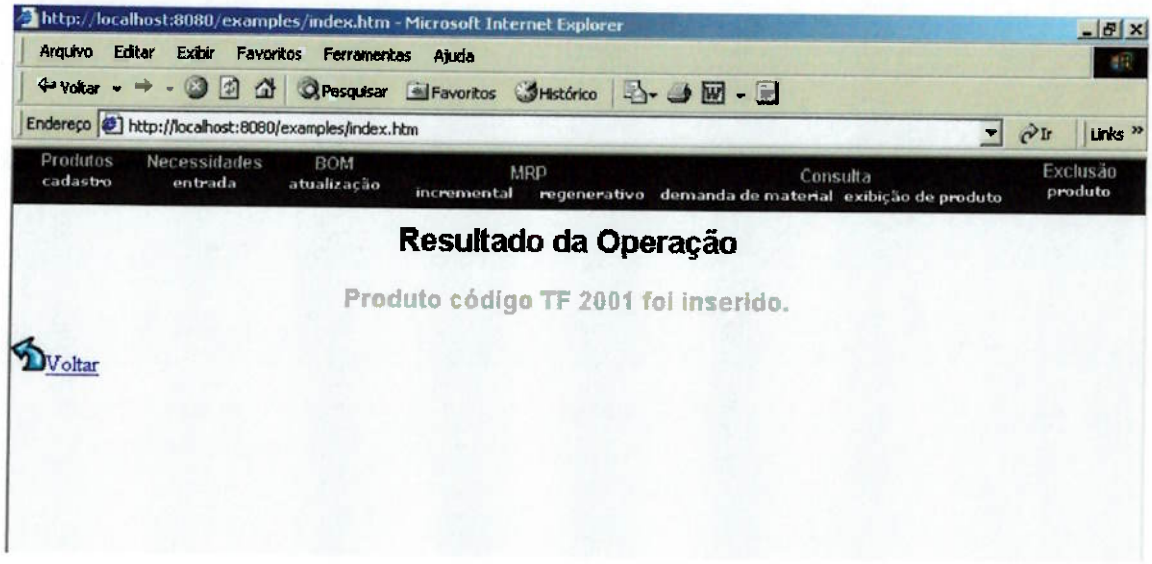
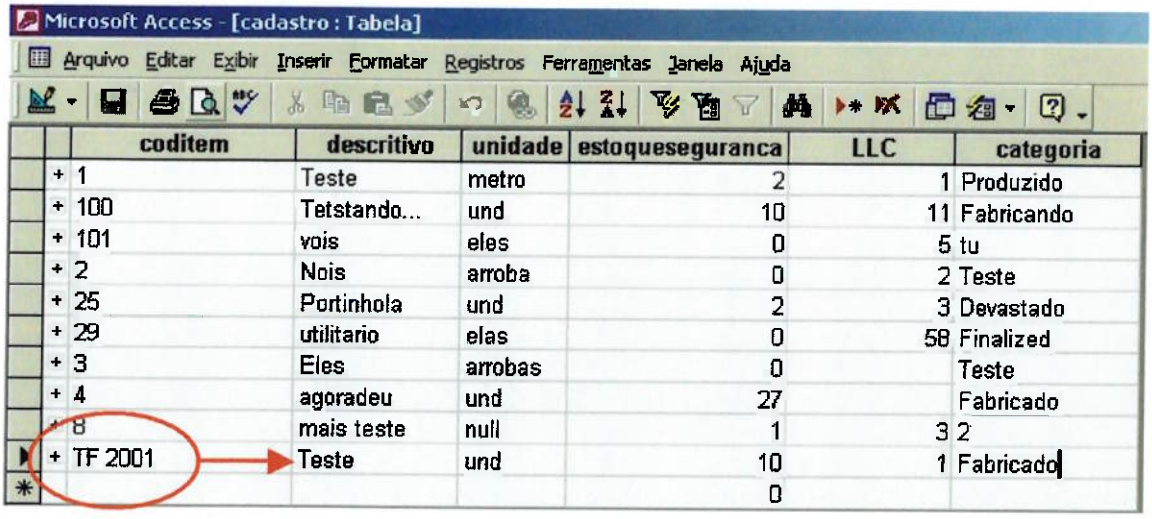



*Figura 7.3 – Saída da Inserção de Produtos*

A inserção pode ser conferida na leitura do banco de dados no MS Access. Veja a figura 7.4.

Note que a inserção de um registro numa tabela do banco de dados é sempre feita na última linha desse registro (a não ser que haja um tratamento especial da inserção).




| coditem | descritivo | unidade | estoqueseguranca | LLC | categoria |
|---|---|---|---|---|---|
| 1 | Teste | metro | 2 | 1 | Produzido |
| 100 | Tetstando... | und | 10 | 11 | Fabricando |
| 101 | vois | eles | 0 | 5 | tu |
| 2 | Nois | arroba | 0 | 2 | Teste |
| 25 | Portinhola | und | 2 | 3 | Devastado |
| 29 | utilitario | elas | 0 | 58 | Finalized |
| 3 | Eles | arrobas | 0 | | Teste |
| 4 | agoradeu | und | 27 | | Fabricado |
| 8 | mais teste | null | 1 | 3 | 2 |
| TF 2001 | Teste | und | 10 | 1 | Fabricado |
| | | | 0 | | |

*Figura 7.4 – Tabela Access de Cadastro*

## 7.3 Entrada das Necessidades

A entrada do sistema é realizada artificialmente a partir da digitação da demanda pelo usuário, indicando o código do produto, quantidade e data pretendida para obtenção do mesmo.

Ao clicar o link entrada, abaixo de Necessidades, na página inicial vem a página deste item. Veja a figura 7.5.

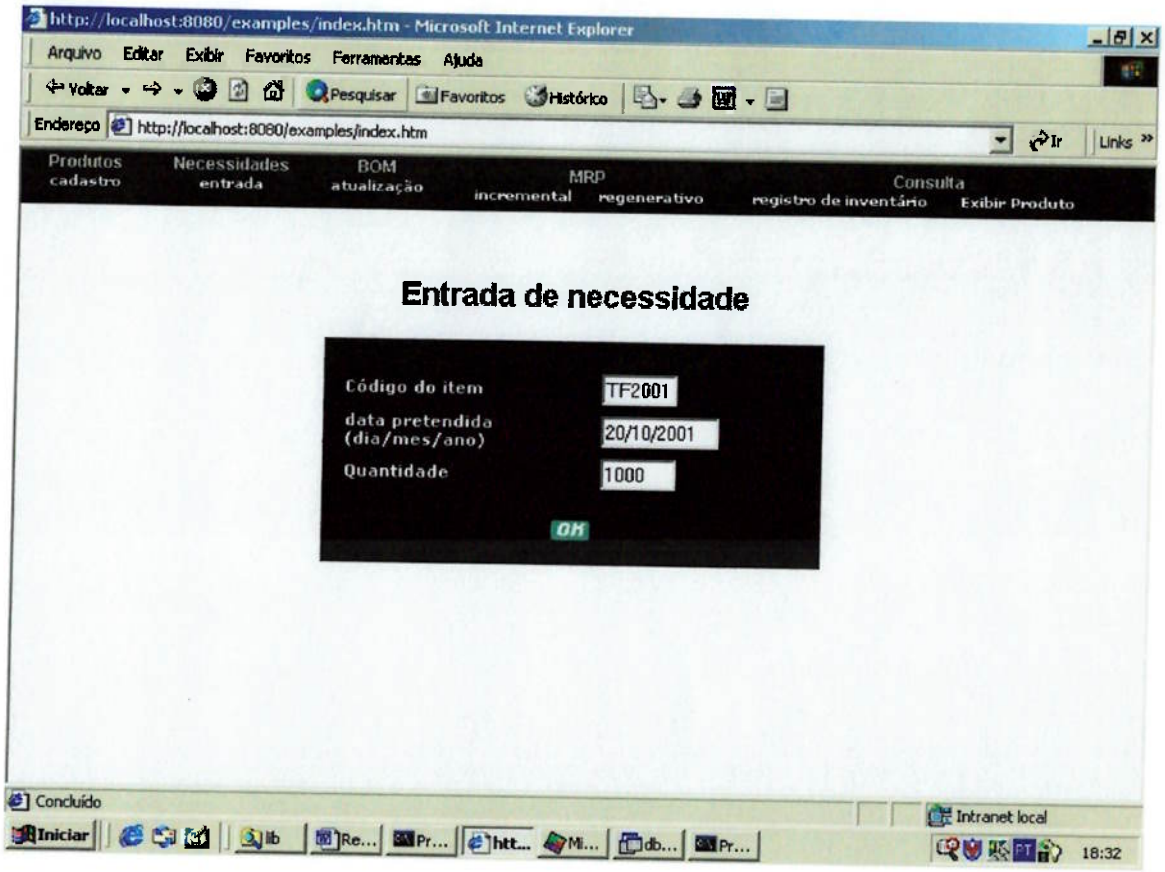



*Figura 7.5 - Página da Entrada das Necessidades*

## 7.3.1 FUNCIOAMENTO DO SERVLET

A inserção é realizada quando o usuário clica no botão "ok", conforme a figura acima.

A saída da requisição de inserção da necessidade ao usuário é mostrada na figura 7.6.

A inserção pode ser conferida na leitura do banco de dados no MS Access. Veja a figura 7.7.

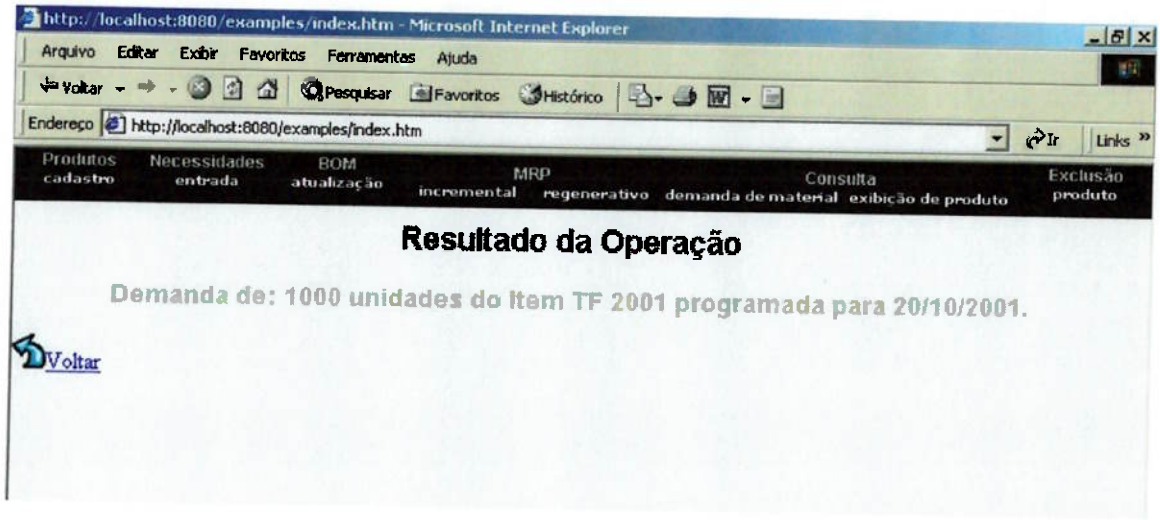



*Figura 7.6 – Saída da Inserção de Necessidades*

Microsoft Access - [RegInvent : Tabela]

Arquivo Editar Exibir Inserir Formatar Registros Ferramentas Janela Ajuda

| cod_item | data | necess_indep | necess_dep | recebntoprog | estoque disp | necliquida |
|---|---|---|---|---|---|---|
| PF 39 | 5/2/1900 | 2 | 0 | 0 | 0 | 2 |
| PF 40 | 8/2/1900 | 4 | 0 | 0 | 0 | 4 |
| PF 40 | 15/2/1900 | 4 | 0 | 0 | 0 | 4 |
| PF 41 | 8/2/1900 | 2 | 0 | 0 | 0 | 2 |
| PF 43 | 14/2/1900 | 4 | 0 | 0 | 0 | 4 |
| PF 44 | 15/2/1900 | 1 | 0 | 0 | 0 | 1 |
| PF 45 | 5/2/1900 | 2 | 0 | 0 | 0 | 2 |
| PF 45 | 8/2/1900 | 1 | 0 | 0 | 0 | 1 |
| PF 45 | 17/2/1900 | 4 | 0 | 0 | 0 | 4 |
| PF 47 | 14/2/1900 | 4 | 0 | 0 | 0 | 4 |
| PF 49 | 15/2/1900 | 4 | 0 | 0 | 0 | 4 |
| PF 5 | 7/2/1900 | 2 | 0 | 0 | 0 | 2 |
| PF 6 | 12/2/1900 | 5 | 0 | 0 | 0 | 5 |
| PF 9 | 13/2/1900 | 3 | 0 | 0 | 0 | 3 |
| PF 9 | 17/2/1900 | 1 | 0 | 0 | 0 | 1 |
| teste | 17/10/2001 | 1000 | 0 | 0 | 0 | 0 |
| TF2001 | 20/10/2001 | 1000 | 0 | 0 | 0 | 0 |
| * | | 0 | 0 | 0 | 0 | 0 |

*Figura 7.7 – Tabela de Registro de Inventário*

## 7.4 Cadastro do Bill of Material

A estrutura de um novo produto pode ser inserida no banco de dados por meio da página descrita na figura 7.8. Ela vem da opção pelo link atualização, abaixo de BOM, vista na figura 7.1.


*Figura 7.8 Cadastro do Bill of Material*

O usuário deve entrar com o código do produto, o código de um dos seus filhos, a quantidade em que ele aparece, o lead time de fabricação deste filho em função do item pai e o código LLC.

## 7.4.1 FUNCIONAMENTO DO SERVLET

Clicando o botão "OK", o usuário enxerga a saída expressa pela figura 7.9 abaixo.

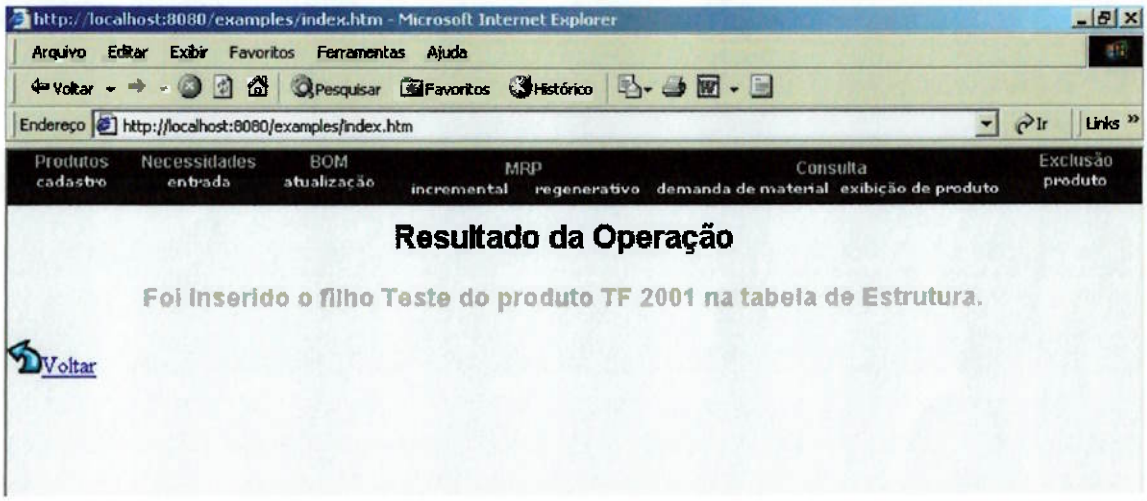
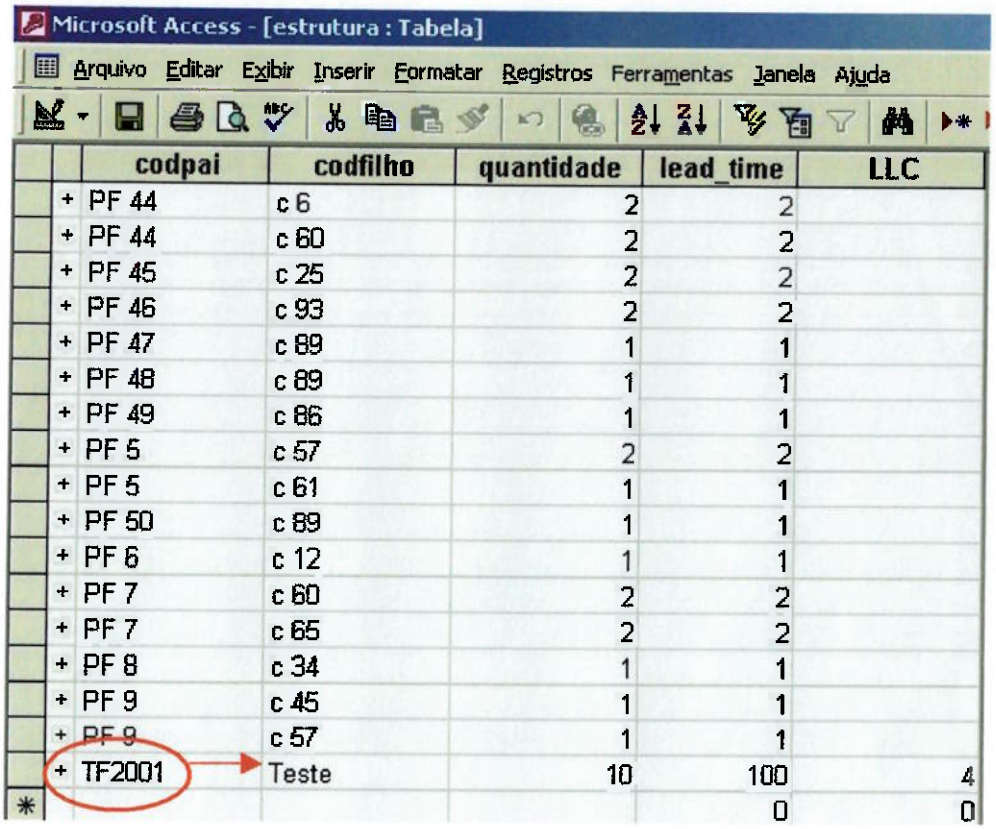



*Figura 7.9 – Saída da Inserção de Estrutura do Produto*

Da mesma forma como nos anteriores, é possível observar o resultado da execução do sistema nas tabelas MS Access. Veja a figura 7.10 abaixo.




| | codpai | codfilho | quantidade | lead_time | LLC |
|---|---|---|---|---|---|
| + | PF 44 | c 6 | 2 | 2 | |
| + | PF 44 | c 60 | 2 | 2 | |
| + | PF 45 | c 25 | 2 | 2 | |
| + | PF 46 | c 93 | 2 | 2 | |
| + | PF 47 | c 89 | 1 | 1 | |
| + | PF 48 | c 89 | 1 | 1 | |
| + | PF 49 | c 86 | 1 | 1 | |
| + | PF 5 | c 57 | 2 | 2 | |
| + | PF 5 | c 61 | 1 | 1 | |
| + | PF 50 | c 89 | 1 | 1 | |
| + | PF 6 | c 12 | 1 | 1 | |
| + | PF 7 | c 60 | 2 | 2 | |
| + | PF 7 | c 65 | 2 | 2 | |
| + | PF 8 | c 34 | 1 | 1 | |
| + | PF 9 | c 45 | 1 | 1 | |
| + | PF 9 | c 57 | 1 | 1 | |
| + | TF2001 | Teste | 10 | 100 | 4 |
| * | | | | 0 | 0 |

*Figura 7.10 - Tabela de Bill of Material*

## 7.5 MRP

A opção descrita por MRP, leva à atualização dos dados existentes na tabela registro de inventário, isto é, leva à funcionalidade básica e principal do sistema, a execução da lógica MRP.

Há duas possibilidades para tal execução, a incremental e absoluta, para cada uma delas há um respectivo link na barra inicial de opções.

Estas opções não levam a uma página intermediária como nos itens anteriores, mas à atualização direta da tabela Access e desta maneira, a consulta *on line* é a maneira de verificar seu resultado.

Existe obviamente a possibilidade de serem realizadas consultas diretamente às tabelas de dados no MS Access com o intuito de verificação da resposta do sistema.

Utilizando o item da figura 7.19 seria possível uma tentativa de confirmação da eficácia do trabalho.

Inicialmente existem em Registro de Inventário necessidades independentes do item 10 – Cilindro, de 40 para o dia 23/09/2001 e 100 para o dia 12/10/2001.

Ao rodar o sistema obter-se-ia o seguinte resultado (vide figura 7.11):

Microsoft Access - [RegInvent : Tabela]

Arquivo Editar Exibir Inserir Formatar Registros Ferramentas Janela Ajuda

| cod_item | data | necess_indep | necess_dep | recebntoprog | estoque disp | necliquida |
|---|---|---|---|---|---|---|
| 10 | 23/9/2001 | 40 | 0 | 10 | 20 | 20 |
| 10 | 13/11/2001 | 100 | 0 | 90 | 0 | 20 |
| 20 | 21/9/2001 | 0 | 20 | 0 | 0 | 35 |
| 20 | 11/11/2001 | 0 | 20 | 0 | 0 | 35 |
| 30 | 21/9/2001 | 0 | 20 | 0 | 0 | 20 |
| 30 | 11/11/2001 | 0 | 20 | 0 | 0 | 20 |
| 40 | 16/9/2001 | 0 | 40 | 0 | 0 | 80 |
| 40 | 8/11/2001 | 0 | 40 | 0 | 0 | 80 |
| 50 | 17/9/2001 | 0 | 400 | 0 | 0 | 450 |
| 50 | 7/11/2001 | 0 | 400 | 0 | 0 | 450 |
| 60 | 16/9/2001 | 0 | 160 | 0 | 0 | 180 |
| 60 | 6/11/2001 | 0 | 160 | 0 | 0 | 180 |
| 70 | 16/9/2001 | 0 | 80 | 0 | 0 | 230 |
| 70 | 6/11/2001 | 0 | 80 | 0 | 0 | 230 |
| 80 | 15/9/2001 | 0 | 11500 | 0 | 0 | 21500 |
| 80 | 5/11/2001 | 0 | 11500 | 0 | 0 | 21500 |
| 99 | 12/12/2001 | 12 | 0 | 0 | 0 | 12 |
| | | 0 | 0 | 0 | 0 | 0 |

*Figura 7.11 - Tabela de Registro de Inventário*

Os códigos são números. Para entender que produto é o item 10, seria interessante olhar a figura 7.19.

Existe uma configuração inicial de demanda para o item 10 em dois períodos distintos e o sistema calcula a necessidade dependente de cada filho do nível abaixo ao pai atualizando, ao final, as necessidades líquidas do pai segundo a equação mostrada no item 3.4.2.

Esta etapa é realizada novamente para os itens do próximo nível (segundo o LLC – *Low Level Code*) e assim por diante até o último nível.

Observa-se na figura que cada conjunto [data+componente] possui sua respectiva necessidade dependente, baseada na independente do pai e no lead-time de fabricação, e necessidade líquida onde o balanço de estoque é mostrado.

A demonstração formal será melhor realizada em momento de apresentação do trabalho à banca examinadora da Escola Politécnica.

## *7.6 Consulta ao Registro de Inventário*

Esta opção é escolhida através do link abaixo de Consulta na barra de opções. Sua função é exibir as necessidades líquidas do item escolhido pelo usuário na página visualizada pela figura abaixo.

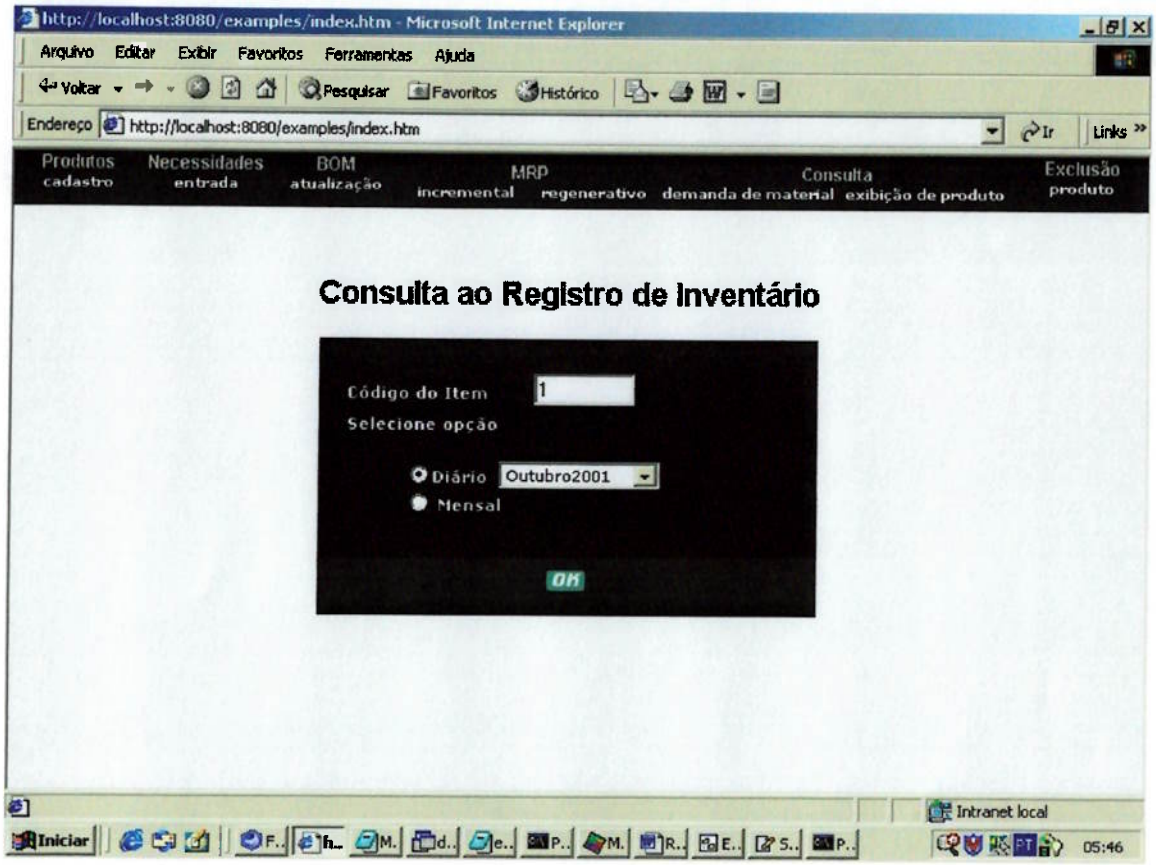



*Figura 7.12 – Página de Entrada para Consulta ao Registro de Inventário*

Nota-se que há duas opções para exibição das necessidades. O usuário pode escolher entre **consulta diária** - e neste caso deve entrar com o mês desejado – e também **mensal**, isto é, consulta das necessidades de cada mês de um determinado ano.

Para efeito de demonstração restringiu-se os dados em banco de dados a apenas Setembro de 2001 a Janeiro de 2002.

## 7.6.1 FUNCIONAMENTO DO SERVLET

Veja a saída na figura 7.13, para a opção, "Diária", escolhida acima.

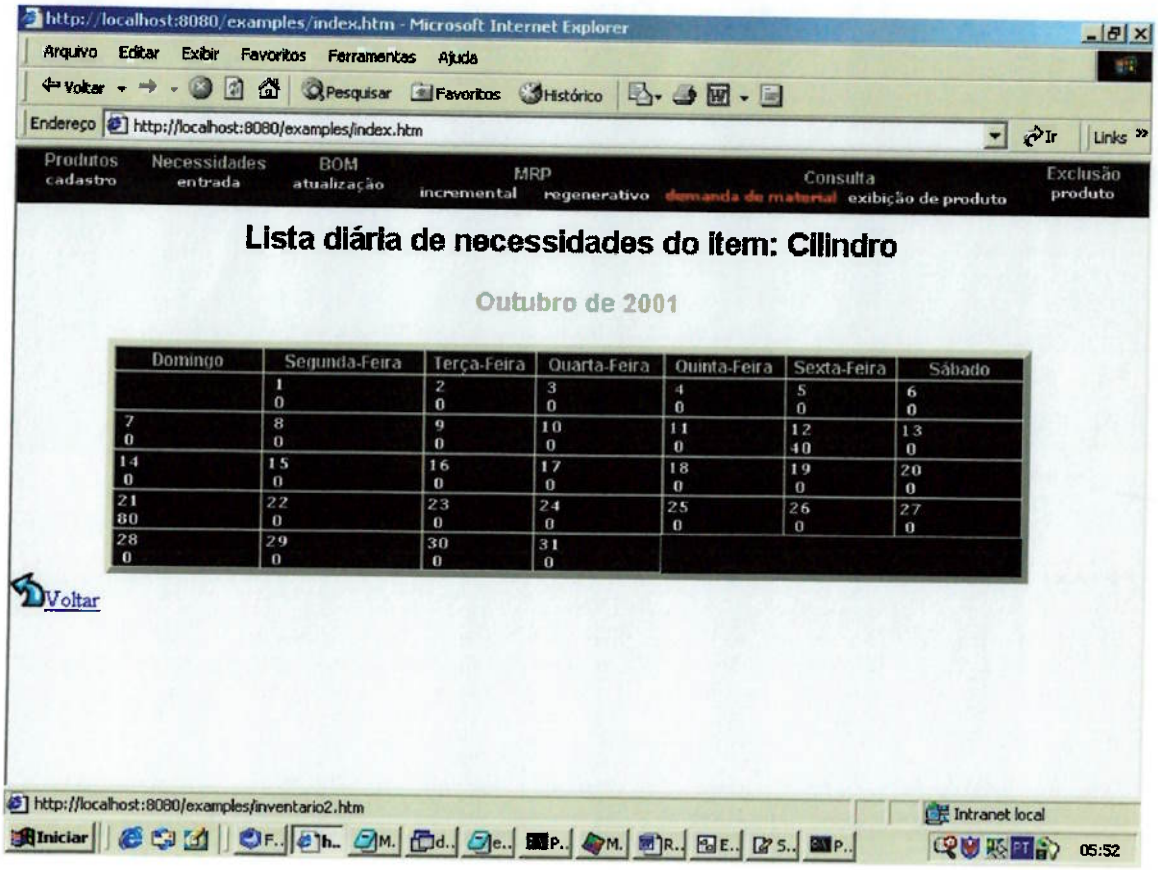



*Figura 7.13 – Demanda diária de material (para Outubro de 2001)*

Note que a saída é dada no formato **calendário** para melhor posicionamento do usuário e que a descrição do item desejado é impresso na tela para evidenciar eventuais erros de digitação e o código errado exista em banco de dados.

Caso a opção do usuário fosse pela demanda mensal de material, seria impresso na tela o visualizado na figura 7.14.

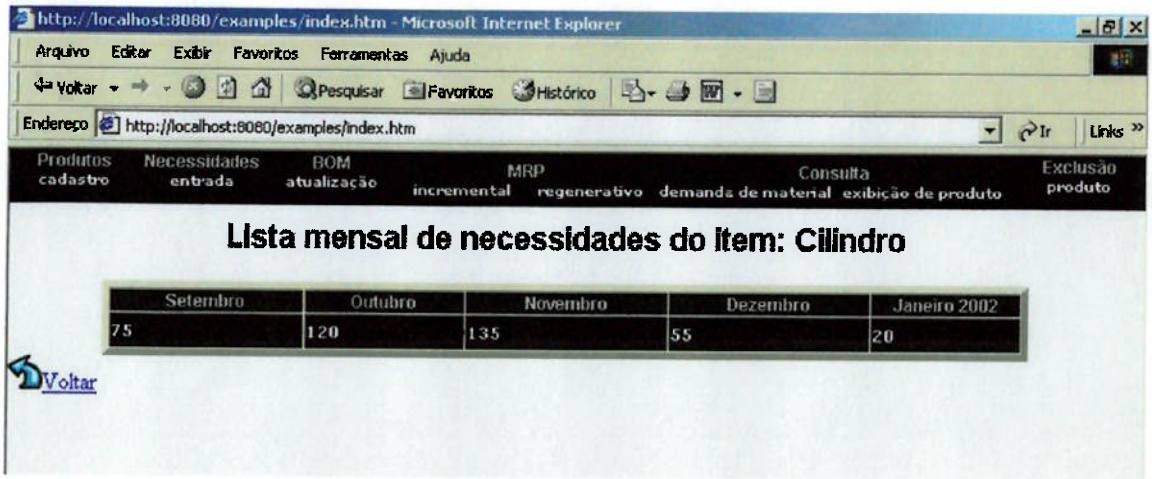
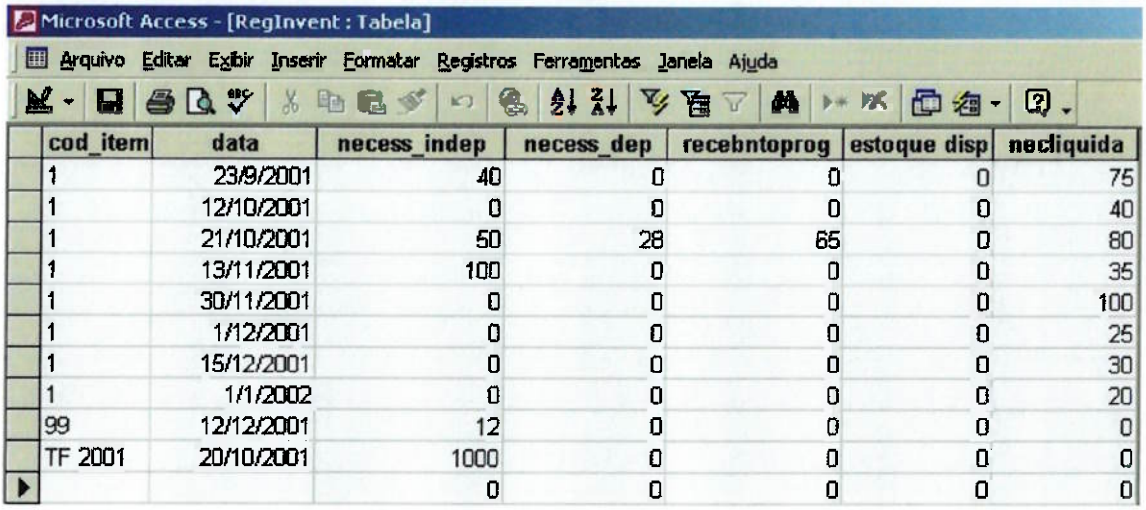



*Figura 7.14 – Demanda mensal de material*

Perceba, para efeito de conferência, que as necessidades do mês de outubro da figura 7.13 foram realmente somadas na exibição da demanda mensal.

O banco de dados que serviu esta demonstração é mostrado na figura 7.15.




| cod_item | data | necess_indep | necess_dep | recebntoprog | estoque disp | necliquida |
|---|---|---|---|---|---|---|
| 1 | 23/9/2001 | 40 | 0 | 0 | 0 | 75 |
| 1 | 12/10/2001 | 0 | 0 | 0 | 0 | 40 |
| 1 | 21/10/2001 | 50 | 28 | 65 | 0 | 80 |
| 1 | 13/11/2001 | 100 | 0 | 0 | 0 | 35 |
| 1 | 30/11/2001 | 0 | 0 | 0 | 0 | 100 |
| 1 | 1/12/2001 | 0 | 0 | 0 | 0 | 25 |
| 1 | 15/12/2001 | 0 | 0 | 0 | 0 | 30 |
| 1 | 1/1/2002 | 0 | 0 | 0 | 0 | 20 |
| 99 | 12/12/2001 | 12 | 0 | 0 | 0 | 0 |
| TF 2001 | 20/10/2001 | 1000 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 |

*Figura 7.15 – Tabela de dados que serve a demonstração neste item*

## 7.6.2 TESTE DE CONSISTÊNCIA

Se o usuário digitar algo não existente na tabela de cadastro, seria visto a saída mostrada pela figura 7.16.

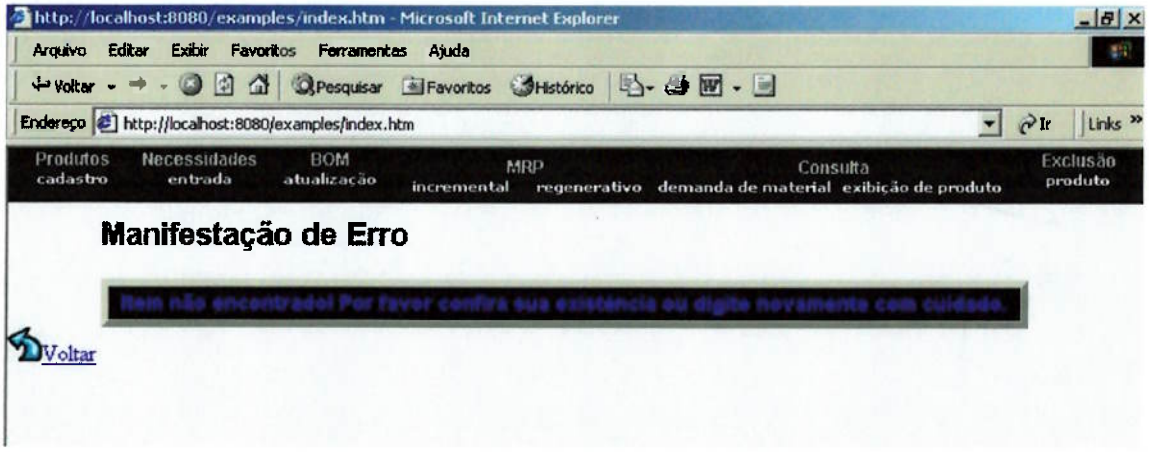



*Figura 7.16 – Erro devido à má digitação do usuário*

## 7.7 Exibição do Produto

Este item remete-se a necessidade que determinados funcionários de fábricas possuem de explodir um item em seus componentes para compor um orçamento ou diagnosticar um defeito, etc.

O usuário deve escolher a opção abaixo ao item Consulta da barra de opções para aparecimento da seguinte página.

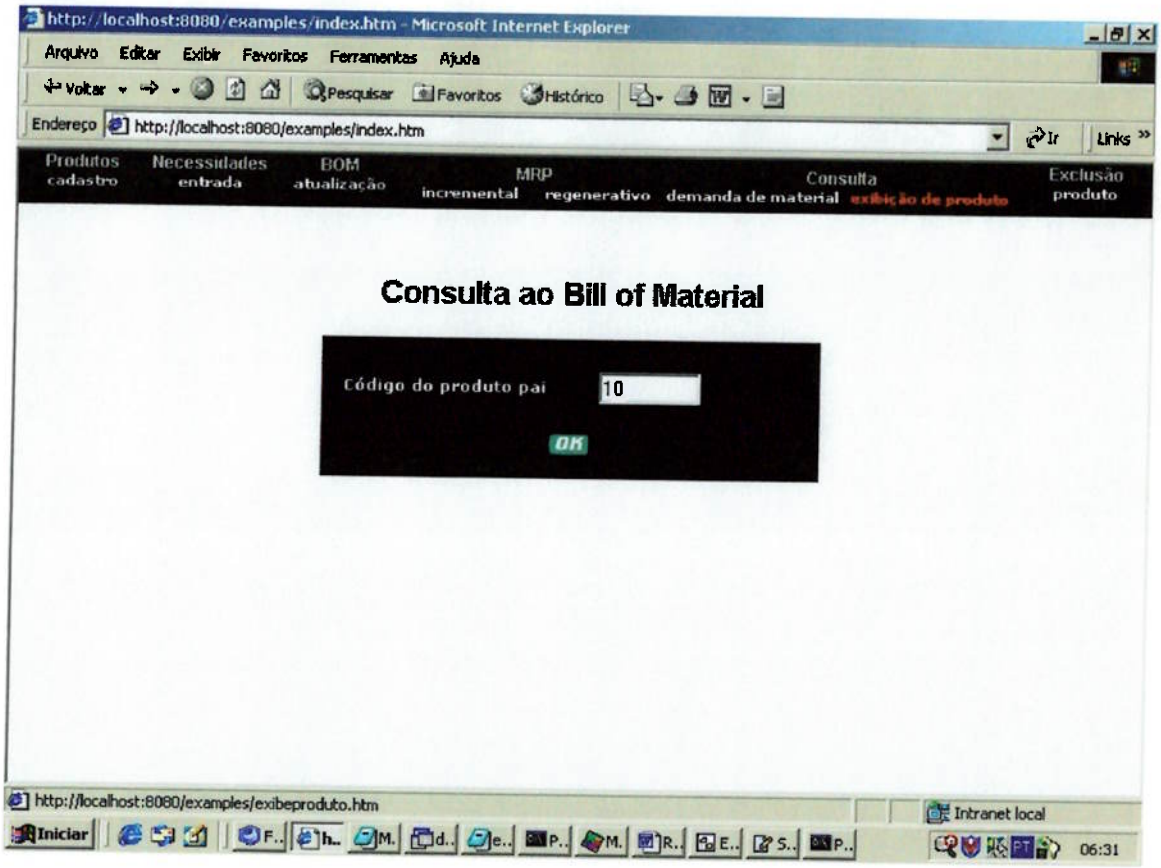



*Figura 7.17 – Página de Exibição do Produto*

O código inserido é o do produto pai, o qual se pesquisa neste momento.

## 7.7.1 FUNCIONAMENTO DO SERVLET

Mostra-se dois tipos de informação para o item selecionado: seus dados cadastrais e seus filhos - quantidade que aparecem, lead-time de fabricação e categoria – caso estes existam.

Caso o item não possua filhos é exibida uma mensagem ao usuário dizendo que o item é matéria prima.

As figuras seguintes mostram a possibilidade de se navegar pelo sistema em busca dos filhos do filho exibido, pois estes aparecem em formas de ***hyperlinks*** que permitem ao usuário novas buscas sem precisar voltar á tela inicial.

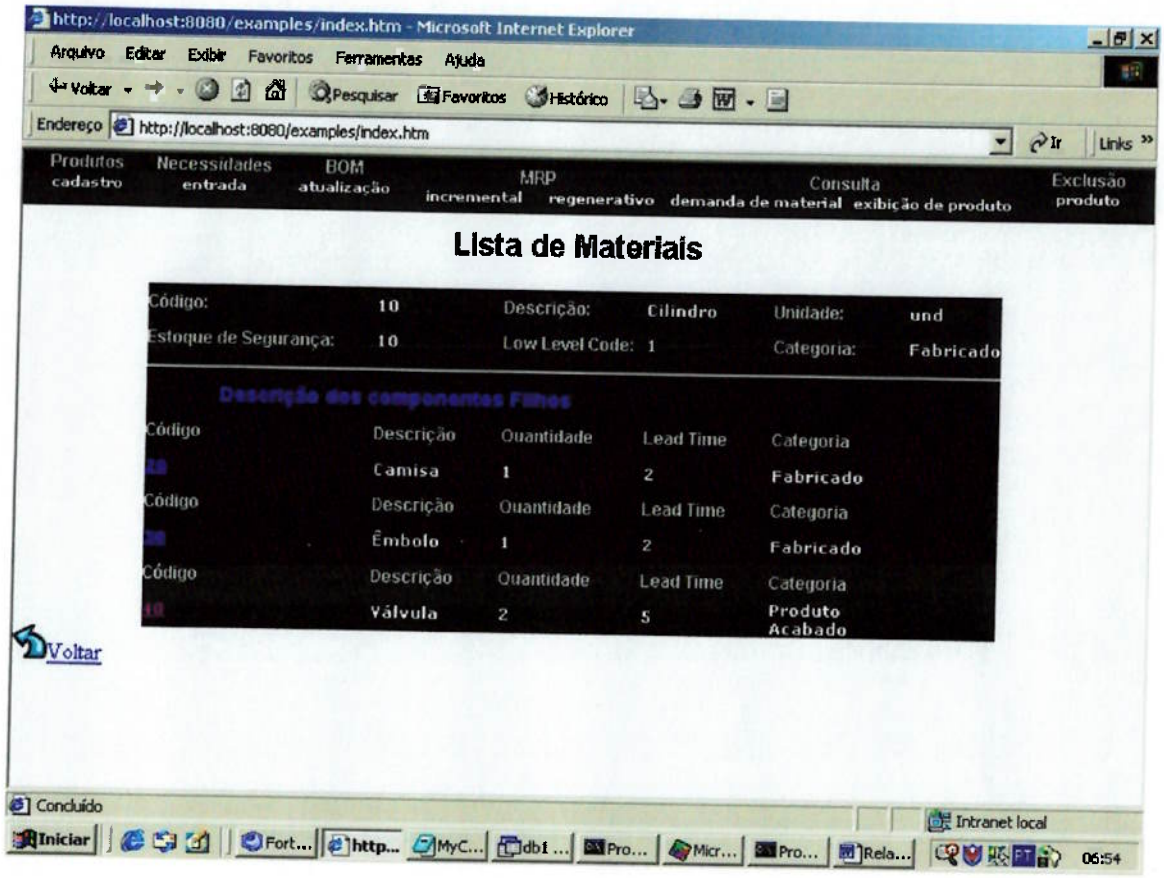



*Figura 7.18 – Saída da exibição do item 10*

Clicando por exemplo em **40** seriam encontradas as informações do item 40 – Válvula – como mostra a figura 7.19.

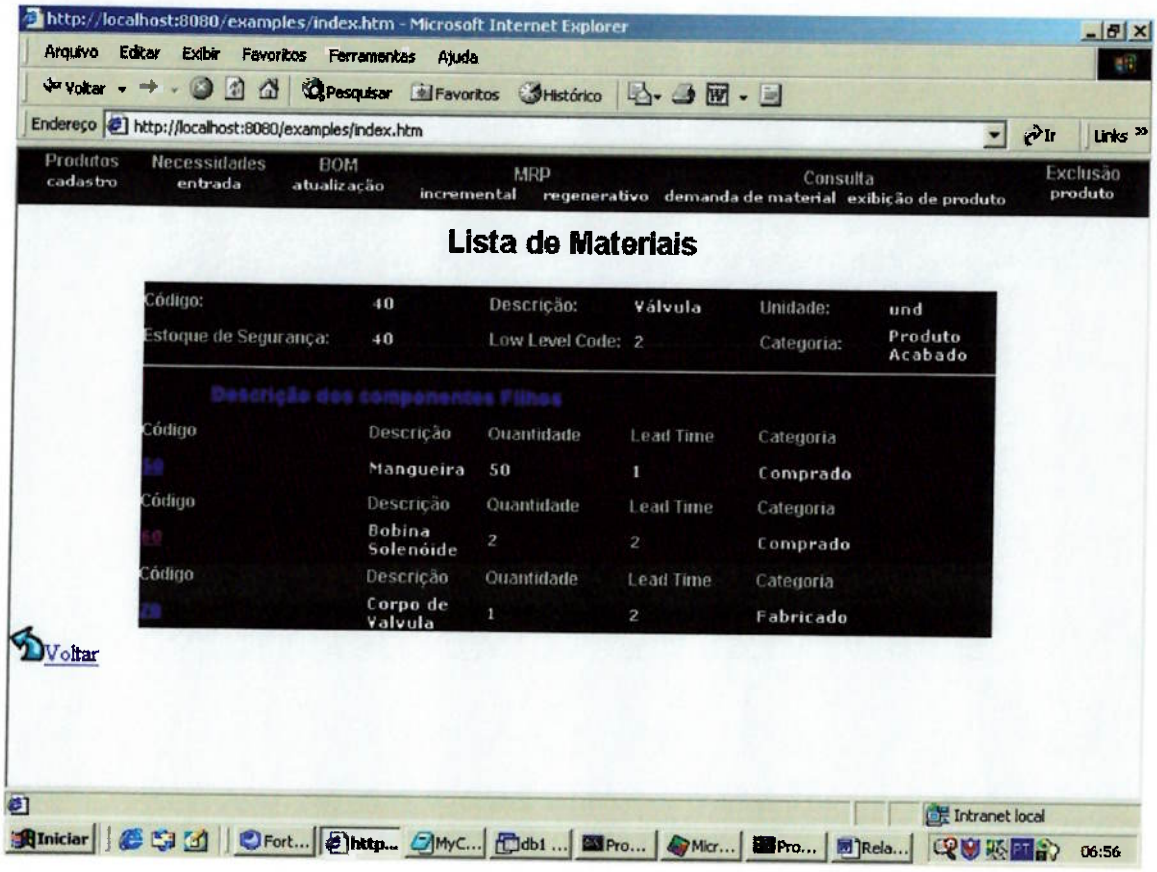



*Figura 7.19 – Saída da exibição do item 40*

O usuário, vendo que o item 60 é comprado, percebe que não há filhos a serem mostrados, mas uma vez que as informações cadastrais são importantes, clica no link **60** - item Bobina Solenóide - e obtém a saída visualizada na figura 7.20.

Note que neste caso é apresentada a mensagem identificando o item como uma matéria prima da fábrica, já que não foi encontrado nenhum filho em banco de dados.

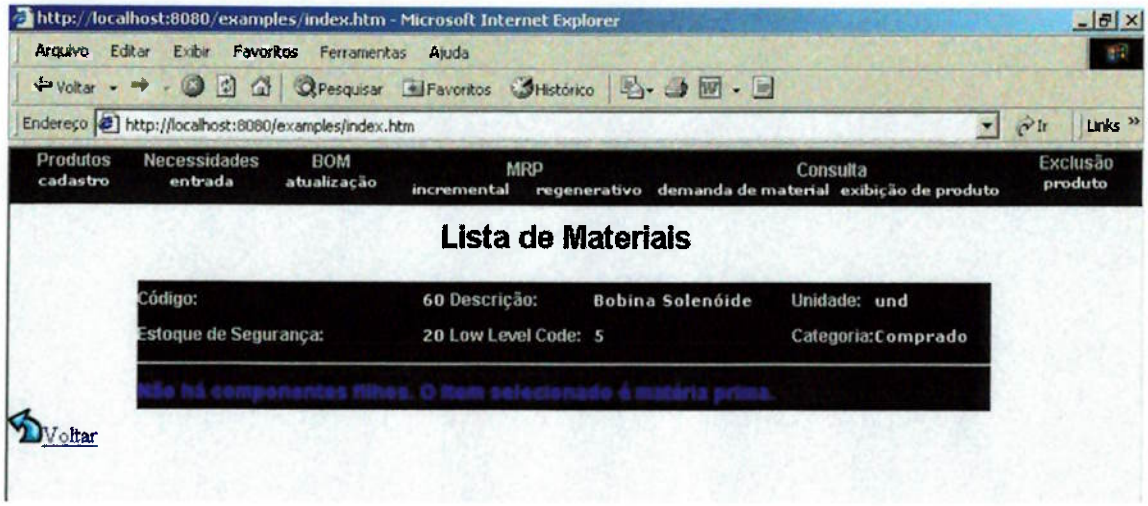



*Figura 7.20 – Saída da exibição do item 60*

Para melhor compreensão deste exemplo o item 10 é um cilindro e sua lista de materiais pode ser esquematizada pela figura 7.21.

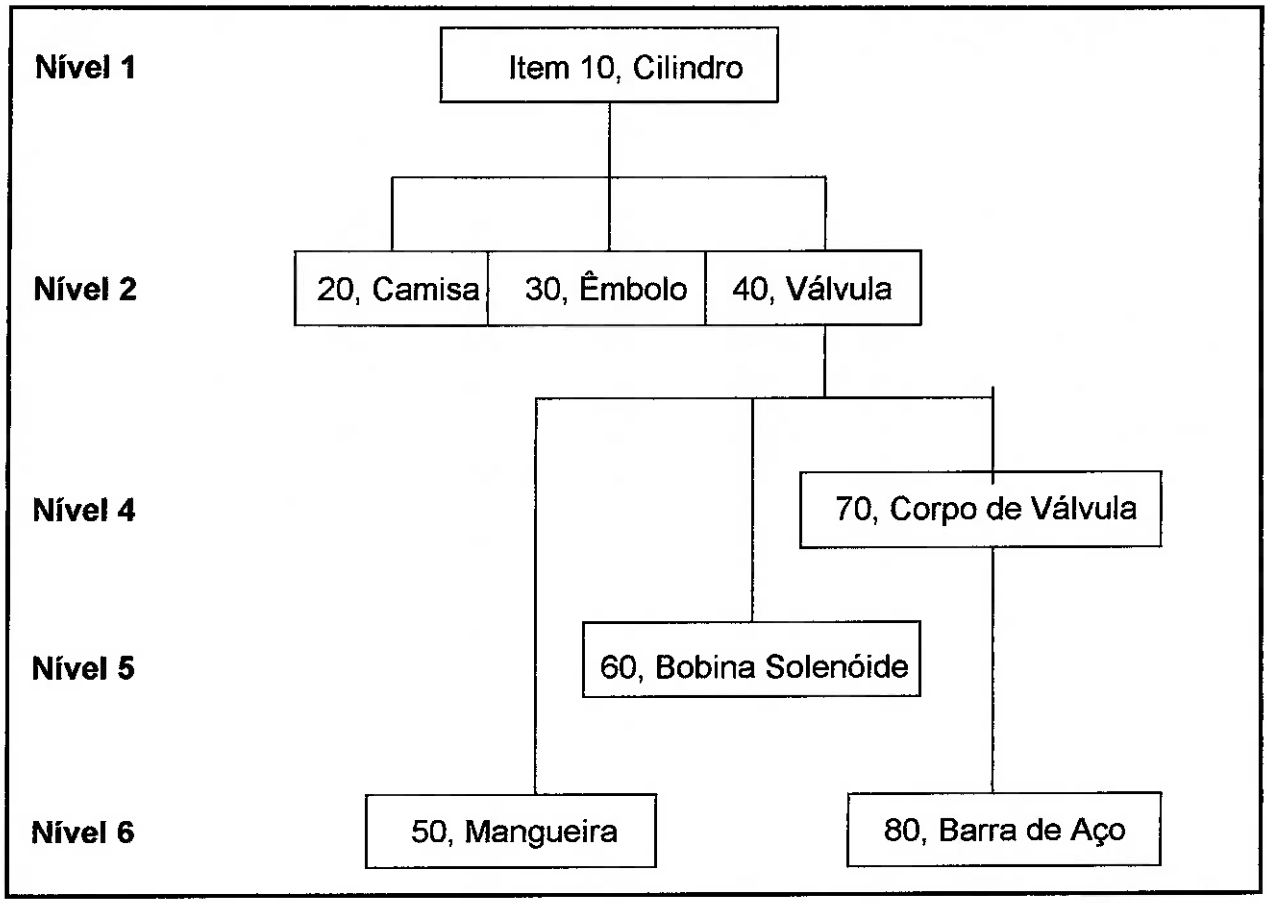



*Figura 7.21 – Visualização do Item 10*

Para efeito de conferência, a figura 7.22 mostra o banco de dados de estrutura do qual o sistema extraiu dados para a montagem da apresentação anterior.

Microsoft Access - [cadastro : Tabela]

Arquivo Editar Exibir Inserir Formatar Registros Ferramentas Janela Ajuda

| coditem | descritivo | unidade | estoquesegurança | LLC | categoria |
|---|---|---|---|---|---|
| 10 | Cilindro | und | 10 | 1 | Fabricado |
| 100 | Teste | und | 10 | 11 | Fabricado |
| 101 | Teste | und | 0 | 5 | Comprado |
| 20 | Camisa | und | 15 | 2 | Fabricado |
| 25 | Portinhola | und | 2 | 3 | Devastado |
| 29 | utilitario | und | 0 | 7 | Comprado |
| 30 | Êmbolo | und | 0 | 2 | Fabricado |
| 40 | Válvula | und | 40 | 2 | Produto Acabado |
| 50 | Mangueira | cm | 50 | 5 | Comprado |
| 60 | Bobina Solenóide | und | 20 | 4 | Comprado |
| 70 | Corpo de Valvula | und | 150 | 3 | Fabricado |
| 80 | Barra de Aço | mm | 10000 | 5 | Comprado |
| TF 2001 | Teste | und | 10 | 2 | Fabricado |
| | | | 0 | | |

*Figura 7.22 – Tabela de dados de Estrutura*

## 7.7.2 TESTE DE CONSISTÊNCIA

Caso o item digitado não exista em banco de dados, o programa produz a seguinte saída.

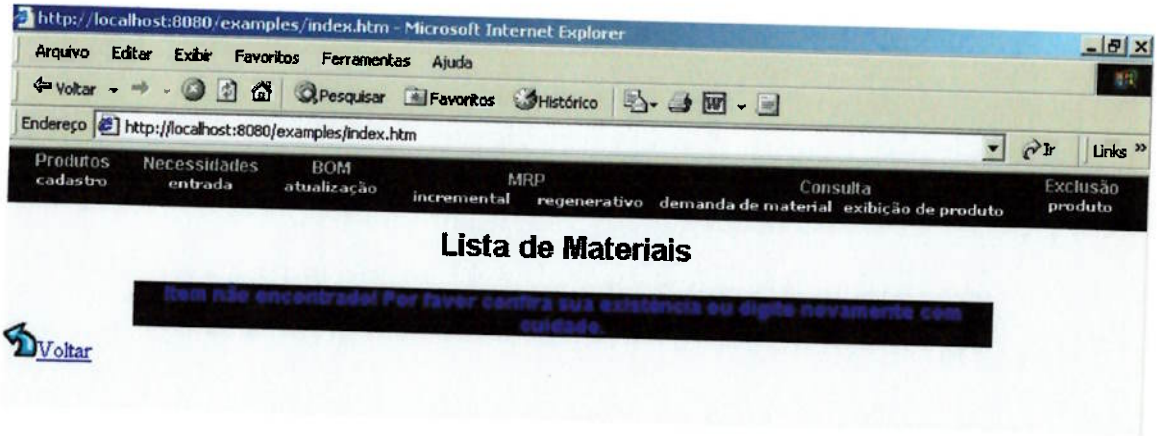



*Figura 7.23 – Saída para má digitação do usuário*

## *7.8 Listagem de Ordens de Obtenção de Produtos*

O modelo de dados adotado reflete as necessidades líquidas de cada item no período em que existam necessidades independentes ou dependentes.

As ordens foram um artifício para determinação da data em que o item deve **começar a ser produzido**, ou que deve ser feito **o pedido de compra** caso o mesmo seja comprado.

Para tanto é considerado um lead time diferente dos utilizados para cálculo da demanda dos itens filhos. Neste caso utiliza-se o **tempo crítico de obtenção de um produto** que por simplificação foi tomado como sendo o maior dos lead times de uma relação item pai – item filho.

Este tempo crítico é uma entidade fornecida por um processista de uma fábrica que consegue dizer para cada item quanto leva-se para ser produzido considerando-se os processos de fabricação e montagem dos filhos no pai e se estes processos são seqüenciais, paralelos ou ambos. Neste trabalho o tempo foi inserido manualmente.

### 7.8.1 FUNCIONAMENTO DO SERVLET

Este item é acionado pelo link, "listagem de ordens", abaixo a Consulta.

A figura 7.24 mostra a saída para necessidades e itens hipotéticos.

Note que não especificação de item nem de período para exibição das ordens. Elas são ordenadas por data.

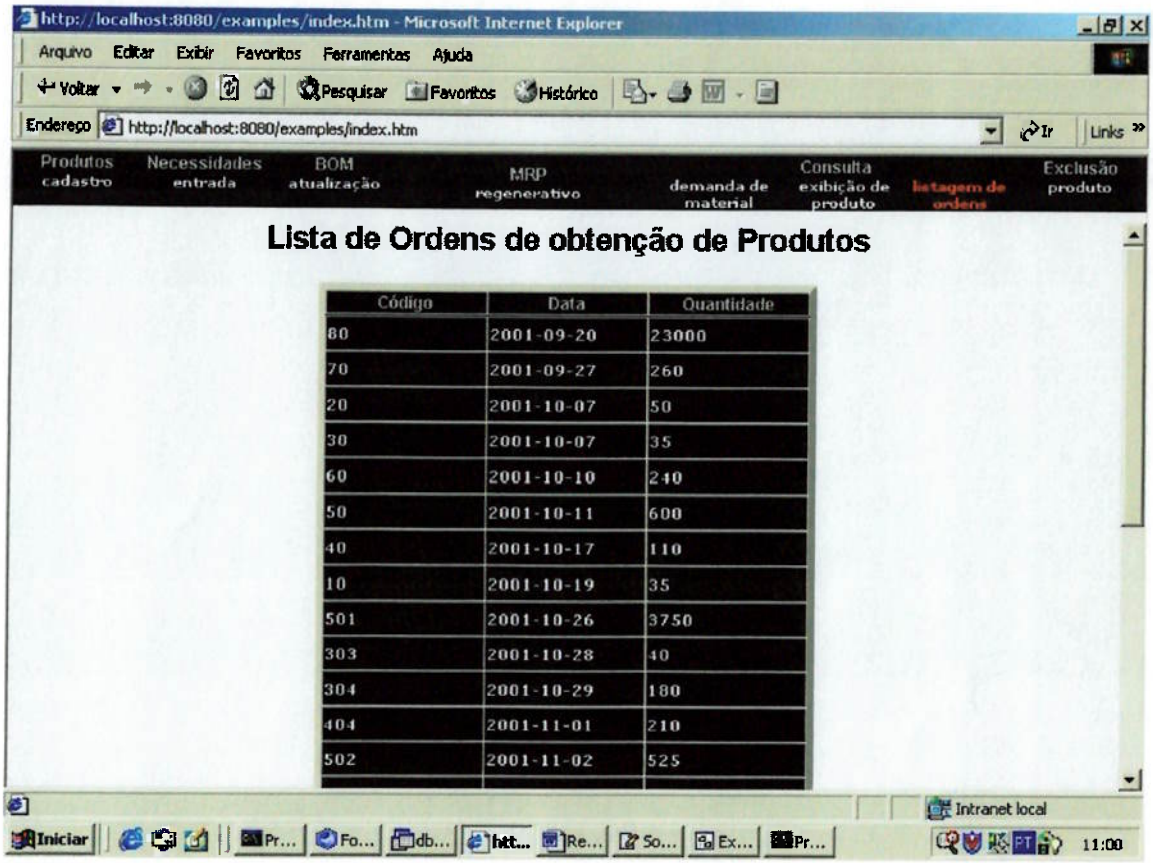



*Figura 7.24 – Listagem de Ordens de Obtenção de Produtos*

## 7.9 Exclusão de Produtos

Última funcionalidade do sistema, a opção de exclusão de itens do banco de dados é importante para, por exemplo, renovação de linhas de produtos com eliminação daqueles que sofreram a obsolescência.

Veja a página de entrada desta função quando o link da barra de opções é clicado.

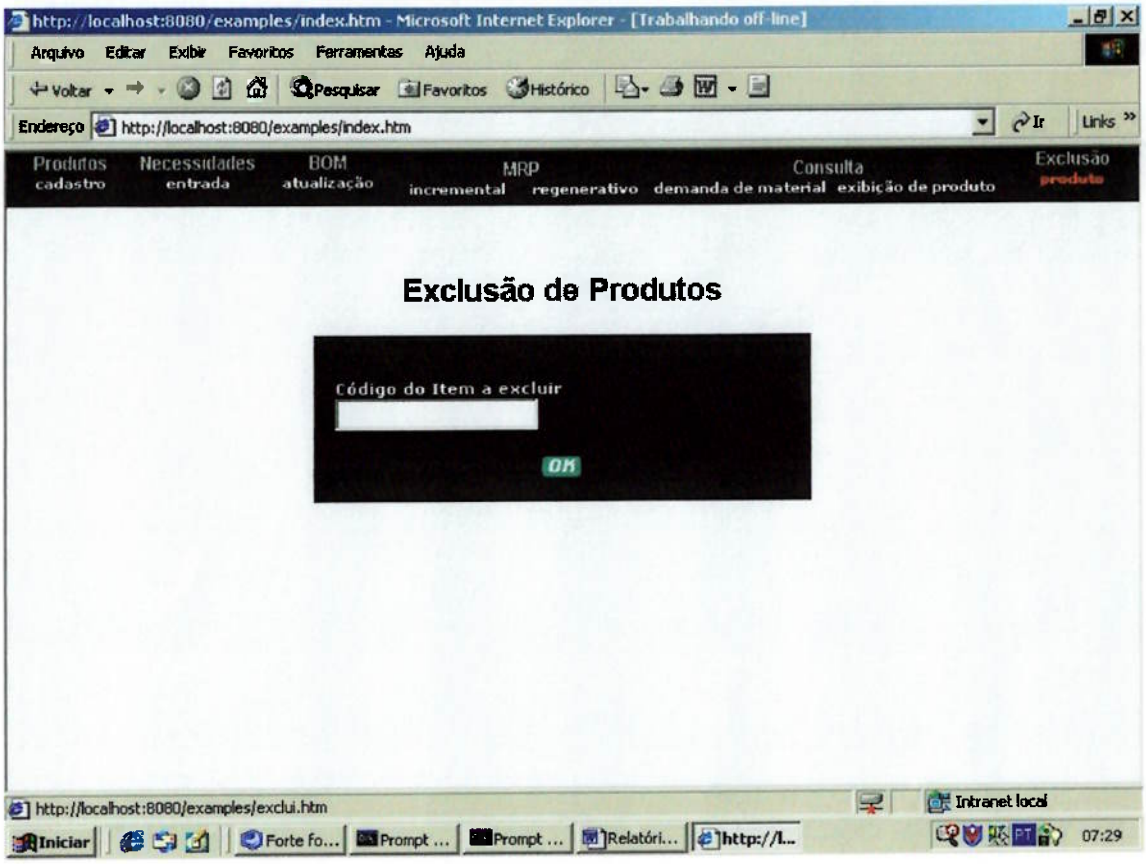



*Figura 7.25 – Página de entrada de Exclusão de Produtos*

## 7.9.1 FUNCIONAMENTO DO SERVLET

Neste trabalho o *data basis* utilizado – MSAccess – não faz a exclusão de um item de todas as tabelas onde o mesmo constitui um relacionamento. Desta maneira esta operação foi realizada "manualmente". Quando o item selecionado existe em tabela de BOM ou de Registro de Inventário, sua exclusão é mostrada ao usuário.

Veja a figura 7.26 para uma demonstração.

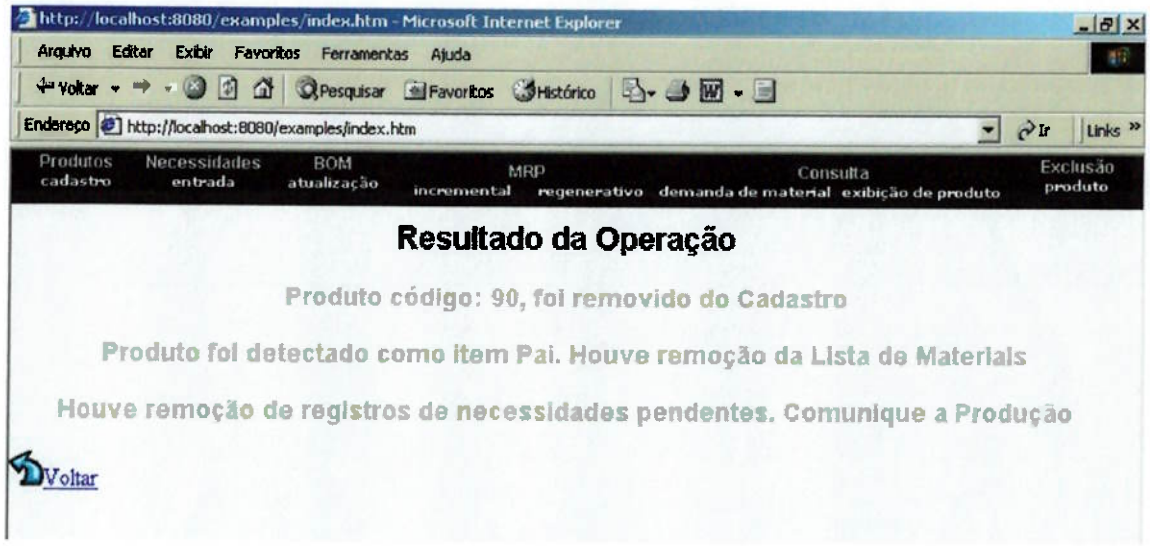



*Figura 7.26 – Saída da Exclusão de Produtos*

## 7.9.2 TESTE DE CONSISTÊNCIA

Caso o usuário, mais uma vez, digite o código do item a ser excluído errado e este não exista em banco de dados, o sistema mostra uma mensagem de erro conforme é visto na figura 7.27.

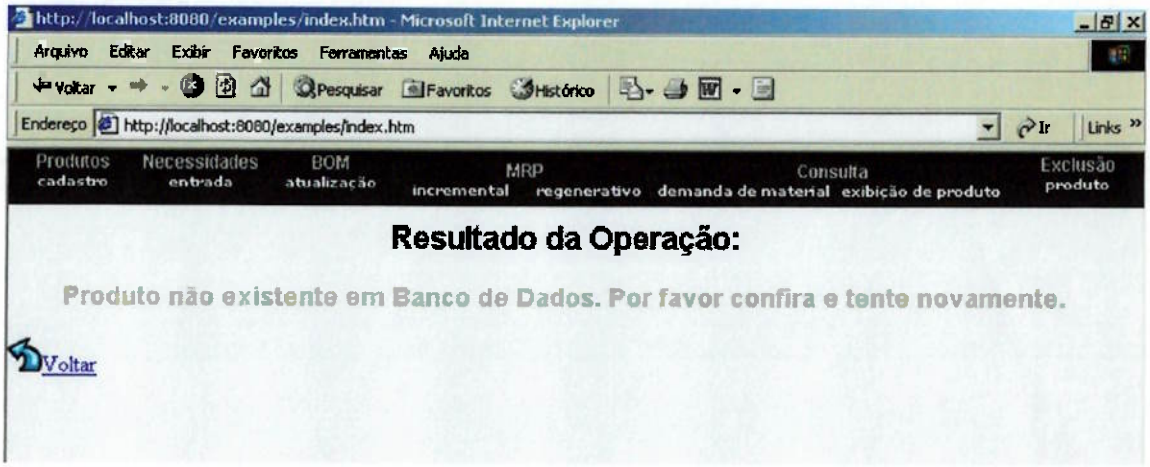



*Figura 7.27 – Mensagem de erro em função de má digitação do usuário*

# 8 Conclusões e Recomendações

## *8.1 Conclusões*

São conhecidos os conceitos de MRP, tecnologia *servlet*, ambiente cliente servidor, linguagem orientada a objetos e parte da linguagem SQL.

Foram definidos: o escopo do trabalho, os requisitos de projeto e foram criados os registros em banco de dados e seus relacionamentos e as interfaces HTML do sistema com o usuário.

Com a inteligência do projeto praticamente definida, propôs-se para o segundo semestre, em curso da disciplina Projeto Mecânico II, a codificação do sistema em JAVA e a integração das três partes chave: Banco de dados, Servidor e programa Java.

Foram construídos os *servlets* responsáveis pela inteligência e dinâmica do projeto onde foram feitos os ajustes no projeto básico devido problemas encontrados na lógica desenhada, restrições de tempo e formalizações ainda não conhecidas da linguagem de programação.

Ao finalizar este relatório o leitor estará simpatizado com **parte** do método de cálculo MRP e terá uma ótima noção de desenvolvimento de *servlets* ao examinar os anexos postos no item 10.

## 8.2 Recomendações e Sugestões

Recomenda-se para o estudante interessado no assunto:

- Estender o método do MRP, bem resumido no item 3, porém apenas parcialmente implementado neste trabalho.
- Desenvolver a modalidade de cálculo MRP – Incremental, onde apenas os dados modificados desde a última execução, são aqueles considerados para cálculo.
- Implementar a política de estoque, lote mínimo de produção, onde são produzidos um número mínimo de unidades independentemente da demanda devido a fatores econômicos e de planejamento.
- Instituir decisões inteligentes no sistema de forma a ser possível produzir os materiais no momento exato em que forem necessários considerando o tempo de produção dos itens componentes para por exemplo economizar mais capital de giro na empresa.

# 9 Bibliografia


- C. Fullmann, L. Ritzman, L. Krajewski, "M. Machado e R. Moura, MRP, MRP II, MRP II <MRP + JIT / KANBAN>, OPT, GDR", IMAN, 1989.
- Orlicky, Joseph, "Material Requirements Planning", McGraw-Hill, 1975.
- Gane, Chris; Sarson,Trish, "Análise Estruturada de Sistemas, Livros Técnicos e Científicos Editora S.A., 1986.
- Sun Microsystems, "Tutoriais Java", internet, site: http://www.javasoft.com
- Lemay, Laura; Cadenhead, Rogers; "Aprenda em 21 dias Java 2", Editora Campus, 2000.
- Chen, Peter; "Gerenciando Banco de Dados", McGraw-Hill: Newstec, 1990.
- Ramalho, José Antônio Alves, "SQL – A Linguagem dos Bancos de Dados", Editora Berkley, 1999.

# 10 Anexo – Listagens

## *10.1 Listagens HTML*

### 10.1.1 INICIO.HTM

```
<html>
<head>
        <title>entrada</title>

<link rel="StyleSheet" href="estilo.css">
</head>

<body>
<br><br><br><br><br>


<p align="center" class=TituloEntrada> Apresentação do Trabalho de
Formatura 2001  MRP Didático</p>

</body>
</html>
```

### 10.1.2 INDEX.HTM

```
<HTML>


<HEAD>

<title></title>

<!-- frames -->
<frameset  rows="45,*">
   <frame name="topo" src="barra2.htm" marginwidth="0" marginheight="0"
scrolling="no" frameborder="0">
   <frame name="main" src="inicio.htm" marginwidth="0" marginheight="0"
scrolling="auto" frameborder="0">
</frameset>
```

```
</html>
```

### 10.1.3 BARRA2.HTM

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
        <title>Untitled</title>

<link rel="StyleSheet" href="estilo.css">

</head>

<body>
<table cellpadding=0 cellspacing=0 border=0 width=100% bgcolor=black>
  <tr>
    <td width=90 align="center" class=BarraTituloLink
valign="top">Produtos<br><a href=produtos.htm class=BarraSubTituloLink
target="main">cadastro</td>
    <td width=90 align="center" class=BarraTituloLink
valign="top">Necessidades<br><a href=necessidade.htm
class=BarraSubTituloLink target="main">entrada</a></td>
    <td width=90 align="center" class=BarraTituloLink valign="top">BOM<br><a
href=bom.htm class=BarraSubTituloLink target="main">atualização</td>
    <td width=180>
          <table cellpadding=0 cellspacing=0 border=0 width=100%>
                <tr>
          <td align="center" colspan=3 class=BarraTituloLink
valign="top">MRP</td>
                </tr>
                <tr>
                 <td width=50% align="center"><a href=mrp.htm?var=incremental
class=BarraSubTituloLink target="main">incremental</td>
                 <td width=50% align="center"><a href=mrp.htm?var=regenerativo
class=BarraSubTituloLink target="main">regenerativo</td>
                </tr>
          </table>
         </td>
    <td width=250>
          <table cellpadding=0 cellspacing=0 border=0 width=100%>
                <tr>
```

```
        <td align="center" colspan=3 class=BarraTituloLink
valign="top">Consulta</td>
                </tr>
                <tr>
                 <td width=50% align="center"><a href=inventario.htm
class=BarraSubTituloLink target="main">registro de inventário</td>
                 <td width=50% align="center"><a href=exibeproduto.htm
class=BarraSubTituloLink target="main">Exibir Produto</td>
                </tr>
         </table>
        </td>
   <td><br><br></td>
        </tr>
</table>
</body>
</html>
```

### 10.1.4 CADASTRO.HTM

```
<html>
<head>
        <title>Produtos</title>

<link rel="StyleSheet" href="estilo.css">
</head>

<body>
<br><br>
<form action="http://localhost:8080/examples/servlet/cadastroproduto"
method="post"
<p align="center" class=TituloPagina>Cadastro de Produtos</p>
<table cellpadding=0 cellspacing=0 border=0 width=350 bgcolor=black
align="center">
<tr>
        <td colspan="2" height="15"> </td>
</tr>
<tr>
        <td width=15></td>
        <td class=CamposPagina valign="bottom" height="25">Código</td>
</tr>
<tr>
        <td width=15></td>
        <td><input type="Text" name=codigo size="20" class=inputs></td>
</tr>
```

```
<tr>
        <td width=15></td>
        <td class=CamposPagina valign="bottom" height="25">Descrição</td>
</tr>
<tr>
        <td width=15></td>
        <td><input type="Text" size="20" name=descricao class=inputs></td>
</tr>
<tr>
        <td width=15></td>
        <td class=CamposPagina valign="bottom" height="25">Unidade</td>
</tr>
<tr>
        <td width=15></td>
        <td><input type="Text" size="5" name=unidade class=inputs></td>
</tr>
<tr>
        <td colspan=2>
         <table cellpadding=0 cellspacing=0 border=0>
                <tr>
                        <td width=25></td>
                        <td class=CamposPagina valign="bottom"
height="25">Estoque de segurança</td>
                        <td width=15></td>
                        <td class=CamposPagina valign="bottom"
height="25">Categoria</td>
                </tr>
                        <td width=25></td>
                        <td><input type="Text" name=estoque size="5"
class=inputs></td>
                        <td width=15></td>
                        <td><input type="Text" name=catg size="15"
class=inputs></td>
                </tr>
   </table>
  </td>
</tr>
<tr>
        <td colspan="2" align="center" height="50"><input type="image"
value="Cadastrar" src=images/b_ok.gif border=0></td>
</tr>
</table>

</form>
</body>
```

```
</html>
```

## 10.1.5 ENTRADA.HTM

```
<html>
<head>
        <title>Entrada de necessidade</title>

<link rel="StyleSheet" href="estilo.css">
</head>

<body>
<br><br>
<form action="http://localhost:8080/examples/servlet/InsertNeedServlet"
method="post">
<p align="center" class=TituloPagina>Entrada de necessidade</p>
<table cellpadding=0 cellspacing=0 border=0 width=350 bgcolor=black
align="center">
<tr>
        <td colspan="2" height="15"> </td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">Código do item</td>
        <td><input type="Text" name=cod_produto size="5" class=inputs></td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina height="25" width="180">data pretendida
(dia/mes/ano)</td>
        <td class=CamposPagina><input type="Text" name=data size="10"
class=inputs></td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">Quantidade</td>
        <td><input type="Text" name=qtd size="5" class=inputs></td>
</tr>
<tr>
        <td colspan="3" align="center" height="50"><input type="image"
value="Cadastrar" src=images/b_ok.gif border=0></td>
```

```
</tr>
</table>

</form>
</body>
</html>
```

### 10.1.6 ATUALIZAÇÃO.HTM

```
<html>
<head>
        <title>BOM</title>

<link rel="StyleSheet" href="estilo.css">
</head>

<body>
<br><br>
<form action="http://localhost:8080/examples/servlet/cadastrobom"
method="post">
<p align="center" class=TituloPagina>Cadastro de BOM</p>
<table cellpadding=0 cellspacing=0 border=0 width=350 bgcolor=black
align="center">
<tr>
        <td colspan="2" height="15"> </td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">Código do
produto</td>
        <td><input type="Text" name=cod_produto size="5"  class=inputs></td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina height="25" width="180">Código do produto
filho</td>
        <td><input type="Text" name=cod_produto_filho size="5"
class=inputs></td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">Quantidade</td>
        <td><input type="Text" name=qtd size="5" class=inputs></td>
</tr>
<tr>
```

```
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">Lead time</td>
        <td><input type="Text" name=lead_time size="5" class=inputs></td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">LLC</td>
        <td><input type="Text" name=llc size="5" class=inputs></td>
</tr>
<tr>
        <td colspan="3" align="center" height="50"><input type="image"
value="Cadastrar" src=images/b_ok.gif border=0></td>
</tr>
</table>

</form>
</body>
</html>
```

### 10.1.7 EXIBIRPRODUTO.HTM

```
<html>
<head>
        <title>ConsultaBom</title>

<link rel="StyleSheet" href="estilo.css">
</head>

<body>
<br><br>
<form action="http//localhost:8080/examples/servlet/ConEstruturaServlet>
<p align="center" class=TituloPagina>Consulta ao Bill of Material</p>
<table cellpadding=0 cellspacing=0 border=0 width=350 bgcolor=black
align="center">
<tr>
        <td colspan="2" height="15"> </td>
</tr>
<tr>
        <td width=15 height="30"></td>
        <td class=CamposPagina  height="25" width="180">Código do produto
pai </td>
        <td><input type="Text" name=cod_produto_pai size="8"
class=inputs></td>
</tr>
```

```
<tr>
        <td colspan="3" align="center" height="50"><input type="image"
value="Consultar" src=images/b_ok.gif border=0></td>
</tr>
</table>

</form>
</body>
</html>
```

### 10.1.8 INVENTÁRIO2.HTM

```
<html>
<head>
        <title>Inventario</title>

<link rel="StyleSheet" href="estilo.css">
</head>

<body>
<form action="http://localhost:8080/examples/servlet/ListNecessidadeServlet"
method="post">

<br><br>


<p align="center" class=TituloPagina>Consulta ao Registro de Inventário</p>
<table cellpadding=0 cellspacing=0 border=0 width=350 bgcolor=black
align="center">
<tr>
        <td colspan="1" height="15"> </td>
</tr>
<tr>
        <td width=10 height="35"></td>
        <td class=CamposPagina height="25" width="250">Código do
Item        <input type="Text"
name=codigo size="8"  class=inputs></td>
</tr>
<tr>
      <td width=15 height="35"></td>
      <td class=CamposPagina height="30" width="250">Selecione
opção<br><br><dl><dd><dd><dd><input type=radio name="escolha"
value=d>Diário  <SELECT NAME="mes"></dl>
```

```
                    <option value="-1"> </option>
                    <option value=1>Outubro2001</option>
                    <option value=4>Novembro2001</option>
                <option value=6>Dezembro2001</option>
                    <option value=2>Janeiro2002</option></td>
</tr>
<tr>
        <td width=10 height="30"></td>
        <td class=CamposPagina height="30"
width="180"><dl><dd><dd><dd><input type=radio name=escolha value=m>
Mensal</dl>

        </td>
</tr>
<tr>
        <td colspan="3" align="center" height="50"><input type="image"
value="Listar" src=images/b_ok.gif border=0></td>
</tr>
</table>
</form>
</body>
</html>
```

## *10.2 Listagens Java*

### 10.2.1 CADASTROPRODUTOSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class cadastroproduto extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
   static final String PWD = "";
        public void init(ServletConfig config) throws ServletException
        {
                super.init(config);
```

```
            // open database connection
            try {
              Class.forName(ODBC_DRIVER).newInstance();
              m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
            } catch (Exception e) {
               e.printStackTrace();
               throw new ServletException("error in ODBC");
            }
      } // init

      public void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
            doPost(req, resp);
      } // doGet

      public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
            resp.setContentType("text/html");
            PrintWriter out = new PrintWriter(resp.getOutputStream());

            // get request parameters
            String codigo = "null";
            if (req.getParameterValues("codigo") != null) {
                  codigo= req.getParameterValues("codigo")[0];
            }
            String descricao = "null";
            if (req.getParameterValues("descricao") != null) {
                  descricao = req.getParameterValues("descricao")[0];
            }
            String unidade = "null";
            if (req.getParameterValues("unidade") != null) {
                  unidade= req.getParameterValues("unidade")[0];
            }
            String estoque = "null";
            if (req.getParameterValues("estoque") != null) {
                  estoque = req.getParameterValues("estoque")[0];
            }
            String catg = "null";
            if (req.getParameterValues("catg") != null) {
                  catg = req.getParameterValues("catg")[0];
            }
      try {
```

```
        PreparedStatement stmt= m_dbConnection.prepareStatement("insert into
cadastro (coditem, descritivo, unidade, estoqueseguranca, categoria) values
(?,?,?,?,?)");
        stmt.setString(1, codigo);
        stmt.setString(2, descricao);
        stmt.setString(3, unidade);
        stmt.setString(4, estoque);
        stmt.setString(5, catg);

        if (stmt.executeUpdate() != 1) {
          out.println("<HTML>");
          out.println("<TITLE>PMC423-Insert error</TITLE>");
          out.println("<BODY>");
          out.println("<h1>Error during insert</h1>");
          out.println("</BODY>");
          out.println("</HTML>");
        } else {
          out.println("<HTML>");
          out.println("<TITLE>PMC423-Insert error</TITLE>");
          out.println("<BODY>");
          out.println("<h1>Produto inserido!</h1>");
          out.println("</BODY>");
          out.println("</HTML>");
        }
      } catch (Exception e) {
          out.println("<HTML>");
          out.println("<TITLE>PMC423-Insert error</TITLE>");
          out.println("<BODY>");
          out.println(e.toString());
          out.println("</BODY>");
          out.println("</HTML>");
      }
              out.close();
        } // doPost

} //cadastroproduto
```

### 10.2.2 CADASTROBOMSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class cadastrobom extends HttpServlet
```

```
{
  Connection m_dbConnection = null;
  static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
  static final String DSN = "jdbc:odbc:PMC423";
  static final String USER = "";
  static final String PWD = "";
      public void init(ServletConfig config) throws ServletException
      {
            super.init(config);

            // open database connection
            try {
              Class.forName(ODBC_DRIVER).newInstance();
              m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
            } catch (Exception e) {
               e.printStackTrace();
               throw new ServletException("error in ODBC");
            }
      } // init

      public void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
            doPost(req, resp);
      } // doGet

      public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
            resp.setContentType("text/html");
            PrintWriter out = new PrintWriter(resp.getOutputStream());

            // get request parameters
            String cod_produto = "null";
            if (req.getParameterValues("cod_produto") != null) {
                  cod_produto= req.getParameterValues("cod_produto")[0];
            }
            String cod_produto_filho = "null";
            if (req.getParameterValues("cod_produto_filho") != null) {
                  cod_produto_filho
=
req.getParameterValues("cod_produto_filho")[0];
            }
            String qtd = "null";
            if (req.getParameterValues("qtd") != null) {
```

```
                qtd= req.getParameterValues("qtd")[0];
            }
            String lead_time = "null";
            if (req.getParameterValues("lead_time") != null) {
                lead_time = req.getParameterValues("lead_time")[0];
            }
            String llc = "null";
            if (req.getParameterValues("llc") != null) {
                llc = req.getParameterValues("llc")[0];
            }
    try {
      PreparedStatement stmt= m_dbConnection.prepareStatement("insert into
estrutura (codpai, codfilho, quantidade, lead_time, LLC) values (?,?,?,?,?)");
      stmt.setString(1, cod_produto);
      stmt.setString(2, cod_produto_filho);
      stmt.setString(3, qtd);
      stmt.setString(4, lead_time);
      stmt.setString(5, llc);

      if (stmt.executeUpdate() != 1) {
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Insert error</TITLE>");
        out.println("<BODY>");
        out.println("<h1>Error during insert</h1>");
        out.println("</BODY>");
        out.println("</HTML>");
      } else {
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Insert error</TITLE>");
        out.println("<BODY>");
        out.println("<h1>Estrutura inserida!</h1>");
        out.println("</BODY>");
        out.println("</HTML>");
      }
    } catch (Exception e) {
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Insert error</TITLE>");
        out.println("<BODY>");
        out.println(e.toString());
        out.println("</BODY>");
        out.println("</HTML>");
    }
            out.close();
      } // doPost
```

```
} //cadastro_bom
```

### 10.2.3 INSERTNEEDSERVLET

```
/*
 * InsertNeedServlet.java
 *
 * Created on 15 de Outubro de 2001, 21:10
 */

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class InsertNeedServlet extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
   static final String PWD = "";
        public void init(ServletConfig config) throws ServletException
        {
                super.init(config);

                // open database connection
                try {
                  Class.forName(ODBC_DRIVER).newInstance();
                  m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
                } catch (Exception e) {
                   e.printStackTrace();
                   throw new ServletException("error in ODBC");
                }
        } // init

        public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
                doPost(req, resp);
        } // doGet

public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
```

```
{
        resp.setContentType("text/html");
        PrintWriter out = new PrintWriter(resp.getOutputStream());

        // get request parameters
        String cod_produto = "null";
        if (req.getParameterValues("cod_produto") != null) {
                cod_produto= req.getParameterValues("cod_produto")[0];
        }
        String data = "null";
        if (req.getParameterValues("data") != null) {
                data = req.getParameterValues("data")[0];
        }
        String qtd = "null";
        if (req.getParameterValues("qtd") != null) {
                qtd= req.getParameterValues("qtd")[0];
        }

    try {
      PreparedStatement stmt= m_dbConnection.prepareStatement("insert into
RegInvent (cod_item, data, necess_indep) values (?,?,?)");
      stmt.setString(1, cod_produto);
      stmt.setString(2, data);
      stmt.setString(3, qtd);

      if (stmt.executeUpdate() != 1) {
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Insert error</TITLE>");
        out.println("<BODY>");
        out.println("<h1>Error during insert</h1>");
        out.println("</BODY>");
        out.println("</HTML>");
      } else {
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Insert error</TITLE>");
        out.println("<BODY>");
        out.println("<h1>Necessidade Independente inserida!</h1>");
        out.println("</BODY>");
        out.println("</HTML>");
      }
    } catch (Exception e) {
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Insert error</TITLE>");
        out.println("<BODY>");
        out.println(e.toString());
```

```
        out.println("</BODY>");
        out.println("</HTML>");
      }
            out.close();
      } // doPost

} //InsertNeedServlet
```

### 10.2.4 REMOVESERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.lang.*;
import java.math.*;

public class RemoveServlet extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
   static final String PWD = "";
      public void init(ServletConfig config) throws ServletException
      {
            super.init(config);

            // open database connection
            try {
              Class.forName(ODBC_DRIVER).newInstance();
              m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
            } catch (Exception e) {
               e.printStackTrace();
               throw new ServletException("error in ODBC");
            }
      } // init

      public void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
            doPost(req, resp);
```

```
    } // doGet

    public void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
    {
            resp.setContentType("text/html");
            PrintWriter out = new PrintWriter(resp.getOutputStream());

            // get request parameters
            String codigo = "null";
            if (req.getParameterValues("codigo") != null) {
                    codigo = req.getParameterValues("codigo")[0];
            }
            int intcodigo = new Integer(codigo).intValue();
    try {
     PreparedStatement stmt= m_dbConnection.prepareStatement("select *
from cadastro where coditem = ?");
     stmt.setInt(1, intcodigo);
     ResultSet rs = stmt.executeQuery();

     if(rs.next()){

         stmt= m_dbConnection.prepareStatement("delete from cadastro
where coditem = ?");
         stmt.setInt(1, intcodigo);
         if (stmt.executeUpdate() != 1) {
         out.println("<HTML>");
         out.println("<TITLE>PMC423-Remove error</TITLE>");
         out.println("<BODY>");
         out.println("<h1>Error during remove</h1>");
         out.println("</BODY>");
         out.println("</HTML>");
      } else {
         out.println("<HTML>");
         out.println("<TITLE>PMC423-Insert error</TITLE>");
         out.println("<link rel=StyleSheet href=../../../../examples/estilo.css>");
         out.println("<BODY>");
         out.println("<p align=center class=TituloPagina>Resultado da
Operação</p>");
         out.println("<p align=center class=SubTituloPagina>Produto código: "
+ codigo + ", foi removido do Cadastro</p>");

      }
```

```
            stmt= m_dbConnection.prepareStatement("select * from estrutura
where codpai = ?");
            stmt.setInt(1, intcodigo);
            ResultSet rsestrutura = stmt.executeQuery();

            if(rsestrutura.next()){
               stmt= m_dbConnection.prepareStatement("delete from estrutura
where codpai = ?");
               stmt.setInt(1, intcodigo);
               stmt.executeUpdate();
               out.println("<p align=center class=SubTituloPagina>Produto foi
detectado como item Pai. Houve remoção da Lista de Materiais</p>");

               stmt= m_dbConnection.prepareStatement("select * from RegInvent
where cod_item = ?");
               stmt.setInt(1, intcodigo);
               ResultSet rsreginvent = stmt.executeQuery();

               if(rsreginvent.next()){
                  stmt= m_dbConnection.prepareStatement("delete from
RegInvent where cod_item = ?");
                  stmt.setInt(1, intcodigo);
                  stmt.executeUpdate();

                  out.println("<p align=center class=SubTituloPagina>Houve
remoção de registros de necessidades pendentes. Comunique a
Produção</p>");

               }//if reginvent.next

            }//if rsestrutura.next

         }//if rs.next
         else{
            out.println("<HTML>");
            out.println("<TITLE>PMC423-Insert error</TITLE>");
            out.println("<link rel=StyleSheet href=../../../../examples/estilo.css>");
            out.println("<BODY>");
            out.println("<p align=center class=TituloPagina>Resultado da
Operação:</p>");
            out.println("<p align=center class=SubTituloPagina>Produto não
existente em Banco de Dados. Por favor confira e tente novamente.</p>");

         }//else rs.next
```

```
        out.println("<a href=javascript:history.back()><img
src=../../../../examples/images/return.gif border=0>Voltar</a>");
        out.println("</BODY>");
        out.println("</HTML>");

        } catch (Exception e) {
            out.println("<HTML>");
            out.println("<TITLE>PMC423-Remove error</TITLE>");
            out.println("<BODY>");
            out.println(e.toString());
            out.println("</BODY>");
            out.println("</HTML>");
        }
                out.close();
          } // doPost

} //RemoveServlet
```

## 10.2.5 EXIBEPRODUTOSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class ExibeProdutoServlet extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
   static final String PWD = "";

   public void init(ServletConfig config) throws ServletException
        {
                super.init(config);

                // open database connection
                try {
                  Class.forName(ODBC_DRIVER).newInstance();
                  m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
                } catch (Exception e) {
                   e.printStackTrace();
```

```
                throw new ServletException("error in ODBC");
            }
        } // init

        public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
                doPost(req, resp);
        } // doGet

        public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
                resp.setContentType("text/html");
                PrintWriter out = new PrintWriter(resp.getOutputStream());


            // get request parameters

            String aux = "null";
                String codpai = "null";
                if (req.getParameterValues("codpai") != null) {
                        codpai = req.getParameterValues("codpai")[0];
                }

      try {

        PreparedStatement stmt = m_dbConnection.prepareStatement("Select
descritivo, unidade, estoqueseguranca, LLC, categoria from cadastro where
coditem=? ");
        stmt.setString(1, codpai);//verificar este indice 1
        ResultSet rscadastro = stmt.executeQuery();


        out.println("<html>");
        out.println("<title>PMC423 - Exibir Produto</title>");
        out.println("<link rel=StyleSheet href=../../../../examples/estilo.css>");
        out.println("<body>");
        out.println("<p align=center class=TituloPagina> Lista de Materiais</p>");
        out.println("<table cellpadding=0 cellspacing=0 border=0 width=600
bgcolor=black align=center>");

        if(rscadastro.next()) {

         do{
```

```
        out.println("<tr><td class=BarraTituloLink height=25
width=50>Código: </td><td class=CamposPagina>"+codpai+"<td
class=BarraTituloLink width=50>Descrição:</td><td
class=CamposPagina>"+rscadastro.getString(1)+"</td><td
class=BarraTituloLink height=25 width=40>Unidade: </td><td
class=CamposPagina>"+rscadastro.getString(2)+"</td></tr>");
        out.println("<tr><td class=BarraTituloLink height=25
width=160>Estoque de Segurança: </td><td
class=CamposPagina>"+rscadastro.getString(3)+"</td><td
class=BarraTituloLink width=100>Low Level Code: </td><td
class=CamposPagina>"+rscadastro.getString(4)+"</td><td
class=BarraTituloLink width=50>Categoria:</td><td
class=CamposPagina>"+rscadastro.getString(5)+"</td></tr>");
      }while(rscadastro.next());



      out.println("<tr><td colspan=6><hr></td></tr>");

      stmt = m_dbConnection.prepareStatement("Select estrutura.codfilho,
cadastro.descritivo, estrutura.quantidade, estrutura.lead_time,
cadastro.categoria from estrutura,cadastro where estrutura.codpai=? and
cadastro.coditem=estrutura.codfilho");
      stmt.setString(1, codpai);
      ResultSet rsestrutura = stmt.executeQuery();


      if(rsestrutura.next()){
       out.println("<tr><td align=center colspan=3 class=JabaTituloLink
height=25>Descrição dos componentes Filhos</td></tr>");

          do{

            aux=rsestrutura.getString(1); //variável que permite a construção
dos links de recursão
            out.println("<tr><td class=BarraTituloLink height=25
width=90>Código</td><td class=BarraTituloLink height=25
width=90>Descrição</td><td class=BarraTituloLink
width=90>Quantidade</td><td class=BarraTituloLink width=90>Lead
Time</td><td class=BarraTituloLink width=90>Categoria</td></tr>");
            out.println("<tr><td class=CamposPagina height=25><a
href=http://localhost:8080/examples/servlet/ExibeProdutoServlet?codpai="+aux
+">"+aux+"</a></td><td
class=CamposPagina>"+rsestrutura.getString(2)+"</td><td
class=CamposPagina>"+rsestrutura.getString(3)+"</td><td
```

```
class=CamposPagina>"+rsestrutura.getString(4)+"</td><td
class=CamposPagina>"+rsestrutura.getString(5)+"</td></tr>");


            }while(rsestrutura.next());

          }//if
          else{
            out.println("<tr><td colspan=5 class=JabaTituloLink height=25>Não
há componentes filhos. O Item selecionado é matéria prima.");
          }
        }//if
         else{
           out.println("<tr><td align=center colspan=5 class=JabaTituloLink
height=25>Item não encontrado! Por favor confira sua existência ou digite
novamente com cuidado.");
         }
        out.println("</table>");
        out.println("<a                              href=javascript:history.back()><img
src=../../../../examples/images/return.gif border=0>Voltar</a>");
        out.println("</body>");
        out.println("</html>");




      }//try
         catch (Exception e) {
         out.println("<HTML>");
         out.println("<TITLE>PMC423-List error</TITLE>");
         out.println("<BODY>");
         out.println(e.toString());
         out.println("</BODY>");
         out.println("</HTML>");
      }
               out.close();
        } // doPost

} //ExibeProdutoServlet
```

## 10.2.6 LISTNECESSIDADESERVLET

```
import java.io.*;
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;


public class ListNecessidadeServlet extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
   static final String PWD = "";

   /** Initializes the servlet.
   */
    public void init(ServletConfig config) throws ServletException
        {
                super.init(config);

                // open database connection
                try {
                  Class.forName(ODBC_DRIVER).newInstance();
                  m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
                } catch (Exception e) {
                   e.printStackTrace();
                   throw new ServletException("error in ODBC");
                }
        } // init


   /** Handles the HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   */
   public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
                doPost(req, resp);
        } // doGet

   /** Handles the HTTP <code>POST</code> method.
   * @param request servlet request
   * @param response servlet response
   */
```

```
  public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
      {
            resp.setContentType("text/html");
            PrintWriter out = new PrintWriter(resp.getOutputStream());

          // get request parameters
            String codigo = "null";
            if (req.getParameterValues("codigo") != null) {
                  codigo = req.getParameterValues("codigo")[0];
          }
          int intcodigo = new Integer(codigo).intValue();

          String escolha = "null";
            if (req.getParameterValues("escolha") != null) {
                  escolha = req.getParameterValues("escolha")[0]; //valor de
seleção diário X mensal
          }
          String mes = "null";
            if (req.getParameterValues("mes") != null) {
                  mes = req.getParameterValues("mes")[0];  //valor de
seleção do mes pretendido
          }
            int intmes = new Integer(mes).intValue();

     try {

       int dataAtual=0;
       int ultimo=0;
       int i=0;
       Date MesMaior;
       Date MesMenor;

       PreparedStatement stmt = m_dbConnection.prepareStatement("Select
descritivo from cadastro where coditem=?");
       stmt.setInt(1, intcodigo);//verificar este indice 1
       ResultSet rsname = stmt.executeQuery();


       out.println("<html>");
       out.println("<title>PMC423 - Listar Necessidades</title>");
       out.println("<link rel=StyleSheet href=../../../../examples/estilo.css>");
       out.println("<body>");

       if (rsname.next()){
```

```
        if (escolha.equals("d")) {

            out.println("<p align=center class=TituloPagina> Lista diária de
necessidades do item: "+rsname.getString(1)+"</p>");

            switch (intmes) {
               case 1:
                  out.println("<p align=center
class=SubTituloPagina>  Outubro de 2001</p>");
                  ultimo=31;

                  MesMaior= new Date (101,9,31);
                  MesMenor= new Date (101,9,1);
                  break;
               case 2:
                  out.println("<p align=center
class=SubTituloPagina>  Janeiro de 2002</p>");
                  ultimo=31;
                  MesMaior= new Date(102,0,31);
                  MesMenor= new Date(102,0,1);
                  break;
               case 4:
                  out.println("<p align=center
class=SubTituloPagina>  Novembro de 2001</p>");
                  ultimo=30;
                  MesMaior= new Date(101,10,30);
                  MesMenor= new Date(101,10,1);
                  break;
               case 6:
                  out.println("<p align=center
class=SubTituloPagina>  Dezembro de 2001</p>");
                  ultimo=31;
                  MesMaior= new Date(101,11,31);
                  MesMenor= new Date(101,11,1);
                  break;
               default: MesMaior = new Date (101,8,30);
                     MesMenor = new Date (101,8,1);

            }//switch

            out.println("<table cellpadding=0 cellspacing=0 border=5 width=650
bgcolor=black align=center>");
            out.println("<tr><td align=center
class=BarraTituloLink>Domingo</td><td align=center
```

```
class=BarraTituloLink>Segunda-Feira</td><td align=center
class=BarraTituloLink>Terça-Feira</td><td align=center
class=BarraTituloLink>Quarta-Feira</td><td align=center
class=BarraTituloLink>Quinta-Feira</td><td align=center
class=BarraTituloLink>Sexta-Feira</td><td align=center
class=BarraTituloLink>Sábado</td></tr>");
                out.println("<tr>");
                for (i=1; i<=intmes; i++){
                   out.println("<td width=20 height=25 >");
                   out.println("<br> </td>");
                }

                stmt = m_dbConnection.prepareStatement("Select data,necliquida
from RegInvent where cod_item=? and data >= ? and data <= ? order by 1");
                stmt.setInt(1, intcodigo);
                stmt.setDate(2, MesMenor);
                stmt.setDate(3, MesMaior);
                ResultSet rsnecessidade = stmt.executeQuery();

                int teste = 0;// variável auxiliar p/ o teste de entrada no if abaixo

                if (rsnecessidade.next()){

                  do{
                     Date data = rsnecessidade.getDate(1);
                     int d = data.getDate();

                     for(dataAtual=1; dataAtual<=ultimo; dataAtual++){
                        out.println("<td align=center width=20 height=25
class=CamposPagina>"+dataAtual);
                        if(d == dataAtual){
                          out.println("<br>"+rsnecessidade.getString(2)+"</td>");
                          if(rsnecessidade.next()){;//avança uma linha do bco de
dados
                             data = rsnecessidade.getDate(1);
                             d = data.getDate();
                          }//if
                        }//if
                          else{
                            out.println("<br>0</td>");
                          }//else

                        if((dataAtual+intmes)==7){
                          out.println("</tr>");
                          out.println("<tr>");
```

```
                    }//if
                    if((dataAtual+intmes)==14){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==21){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==28){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==35){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                 }//for
              } while(rsnecessidade.next());//do
           }//if rsnecessidade.next()
           else{
              for(dataAtual=1; dataAtual<=ultimo; dataAtual++){
                    out.println("<td align=center width=20 height=25
class=CamposPagina>"+dataAtual);
                    out.println("<br>0</td>");
                    if((dataAtual+intmes)==7){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==14){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==21){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==28){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
                    if((dataAtual+intmes)==35){
                       out.println("</tr>");
                       out.println("<tr>");
                    }//if
```

```
            }//for

          }//else
          out.println("</table>");

        }//if escolha == d
        else if(escolha.equals("m")){

          out.println("<p align=center class=TituloPagina> Lista mensal de
necessidades do item: "+rsname.getString(1)+"</p>");

          stmt = m_dbConnection.prepareStatement("Select data,necliquida
from RegInvent where cod_item=? order by 1");
          stmt.setInt(1, intcodigo);
          ResultSet rsnecessidade = stmt.executeQuery();

          out.println("<table cellpadding=0 cellspacing=0 border=5 width=650
bgcolor=black align=center>");
          out.println("<tr><td align=center
class=BarraTituloLink>Setembro</td><td align=center
class=BarraTituloLink>Outubro</td><td align=center
class=BarraTituloLink>Novembro</td><td align=center
class=BarraTituloLink>Dezembro</td><td align=center
class=BarraTituloLink>Janeiro 2002</td></tr>");
          out.println("<tr>");

          Date setinicio = new Date(101,8,0);
          Date setfim = new Date(101,8,31);
          int NecSet=0;
          Date dataset = new Date(101,0,1);//inicialização de dataset com valor
arbitrário
          if (rsnecessidade.next())
          dataset = rsnecessidade.getDate(1);

          while(dataset.after(setinicio) && dataset.before (setfim)){

            String AuxSet=rsnecessidade.getString(2);
            int intAuxSet= new Integer(AuxSet).intValue();
            NecSet=NecSet+intAuxSet;
            if (rsnecessidade.next())
              dataset=rsnecessidade.getDate(1);
            else
              break;

          }// while Setembro
```

```
        Date outinicio = new Date(101,9,0);
        Date outfim = new Date(101,9,32);
        Date dataout = dataset;//falha no while anterior leva à tentativa de
teste com outro mês
        int NecOut=0;

        while(dataout.after(outinicio) && dataout.before (outfim)){

           String AuxOut=rsnecessidade.getString(2);
           int intAuxOut= new Integer(AuxOut).intValue();
           NecOut=NecOut+intAuxOut;
           if (rsnecessidade.next())
              dataout = rsnecessidade.getDate(1);
           else
              break;

        }// while Outubro

        Date novinicio = new Date(101,10,0);
        Date novfim = new Date(101,10,31);
        Date datanov = dataout;//inicialização que satisfaz o while em seguida
        int NecNov=0;

        while(datanov.after(novinicio) && datanov.before (novfim)){

           String AuxNov=rsnecessidade.getString(2);
           int intAuxNov= new Integer(AuxNov).intValue();
           NecNov=NecNov+intAuxNov;
           if (rsnecessidade.next())
              datanov = rsnecessidade.getDate(1);
           else
              break;

        }// while Novembro

        Date dezinicio = new Date(101,11,0);
        Date dezfim = new Date(101,11,32);
        Date datadez = datanov;//inicialização que satisfaz o while em seguida
        int NecDez=0;

        while(datadez.after(dezinicio) && datadez.before (dezfim)){

           String AuxDez=rsnecessidade.getString(2);
           int intAuxDez= new Integer(AuxDez).intValue();
```

```
            NecDez=NecDez+intAuxDez;
             if (rsnecessidade.next())
               datadez = rsnecessidade.getDate(1);
            else
               break;

         }//while Dezembro

         Date janinicio = new Date(102,0,0);
         Date janfim = new Date(102,0,32);
         Date datajan = datadez;//inicialização que satisfaz o while em seguida
         int NecJan=0;

         while(datajan.after(janinicio) && datajan.before (janfim)){

            String AuxJan=rsnecessidade.getString(2);
            int intAuxJan= new Integer(AuxJan).intValue();
            NecJan=NecJan+intAuxJan;
            if (rsnecessidade.next())
               datajan = rsnecessidade.getDate(1);
            else
               break;

         }//while Janeiro2002

         out.println("<td width=20 height=25
class=CamposPagina>"+NecSet+"</td>");
         out.println("<td width=20 height=25
class=CamposPagina>"+NecOut+"</td>");
         out.println("<td width=20 height=25
class=CamposPagina>"+NecNov+"</td>");
         out.println("<td width=20 height=25
class=CamposPagina>"+NecDez+"</td>");
         out.println("<td width=20 height=25
class=CamposPagina>"+NecJan+"</td>");
         out.println("</tr>");
         out.println("</table>");

       }//else if
      }//if rsname.next()

      else{
         out.println("<table cellpadding=0 cellspacing=0 border=5 width=650
bgcolor=black align=center>");
```

```
            out.println("<p align=center class=TituloPagina> Manifestação de
Erro</p>");
            out.println("<tr><td align=center colspan=5 class=JabaTituloLink
height=25>Item não encontrado! Por favor confira sua existência ou digite
novamente com cuidado.</td></tr>");
            out.println("</table>");
          }// caso de não existencia do item digitado


           out.println("<a href=javascript:history.back()><img
src=../../../../examples/images/return.gif border=0>Voltar</a>");
           out.println("</body>");
           out.println("</html>");

        } catch (Exception e) {
           out.println("<HTML>");
           out.println("<TITLE>PMC423-List error</TITLE>");
           out.println("<BODY>");
           out.println(e.toString());
           out.println("</BODY>");
           out.println("</HTML>");
        }
                out.close();
          } // doPost

} //ListNecessidadeServlet
```

## 10.2.7 MRPSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;



public class MrpServlet extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
```

```
  static final String PWD = "";
      public void init(ServletConfig config) throws ServletException
      {
            super.init(config);

            // open database connection
            try {
              Class.forName(ODBC_DRIVER).newInstance();
              m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
            } catch (Exception e) {
               e.printStackTrace();
               throw new ServletException("error in ODBC");
            }
      } // init

      public java.sql.Date CalculaData(java.sql.Date datapai, int ltime) {
       // rotina que calcula o período onde deve ser inserida a demanda do
item filho

       String sdatapai;
       sdatapai = datapai.toString(); // atribui a data do pai a um string

       // Decomposição do String em ano, mes e dia
       String year = sdatapai.substring(0,4);
       String month = sdatapai.substring(5,7);
       String day = sdatapai.substring(8);

       // Conversão de String para inteiro
       int intyear=Integer.parseInt(year);
       intyear=intyear-1900;
       int intmonth=Integer.parseInt(month);
       intmonth--;
       int intday=Integer.parseInt(day);

       //Criação de data no formato java.util.Date
       java.util.Date udatapai = new java.util.Date(intyear,intmonth,intday);
       Calendar cdatapai = Calendar.getInstance();
       cdatapai.setTime(udatapai);//Conversão de util.Date p/ util.Calendar
       Calendar cdatafilho= cdatapai;
       cdatafilho.add(Calendar.DATE,   -ltime);//calcula   o   período   da
necessidade

       //Conversão de Calendar p/ java.sql.Date
       java.util.Date udatafilho=cdatafilho.getTime();
```

```
        int dia = udatafilho.getDate();
        int mes = udatafilho.getMonth();
        int ano = udatafilho.getYear();
        java.sql.Date datafilho = new java.sql.Date(ano, mes, dia);

        return datafilho;
      }//CalculaData

      public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
              doPost(req, resp);
        } // doGet

        public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
         resp.setContentType("text/html");
         PrintWriter out = new PrintWriter(resp.getOutputStream());
         try {
            PreparedStatement                                          stmt=
m_dbConnection.prepareStatement("update  RegInvent  set  necliquida=0,
necess_dep=0 where necliquida>0 or necess_dep>0");

              if (stmt.executeUpdate() != 1) {
                   out.println("<HTML>");
                   out.println("<TITLE>PMC423-Update error</TITLE>");
                   out.println("<BODY>");
                   out.println("<h1>Error during update</h1>");
                   out.println("</BODY>");
                   out.println("</HTML>");
              }//if
          stmt=    m_dbConnection.prepareStatement("update   Ordens   set
Quantidade=0 where Quantidade>0");

              if (stmt.executeUpdate() != 1) {
                   out.println("<HTML>");
                   out.println("<TITLE>PMC423-Update error</TITLE>");
                   out.println("<BODY>");
                   out.println("<h1>Error during update</h1>");
                   out.println("</BODY>");
                   out.println("</HTML>");
              }//if
          int i=0;
          for(i=0; i<6; i++){
```

```
            doByLevel(req, resp, i);
          }//for

        }//try
        catch (Exception e) {
          e.printStackTrace();
          out.println("<HTML>");
          out.println("<TITLE>PMC423-Update error</TITLE>");
          out.println("<BODY>");
          out.println(e.toString());
          out.println("</BODY>");
          out.println("</HTML>");
        }
        out.close();

      }//doPost
      public void doByLevel(HttpServletRequest req, HttpServletResponse resp,
int level)
        throws ServletException, IOException
      {

        resp.setContentType("text/html");
        PrintWriter out = new PrintWriter(resp.getOutputStream());

        try {
          PreparedStatement stmt = m_dbConnection.prepareStatement("Select
RegInvent.*, cadastro.estoqueseguranca from RegInvent,cadastro where
(RegInvent.necess_indep>0 or RegInvent.necess_dep>0) and cadastro.LLC=?
and cadastro.coditem=RegInvent.cod_item");
          stmt.setInt(1,level);
          ResultSet rsinicial = stmt.executeQuery();//todos os itens do nível "level"
que possuam necessidades dep e/ou indep maiores que zero

          int excedente = 0; //variável que mede sobra de estoque frente a
demanda
          int codigovelho = 0;
          int tcritico = 0;

          while(rsinicial.next()){


            int codigo = rsinicial.getInt(1);

            out.println("codigo"+codigo);
            java.sql.Date datapai = rsinicial.getDate(2);
```

```
            int nindep = rsinicial.getInt(3);
            int ndep = rsinicial.getInt(4);
            int recprog = rsinicial.getInt(5);
            int estdisp = rsinicial.getInt(6);
            int nliqpai = rsinicial.getInt(7);
            int estseg = rsinicial.getInt(8);

/***************conta de balanço de inventário**********************/
            if(codigovelho==codigo){
            recprog= recprog + excedente;// joga a sobra do excedente no
próximo período
            excedente=0;
            }//if
            nliqpai=nliqpai + nindep + ndep - recprog - estdisp + estseg;//conta de
balanço de inventário
/*********************************************************************/

            if(nliqpai >0){
              if(9999!=codigo){
                stmt = m_dbConnection.prepareStatement("Select codfilho,
quantidade, lead_time, TCritico from estrutura where codpai = ? ");
                stmt.setInt(1, codigo);
                ResultSet rsestrutura = stmt.executeQuery();

                if(rsestrutura.next()){// Teste de existência de filhos
                   do{
                      int filho = rsestrutura.getInt(1);
                      int qtd = rsestrutura.getInt(2);
                      int ltime = rsestrutura.getInt(3);
                      tcritico = rsestrutura.getInt(4);
                      int ndepfilho = 0;

                      if (9999!=filho){

                      ndepfilho=nliqpai*qtd;//atribui ao filho a necessidade do pai
                      java.sql.Date datafilho = CalculaData(datapai, ltime);

                      stmt = m_dbConnection.prepareStatement("Select
necess_dep from RegInvent where cod_item=? and data = ? ");
                      stmt.setInt(1, filho);// sonda a existência anterior de
necessidades no período calculado
                      stmt.setDate(2, datafilho);
                      ResultSet rsnecfilho = stmt.executeQuery();
```

```
                if (rsnecfilho.next()){//teste existência de necessidade
dependente p/ o período
                    int ndepfilhovelha=rsnecfilho.getInt(1);
                    out.println("Alo entrou nop esqime");
                    if(ndepfilhovelha != 0){
                       ndepfilho=ndepfilho + ndepfilhovelha;
                    }//if
                    stmt=          m_dbConnection.prepareStatement("update
RegInvent set necess_dep=? where cod_item=? and data=?");
                    stmt.setInt(1, ndepfilho);
                    stmt.setInt(2, filho);
                    stmt.setDate(3, datafilho);
                    if (stmt.executeUpdate() != 1) {
                       out.println("<HTML>");
                       out.println("<TITLE>PMC423-Update error</TITLE>");
                       out.println("<BODY>");
                       out.println("<h1>Error during update</h1>");
                       out.println("</BODY>");
                       out.println("</HTML>");
                    }//if
                 }//if
                 else{

                    stmt=    m_dbConnection.prepareStatement("insert    into
RegInvent (cod_item, data, necess_dep) values (?,?,?)");
                    stmt.setInt(1, filho);
                    stmt.setDate(2, datafilho);
                    stmt.setInt(3, ndepfilho);
                    stmt.executeUpdate();
                 }

               }//if filho!=9999

             }while(rsestrutura.next());

           }//if teste de existencia de filhos


           //proximo passo: atualiza necessidade liquida do item que pode ter
filhos ou não

           stmt= m_dbConnection.prepareStatement("update RegInvent set
necliquida=?, recebntoprog=? where cod_item=? and data=?");
           stmt.setInt(1, nliqpai);
           stmt.setInt(2, recprog);
```

```
            stmt.setInt(3, codigo);
            stmt.setDate(4, datapai);

            if (stmt.executeUpdate() != 1) {
               out.println("<HTML>");
               out.println("<TITLE>PMC423-Update error</TITLE>");
               out.println("<BODY>");
               out.println("<h1>Error during update</h1>");
               out.println("</BODY>");
               out.println("</HTML>");
             }//if
             else{
                out.println("<HTML>");
                out.println("<TITLE>PMC423-Update error</TITLE>");
                                                              rel=StyleSheet
             out.println("<link
href=../../../../examples/estilo.css>");
             out.println("<BODY>");
             out.println("<p  align=center  class=TituloPagina>Resultado  da
Operação:</p>");
             out.println("<p      align=center      class=SubTituloPagina>MRP
Regenerativo concluído.</p>");
             out.println("</BODY>");
             out.println("</HTML>");
             }//else

             //ultimo passo: libera ordem de obtenção do item, inserindo data,
quantidade e o codigo na tabela Ordens
             java.sql.Date dataOrdem = CalculaData(datapai, tcritico);

             stmt  =  m_dbConnection.prepareStatement("Select  Quantidade
from Ordens where CodigoItem=? and Data = ? ");
             stmt.setInt(1, codigo);// sonda a existência anterior de ordem no
período calculado
             stmt.setDate(2, dataOrdem);
             ResultSet rsOrdem = stmt.executeQuery();

             if(rsOrdem.next()){

                int qtdeOrdem = rsOrdem.getInt(1);
                qtdeOrdem = qtdeOrdem + nliqpai;

                stmt=  m_dbConnection.prepareStatement("update  Ordens  set
 Quantidade=? where CodigoItem=? and Data=?");
                stmt.setInt(1, qtdeOrdem);
                stmt.setInt(2, codigo);
```

```
                stmt.setDate(3, dataOrdem);
                stmt.executeUpdate();
            }//if

            else{
                stmt= m_dbConnection.prepareStatement("insert into Ordens
(CodigoItem, Data, Quantidade) values (?,?,?)");
                stmt.setInt(1, codigo);
                stmt.setDate(2, dataOrdem);
                stmt.setInt(3, nliqpai);
                stmt.executeUpdate();
            }

          }//if 9999!=codigo
        }//if nliq>0
        else{

            stmt= m_dbConnection.prepareStatement("update RegInvent set
recebntoprog=? where cod_item=? and data=?");
            stmt.setInt(1, recprog);
            stmt.setInt(2, codigo);
            stmt.setDate(3, datapai);
            if (stmt.executeUpdate() != 1) {
                out.println("<HTML>");
                out.println("<TITLE>PMC423-Update error</TITLE>");
                out.println("<BODY>");
                out.println("<h1>Error during update</h1>");
                out.println("</BODY>");
                out.println("</HTML>");
            }//if

            excedente = (-1)*nliqpai;
            codigovelho=codigo;

        }//else nliq>0

    }//while

    out.println("<HTML>");
    out.println("<TITLE>PMC423-Update error</TITLE>");
    out.println("<link rel=StyleSheet href=../../../../examples/estilo.css>");
    out.println("<BODY>");
    out.println("<p align=center class=TituloPagina>Resultado da
Operação:</p>");
```

```
        out.println("<p align=center class=SubTituloPagina>MRP Regenerativo
concluído.</p>");
        out.println("</BODY>");
        out.println("</HTML>");

      }//try


      catch (Exception e) {
        e.printStackTrace();
        out.println("<HTML>");
        out.println("<TITLE>PMC423-Update error</TITLE>");
        out.println("<BODY>");
        out.println(e.toString());
        out.println("</BODY>");
        out.println("</HTML>");
      }
            out.close();

    } // doByLevel

} //MrpServlet
```

## 10.2.8 LISTORDENSSERVLET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.Calendar;


public class ListOrdensServlet extends HttpServlet
{
   Connection m_dbConnection = null;
   static final String ODBC_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
   static final String DSN = "jdbc:odbc:PMC423";
   static final String USER = "";
   static final String PWD = "";

   /** Initializes the servlet.
   */
    public void init(ServletConfig config) throws ServletException
        {
```

```
            super.init(config);

            // open database connection
            try {
              Class.forName(ODBC_DRIVER).newInstance();
              m_dbConnection = DriverManager.getConnection(DSN, USER,
PWD);
            } catch (Exception e) {
               e.printStackTrace();
               throw new ServletException("error in ODBC");
            }
      } // init


   /** Handles the HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   */
   public void doGet(HttpServletRequest req, HttpServletResponse resp)
         throws ServletException, IOException
         {
               doPost(req, resp);
         } // doGet

   /** Handles the HTTP <code>POST</code> method.
   * @param request servlet request
   * @param response servlet response
   */
   public void doPost(HttpServletRequest req, HttpServletResponse resp)
         throws ServletException, IOException
         {
               resp.setContentType("text/html");
               PrintWriter out = new PrintWriter(resp.getOutputStream());



      try {

        int dataAtual=0;
        int ultimo=0;
        int i=0;
        Date MesMaior;
        Date MesMenor;
```

```
        PreparedStatement stmt = m_dbConnection.prepareStatement("Select *
from Ordens Order by Data");
        ResultSet rsOrdens = stmt.executeQuery();


        out.println("<html>");
        out.println("<title>PMC423 - Listar Ordens</title>");
        out.println("<link rel=StyleSheet href=../../../../examples/estilo.css>");
        out.println("<body>");

        if (rsOrdens.next()){

            out.println("<p align=center class=TituloPagina>Lista de Ordens de
obtenção de Produtos</p>");

            out.println("<table cellpadding=0 cellspacing=0 border=5 width=350
bgcolor=black align=center>");
            out.println("<tr><td align=center class=BarraTituloLink>Código</td><td
align=center class=BarraTituloLink>Data</td><td align=center
class=BarraTituloLink>Quantidade</td></tr>");
            out.println("<tr></tr>");
            do{
            out.println("<tr>");
            out.println("<td width=100 height=25
class=CamposPagina>"+rsOrdens.getInt(1)+"</td>");
            out.println("<td width=100 height=25
class=CamposPagina>"+rsOrdens.getDate(2)+"</td>");
            out.println("<td width=100 height=25
class=CamposPagina>"+rsOrdens.getInt(3)+"</td>");
            out.println("</tr>");

            }while(rsOrdens.next());


            out.println("</table>");


        }//if rsOrdens.next()

        else{
            out.println("<p align=center class=TituloPagina> Manifestação de
Erro</p>");
            out.println("<table cellpadding=0 cellspacing=0 border=5 width=350
bgcolor=black align=center>");
```

```
            out.println("<tr><td align=center colspan=5 class=JabaTituloLink
height=25>Não há Ordens em Banco de Dados.</td></tr>");
            out.println("</table>");
          }// caso de não existencia do item digitado


          out.println("<a href=javascript:history.back()><img
src=../../../../examples/images/return.gif border=0>Voltar</a>");
          out.println("</body>");
          out.println("</html>");

        } catch (Exception e) {
            out.println("<HTML>");
            out.println("<TITLE>PMC423-List error</TITLE>");
            out.println("<BODY>");
            out.println(e.toString());
            out.println("</BODY>");
            out.println("</HTML>");
        }
                out.close();
          } // doPost

} //ListOrdensServlet
```