UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB

CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT

DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO - DSC

RODRIGO FIGUEIREDO DE ALBUQUERQUE

"DESENVOLVIMENTO DE FERRAMENTA DE INTEGRAÇÃO DO PANIFICADOR"

"REFATORAMENTO DO PANIFICADOR"

Relatório final para a conclusão da disciplina Estágio Supervisionado

Local: ByteCom Sistemas Ltda.
Orientadora: Francilene Procópio Garcia
Supervisor: Robert Kalley C. Menezes

Campina Grande, Maio/2001

Biblioteca Setorial do CDSA. Maio de 2021.

Sumé - PB

Apresentação

O presente relatório tem por objetivo descrever de forma detalhada o trabalho realizado por Rodrigo Figueiredo de Albuquerque, aluno concluínte do curso de Ciência da Computação, durante seu Estágio Supervisionado. O referido estágio fora realizado na empresa ByteCom Sistemas durante o período de dezembro de 2000 a março de 2001 e a supervisão técnica foi feita pelo Sr. Roberto Kalley Cavalcanti de Menezes. A orientação acadêmica ficou sob a responsabilidade da Sra. Francilene Procópio Garcia, professora do Departamento de Sistemas e Computação - DSC da Universidade Federal da Paraíba - Campus II.

O corpo do presente relatório fora dividido em duas partes devido ao fato de terem sido realizadas, durante o decorrer do estágio, duas tarefas de naturezas bastante distintas cujos objetivos eram diferentes. Porém, ambas se encaixam perfeitamente no contexto da Ciência da Computação. Cada uma destas partes do relatório foi subdividida da seguinte forma:

- Introdução: trata basicamente dos problemas encontrados e dos objetivos a serem alcançados;
- Desenvolvimento: trata da metodologia adotada e das atividades desenvolvidas;
- Conclusão: trata dos resultados alcançados e os compara com os objetivos propostos;

O ambiente de estágio, que é comum para as duas atividades desenvolvidas, será descrito uma única vez no início do relatório.

ÍNDICE ANALÍTICO

Ambiente de Estágio

- ***Dados da Empresa***

Razão Social: ByteCom Sistemas Ltda.

CNPJ: 03.688.196/0001-47

Endereço: Av. Aprígio Veloso,882 - Bodocongó, Campina Grande, Paraíba

Tele/Fax: (83)310-1438

E-mail: bytecom@bytecom.com.br

URL: http://www.bytecom.com.br

- ***Nicho de mercado***

A ByteCom Sistemas é uma empresa incubada no CENTRO SOFTEX GENESIS de Campina Grande - POLIGENE que atua principalmente na informatização de indústrias/empresas do setor de Alimentação. Os principais clientes da ByteCom Sistemas são panificadoras, confeitarias, restaurantes, fast foods e pizzarias. Além desses, a empresa desenvolve soluções customizadas instituições publicas e privadas de outros setores de atividade.

- ***Organograma da Empresa***

ByteCom Sistemas - Organograma

- ***Estado da Informática na Empresa***

A empresa atualmente utiliza a estrutura do laboratório de prototipagem de software do centro SOFTEX GENESIS de Campina Grande – POLIGENE. No total são 08 (oito) computadores Compaq com processadores Pentium II Celeron 466 MHz, com 64 MB de memória RAM rodando sobre o sistema operacional Windows 98® e um Servidor Netfinity 3500 da IBM rodando o sistemas operacional Windows NT® Server. O laboratório dispõe de uma linha telefônica e fax para auxiliar os negócios da empresa. Além disso, o mesmo está disponível 24 horas por dia incluindo sábados, domingos e feriados. Todos os computadores possuem acesso à Internet 24 horas por dia.

A empresa adotou o Borland Delphi 5.0 na versão Enterprise para desenvolver suas aplicações convencionais e utiliza a linguagem Java® para desenvolver aplicações web. Os pacotes de software instalados nas máquinas são em sua grande maioria registrados e o restante são cópias grátis de avaliação adquiridas na Internet.

A equipe de desenvolvimento é formada basicamente por alunos da graduação do curso de Ciências da Computação e do curso de Desenho Industrial da UFPB, Campus II. O supervisor técnico é **Robert Kalley Menezes**, coordenador do centro SOFTEX GENESIS de Campina Grande – POLIGENE e professor do Departamento de Sistemas e Computação – DSC.

- ***Aspectos Positivos***

Apesar de não ter , ainda, em seu quadro de pessoal nenhum profissional formado na área de informática, a empresa, pelo fato de ser incubada, usufrui de toda a qualificação, dedicação e experiência do quadro de professores do Departamento de Sistemas e Computação, que sempre se mostrou receptivo quando a equipe de desenvolvimento não conseguia resolver problemas de natureza técnica por falta de experiência. Assim, essa "deficiência" sempre pôde ser rapidamente resolvida.

Um outro aspecto positivo foi o fato de ser um dos sócios da empresa em questão. Já que a disciplina Estágio Supervisionado exige que o aluno desenvolva um projeto em uma empresa para que o mesmo possa concluir a graduação, achamos por bem que este estágio fosse realizado na nossa empresa, pois à medida que fosse sendo desenvolvido,

os resultados seriam assimilados imediatamente pela mesma. Como estaríamos trabalhando em algo nosso, a motivação seria muito maior do que se estivéssemos em um outro ambiente, onde provavelmente não estaríamos sendo remunerados e a única motivação seria concluir a graduação, o que poderia afetar a qualidade do trabalho .

- ***Aspectos Negativos***

Apesar da moderna estrutura disponibilizada e facilidade de acesso à mesma, o ambiente utilizado pela ByteCom Sistemas é compartilhado com outras empresas incubadas, o que reduz os recursos disponíveis por empresa e muitas vezes limita as ações de cada empresa no sentido de desenvolver mais rapidamente seus negócios.

Ferramenta de Integração do Panificador

1 Introdução

1.1 Problema Encontrado

A ByteCom Sistemas Ltda. desenvolveu um sistema especialmente projetado para prover um gerenciamento completo e eficaz para casas de panificação e confeitaria. Este software, chamado Panificador, está em sua primeira versão e atualmente vem sendo comercializado pela empresa no mercado local. O software conta com todas as ferramentas necessárias para o completo gerenciamento de panificadoras. Porém, não é capaz de gerenciar uma rede de lojas, até mesmo porque não foi projetado para tal atividade.

Através de conversas com clientes em potencial, membros da ByteCom identificaram a necessidade de desenvolver uma ferramenta que fosse capaz de gerenciar, quando em funcionamento conjunto com o Panificador, uma rede de panificadoras. A idéia foi desenvolver tal ferramenta de forma a permitir que a empresa não precisasse efetuar alterações no sistema que já está pronto. Analisar, projetar, implementar e testar a primeira versão desta ferramenta são objetivos desta parte do meu estágio supervisionado. Trata-se de uma ferramenta que tem como objetivo fundamental o de centralizar, em uma única loja, informações de todas as lojas da rede. O tratamento destas informações deverá gerar relatórios capazes de demonstrar a realidade de toda a rede de lojas. Nesta primeira versão serão centralizadas apenas informações relativas à movimentações de caixa e de estoque da rede de panificadoras.

1.2 Revisão Bibliográfica

1.2.1 Da Modelagem de Sistemas

Um dos maiores problemas encontrados pelos profissionais de informática é o da ineficácia na comunicação com seus clientes. Grande parte das deficiências encontradas nas especificações de sistemas se deve ao problema da comunicação entre analistas de sistemas e usuários. Por este motivo, um esforço considerável foi e continua sendo realizado no sentido de propor metodologias que utilizem linguagens que possam atender as

necessidades de modelagem dos analistas mas que também possam ser entendidas pelo cliente/usuário do sistema. Em outras palavras, as linguagens utilizadas por estas metodologias devem manter-se suficientemente precisas para permitir que o analista produza uma especificação que seja a base para o sistema de informação de que o cliente necessita e, por outro lado, também devem ser mais inteligíveis ao usuário não-técnico. Nesta linha, foram desenvolvidas linguagens que passaram a incluir os recursos necessários para definir as necessidades do usuário, independentemente de qualquer restrição relativa à implementação do sistema. O principal objetivo destas linguagens é o de possuir recursos capazes de permitir a produção de uma especificação que possa ser apresentada ao usuário e com ele discutida. A boa técnica de análise dos sistemas de qualquer empresa preconiza uma boa interação - vale dizer, comunicação - entre os usuários e os técnicos de informática.

O principal produto da análise são modelos que darão origem aos sistemas. Estes modelos, gerados pela análise do sistema, variam de acordo com a metodologia utilizada. Porém, independente da metodologia adotada, os modelos gerados possuem como principais as seguintes utilidades:

- Estabelecer uma visão comum, entre usuário e analista, do ambiente antes da automação;
- Servir como suporte para negociação e especificação de requisitos e possibilidades futuras para o sistema;
- Escalonar a informatização em fases, com produtos bem definidos e dependência mínima entre as fases;
- Tratar a complexidade do problema por níveis de abstração, começando pela visão mais abstrata e descendo a visões progressivamente mais detalhadas;
- Promover indicações quantitativas do escopo e da complexidade do problema;
- Prover facilidades para a geração de testes de aceitação;

Além disso, as mais modernas abordagens enfatizam a perspectiva dos dados, não ignorando, entretanto, a importância da perspectiva das funções bem como a dos controles do sistema. Cada uma destas perspectivas é suportada por uma maneira de expressar a

realidade registrada por meio de um modelo que a descreve, segundo uma forma de representação.

1.2.2 Da Análise Essencial

A metodologia adotada para o desenvolvimento do sistema objeto do estágio supervisionado foi a Análise Essencial. A grande vantagem desta metodologia sobre as análises tradicional e estruturada reside no fato de a mesma ter como proposta fundamental o uso de eventos como base para o particionamento do sistema. Esta abordagem tem mostrado ao longo do tempo ser mais efetiva do que a abordagem *"top-down"*, por tornar mais fácil a identificação das funções e entidades que compõem o sistema. Assim, a verdadeira essência do sistema pode ser mais facilmente modelada pelo analista de sistemas. Outra grande vantagem desta metodologia é o suporte dados à construção dos modelos de dados e de funções concomitantemente, o que certamente garante uma maior correspondência entre os dois modelos.

Em relação ao grau de abstração, a Análise Essencial considera dois níveis: o nível essencial e o nível de implementação. O nível essencial é representado pelo modelo essencial e o nível de implementação pelo modelo de implementação.

O **modelo essencial** apresenta o sistema num grau de abstração completamente independente de restrições tecnológicas. Antes que um sistema seja implementado, é necessário conhecer-se a sua verdadeira essência, não importando saber se a sua implementação vai ser manual ou automatizada, e nem mesmo que tipo de "hardware" ou "software" vai ser utilizado quando da implementação do mesmo. Assim, na confecção do modelo essencial devemos considerar a existência de uma tecnologia perfeita. Esta expressão foi definida por S. McMenmin e J. Palmer e deve ser entendida como uma abstração onde se supõe uma tecnologia ideal onde:

- Não há custos com equipamentos;
- A velocidade de armazenamento de dados do sistema é infinita;
- A velocidade de processamento dos dados é infinita;
- O tempo de acesso aos dados é instantâneo;

- Não ocorrem falhas de qualquer espécie;

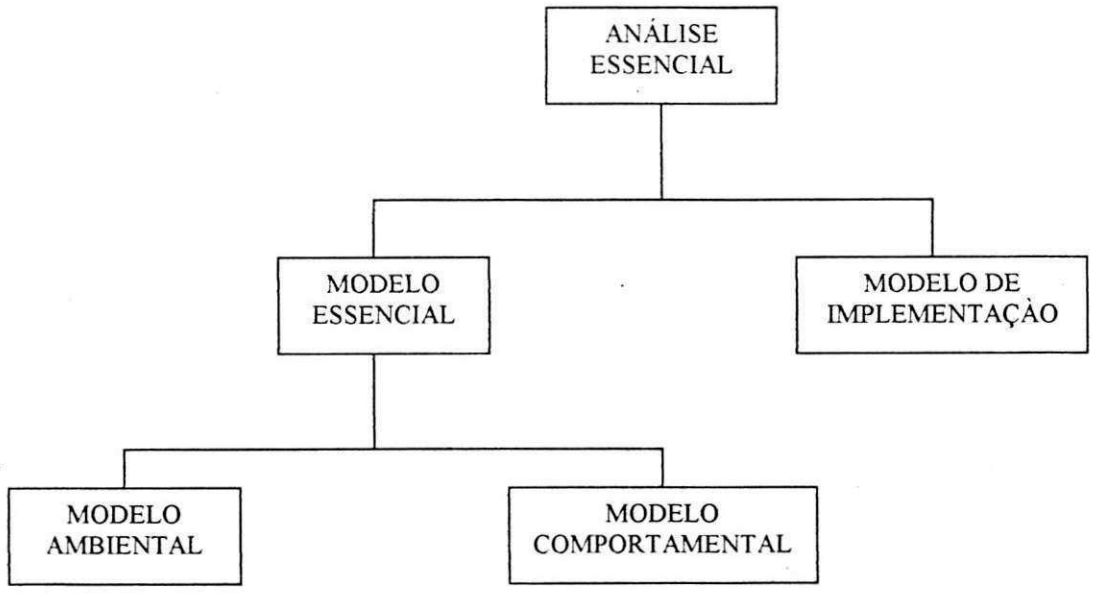
O **modelo de implementação** apresenta o sistema num grau de abstração completamente dependente de restrições tecnológicas. É derivado do modelo essencial e diz respeito à implementação do sistema. Neste modelo são levadas em consideração todas as características tecnológicas, importando saber se parte de sua implementação vai ser manual ou automatizada, que tipo de "hardware" ou "software" vai ser usado etc.

O modelo essencial se subdivide em dois outro modelos: o modelo ambiental e o modelo comportamental.

O **modelo ambiental** é voltado para fora do sistema, para o ambiente em que está inserido. Trata da representação da interface do sistema com o mundo exterior e de interação do sistema com os elementos externos a ele.

O **modelo comportamental** é voltado para o interior do sistema, para o comportamento de suas partes internas. Mostra basicamente como ele deve reagir a estímulos do ambiente exterior.

A figura que segue mostra a decomposição da modelagem proposta pela Análise Essencial, em seus respectivos modelos:



1.2.2.1 Modelo Ambiental

A definição do ponto de vista externo apresenta um componente importante denominado "ambiente". Portanto é de fundamental importância, na especificação de um sistema, definir quais os elementos do ambiente em que o sistema se acha inserido, a que necessidades ele deve atender, independente da maneira que será implementado. A esse conjunto de elementos chama-se modelo ambiental.

O modelo ambiental define as fronteiras do sistema, sua interface com o mundo exterior, o que é parte integrante do sistema e o que não é, com quem ele interage, quais suas entradas e de onde elas vêm, quais as suas saídas e para onde vão. Quais são as finalidades a que o sistema deve atender e a que estímulos ele deve reagir.

Os componentes do modelo ambiental são:

- Lista de Eventos que afetam o sistema;
- Diagrama de Contexto;
- Declaração dos objetivos do sistema;

- **Lista de Eventos que afetam o sistema:** o primeiro passo na especificação de um sistema é identificar a quais eventos do mundo exterior ele deverá responder. Isto, por si só, já ajuda a delimitar as fronteiras do problema de que estamos tratando. Pode-se dizer também que a lista de eventos do sistema está intrinsecamente ligada às finalidades do sistema, uma vez que as finalidades do sistema são atender a determinadas necessidades e estas necessidades, por sua vez, são decorrentes de eventos que acontecem no mundo exterior;

- **Diagrama de Contexto:** representa o sistema por um único processo e suas interligações com as entidades externas, mostrando apenas as interfaces do sistema com o ambiente em que ele está inserido;

- **Declaração dos Objetivos do sistema:** é uma definição externa relativa ao ambiente do sistema, onde se procura responder questões como: finalidade do

sistema, problemas a serem resolvidos com a implantação do sistema, requisitos que devem ser atendidos. A declaração deve ser elaborada em poucas frases e com uma linguagem simples e precisa de modo a ser entendida pelos usuários do sistema e pela administração da empresa;

1.2.2.2 Modelo Comportamental

O modelo comportamental é definido do ponto de vista interno. É o modelo do interior do sistema. Descreve de que maneira o sistema, enquanto um conjunto de elementos inter-relacionados, reage internamente aos estímulos do exterior. Este modelo tem como finalidade, mostrar as ações que o sistema deve executar para responder adequadamente aos eventos previstos no modelo ambiental.

O modelo comportamental tem como objetivo decompor o sistema segundo dois ponto de vistas diferentes que são: funções e dados. Ele procura descrever as funções do sistema e seus arquivos ou depósitos de dados. Para isso, deve-se antes saber: o que é produzido pelo sistema e a que estímulos o sistema deve responder. Na verdade, dados armazenados e funções são meios para atingir-se o verdadeiro objetivo do sistema, que é apresentar as respostas adequadas ao ambiente em que está contido. Portanto, a decomposição deve ser feita a partir da necessidade de resposta aos eventos.

Os componentes do modelo comportamental são:

- Diagrama de Entidade-Relacionamento - DER;
- Diagrama de Fluxo de Dados - DFD;
- Miniespecificações;
- Dicionário de Dados;

- **Diagrama de Entidade-Relacionamento -DER:** representação gráfica do modelo conceitual de dados. Este diagrama é formado pelos quatro componentes primitivos do modelo, que representam o mundo real, que são: entidades, relacionamentos, atributos e domínios;

- **Diagrama de Fluxo de Dados -DFD:** representação gráfica do modelo funcional. Mostra a interdependência das funções que compõem o sistema, apresentando fluxo de dados entre elas. Mostra também arquivos lógicos ou depósitos de dados e as entidades externas que tanto podem ser origem como destino dos fluxos de dados;

- **Miniespecificações:** descrição das funções primitivas do modelo funcional. As principais técnicas de especificações são: português estruturado, pseudocódigo, tabela de decisão e árvore de decisão;

- **Dicionário de Dados:** repositório de informações sobre os componentes do sistema;

2 Desenvolvimento

2.1 Estratégia de Análise

Por se tratar de um sistema com a funcionalidade básica de gerar relatórios a partir dos dados das lojas de uma rede de panificadoras, não houve a necessidade de dividir o sistema em subsistemas para facilitar a análise.

A estratégia adotada para a análise do sistema foi a definição de duas fases bem distintas: fase de levantamento de requisitos e fase de análise do sistema. Cada uma dessas fases é formada por um conjunto de procedimentos que quando seguidos levam a obtenção de produtos específicos de cada uma delas. Os procedimentos e produtos de cada fase estão de acordo com a metodologia adotada, Análise Essencial.

A seguir estão descritos as duas fases com seus procedimentos e produtos.

2.1.1 Levantamento de Requisitos

Procedimentos:

Entrevistas

Diagramação Preliminar

Produtos:

Anotações

Esboços

Lista de requisitos

2.1.2 Análise do Sistema

2.1.2.1 Modelo Ambiental

Procedimentos:

DER preliminar

Lista de Eventos

Produtos:

DER, preliminar

Declaração de Objetivos

Lista de Eventos

2.1.2.2 Modelo Comportamental

Procedimentos:

Diagrama de Entidade-Relacionamento - DER

Dicionário de Fluxo de Dados com Miniespecificações

Dicionarização do DER com definição de restrições

Produtos:

Diagrama de Entidade-Relacionamento completo

Dicionário de Fluxo de Dados completo

Miniespecificações de processos

Dicionário de Dados

2.2 Atividades Desenvolvidas

A fase de levantamento de requisitos tem como objetivo identificar os problemas e as deficiências mais comuns dentro de um sistema de gerenciamento de redes de panificadoras e confeitarias e que não podem ser resolvidos atualmente com o uso do Panificador V 1.0, sistema desenvolvido pela ByteCom Sistemas Ltda. e em fase de comercialização. Esta primeira versão do sistema de integração tem por objetivo focar basicamente as operações de movimentações de caixa e estoque.

Já a fase de análise tem o objetivo de especificar o modelo ambiental e o modelo comportamental para o mesmo sistema.

A seguir serão descritas as atividades que foram realizadas dentro de cada fase e os produtos gerados por tais atividades.

2.2.1 Fase de Levantamento de Requisitos

Esta fase do trabalho de estágio consistiu em analisar o domínio do problema com o intuito de identificar deficiências , conhecer as atividades envolvidas e as rotinas de trabalho. Durante esta fase foram feitas diversas entrevistas com gerentes e funcionários de panificadoras. As entrevistas seguiam um roteiro elaborado previamente, observando o cargo que o entrevistado ocupava.

Os produtos gerados durante esta fase foram versões preliminares do diagrama de entidade-relacionamento, anotações, esboços, além de um relato dos principais problemas e deficiências do sistema.

Devido à natureza do sistema de integração, todos os problemas encontrados na fase de levantamento de requisitos estão diretamente ligados com a falta de informações conjuntas que expressem a realidade da rede de panificadoras e não apenas de uma única loja da rede. Os problemas encontrados foram os seguintes:

- Não existe um controle eficaz de caixa da rede de lojas. As informações são obtidas para lojas da rede, mas não para a rede de lojas como um todo. A análise dos relatórios gerados em cada uma das lojas para obter informações da rede é muito demorada e por isso se torna inviável. As informações relativas ao caixa das quais se necessita são, por exemplo:
  - Número de atendimentos na rede de lojas por período de tempo;
  - Média de atendimentos por loja por período de tempo;
  - Faturamento da rede de lojas por período de tempo;
  - Média de faturamento por loja por período de tempo;
  - Faturamento por forma de pagamento (dinheiro, cheque, cartão);
  - Total de retiradas para pagamentos (sangrias);

- Não existe um controle eficaz de movimentações de estoque da rede de lojas. As informações são obtidas para lojas da rede, mas não para a rede de lojas como um todo. A análise dos relatórios gerados em cada uma das lojas para obter informações da rede é muito demorada e por isso se torna inviável. As

informações relativas às movimentações de estoque das quais se necessita são, por exemplo:

- Listagem de produtos vendidos na rede de lojas por período de tempo;
- Listagem de produtos desperdiçados na rede de lojas;
- Listagem de produtos retornados ao estoque por sobra na rede de lojas por período de tempo;
- Média de vendas de cada produto por loja da rede de lojas e por período de tempo;
- Listagem de produtos adquiridos para a rede de lojas por um período de tempo;
- Movimentação de estoque de um produto específico na rede de lojas por período específico e por tipo(s) de movimentação específico(s);

Diante do requisitos levantados percebemos que para um sistema de gerenciamento de redes de panificadoras e confeitarias ser eficaz, confiável e lucrativo, é extremamente necessário a automatização dos seus processos, pois esse sistema trabalha com um numero muito grande de informações e o trabalho manual seria dispendioso.

2.2.2 Fase de Análise do Sistema

Seguindo a estratégia de análise definida anteriormente, o passo seguinte à fase de levantamento de requisitos é a fase de análise do sistema. Ao final desta fase o sistema terá um modelo ambiental e um modelo comportamental definidos e documentados.

Esta fase serviu para solidificar os conhecimentos adquiridos do sistema que foram obtidos durante a fase de levantamento de requisitos e possibilitou a especificação precisa das atividades e detalhamento do modelo de dados. As ferramentas utilizadas nesta fase foram: Diagrama de Entidade-Relacionamento, Diagrama de Fluxo de Dados, Dicionário de Dados , Miniespecificações e Diagrama de Contexto.

2.2.2.1 Construção dos Modelos Ambientais

Como já se sabe, os componentes do modelo ambiental são: Lista de Eventos, Diagrama de Contexto e Declaração de Objetivos.

A lista de eventos foi construída baseado nas informações que foram obtidas durante a fase de levantamento de requisitos. Esta lista de eventos está descrita no anexo 1D. A declaração de objetivos, por sua vez, foi baseada na lista de eventos externos e está descrita logo a seguir:

- O sistema de integração tem como objetivo básico e fundamental prover informações que possibilitem ao empresário panificador fazer um acompanhamento eficaz da sua rede de lojas. O Panificador V. 1.0 já provê este controle para uma única loja, porém reunir estas informações para que se tenha informações reais da rede como um todo através das informações isoladas fornecidas pelo Panificador é um processo que, se feito manualmente, se torna extremamente inviável;

A elaboração do diagrama de contexto foi baseada na lista de eventos do sistema, sendo assim esse diagrama só poderia ser construído após a finalização da lista de eventos. O diagrama de contexto está descrito no anexo 1A.

2.2.2.2 Construção dos Modelos Comportamentais

A construção de um modelo comportamental envolve a elaboração de um modelo conceitual de dados que descreve a perspectiva dos dados do sistema e a elaboração de um modelo funcional que, por sua vez, descreve a perspectiva das funções. Essa duas perspectivas do sistema, segundo a Análise Essencial, são integradas através dos eventos externos listados no modelo ambiental, desta forma se torna possível obter, por construção, esses dois modelos de maneira integrada, elaborando-os simultaneamente.

Método adotado para obtenção do Modelo de Dados

A fim de facilitar a percepção de quais são os principais componentes do modelo conceitual de dados, seguimos os seguintes passos:

Passo 1: Classificar os eventos em ordem cronológica

Passo 2: Para cada evento

2.1 Identificar as entidades envolvidas
1.1 Identificar os atributos das entidades
1.2 Eleger os atributos identificadores das entidades
1.3 Identificar os relacionamentos entre as classes de entidades
1.4 Identificar o tipo de mapeamento de cada relacionamento

Os dois passos apresentados acima, foram utilizados na construção do modelo de dados de cada subsistema de modo que no final destas análises estávamos com o diagrama de entidade e relacionamento completo do sistema

A descrição do Diagrama de Entidadade-Relacionamento está presente no anexo 1B.

Método adotado para obtenção do Modelo Funcional

Para a obtenção dos componentes do modelo funcional, seguimos os seguintes passos:

Passo 1: Classificar os eventos em ordem cronológica

Passo 2: Para cada evento

2.1 Construir o DFD de resposta a cada evento

2.2 Elaborar as miniespecificações dos processos do DFD

2.3 Identificar a estrutura dos depósitos de dados, a partir das estruturas de dados dos fluxos de entrada e saída de cada função

Seguindo os dois passos acima obtivemos para o sistema, os diagramas de fluxo de dados e as miniespecificações dos seus processos.

A descrição dos diagramas de fluxo de dados e das miniespecificações do sistema está no anexo 1D.

A construção do Dicionário de Dados foi feita em paralelo com a construção dos modelos funcionais e dos modelos de dados. Sua descrição consta no anexo 1C.

3 Conclusão

O estágio supervisionado é uma atividade bastante importante e necessária para um aluno concluínte de um curso universitário. É importante porque o aluno tem a oportunidade de colocar em prática os seus conhecimentos adquiridos durante sua vida acadêmica e com isso ganhar experiência profissional. E é necessário porque o aluno tem a possibilidade de conhecer suas deficiências e suas qualidades profissionais, tentando, assim, aperfeiçoar-se profissionalmente.

Não encontrei muitas dificuldades no desenvolvimento do estágio supervisionado no que diz respeito ao conhecimento da metodologia adotada e do uso das ferramentas de desenvolvimento. Isto porque desde meados de 1998 tenho estado engajado em diversos projetos de natureza semelhante ao desenvolvido neste estágio, inclusive pelo caráter comercial do mesmo. A principal dificuldade foi encontrada principalmente na fase de levantamento de requisitos do sistema e está diretamente relacionada com a comunicação entre analista do sistema e o cliente/usuário. Por outro lado, considero esta dificuldade também como um ponto positivo. Lidar com pessoas, além de ser extremamente necessário para se identificar os verdadeiros requisitos do sistema é também uma tarefa muito gratificante pois permite a troca de conhecimentos e experiências, contribuindo, assim, para o enriquecimento pessoal e profissional.

Esta é a conclusão da primeira etapa de um trabalho que certamente terá continuidade. O sistema de integração do Panificador, produto da ByteCom Sistemas Ltda., tem sido demandado pelo mercado. Em geral, o cliente manifesta o interesse e ressalta a importância da existência de tal ferramenta sempre que o mesmo possui uma rede de lojas. Portanto, esta ferramenta de integração deverá ser brevemente expandida para atender a novos requisitos funcionais demandados pelo mercado-alvo da ByteCom. É bastante provável que eu esteja engajado, também, nesta nova etapa deste projeto.

4 Referências Bibliográficas

[Pompilho, 1994] Análise Essencial, S. Pompilho, Editora IBPI, 1994

[McMenamin e Palmer, 1984] Análise Essencial de Sistemas, Stephen M. McMenamin e John F. Palmer, São Paulo, Editora McGrawn Hill, 1991

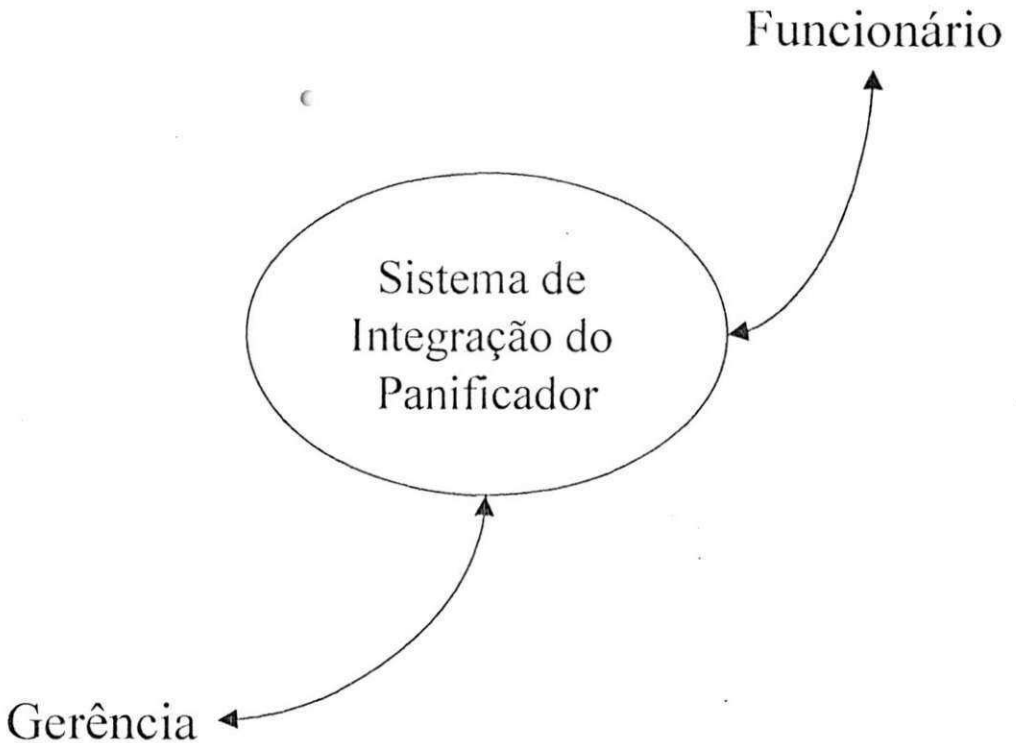
[PRESSMAN, 1995] Engenharia de Software, Roger S. Pressman - São Paulo, Editora Makron Books, 1995.

[Fairley, 1985] Software Enginnering Concepts, Richard E. Fairley, Editora McGraw-Hill, 1985

[Korth, 1995] Sistema de Banco de Dados, Henry F. Korth e Abrahm Silberschatz, Editora Makron books, 1995

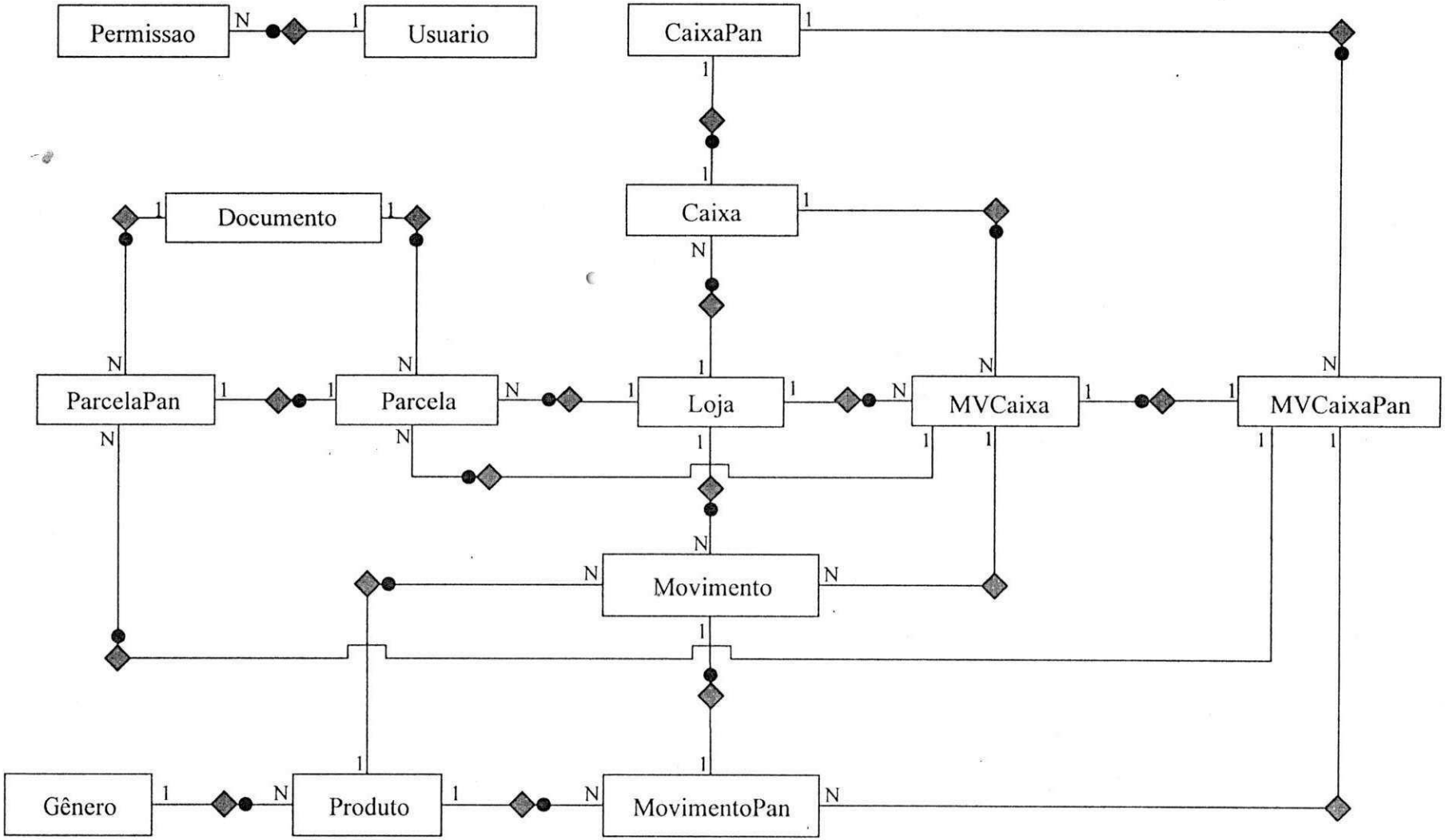
Anexo 1A

Diagrama de Contexto - Sistema de Integração do Panificador



Anexo 1B

D.E.R. - Sistema de Integração do Panificador



Anexo 1C

Depósito de Dados: MVCAIXA

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_MVCAIXA	Código da movimentação de caixa	N	10	9.999.999.999	Chave primária composta
*CODIGO_LOJA	Código da loja de origem da movimentação	N	5	99.999	Chave primária composta / Chave estrangeira
DESCONTO	Desconto dado para a movimentação	N	5,2	999,99	Não Nulo
DATA_MVCAIXA	Data da movimentação de caixa	DATA	8	dd/mm/aaaa	Não Nulo
HORA_MVCAIXA	Hora da movimentação de caixa	HORA	6	hh:mm:ss	Não Nulo
VALOR	Valor da movimentação de caixa	N	9,2	9.999.999,99	Não Nulo
*CODIGO_CAIXA	Código do caixa da movimentção	N	10	9.999.999.999	Chave estrangeira / Não Nulo

Depósito de Dados: MOVIMENTO

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_MOVIMENTO	Código da movimentação de estoque	N	10	9.999.999.999	Chave primária composta
*CODIGO_LOJA	Código da loja de origem da movimentação	N	5	99.999	Chave primária composta / Chave estrangeira
DATA_MOVIMENTO	Data da movimentação de estoque	DATA	8	dd/mm/aaaa	Não Nulo
*CODIGO_PRODUTO	Código do produto movimentado	N	5	99.999	Chave estrangeira / Não Nulo
QUANTIDADE	Quantidade movimentada do produto	N	9,2	9.999.999,99	Não Nulo
VALOR	Valor unitário do produto movimentado	N	9,2	9.999.999,99	Não Nulo
CODIGO_OPERACAO	Código da operação realizada	N	5	99.999	Não Nulo
*CODIGO_MVCAIXA	Código da movimentação de caixa	N	10	9.999.999.999	Chave estrangeira

Depósito de Dados: CAIXA

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_CAIXA	Código do caixa	N	10	9.999.999.999	Chave primária composta
*CODIGO_LOJA	Código da loja de origem do caixa	N	5	99.999	Chave primária composta / Chave estrangeira
HORA_INICIO	Hora em que o caixa foi aberto	HORA	6	hh:mm:ss	Não Nulo
HORA_TERMINO	Hora em que o caixa foi fechado	HORA	6	hh:mm:ss	
CAIXA_INICIAL	Valor com o qual caixa foi aberto	N	9,2	9.999.999,99	Não Nulo
CAIXA_FINAL	Valor com o qual caixa foi fechado	N	9,2	9.999.999,99	
TERMINAL	Nome da máquina usada para o caixa	C	15		Não Nulo
DATA_INICIO	Data em que o caixa foi aberto	DATA	8	dd/mm/aaaa	Não Nulo
DATA_TERMINO	Data em que o caixa foi fechado	DATA	8	dd/mm/aaaa	

Depósito de Dados: DOCUMENTO

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_DOCUMENTO	Código do documento	N	5	99.999	Chave primária
TIPO	Tipo do documento	C	1		Não Nulo
DOCUMENTO	Descrição do documento	C	40		Não Nulo

Depósito de Dados: PARCELA

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_PARCELA	Código da parcela	N	10	9.999.999.999	Chave primária composta
*CODIGO_LOJA	Código da loja de origem da parcela	N	5	99.999	Chave primária composta / Chave estrangeira
*CODIGO_DOCUMENTO	Código do documento usado na parcela	N	5	99.999	Chave estrangeira / Não Nulo
VALOR_PARCELA	Valor da parcela	N	9,2	9.999.999,99	Não Nulo
*CODIGO_MVCAIXA	Código da movimentação de caixa	N	10	9.999.999.999	Chave estrangeira / Não Nulo

Depósito de Dados: LOJA

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_LOJA	Código da loja da rede de lojas	N	5	99.999	Chave primária
LOJA	Nome da loja da rede de lojas	C	40		Não Nulo
CNPJ	CNPJ da loja	C	14		Não Nulo
FAX	Número do FAX da loja	C	14		
TELEFONE	Número do telefone da loja	C	14		
BAIRRO	Bairro em que está localizada a loja	C	25		
CIDADE	Cidade em que está localizada a loja	C	30		
PAIS	País em que está localizada a loja	C	20		
EMAIL	E-Mail da loja	C	30		
ESTADO	Estado em que está localizada a loja	C	20		
RUA	Rua em que está localizada a loja	C	35		
CEP	CEP do local em que está localizada a loja	C	8		
NUMERO	Número de endereço da loja	C	6		
INCRICAO_ESTADUAL	Inscrição Estadual da loja	C	20		

Depósito de Dados: GENERO

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_GENERO	Código do gênero	N	5	99.999	Chave primária
GENERO	Descrição do gênero	C	20		Não Nulo

Depósito de Dados: PRODUTO

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_PRODUTO	Código do produto	N	5	99.999	Chave primária
UNIDADE	Unidade do produto	C	2		Não Nulo
*CODIGO_GENERO	Código do gênero do produto	N	5	99.999	Chave estrangeira / Não Nulo
PRODUTO	Descrição do produto	C	30		Não Nulo
ALIQUOTA_ICMS	Alíquota de ICMS do produto	N	5,2	999,99	Não Nulo
ALIQUOTA_IPI	Alíquota de IPI do produto	N	5,2	999,99	Não Nulo
CODIGO_BARRAS	Código de barras do produto	C	13		Não Nulo
TIPO	Tipo do produto	C	20		Não Nulo
SITUACAO_TRIBUTARIA	Situação tributária do produto	C	1		

Depósito de Dados: USUARIO

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO	Código do usuário do sistema	N	5	99.999	Chave primária
USUARIO	Nome do usuário do sistema	C	35		Não Nulo
LOGIN	Login do usuário do sistema	C	10		Não Nulo
SENHA	Senha do usuário do sistema	C	10		Não Nulo
LIBERA	Permissão do usuário para cadastros	C	1		
EXCLUI	Permissão do usuário para exclusões	C	1		

Depósito de Dados: PERMISSAO

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO	Código da permissão do usuário	N	5	99.999	Chave primária
*USUARIO	Código do usuário dono da permissão	N	5	99.999	Chave estrangeira / Não Nulo
PERMISSAO	Nome da permissão	C	30		Não Nulo

Depósito de Dados: MVCAIXAPAN

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_MVCAIXA	Código da movimentação de caixa	N	10	9.999.999.999	Chave primária
DESCONTO	Desconto dado para a movimentação	N	5,2	999,99	Não Nulo
DATA_MVCAIXA	Data da movimentação de caixa	DATA	8	dd/mm/aaaa	Não Nulo
HORA_MVCAIXA	Hora da movimentação de caixa	HORA	6	hh:mm:ss	Não Nulo
VALOR	Valor da movimentação de caixa	N	9,2	9.999.999,99	Não Nulo
*CODIGO_CAIXA	Código do caixa da movimentção	N	10	9.999.999.999	Chave estrangeira / Não Nulo

Depósito de Dados: MOVIMENTOPAN

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_MOVIMENTO	Código da movimentação de estoque	N	10	9.999.999.999	Chave primária
DATA_MOVIMENTO	Data da movimentação de estoque	DATA	8	dd/mm/aaaa	Não Nulo
*CODIGO_PRODUTO	Código do produto movimentado	N	5	99.999	Chave estrangeira / Não Nulo
QUANTIDADE	Quantidade movimentada do produto	N	9,2	9.999.999,99	Não Nulo
VALOR	Valor unitário do produto movimentado	N	9,2	9.999.999,99	Não Nulo
CODIGO_OPERACAO	Código da operação realizada	N	5	99.999	Não Nulo
*CODIGO_MVCAIXA	Código da movimentação de caixa	N	10	9.999.999.999	Chave estrangeira

Depósito de Dados: CAIXAPAN

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_CAIXA	Código do caixa	N	10	9.999.999.999	Chave primária
HORA_INICIO	Hora em que o caixa foi aberto	HORA	6	hh:mm:ss	Não Nulo
HORA_TERMINO	Hora em que o caixa foi fechado	HORA	6	hh:mm:ss	
CAIXA_INICIAL	Valor com o qual caixa foi aberto	N	9,2	9.999.999,99	Não Nulo
CAIXA_FINAL	Valor com o qual caixa foi fechado	N	9,2	9.999.999,99	
TERMINAL	Nome da máquina usada para o caixa	C	15		Não Nulo
DATA_INICIO	Data em que o caixa foi aberto	DATA	8	dd/mm/aaaa	Não Nulo
DATA_TERMINO	Data em que o caixa foi fechado	DATA	8	dd/mm/aaaa	

Depósito de Dados: PARCELAPAN

Atributo	Descrição	Tipo	Tam.	Máscara	Restrição
CODIGO_PARCELA	Código da parcela	N	10	9.999.999.999	Chave primária
*CODIGO_DOCUMENTO	Código do documento usado na parcela	N	5	99.999	Chave estrangeira / Não Nulo
VALOR_PARCELA	Valor da parcela	N	9,2	9.999.999,99	Não Nulo
*CODIGO_MVCAIXA	Código da movimentação de caixa	N	10	9.999.999.999	Chave estrangeira / Não Nulo

FLUXOS DE DADOS

Dados_Login = LOGIN + SENHA

Dados_Gênero = Novos_Dados_Gênero = GENERO

ID_Gênero = {@CODIGO_GENERO | GENERO}

Dados_Loja = Novos_Dados_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]

ID_Loja = {@CODIGO_LOJA | LOJA | CNPJ}

Dados_Produto = Novos_Dados_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]

ID_Produto = {@CODIGO_PRODUTO | PRODUTO | CODIGO_BARRAS}

Dados_Documento = Novos_Dados_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO

ID_Documento = {@CODIGO_DOCUMENTO | DOCUMENTO}

Dados_Usuario = Novos_Dados_Usuario = USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]

ID_Usuario = {@CODIGO_USUARIO | USUARIO | LOGIN + SENHA}

Dados_Permissao = Novos_Dados_Permissao = PERMISSAO + CODIGO_USUARIO

ID_Permissao = @CODIGO_PERMISSAO

Relatório_Gênero = @CODIGO_GENERO + GENERO

Relatório_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]

Relatório_Produtos_Por_Gênero = GENERO + (@CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA])

Relatório_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO

Relatório_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]

Relatório_Usuário = @CODIGO_USUARIO + USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]

Intervalo_Datas = DataInicial + DataFinal

Dados_Caixa = @CODIGO_CAIXA + HORA_INICIO + [HORA_TERMINO] + CAIXA_INICIAL + [CAIXA_FINAL] + TERMINAL + DATA_INICIO + [DATA_TERMINO]

Dados_MVCaixa = @CODIGO_MVCAIXA + DESCONTO + DATA_MVCAIXA + HORA_MVCAIXA + VALOR + CODIGO_CAIXA

Dados_Movimento = @CODIGO_MOVIMENTO + DATA_MOVIMENTO + CODIGO_PRODUTO + QUANTIDADE + VALOR + CODIGO_OPERACAO + [CODIGO_MVCAIXA]

Dados_Parcela = @CODIGO_PARCELA + CODIGO_DOCUMENTO + VALOR_PARCELA + CODIGO_MVCAIXA

Relatório_Movimento = **Relatório_Movimento_Por_Produto** = @CODIGO_MOVIMENTO + CODIGO_PRODUTO + PRODUTO + DATA_MOVIMENTO + QUANTIDADE + VALOR + TOTAL + CODIGO_OPERACAO

Tipo_Movimento = ["1"] %Entrada por compra% + ["2"] %Saída por venda% + ["3"] %Saída por sobra% + ["4"] %Entrada por sobra% + ["5"] %Saída para produção% + ["6"] %Entrada por produção% + ["7"] %Saída por desperdício%

Relatório_Caixa = @CODIGO_CAIXA + HORA_INICIO + [HORA_TERMINO] + CAIXA_INICIAL + [CAIXA_FINAL] + TERMINAL + DATA_INICIO + [DATA_TERMINO]

Relatório_Histórico_Caixa = QUANT_DE_CAIXAS + FATURAMENTO + TOTAL_DE_ATENDIMENTOS + TOTAL_DE_DESCONTOS + FATURAMENTO_MEDIO + MEDIA_DE_ATENDIMENTOS + MEDIA_DE_DESCONTOS

Relatório_Faturamento_Por_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO + TOTAL_DO_DOCUMENTO

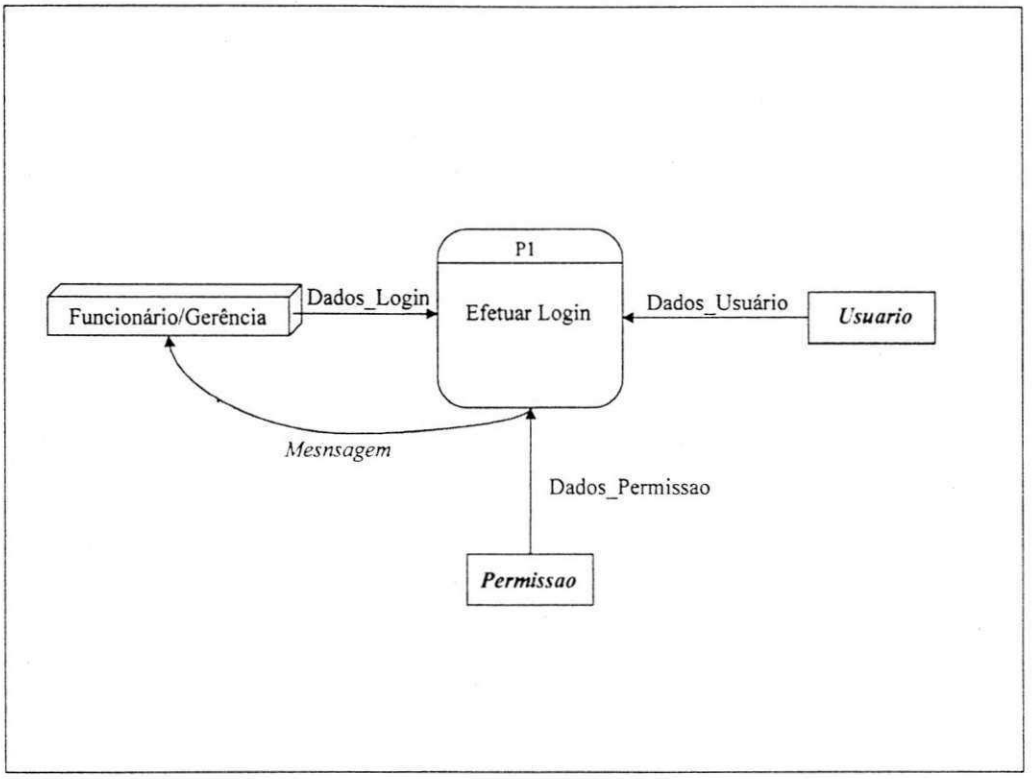
Legenda:

=	É equivalente a
{ }	Ou
*(min - max)	Repetições
[ ]	Opcional
@	Chave
% %	Comentário

Anexo 1D

Lista de Eventos:

1. Funcionário/Gerência Efetua login;
2. Funcionário Cadastra Gênero;
3. Funcionário Edita Gênero;
4. Funcionário Consulta Gênero;
5. Funcionário Cadastra Loja;
6. Funcionário Edita Loja;
7. Funcionário Consulta Loja;
8. Funcionário Cadastra Produto;
9. Funcionário Edita Produto;
10. Funcionário Consulta Produto;
11. Funcionário Cadastra Documento;
12. Funcionário Edita Documento;
13. Funcionário Consulta Documento;
14. Gerência Cadastra Usuário;
15. Gerência Edita Usuário;
16. Gerência Consulta Usuário;
17. Gerência Cadastra Permissão;
18. Gerência Exclui Permissão;
19. Gerência Consulta Permissão;
20. Gerência Solicita Relatório de Gênero;
21. Gerência Solicita Relatório de Produtos;
22. Gerência Solicita Relatório de Produtos por Gênero;
23. Gerência Solicita Relatório de Documentos;
24. Gerência Solicita Relatório de Lojas;
25. Gerência Solicita Relatório de Usuários;
26. Gerência Transfere Dados da Filial para a Matriz;
27. Gerência Solicita Relatório de Movimentação de Estoque;
28. Gerência Solicita Relatório de Movimentação de Estoque por Produto Específico;
29. Gerência Solicita Relatório Resumido da Caixa;
30. Gerência Solicita Relatório de Histórico de Caixa;
31. Gerência Solicita Relatório de Faturamento por Documento;



P1: Funcionário/Gerência Efetua Login

Receba Dados_Login
Pesquise em ***Usuário*** o elemento onde LOGIN de Dados_Login é igual a LOGIN de Dados_Usuário e SENHA de Dados_Login é igual a SENHA de Dados_Usuário
Se existir
    Obtenha Dados_Usuário em ***Usuário***
    Enquanto houver elementos em ***Permissão*** onde CODIGO_USUARIO de Dados_Permissão é igual a @CODIGO_USUARIO de Dados_Usuário faça
        Obtenha PERMISSAO em ***Permissão***
        Libere Permissão_De_Uso_Do_Sistema equivalente à PERMISSAO
    Fim enquanto
Senão
    Imprima "Par Login e Senha inválido. Usuário não encontrado."

P1: Funcionário/Gerência Efetua Login

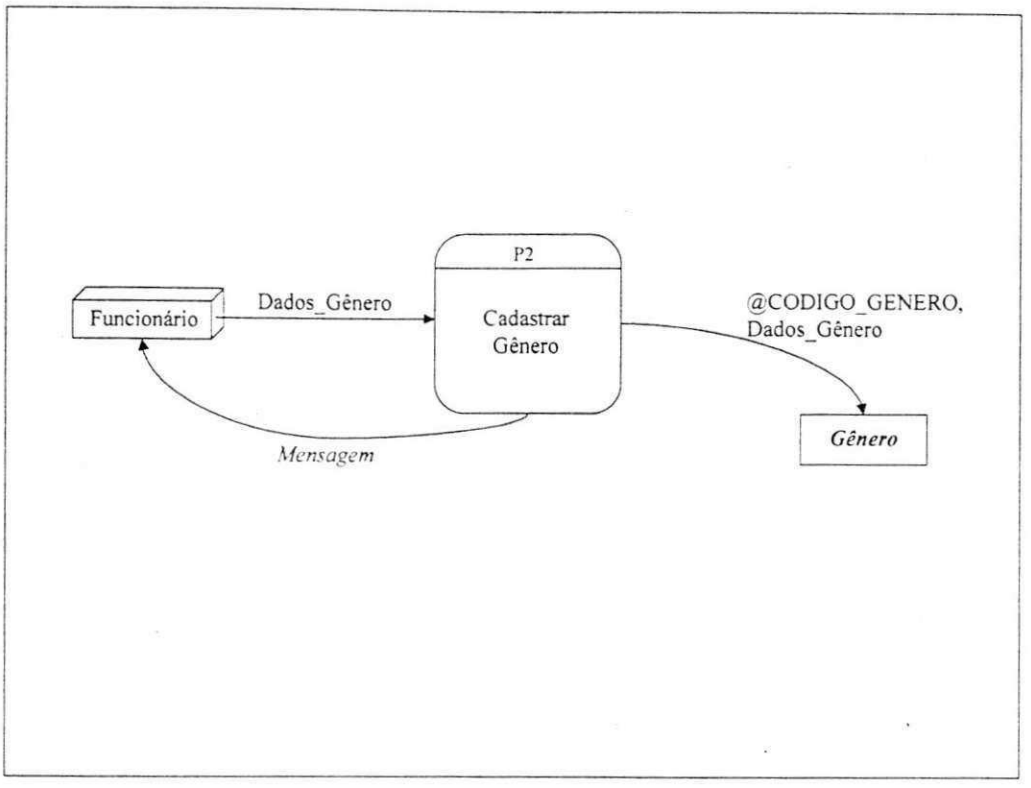
Fluxos de Dados:

Dados_Login = LOGIN + SENHA

Dados_Usuario = USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]

Dados_Permissao = PERMISSAO + CODIGO_USUARIO

D.E.R Parcial:



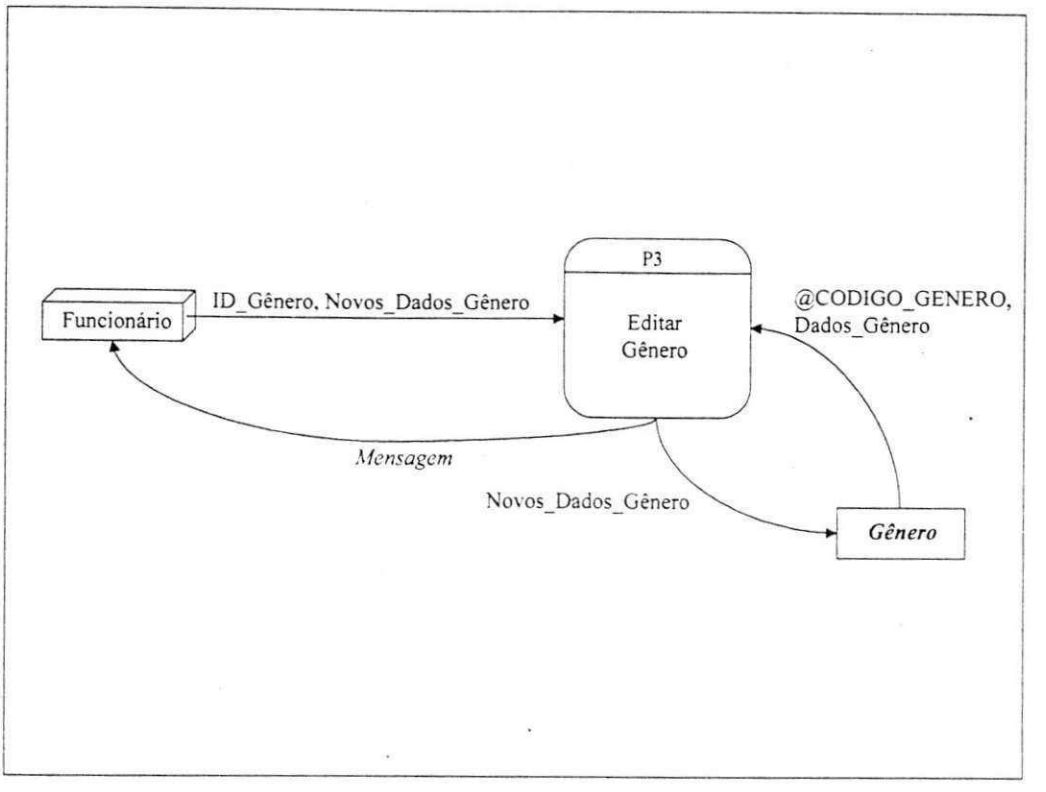
P2: Funcionário Cadastra Gênero

Receba Dados_Gênero
Gere @CODIGO_GENERO
Grave @CODIGO_GENERO e Dados_Gênero em *Gênero*
Imprima "Cadastro realizado com sucesso"

P2: Funcionário Cadastra Gênero

Fluxos de Dados:

Dados_Gênero = GENERO



P3: Funcionário Edita Gênero

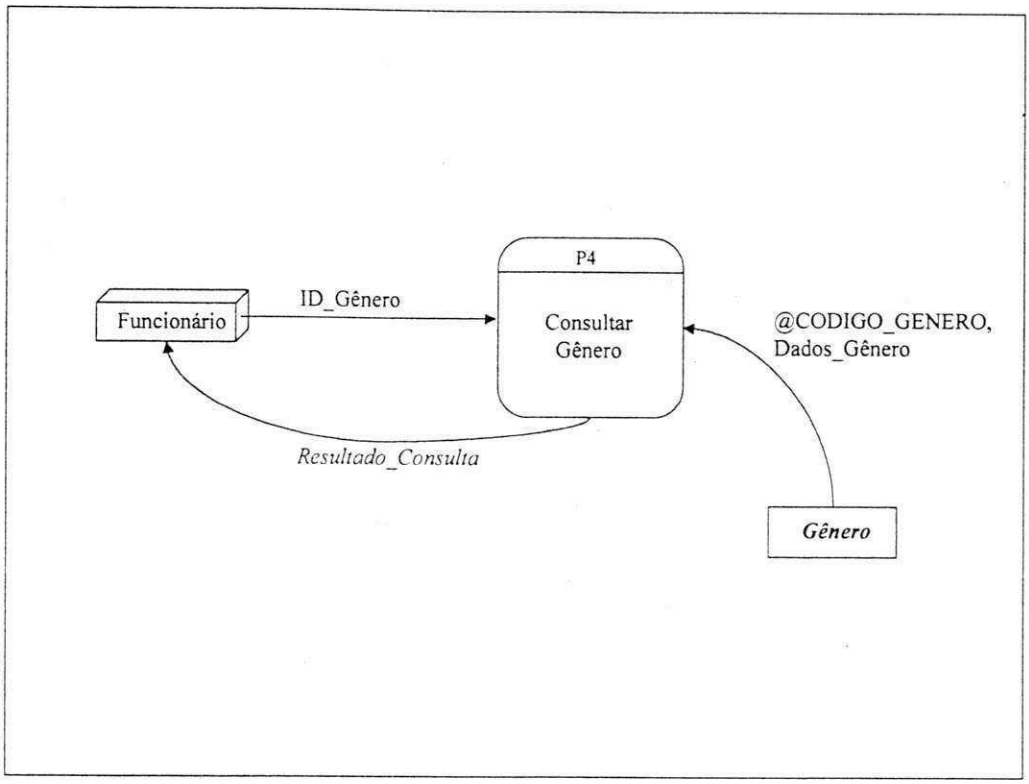
Receba ID_Gênero, Novos_Dados_Gênero
Verifique se ID_Gênero existe em ***Gênero***
Se existir
    Obtenha @CODIGO_GENERO e Dados_Gênero em ***Gênero***
    Grave Novos_Dados_Gênero em ***Gênero***
    Imprima "Edição realizada com sucesso"
Senão
    Imprima "Gênero não cadastrado"
Fim se

P3: Funcionário Edita Gênero

Fluxos de Dados:

Dados_Gênero = Novos_Dados_Gênero = GENERO

ID_Gênero = {@CODIGO_GENERO | GENERO}



P4: Funcionário Consulta Gênero

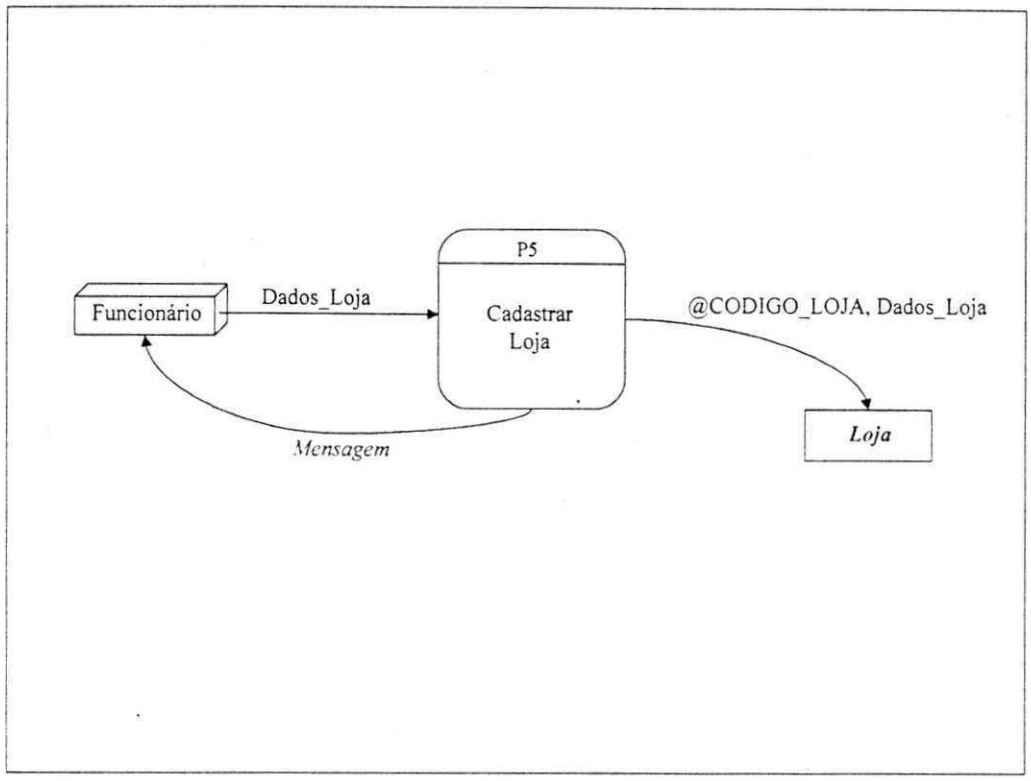
Receba ID_Gênero
Verifique se ID_Gênero existe em ***Gênero***
Se existir
    Obtenha @CODIGO_GENERO e Dados_Gênero em ***Gênero***
    Exiba @CODIGO_GENERO e Dados_Gênero
Senão
    Imprima "Gênero não cadastrado"
Fim se

P4: Funcionário Consulta Gênero

Fluxos de Dados:

Dados_Gênero = GENERO

ID_Gênero = {@CODIGO_GENERO | GENERO}



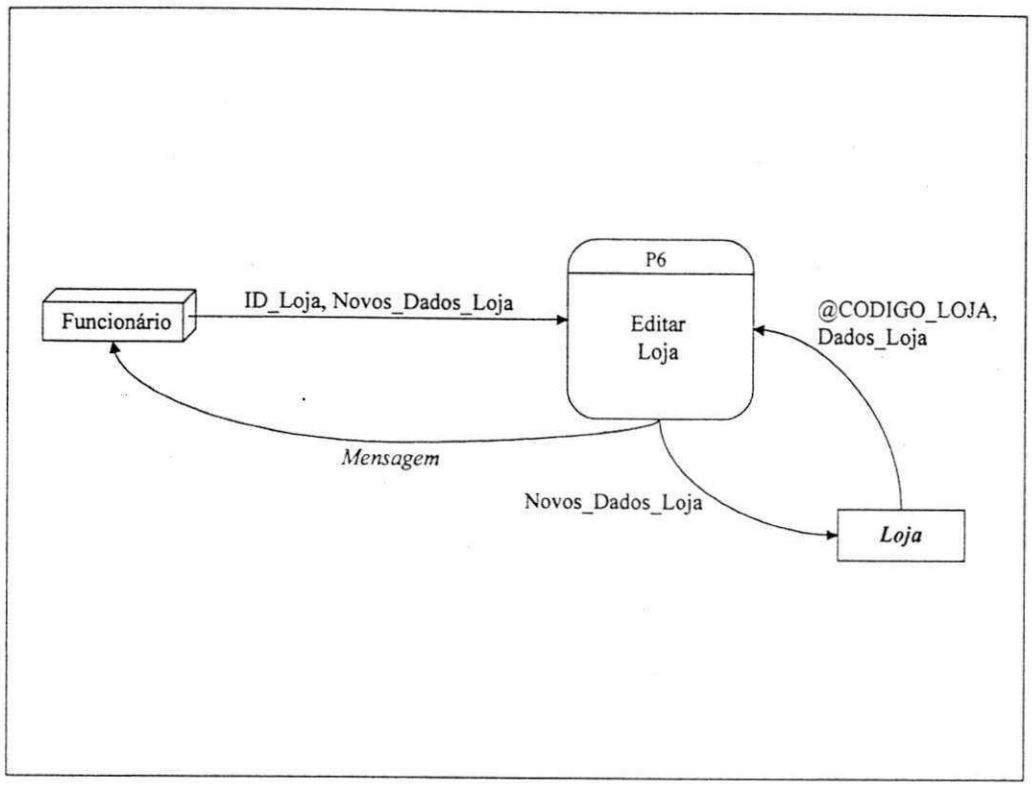
P5: Funcionário Cadastra Loja

Receba Dados_Loja
Gere @CODIGO_LOJA
Grave @CODIGO_LOJA e Dados_Loja em ***Loja***
Imprima "Cadastro realizado com sucesso"

P5: Funcionário Cadastra Loja

Fluxos de Dados:

Dados_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]



P6: Funcionário Edita Loja

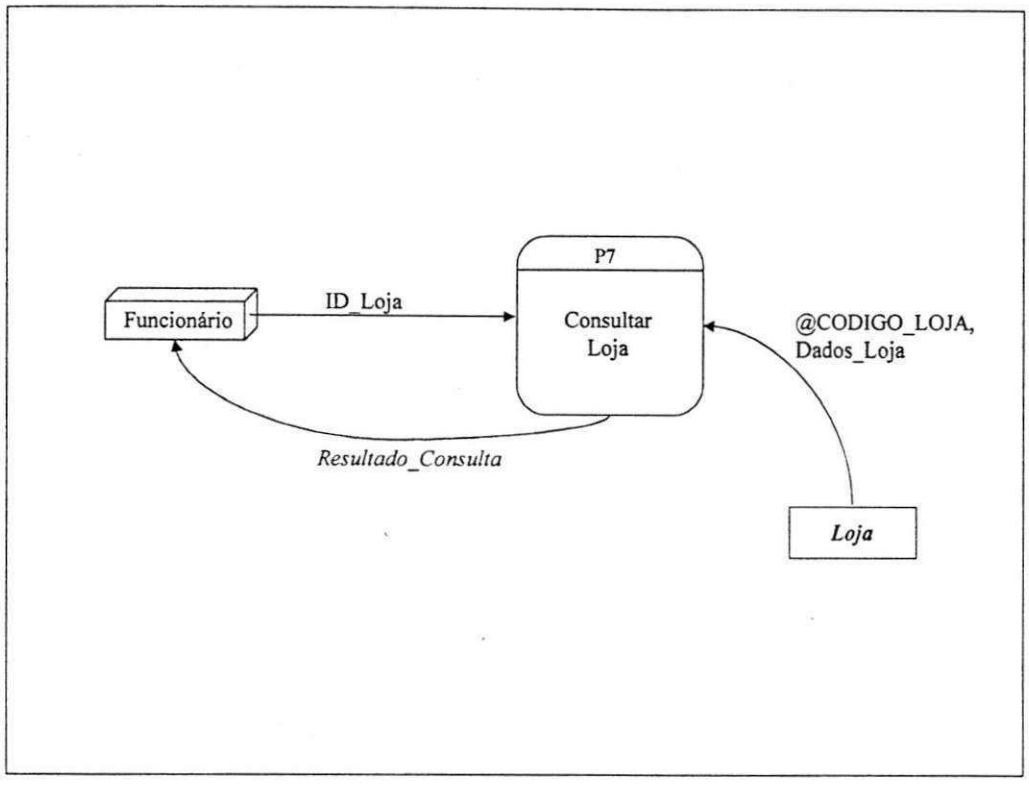
Receba ID_Loja, Novos_Dados_Loja
Verifique se ID_Loja existe em ***Loja***
Se existir
    Obtenha @CODIGO_LOJA e Dados_Loja em ***Loja***
    Grave Novos_Dados_Loja em ***Loja***
    Imprima "Edição realizada com sucesso"
Senão
    Imprima "Loja não cadastrada"
Fim se

P6: Funcionário Edita Loja

Fluxos de Dados:

Dados_Loja = Novos_Dados_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]

ID_Loja = {@CODIGO_LOJA | LOJA | CNPJ}



P7: Funcionário Consulta Loja

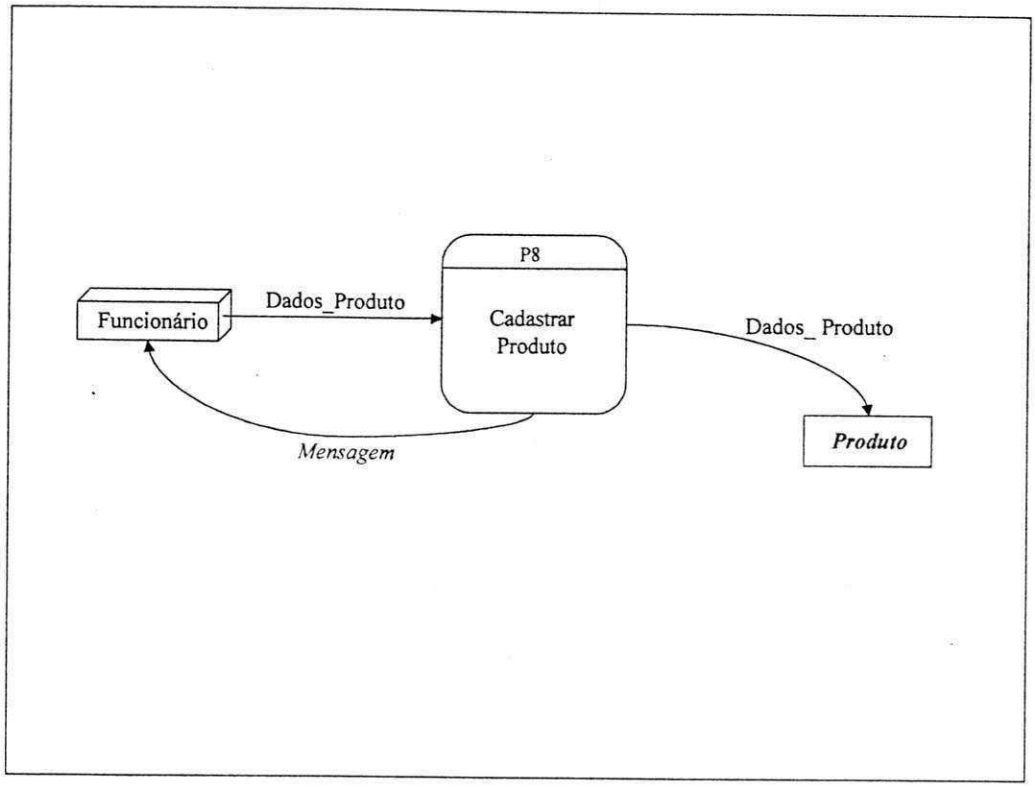
Receba ID_Loja
Verifique se ID_Loja existe em ***Loja***
Se existir
    Obtenha @CODIGO_LOJA e Dados_Loja em ***Loja***
    Exiba @CODIGO_LOJA e Dados_Loja
Senão
    Imprima "Loja não cadastrada"
Fim se

P7: Funcionário Consulta Loja

Fluxos de Dados:

Dados_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]

ID_Loja = {@CODIGO_LOJA | LOJA | CNPJ}



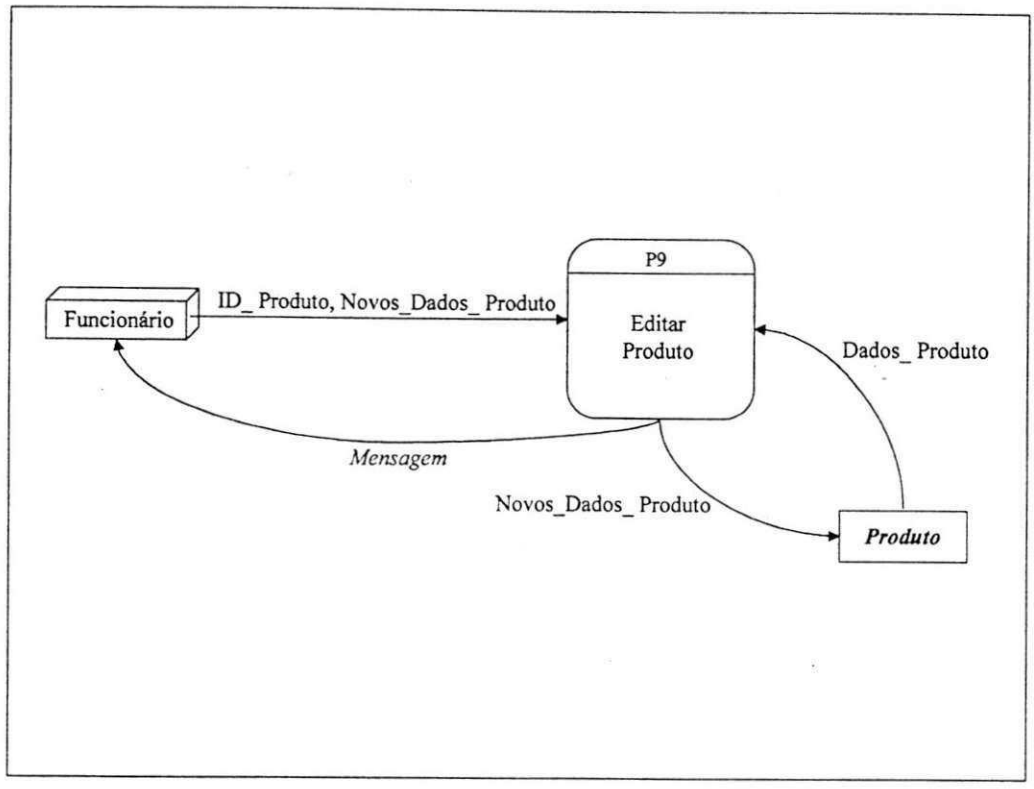
P8: Funcionário Cadastra Produto

Receba Dados_Produto
Grave Dados_Produto em ***Produto***
Imprima "Cadastro realizado com sucesso"

P8: Funcionário Cadastra Produto

Fluxos de Dados:

Dados_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]



P9: Funcionário Edita Produto

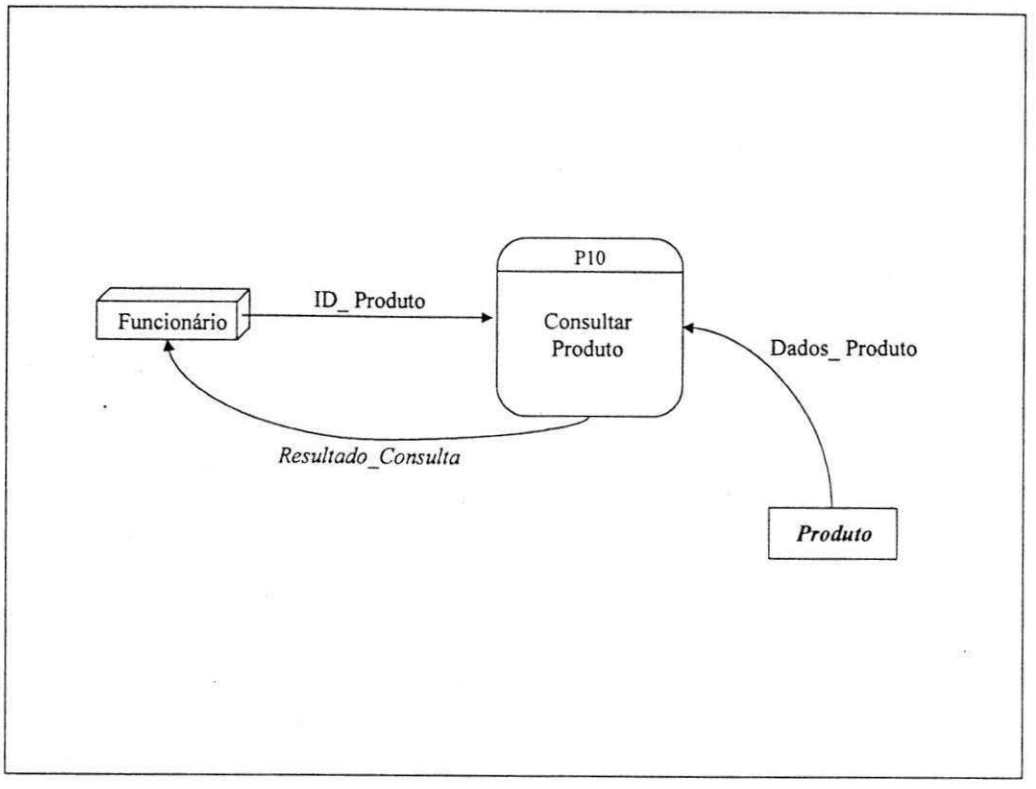
Receba ID_Produto, Novos_Dados_Produto
Verifique se ID_Produto existe em ***Produto***
Se existir
    Obtenha Dados_Produto em ***Produto***
    Grave Novos_Dados_Produto em ***Produto***
    Imprima "Edição realizada com sucesso"
Senão
    Imprima "Produto não cadastrado"
Fim se

P9: Funcionário Edita Produto

Fluxos de Dados:

Dados_Produto = **Novos_Dados_Produto** = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]

ID_Produto = {@CODIGO_PRODUTO | PRODUTO | CODIGO_BARRAS}



P10: Funcionário Consulta Produto

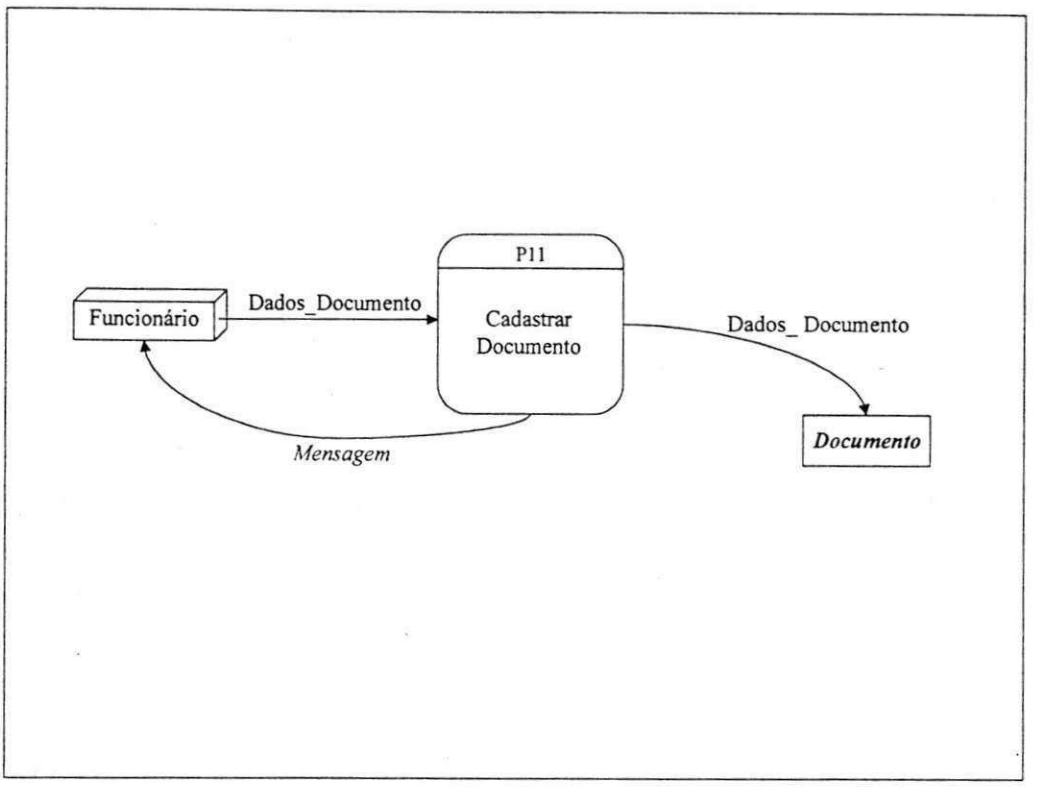
Receba ID_Produto
Verifique se ID_Produto existe em ***Produto***
Se existir
    Obtenha Dados_Produto em ***Produto***
    Exiba Dados_Produto
Senão
    Imprima "Produto não cadastrado"
Fim se

P10: Funcionário Consulta Produto

Fluxos de Dados:

Dados_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]

ID_Produto = {@CODIGO_PRODUTO | PRODUTO | CODIGO_BARRAS}



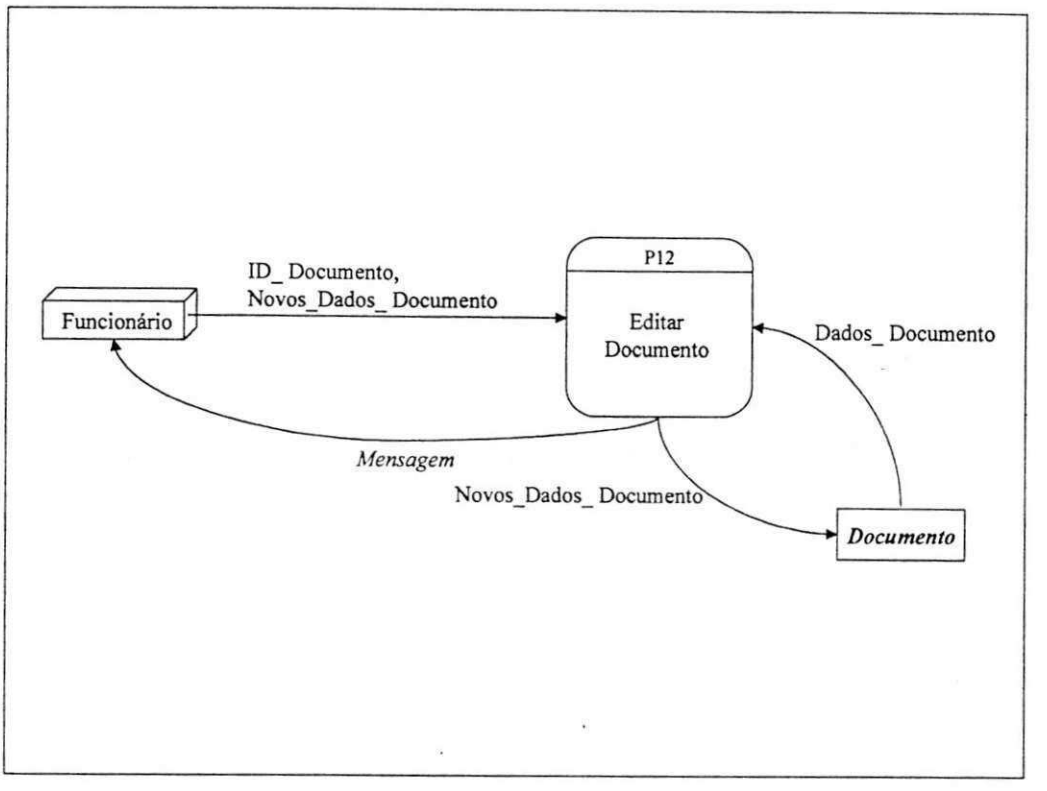
P11: Funcionário Cadastra Documento

Receba Dados_Documento
Grave Dados_Documento em ***Documento***
Imprima "Cadastro realizado com sucesso"

P11: Funcionário Cadastra Documento

Fluxos de Dados:

Dados_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO



P12: Funcionário Edita Documento

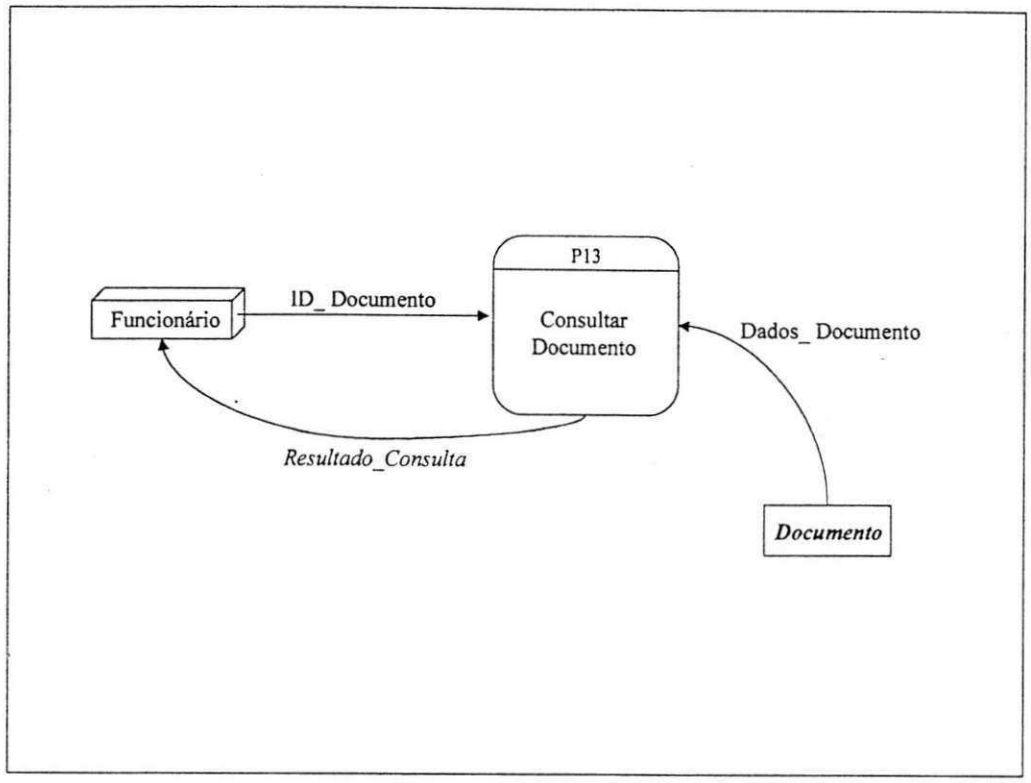
Receba ID_Documento, Novos_Dados_Documento
Verifique se ID_Documento existe em ***Documento***
Se existir
    Obtenha Dados_Documento em ***Documento***
    Grave Novos_Dados_Documento em ***Documento***
    Imprima "Edição realizada com sucesso"
Senão
    Imprima "Documento não cadastrado"
Fim se

P12: Funcionário Edita Documento

Fluxos de Dados:

Dados_Documento = Novos_Dados_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO

ID_Documento = {@CODIGO_DOCUMENTO | DOCUMENTO}



P13: Funcionário Consulta Documento

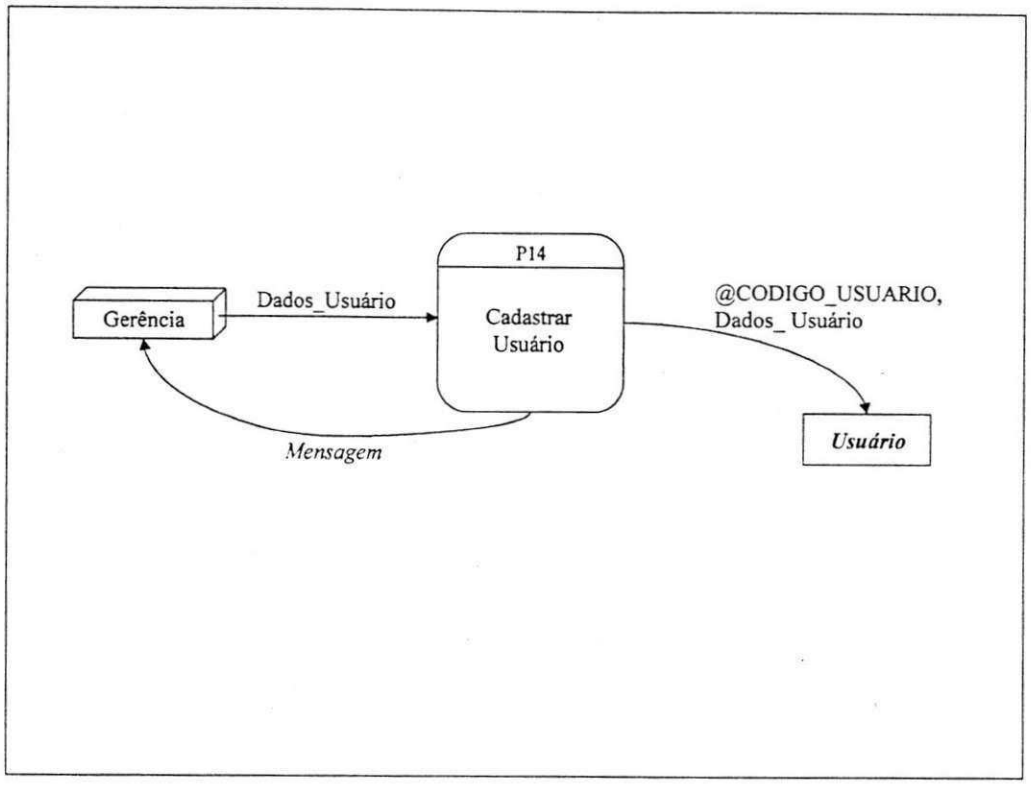
Receba ID_Documento
Verifique se ID_Documento existe em ***Documento***
Se existir
    Obtenha Dados_Documento em ***Documento***
    Exiba Dados_Documento
Senão
    Imprima "Documento não cadastrado"
Fim se

P13: Funcionário Consulta Documento

Fluxos de Dados:

Dados_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO

ID_Documento = {@CODIGO_DOCUMENTO | DOCUMENTO}



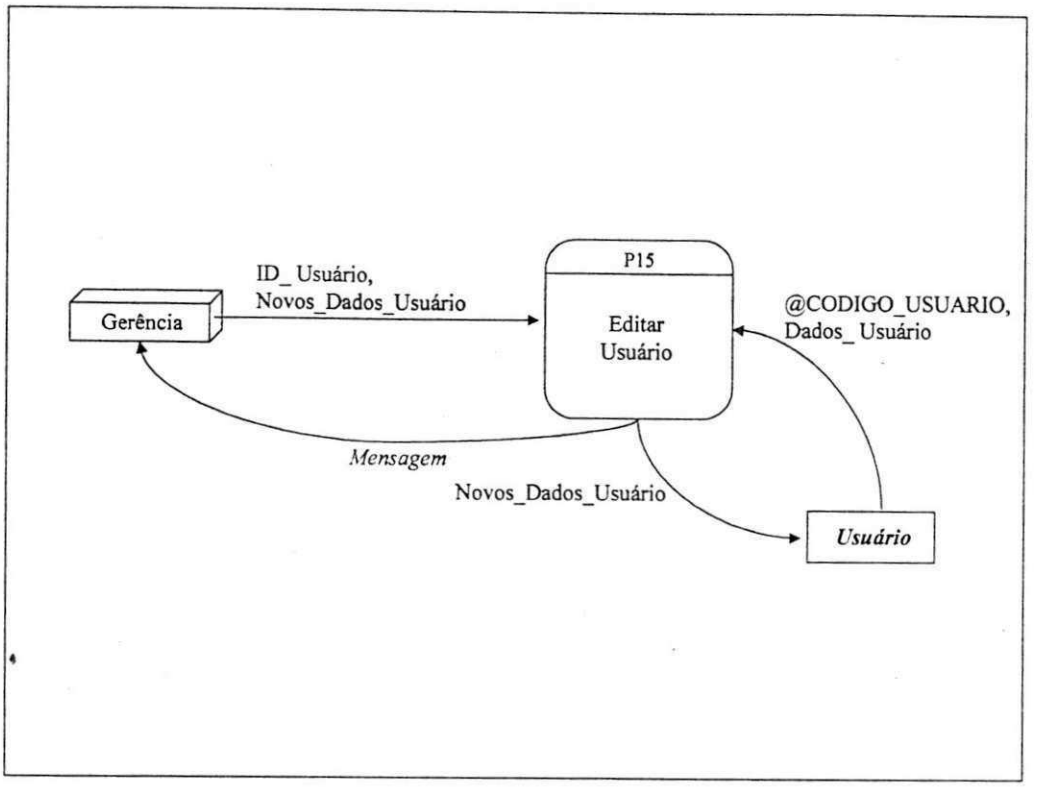
P14: Gerência Cadastra Usuário

Receba Dados_Usuário
Gere @CODIGO_USUARIO
Grave @CODIGO_USUARIO e Dados_Usuário em ***Usuário***
Imprima "Cadastro realizado com sucesso"

P14: Gerência Cadastra Usuário

Fluxos de Dados:

Dados_Usuario = USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]



P15: Gerência Edita Usuário

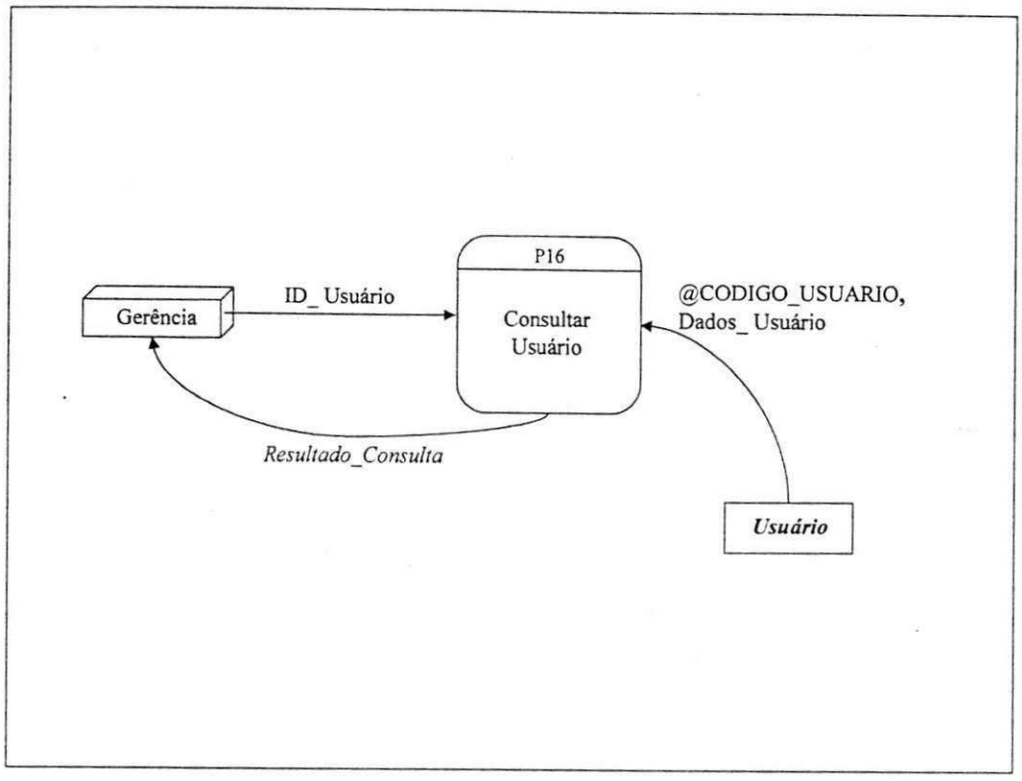
Receba ID_Usuário, Novos_Dados_Usuário
Verifique se ID_Usuário existe em ***Usuário***
Se existir
    Obtenha @CODIGO_USUARIO e Dados_Usuário em ***Usuário***
    Grave Novos_Dados_Usuário em ***Usuário***
    Imprima "Edição realizada com sucesso"
Senão
    Imprima "Usuário não cadastrado"
Fim se

P15: Gerência Edita Usuário

Fluxos de Dados:

Dados_Usuario = **Novos_Dados_Usuario** = USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]

ID_Usuario = {@CODIGO_USUARIO | USUARIO | LOGIN + SENHA}



P16: Gerência Consulta Usuário

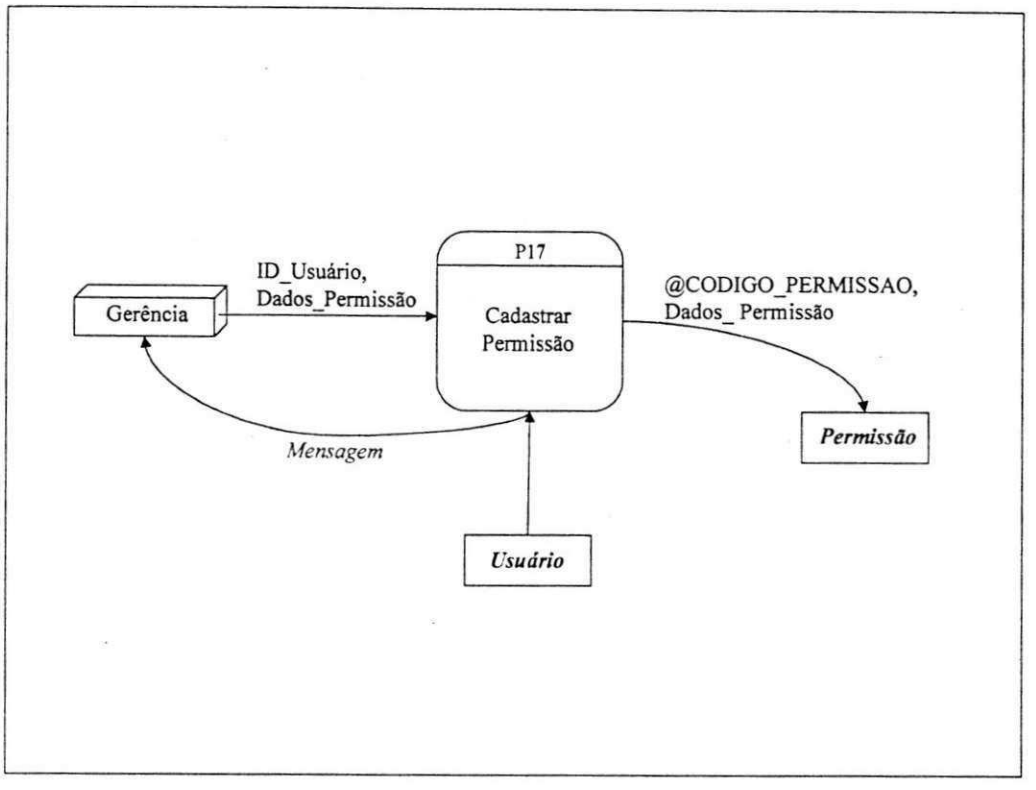
Receba ID_Usuário
Verifique se ID_Usuário existe em ***Usuário***
Se existir
    Obtenha @CODIGO_USUARIO e Dados_Usuário em ***Usuário***
    Exiba @CODIGO_USUARIO e Dados_Usuário
Senão
    Imprima "Usuário não cadastrado"
Fim se

P16: Gerência Consulta Usuário

Fluxos de Dados:

Dados_Usuario = USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]

ID_Usuario = {@CODIGO_USUARIO | USUARIO | LOGIN + SENHA}



P17: Gerência Cadastra Permissão

Receba ID_Usuário, Dados_Permissão
Verifique se ID_Usuário existe em ***Usuário***
Se existir
    Gere @CODIGO_PERMISSAO
    Grave @CODIGO_PERMISSAO e Dados_Permissão em ***Permissão***
    Imprima "Cadastro realizado com sucesso"
Senão
    Imprima "Usuário não cadastrado. Não é possível cadastrar permissão"
Fim se

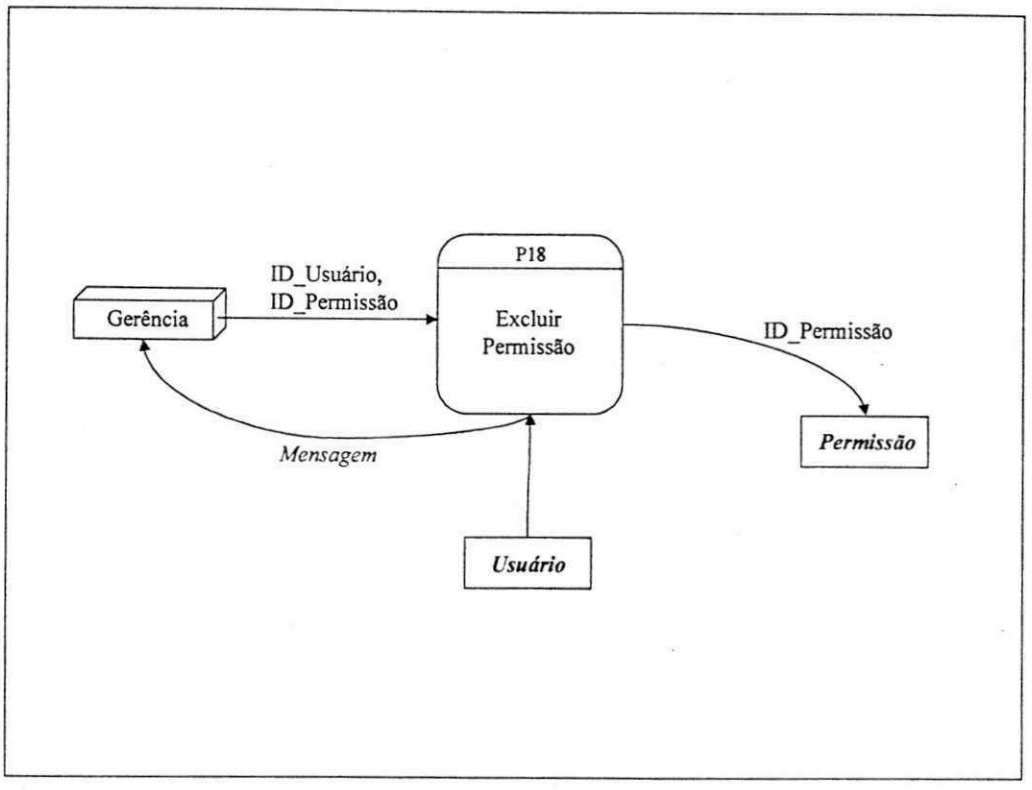
P17: Gerência Cadastra Permissão

Fluxos de Dados:

ID_Usuario = {@CODIGO_USUARIO | USUARIO | LOGIN + SENHA}

Dados_Permissao = Novos_Dados_Permissao = PERMISSAO + CODIGO_USUARIO

D.E.R Parcial:



P18: Gerência Exclui Permissão

Receba ID_Usuário, ID_Permissão
Verifique se ID_Usuário existe em ***Usuário***
Se existir
    Verifique se ID_Permissão existe em ***Permissão***
    Se existir
        Exclua @CODIGO_PERMISSAO e Dados_Permissão de ***Permissão***
        Imprima "Permissão excluída com sucesso."
    Senão
        Imprima "Permissão não cadastrada. Impossível excluí-la."
    Fim se
Senão
    Imprima "Usuário não cadastrado."
Fim se

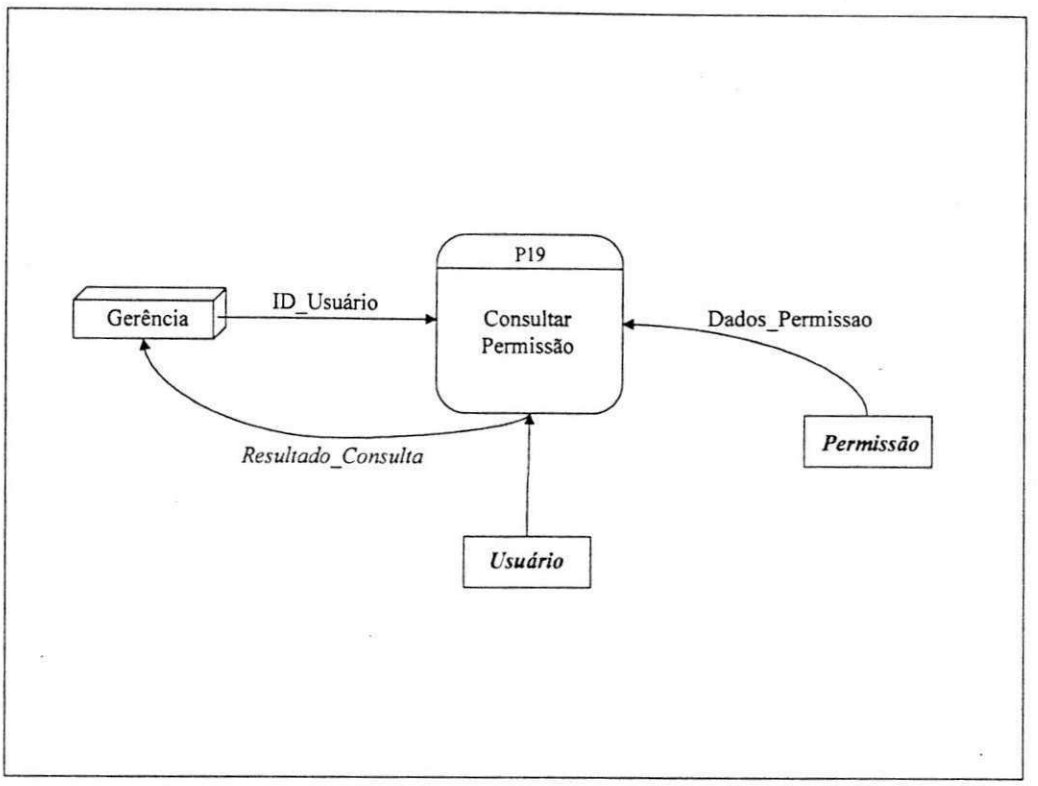
P18: Gerência Exclui Permissão

Fluxos de Dados:

ID_Usuario = {@CODIGO_USUARIO | USUARIO | LOGIN + SENHA}

ID_Permissao = @CODIGO_PERMISSAO

D.E.R Parcial:



P19: Gerência Consulta Permissão

Receba ID_Usuário
Verifique se ID_Usuário existe em ***Usuário***
Se existir
    Pesquise em ***Permissão*** elementos onde CODIGO_USUARIO de Dados_Permissão é igual a @CODIGO_USUARIO de ID_Usuário

    Para cada elemento encontrado faça
        Obtenha Dados_Permissão em ***Permissão***
        Exiba Dados_Permissão
    Fim para

Senão
    Imprima "Usuário não cadastrado."
Fim se

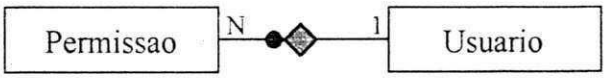
P19: Gerência Consulta Permissão

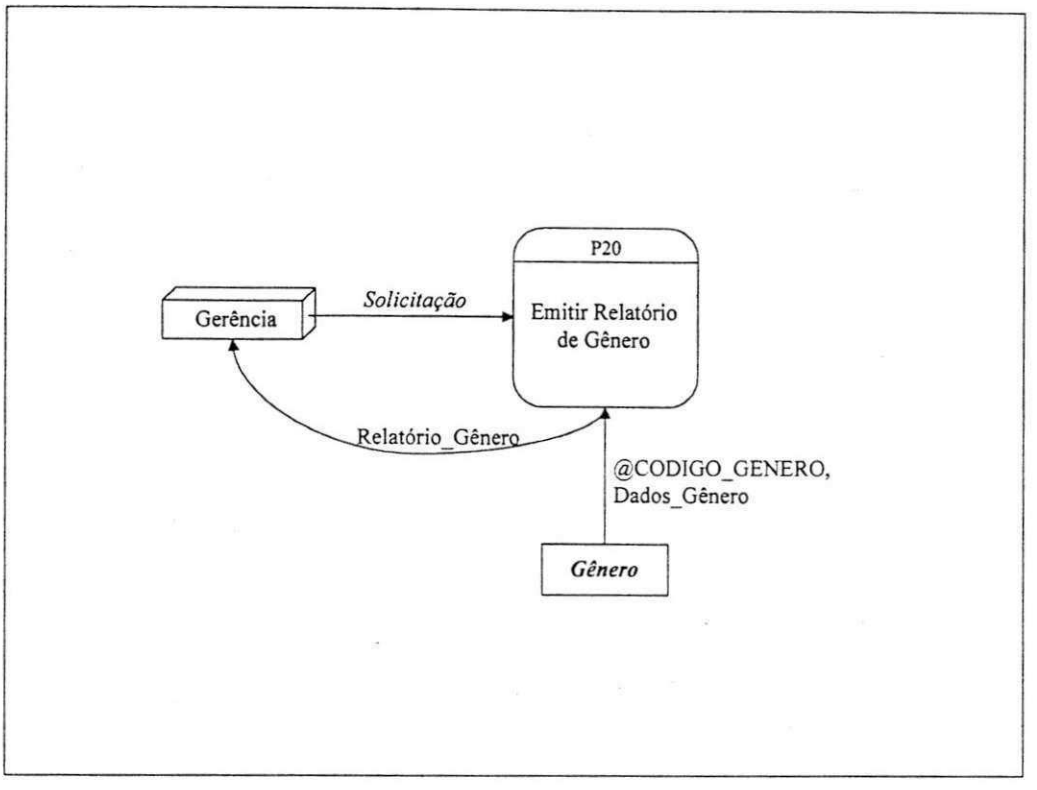
Fluxos de Dados:

ID_Usuario = {@CODIGO_USUARIO | USUARIO | LOGIN + SENHA}

Dados_Permissao = PERMISSAO + CODIGO_USUARIO

D.E.R Parcial:





P20: Gerência Solicita Relatório de Gênero

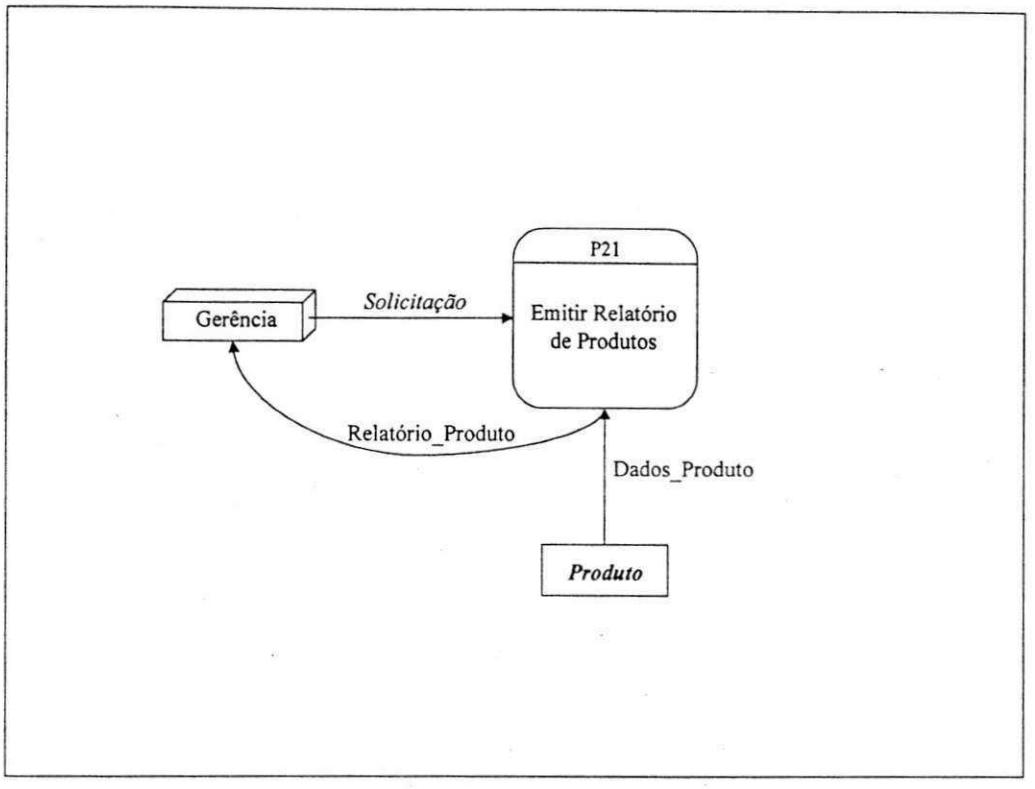
Receba *Solicitação*
Enquanto houver elementos em ***Gênero*** faça
    Obtenha @CODIGO_GENERO e Dados_Gênero em ***Gênero***
Fim enquanto
Exiba Relatório_Gênero

P20: Gerência Solicita Relatório de Gênero

Fluxos de Dados:

Dados_Gênero = **Novos_Dados_Gênero** = GENERO

Relatório_Gênero = @CODIGO_GENERO + GENERO



P21: Gerência Solicita Relatório de Produtos

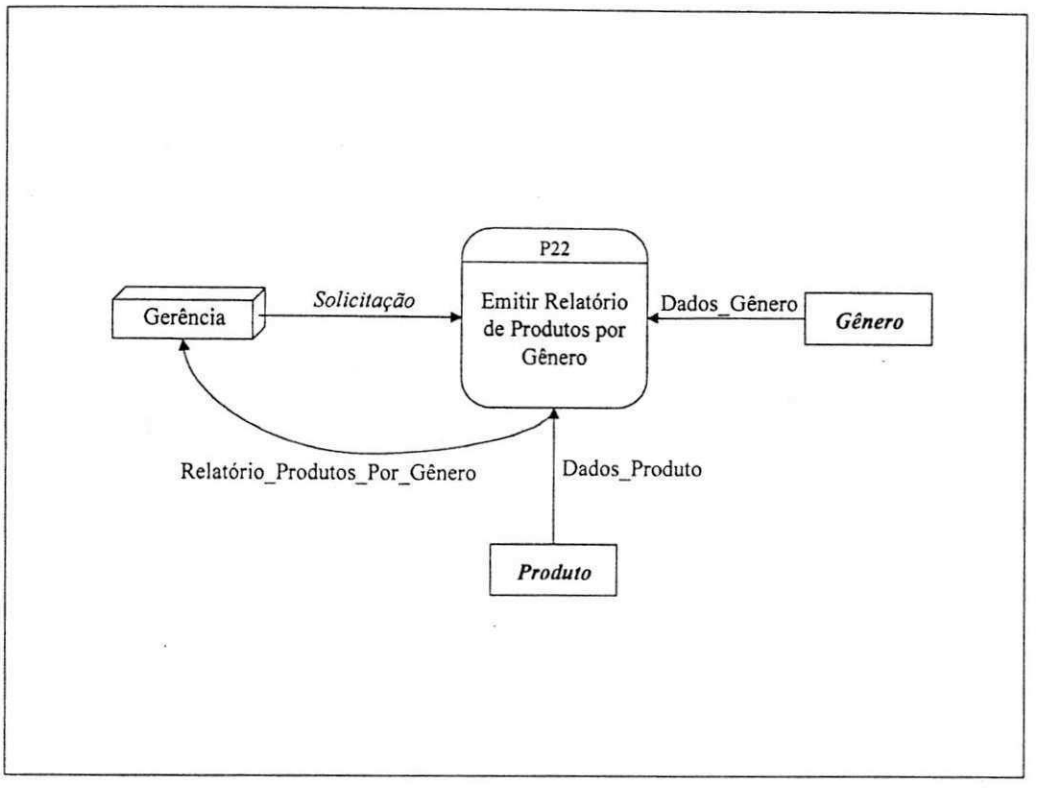
Receba *Solicitação*
Enquanto houver elementos em ***Produto*** faça
    Obtenha Dados_Produto em ***Produto***
Fim enquanto
Exiba Relatório_Produto

P21: Gerência Solicita Relatório de Produtos

Fluxos de Dados:

Relatório_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]

Dados_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]



P22: Gerência Solicita Relatório de Produtos Por Gênero

Receba *Solicitação*
Enquanto houver elementos em ***Gênero*** faça
    Obtenha GENERO em ***Gênero***
    Enquanto houver elementos em ***Produto*** onde CODIGO_GENERO de Dados_Produto é igaul a @CODIGO_GENERO de Dados_Genero faça
        Obtenha Dados_Produto em ***Produto***
    Fim enquanto
Fim enquanto
Exiba Relatório_Produtos_Por_Gênero

P22: Gerência Solicita Relatório de Produtos por Gênero

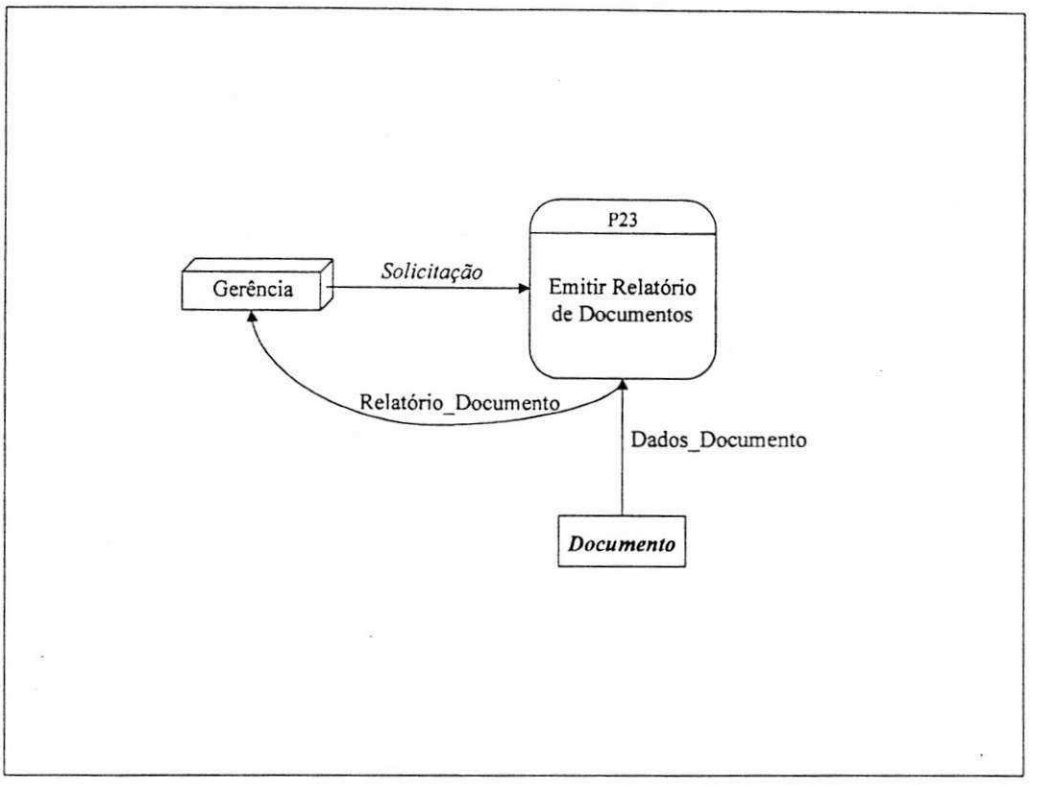
Fluxos de Dados:

Dados_Produto = @CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA]

Dados_Gênero = Novos_Dados_Gênero = GENERO

Relatório_Produtos_Por_Gênero = GENERO + (@CODIGO_PRODUTO + PRODUTO + UNIDADE + CODIGO_GENERO + ALIQUOTA_ICMS + ALIQUOTA_IPI + CODIGO_BARRAS + TIPO + [SITUACAO_TRIBUTARIA])

D.E.R Parcial:



P23: Gerência Solicita Relatório de Documentos

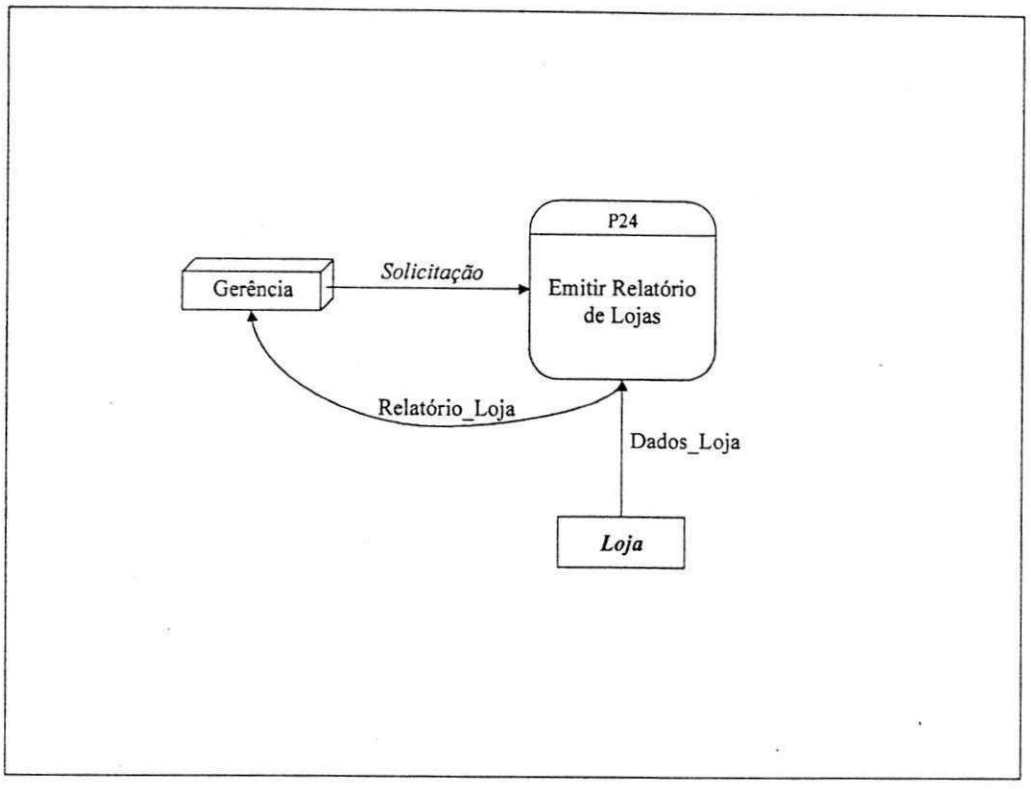
Receba *Solicitação*
Enquanto houver elementos em ***Documento*** faça
    Obtenha Dados_Documento em ***Documento***
Fim enquanto
Exiba Relatório_Documento

P23: Gerência Solicita Relatório de Documentos

Fluxos de Dados:

Relatório_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO

Dados_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO



P24: Gerência Solicita Relatório de Lojas

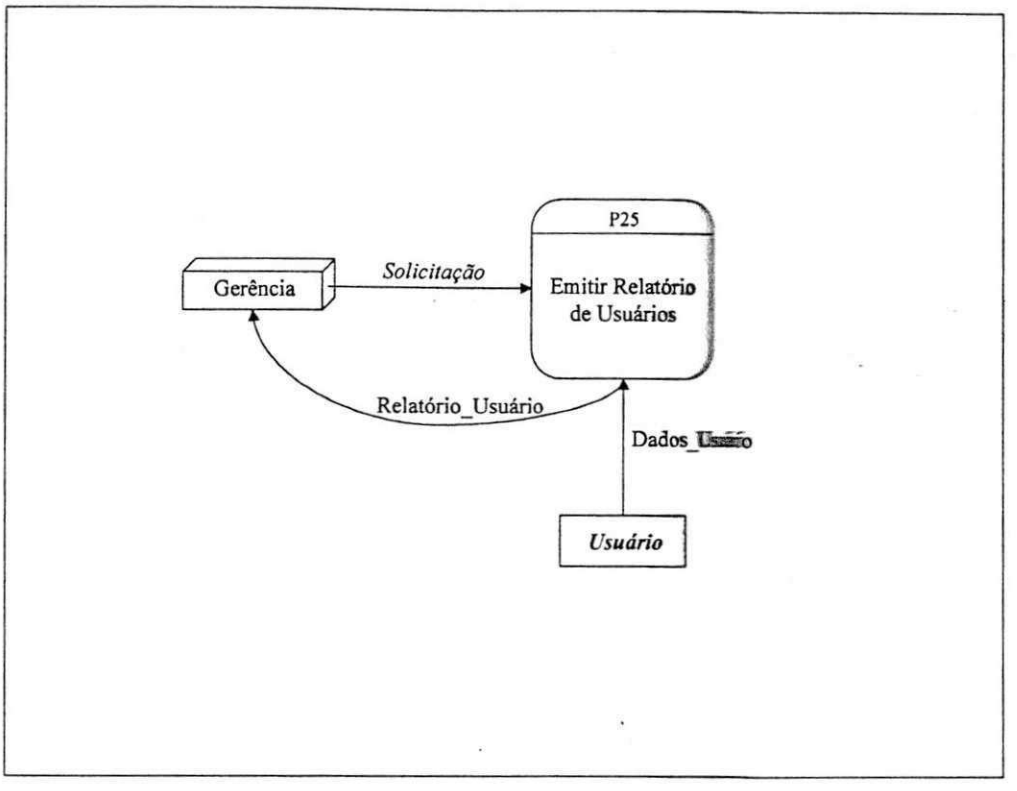
Receba *Solicitação*
Enquanto houver elementos em ***Loja*** faça
    Obtenha @CODIGO_LOJA e Dados_Loja em ***Loja***
Fim enquanto
Exiba Relatório_Loja

P24: Gerência Solicita Relatório de Lojas

Fluxos de Dados:

Relatório_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]

Dados_Loja = LOJA + CNPJ + [FAX] + [TELEFONE] + [BAIRRO] + [CIDADE] + [PAIS] + [EMAIL] + [ESTADO] + [RUA] + [CEP] + [NUMERO] + [INSCRICAO_ESTADUAL]



P25: Gerência Solicita Relatório de Usuários

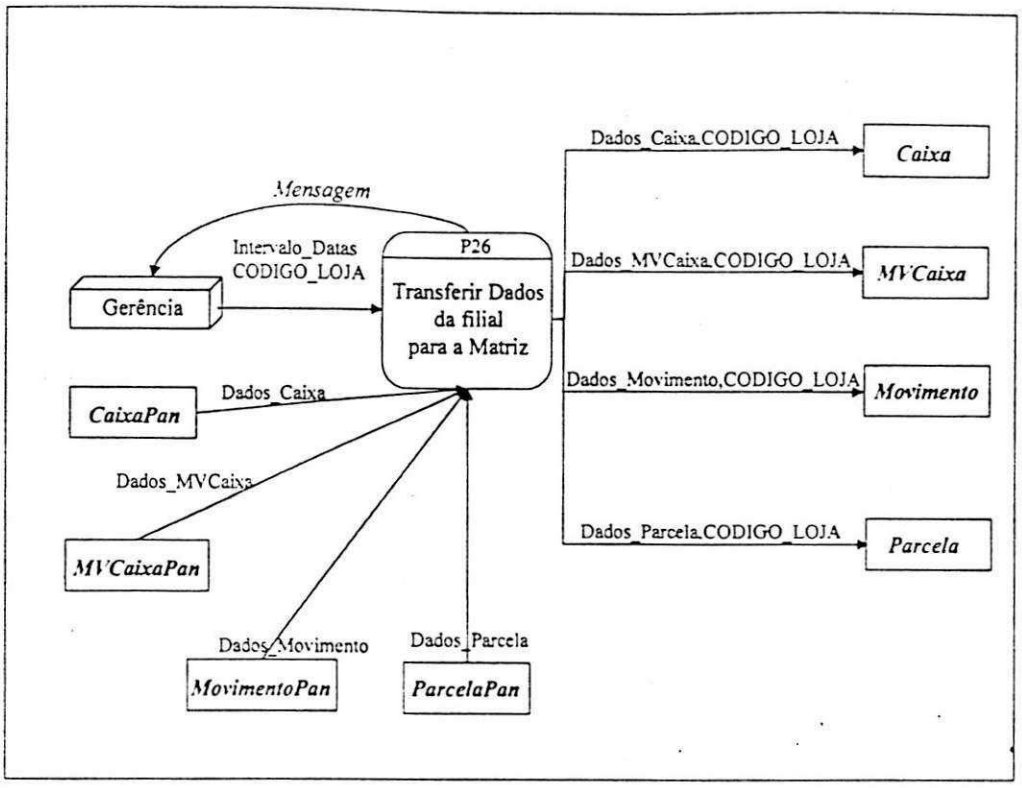
Receba *Solicitação*
Enquanto houver elementos em ***Usuário*** faça
    Obtenha @CODIGO_USUARIO e Dados_Usuário em ***Usuário***
Fim enquanto
Exiba Relatório_Usuário

P25: Gerência Solicita Relatório de Usuários

Fluxos de Dados:

Relatório_Usuário = @CODIGO_USUARIO + USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]

Dados_Usuario = USUARIO + LOGIN + SENHA + [LIBERA] + [EXCLUI]



P26: Gerência Transfere Dados da Filial para a Matriz

Receba Intervalo_Datas, CODIGO_LOJA
Enquanto houver elementos em ***CaixaPan*** onde DATA_INICIO de Dados_Caixa é maior ou igual a Data_Inicial de Intervalo_Datas e DATA_INICIO de Dados_Caixa é menor ou igual a Data_Final de Intervalo_Datas faça
    Obtenha Dados_Caixa em ***CaixaPan***
    Grave Dados_Caixa em Caixa.txt
    Enquanto houver elementos em ***MVCaixaPan*** onde CODIGO_CAIXA de Dados_MVCaixa é igual a @CODIGO_CAIXA de Dados_Caixa faça
        Obtenha Dados_MVCaixa em ***MVCaixaPan***
        Grave Dados_MVCaixa em MVCaixa.txt
        Enquanto houver elementos em ***ParcelaPan*** onde CODIGO_MVCAIXA de Dados_Parcela é igual a @CODIGO_MVCAIXA de Dados_MVCaixa faça
            Obtenha Dados_Parcela em ***ParcelaPan***
            Grave Dados_Parcela em Parcela.txt
        Fim enquanto
    Fim enquanto
Fim enquanto
Enquanto houver elementos em ***MovimentoPan*** onde DATA_MOVIMENTO de Dados_Movimento é maior ou igual a Data_Inicial de Intervalo_Datas e DATA_MOVIMENTO de Dados_Movimento é menor ou igual a Data_Final de Intervalo_Datas faça
    Obtenha Dados_Movimento em ***MovimentoPan***
    Grave Dados_Movimento em Movimento.txt
Fim enquanto

Para cada elemento de Caixa.txt Grave Dados_Caixa + CODIGO_LOJA em ***Caixa***
Para cada elemento de MVCaixa.txt Grave Dados_MVCaixa + CODIGO_LOJA em ***MVCaixa***
Para cada elemento de Parcela.txt Grave Dados_Parcela + CODIGO_LOJA em ***Parcela***
Para cada elemento de Movimento.txt Grave Dados_Movimento + CODIGO_LOJA em ***Movimento***
Imprima "Procedimento de transferência realizado com sucesso."

P26: Gerência Transfere Dados da Filial para a Matriz

Fluxos de Dados:

Intervalo_Datas = DataInicial + DataFinal

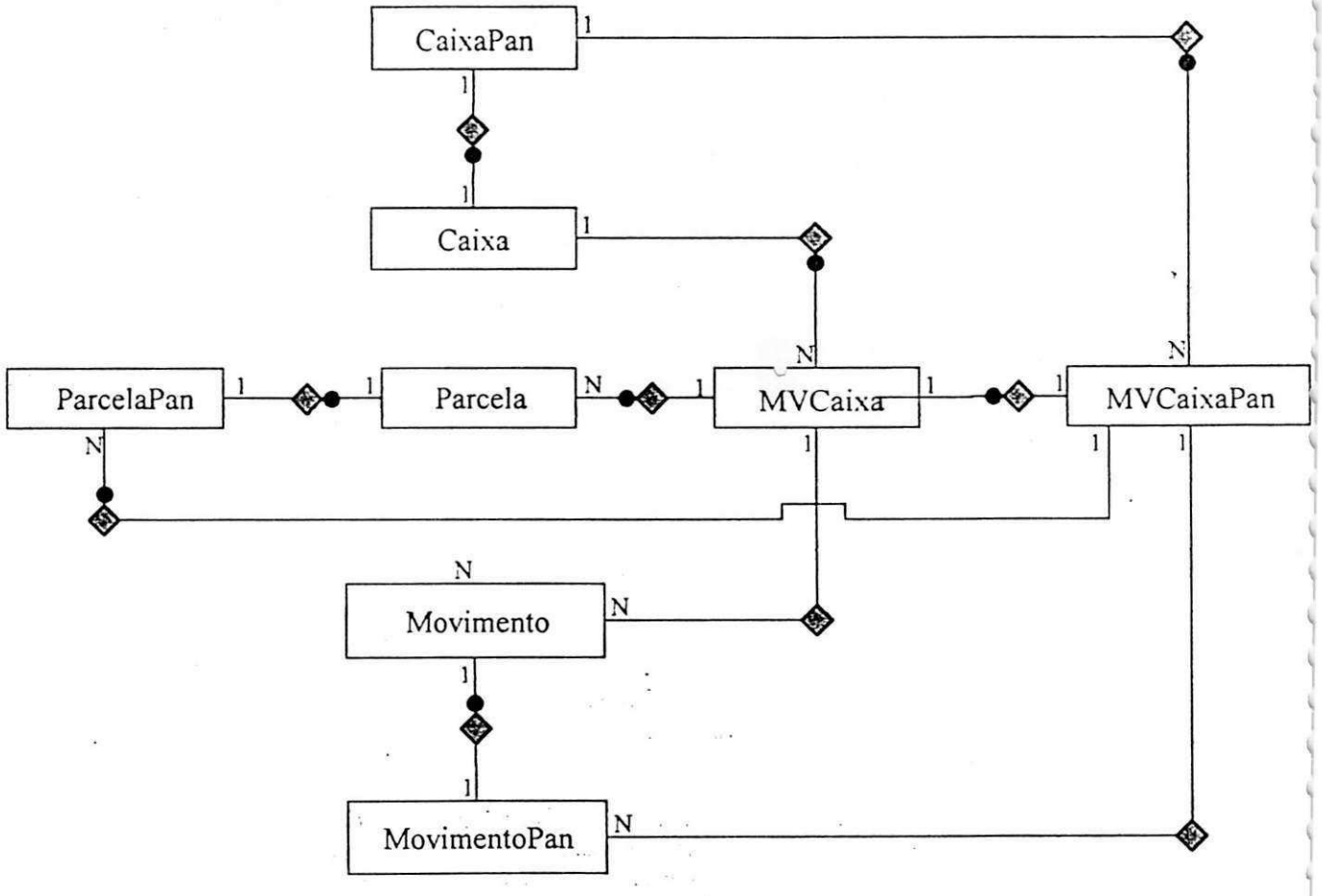
Dados_Caixa = @CODIGO_CAIXA + HORA_INICIO + [HORA_TERMINO] + CAIXA_INICIAL + [CAIXA_FINAL] + TERMINAL + DATA_INICIO + [DATA_TERMINO]

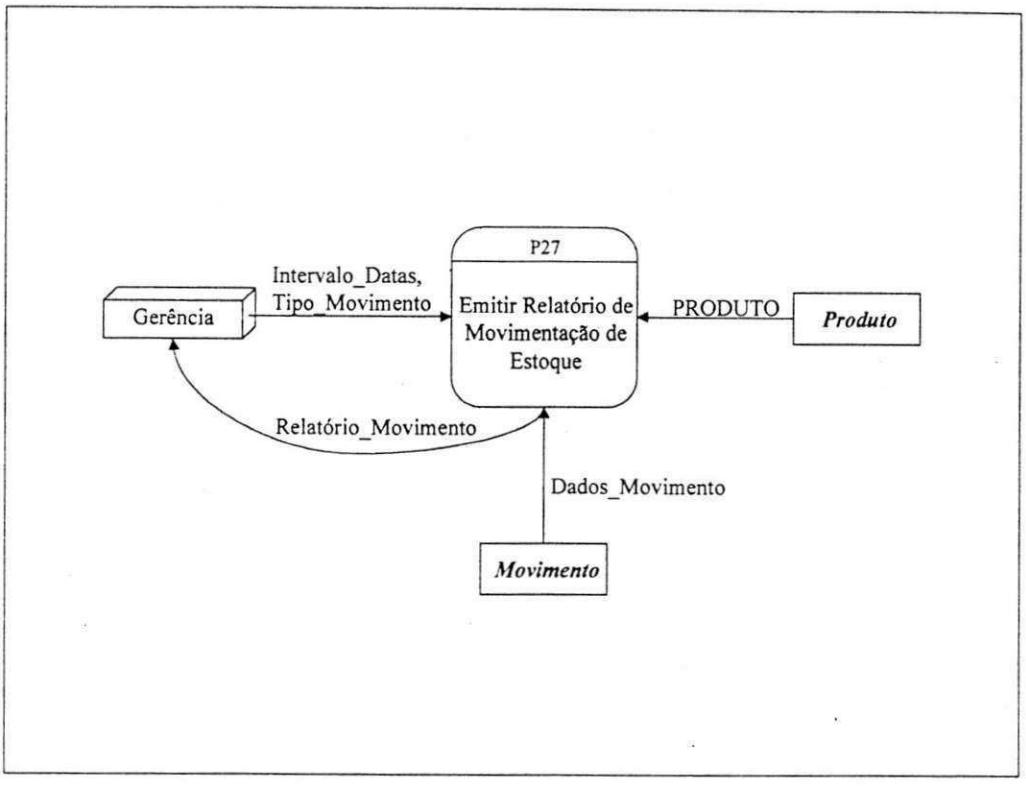
Dados_MVCaixa = @CODIGO_MVCAIXA + DESCONTO + DATA_MVCAIXA + HORA_MVCAIXA + VALOR + CODIGO_CAIXA

Dados_Movimento = @CODIGO_MOVIMENTO + DATA_MOVIMENTO + CODIGO_PRODUTO + QUANTIDADE + VALOR + CODIGO_OPERACAO + [CODIGO_MVCAIXA]

Dados_Parcela = @CODIGO_PARCELA + CODIGO_DOCUMENTO + VALOR_PARCELA + CODIGO_MVCAIXA

D.E.R Parcial:





P27: Gerência Solicita Relatório de Movimentação de Estoque

Receba Intervalo_Datas, Tipo_Movimento
Enquanto houver elementos em ***Movimento*** onde DATA_MOVIMENTO de Dados_Movimento é maior ou igual a Data_Inicial de Intervalo_Datas e DATA_MOVIMENTO de Dados_Movimento é menor ou igual a Data_Final de Intervalo_Datas e CODIGO_OPERACAO de Dados_Movimento está em Tipo_Movimento faça

Obtenha Dados_Movimento em ***Movimento***
Pesquise em ***Produto*** o elemento onde @CODIGO_PRODUTO de Dados_Produto é igual a CODIGO_PRODUTO de Dados_Movimento
Obtenha PRODUTO em ***Produto***

Fim enquanto
Exiba Relatório_Movimento

P27: Gerência Solicita Relatório de Movimentação de Estoque

Fluxos de Dados:

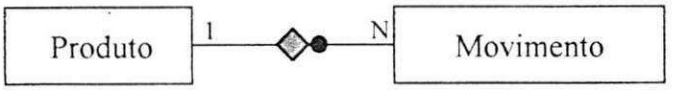
Intervalo_Datas = DataInicial + DataFinal

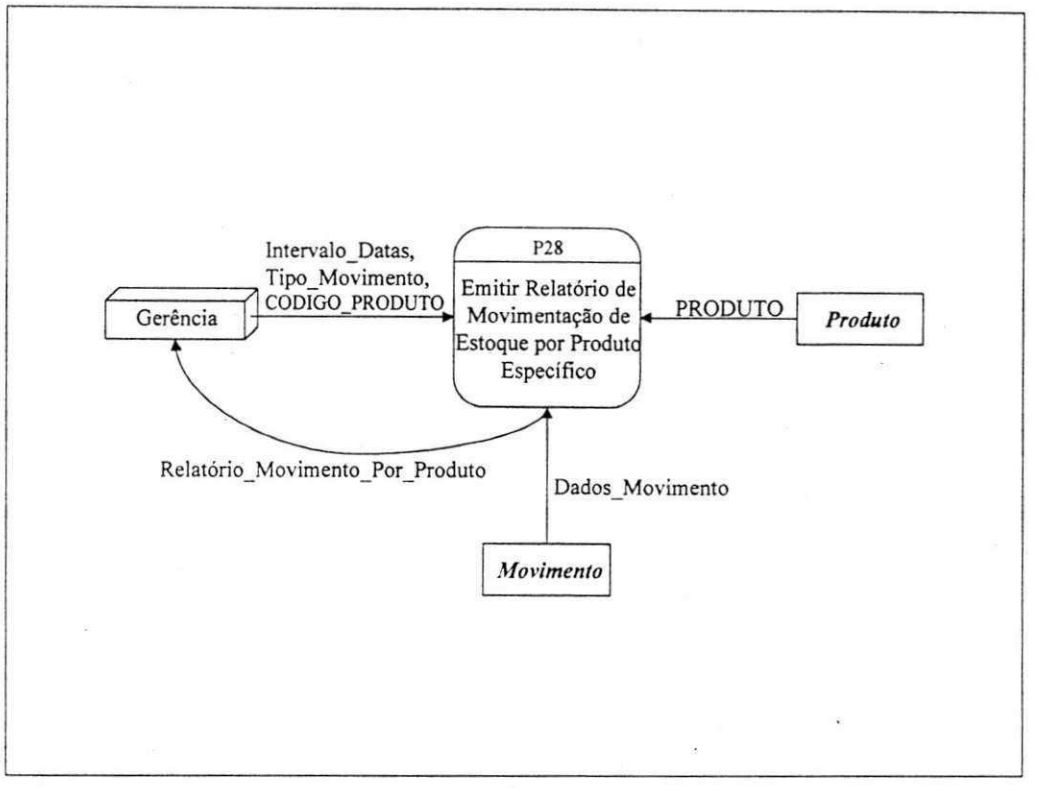
Tipo_Movimento = ["1"] %Entrada por compra% + ["2"] %Saida por venda% + ["3"] %Saida por sobra% + ["4"] %Entrada por sobra% + ["5"] %Saida para produção% + ["6"] %Entrada por produção% + ["7"] %Saida por desperdicio%

Dados_Movimento = @CODIGO_MOVIMENTO + DATA_MOVIMENTO + CODIGO_PRODUTO + QUANTIDADE + VALOR + CODIGO_OPERACAO + [CODIGO_MVCAIXA]

Relatório_Movimento = @CODIGO_MOVIMENTO + CODIGO_PRODUTO + PRODUTO + DATA_MOVIMENTO + QUANTIDADE + VALOR + TOTAL + CODIGO_OPERACAO

D.E.R Parcial:





P28: Gerência Solicita Relatório de Movimentação de Estoque por Produto Específico

Receba Intervalo_Datas, Tipo_Movimento, CODIGO_PRODUTO
Enquanto houver elementos em ***Movimento*** onde DATA_MOVIMENTO de Dados_Movimento é maior ou igual a Data_Inicial de Intervalo_Datas e DATA_MOVIMENTO de Dados_Movimento é menor ou igual a Data_Final de Intervalo_Datas e CODIGO_OPERACAO de Dados_Movimento está em Tipo_Movimento e CODIGO_PRODUTO de Dados_Movimento é igual a CODIGO_PRODUTO faça
    Obtenha Dados_Movimento em ***Movimento***
    Pesquise em ***Produto*** o elemento onde @CODIGO_PRODUTO de Dados_Produto é igual a CODIGO_PRODUTO de Dados_Movimento
        Obtenha PRODUTO em ***Produto***
Fim enquanto
Exiba Relatório_Movimento_Por_Produto

P28: Gerência Solicita Relatório de Movimentação de Estoque por Produto Específico

Fluxos de Dados:

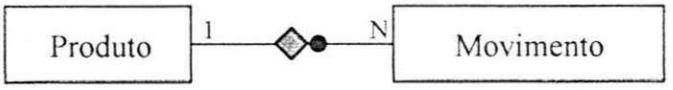
Intervalo_Datas = DataInicial + DataFinal

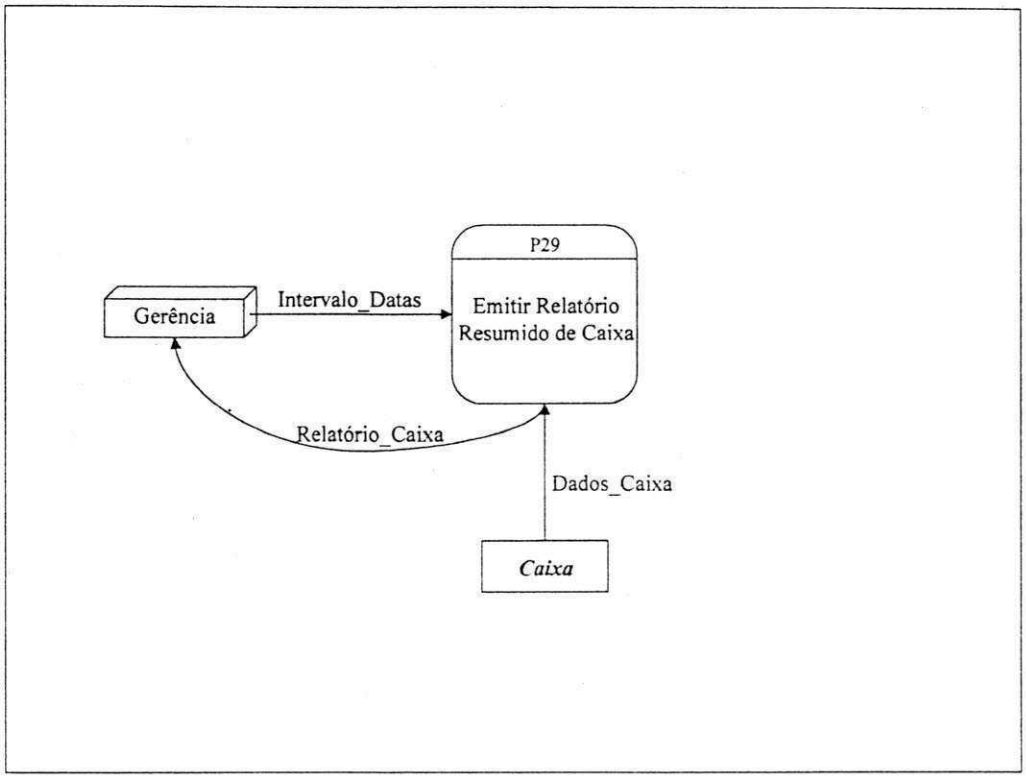
Tipo_Movimento = ["1"] %Entrada por compra% + ["2"] %Saída por venda% + ["3"] %Saída por sobra% + ["4"] %Entrada por sobra% + ["5"] %Saída para produção% + ["6"] %Entrada por produção% + ["7"] %Saída por desperdício%

Dados_Movimento = @CODIGO_MOVIMENTO + DATA_MOVIMENTO + CODIGO_PRODUTO + QUANTIDADE + VALOR + CODIGO_OPERACAO + [CODIGO_MVCAIXA]

Relatório_Movimento_Por_Produto = @CODIGO_MOVIMENTO + CODIGO_PRODUTO + PRODUTO + DATA_MOVIMENTO + QUANTIDADE + VALOR + TOTAL + CODIGO_OPERACAO

D.E.R Parcial:





P29: Gerência Solicita Relatório Resumido de Caixa

Receba Intervalo_Datas
Enquanto houver elementos em ***Caixa*** onde DATA_INICIO de Dados_Caixa é maior ou igual a Data_Inicial de Intervalo_Datas e DATA_INICIO de Dados_Caixa é menor ou igual a Data_Final de Intervalo_Datas faça
    Obtenha Dados_Caixa em ***Caixa***
Fim enquanto
Exiba Relatório_Caixa

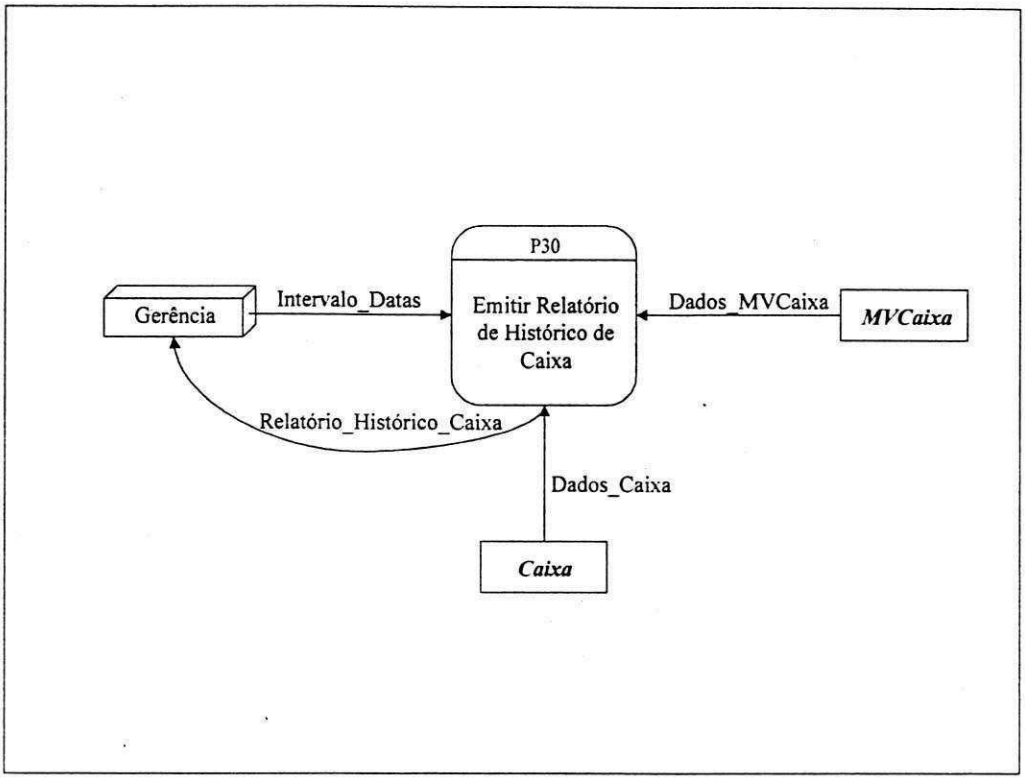
P29: Gerência Solicita Relatório Resumido de Caixa

Fluxos de Dados:

Intervalo_Datas = DataInicial + DataFinal

Dados_Caixa = @CODIGO_CAIXA + HORA_INICIO + [HORA_TERMINO] + CAIXA_INICIAL + [CAIXA_FINAL] + TERMINAL + DATA_INICIO + [DATA_TERMINO]

Relatório_Caixa = @CODIGO_CAIXA + HORA_INICIO + [HORA_TERMINO] + CAIXA_INICIAL + [CAIXA_FINAL] + TERMINAL + DATA_INICIO + [DATA_TERMINO]



P30: Gerência Solicita Relatório de Histórico de Caixa

Receba Intervalo_Datas
Enquanto houver elementos em ***Caixa*** onde DATA_INICIO de Dados_Caixa é maior ou igual a Data_Inicial de Intervalo_Datas e DATA_INICIO de Dados_Caixa é menor ou igual a Data_Final de Intervalo_Datas faça
    Obtenha Dados_Caixa em ***Caixa***
    Obtenha QUANT_DE_CAIXAS Aplicando a Formula
        [QUANT_DE_CAIXAS = QUANT_DE_CAIXAS + 1]
    Obtenha FATURAMENTO Aplicando a Formula
        [FATURAMENTO = FATURAMENTO + CAIXA_FINAL]
    Enquanto houver elementos em ***MVCaixa*** onde CODIGO_CAIXA de Dados_MVCaixa é igual a @CODIGO_CAIXA de Dados_Caixa faça
        Obtenha Dados_MVCaixa em ***MVCaixaPan***
        Obtenha TOTAL_DE_ATENDIMENTOS Aplicando a Formula
            [TOTAL_DE_ATENDIMENTOS = TOTAL_DE_ATENDIMENTOS + 1]
        Obtenha TOTAL_DE_DESCONTOS Aplicando a Formula
            [TOTAL_DE_DESCONTOS = TOTAL_DE_DESCONTOS + ((VALOR * DESCONTO)/100)]
    Fim enquanto
Fim enquanto

Obtenha FATURAMENTO_MEDIO Aplicando a Formula
    [FATURAMENTO_MEDIO = FATURAMENTO / QUANT_DE_CAIXAS]
Obtenha MEDIA_DE_ATENDIMENTOS Aplicando a Formula
    [MEDIA_DE_ATENDIMENTOS = TOTAL_DE_ATENDIMENTOS / QUANT_DE_CAIXAS]
Obtenha MEDIA_DE_DESCONTOS Aplicando a Formula
    [MEDIA_DE_DESCONTOS = TOTAL_DE_DESCONTOS / QUANT_DE_CAIXAS]

Exiba Relatório_Histórico_Caixa

P30: Gerência Solicita Relatório de Histórico de Caixa

Fluxos de Dados:

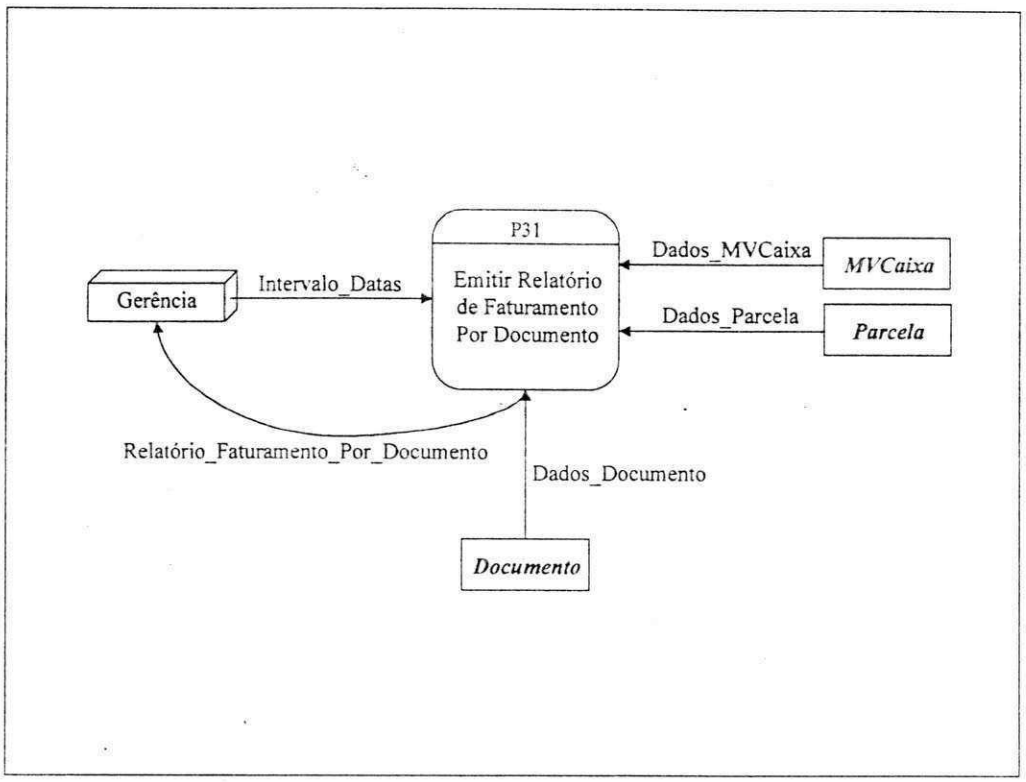
Intervalo_Datas = DataInicial + DataFinal

Dados_Caixa = @CODIGO_CAIXA + HORA_INICIO + [HORA_TERMINO] + CAIXA_INICIAL + [CAIXA_FINAL] + TERMINAL + DATA_INICIO + [DATA_TERMINO]

Dados_MVCaixa = @CODIGO_MVCAIXA + DESCONTO + DATA_MVCAIXA + HORA_MVCAIXA + VALOR + CODIGO_CAIXA

Relatório_Histórico_Caixa = QUANT_DE_CAIXAS + FATURAMENTO + TOTAL_DE_ATENDIMENTOS + TOTAL_DE_DESCONTOS + FATURAMENTO_MEDIO + MEDIA_DE_ATENDIMENTOS + MEDIA_DE_DESCONTOS

D.E.R Parcial:



P31: Emitir Relatório de Faturamento Por Documento

Receba Intervalo_Datas
Pesquise em ***MVCaixa*** elementos onde DATA_MVCAIXA de Dados_MVCaixa é maior ou igual a DataInicial de Intervalo_Datas DATA_MVCAIXA de Dados_MVCaixa é menor ou igual a DataFinal de Intervalo_Datas
Obtenha Dados_MVCaixa em ***MVCaixa***
Para cada elemento em ***Documento*** faça
    Obtenha Dados_Documento em ***Documento***
    Pesquise em ***Parcela*** elementos onde CODIGO_DOCUMENTO de Dados_Parcela é igual a @CODIGO_DOCUMENTO de Dados_Documento e CODIGO_MVCAIXA de Dados_Parcela está em @CODIGO_MVCAIXA de Dados_MVCaixa
    Para cada elemento encontrado faça
        Obtenha TOTAL_DO_DOCUMENTO Aplicando a Formula
            [TOTAL_DO_DOCUMENTO = TOTAL_DO_DOCUMENTO + VALOR_PARCELA]
    Fim para
Fim para
Emitir Relatório_Faturamento_Por_Documento

P31: Gerência Solicita Relatório de Faturamento Por Documento

Fluxos de Dados:

Intervalo_Datas = DataInicial + DataFinal

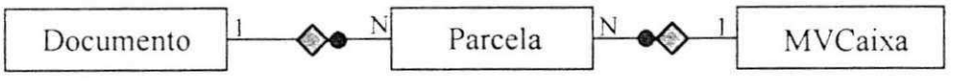
Dados_MVCaixa = @CODIGO_MVCAIXA + DESCONTO + DATA_MVCAIXA + HORA_MVCAIXA + VALOR + CODIGO_CAIXA

Dados_Parcela = @CODIGO_PARCELA + CODIGO_DOCUMENTO + VALOR_PARCELA + CODIGO_MVCAIXA

Dados_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO

Relatório_Faturamento_Por_Documento = @CODIGO_DOCUMENTO + DOCUMENTO + TIPO + TOTAL_DO_DOCUMENTO

D.E.R Parcial:



Anexo 1E

"Script - Banco de Dados da Ferramenta de Integração do Panificador"

```
SET SQL DIALECT 3;

/* CREATE DATABASE 'C:\Bancos de dados\BD Ferramenta de
Integração\IntegracaoPanificador.gdb' PAGE_SIZE 4096

 DEFAULT CHARACTER SET  */

/* Table: CAIXA, Owner: SYSDBA */

CREATE TABLE "CAIXA"
(
  "CODIGO_CAIXA"        INTEGER NOT NULL,
  "CODIGO_LOJA"         SMALLINT NOT NULL,
  "HORA_INICIO"         TIME NOT NULL,
  "CAIXA_INICIAL"       DECIMAL(9, 2) NOT NULL,
  "TERMINAL"            CHAR(15) NOT NULL,
  "DATA_INICIO"         DATE NOT NULL,
  "HORA_TERMINO"        TIME,
  "CAIXA_FINAL"         DECIMAL(9, 2),
  "DATA_TERMINO"        DATE,
 PRIMARY KEY ("CODIGO_CAIXA", "CODIGO_LOJA")
);


/* Table: DOCUMENTO, Owner: SYSDBA */

CREATE TABLE "DOCUMENTO"
(
  "CODIGO_DOCUMENTO"    SMALLINT NOT NULL,
  "TIPO"                CHAR(1) NOT NULL,
  "DOCUMENTO"           CHAR(40) NOT NULL,
 PRIMARY KEY ("CODIGO_DOCUMENTO")
);


/* Table: GENERO, Owner: SYSDBA */

CREATE TABLE "GENERO"
(
  "CODIGO_GENERO"       SMALLINT NOT NULL,
  "GENERO"              CHAR(20) NOT NULL,
 PRIMARY KEY ("CODIGO_GENERO")
);


/* Table: LOJA, Owner: SYSDBA */

CREATE TABLE "LOJA"
(
  "CODIGO_LOJA"         SMALLINT NOT NULL,
  "LOJA"                CHAR(40) NOT NULL,
  "CNPJ"                CHAR(14) NOT NULL,
  "FAX"                 CHAR(14),
  "TELEFONE"            CHAR(14),
```

```
  "BAIRRO"              CHAR(25),
  "CIDADE"              CHAR(30),
  "PAIS"                CHAR(20),
  "EMAIL"               CHAR(30),
  "ESTADO"              CHAR(20),
  "RUA"                 CHAR(35),
  "CEP"                 CHAR(8),
  "NUMERO"              CHAR(6),
  "INSCRICAO_ESTADUAL"  CHAR(20),
 PRIMARY KEY ("CODIGO_LOJA")
);


/* Table: MOVIMENTO, Owner: SYSDBA */

CREATE TABLE "MOVIMENTO"
(
  "CODIGO_MOVIMENTO"    INTEGER NOT NULL,
  "CODIGO_LOJA"         SMALLINT NOT NULL,
  "DATA_MOVIMENTO"      DATE NOT NULL,
  "CODIGO_PRODUTO"      SMALLINT NOT NULL,
  "QUANTIDADE"          DECIMAL(9, 2) NOT NULL,
  "VALOR"               DECIMAL(9, 2) NOT NULL,
  "CODIGO_OPERACAO"     INTEGER NOT NULL,
  "CODIGO_MVCAIXA"      INTEGER,
 PRIMARY KEY ("CODIGO_MOVIMENTO", "CODIGO_LOJA")
);


/* Table: MVCAIXA, Owner: SYSDBA */

CREATE TABLE "MVCAIXA"
(
  "CODIGO_MVCAIXA"      INTEGER NOT NULL,
  "CODIGO_LOJA"         SMALLINT NOT NULL,
  "DESCONTO"            DECIMAL(5, 2) NOT NULL,
  "DATA_MVCAIXA"        DATE NOT NULL,
  "HORA_MVCAIXA"        TIME NOT NULL,
  "VALOR"               DECIMAL(9, 2) NOT NULL,
  "CODIGO_CAIXA"        INTEGER NOT NULL,
 PRIMARY KEY ("CODIGO_MVCAIXA", "CODIGO_LOJA")
);


/* Table: PARCELA, Owner: SYSDBA */

CREATE TABLE "PARCELA"
(
  "CODIGO_PARCELA"      INTEGER NOT NULL,
  "CODIGO_LOJA"         SMALLINT NOT NULL,
  "CODIGO_DOCUMENTO"    SMALLINT NOT NULL,
  "VALOR_PARCELA"       DECIMAL(9, 2) NOT NULL,
  "CODIGO_MVCAIXA"      INTEGER NOT NULL,
 PRIMARY KEY ("CODIGO_PARCELA", "CODIGO_LOJA")
);
```

```
/* Table: PERMISSAO, Owner: SYSDBA */

CREATE TABLE "PERMISSAO"
(
  "CODIGO"              SMALLINT NOT NULL,
  "USUARIO"             SMALLINT NOT NULL,
  "PERMISSAO"           CHAR(30) NOT NULL,
 PRIMARY KEY ("CODIGO")
);


/* Table: PRODUTO, Owner: SYSDBA */

CREATE TABLE "PRODUTO"
(
  "CODIGO_PRODUTO"      SMALLINT NOT NULL,
  "UNIDADE"             CHAR(2) NOT NULL,
  "CODIGO_GENERO"       SMALLINT NOT NULL,
  "PRODUTO"             CHAR(30) NOT NULL,
  "ALIQUOTA_ICMS"       DECIMAL(5, 2) NOT NULL,
  "ALIQUOTA_IPI"        DECIMAL(5, 2) NOT NULL,
  "CODIGO_BARRAS"       CHAR(13) NOT NULL,
  "TIPO"                CHAR(20) NOT NULL,
  "SITUACAO_TRIBUTARIA" CHAR(1),
 PRIMARY KEY ("CODIGO_PRODUTO")
);


/* Table: USUARIO, Owner: SYSDBA */

CREATE TABLE "USUARIO"
(
  "CODIGO"              SMALLINT NOT NULL,
  "NOME"                CHAR(35) NOT NULL,
  "LOGIN"               CHAR(10) NOT NULL,
  "SENHA"               CHAR(10) NOT NULL,
  "LIBERA"              CHAR(1),
  "EXCLUI"              CHAR(1),
 PRIMARY KEY ("CODIGO")
);


ALTER TABLE "CAIXA" ADD CONSTRAINT "FK_CAIXA_LOJA" FOREIGN KEY ("CODIGO_LOJA")
REFERENCES LOJA ("CODIGO_LOJA");

ALTER TABLE "MOVIMENTO" ADD CONSTRAINT "FK_MOVIMENTO_MVCAIXA" FOREIGN KEY
("CODIGO_MVCAIXA", "CODIGO_LOJA") REFERENCES MVCAIXA ("CODIGO_MVCAIXA",
"CODIGO_LOJA");

ALTER TABLE "MOVIMENTO" ADD CONSTRAINT "FK_MOVIMENTO_PRODUTO" FOREIGN KEY
("CODIGO_PRODUTO") REFERENCES PRODUTO ("CODIGO_PRODUTO");

ALTER TABLE "MVCAIXA" ADD CONSTRAINT "FK_MVCAIXA_CAIXA" FOREIGN KEY
("CODIGO_CAIXA", "CODIGO_LOJA") REFERENCES CAIXA ("CODIGO_CAIXA",
"CODIGO_LOJA");
```

```
ALTER TABLE "PARCELA" ADD CONSTRAINT "FK_PARCELA_DOCUMENTO" FOREIGN KEY
("CODIGO_DOCUMENTO") REFERENCES DOCUMENTO ("CODIGO_DOCUMENTO");

ALTER TABLE "PARCELA" ADD CONSTRAINT "FK_PARCELA_LOJA" FOREIGN KEY
("CODIGO_LOJA") REFERENCES LOJA ("CODIGO_LOJA");

ALTER TABLE "PARCELA" ADD CONSTRAINT "FK_PARCELA_MVCAIXA" FOREIGN KEY
("CODIGO_MVCAIXA", "CODIGO_LOJA") REFERENCES MVCAIXA ("CODIGO_MVCAIXA",
"CODIGO_LOJA");

ALTER TABLE "PERMISSAO" ADD CONSTRAINT "FK_PERMISSAO_USUARIO" FOREIGN KEY
("USUARIO") REFERENCES USUARIO ("CODIGO");

ALTER TABLE "PRODUTO" ADD CONSTRAINT "FK_PRODUTO_GENERO" FOREIGN KEY
("CODIGO_GENERO") REFERENCES GENERO ("CODIGO_GENERO");
```

Refatoramento do Panificador V 1.0

1 Introdução

1.1 Problema encontrado

A ByteCom Sistemas Ltda., empresa incubada no Núcleo SOFTEX GENESIS de Campina Grande - POLIGENE, desenvolveu um software chamado Panificador. Destinado ao gerenciamento de panificadoras e afins, este sistema está agora em sua primeira versão e já vem sendo comercializado localmente. O tempo de desenvolvimento total foi de aproximadamente 30 (trinta) meses e durante este período houve algumas substituições de membros da equipe de desenvolvimento. Além disso, no inicio do projeto do software, os membros da equipe de desenvolvimento tinham pouca experiência neste tipo de atividade. Estes aspectos levaram à construção de um código fonte frágil e, muitas vezes, confuso, que precisou ser refeito em sua grande maioria no início do segundo ano do projeto.

Apesar de apresentar-se, atualmente, relativamente padronizado e bem funcional, o sistema vem apresentando a necessidade de melhorias no código fonte. Para resolver este problema optou-se pelo processo de refatoramento. Não muito difundido até o momento, este processo objetiva basicamente alterar o código fonte pronto e funcionando para torná-la mais facilmente legível e suscetível a modificações e adição de novas funcionalidades, sem alterar, entretanto, o seu comportamento. Realizar este processo, seguindo a metodologia proposta por Martin Fowler, é o objetivo desta parte do meu estágio supervisionado.

1.2 Revisão Bibliográfica

1.2.1 Refatatoramento - Definição

Esta técnica surgiu na comunidade Smalltalk, quando Kent Beck e Ward Cunningham iniciaram a produção de um desenvolvimento de software mais adequado à linguagem. Durante estes estudos, perceberam como o refatoramento ajudava a aumentar sua produtividade[Fowler, 2000]. Posteriormente tal processo ficou conhecido como Extreme Programming. Desde então o refatoramento se tornou muito popular entre os

programadores de Smalltalk, porém, com os estudos de William Opdyke sobre sua aplicação no desenvolvimento de frameworks em C++, passou a ser conhecido e utilizado pelos programadores de diversas linguagens orientadas a objeto[Fowler, 2000].

O objetivo do refatoramento é realizar mudanças no código pronto, e funcionando, para torná-lo mais legível e compreensível. Além disto, se apóia em técnicas bem planejadas para tornar estas mudanças seguras e eficientes. Assim, refatoramento deve ser entendido como um conjunto de técnicas bem planejadas que têm o objetivo de melhorar a legibilidade e facilitar a compreensão de um trecho de código.

Martin Fowler, em seu livro "Refactoring Improving the Design of Existing Code", utiliza duas definições para melhorar o entendimento, veja abaixo.

"Refatoramento (substantivo): uma mudança feita na estrutura interna do software para torná-lo mais fácil de entender e barato de modificar sem mudar o seu comportamento."

"Refatorar (verbo): reestruturar um software através da aplicação de uma série de refatoramentos sem mudar o seu comportamento."

1.2.2 Porque Refatorar?

1.2.2.1 Refatoramento ajuda a preservar o projeto do software

No ciclo de vida de um software, mudanças são feitas ao longo do tempo por diversas pessoas. Muitas destas mudanças têm o objetivo de adicionar funcionalidades que não foram previstas na análise e consequentemente não foram contempladas no projeto. Os programadores responsáveis por realizar tais atualizações provavelmente não compreendem o projeto como um todo. Isto os leva a realizar alterações que contribuem para a perda da estrutura do código. Com o uso constante do refatoramento este efeito pode ser diminuído já que as mudanças feitas com o uso da técnica têm o objetivo principal de tornar o código mais compreensível[Fowler, 2000].

Permite corrigir erros de projeto depois do software estar funcionando

Através do refatoramento do software é possível introduzir padrões de projeto com o código já funcionando e também corrigir erros de reuso. É possível que existam decisões tomadas na fase de projeto que sejam tão centrais no funcionamento do software que dificilmente podem ser modificadas, mas estas provavelmente não são maioria em um sistema de software[Fowler, 2000].

1.2.2.2 Ajuda a encontrar erros

Com o código mais fácil de compreender e, consequentemente, seus detalhes mais aparentes, erros cometidos durante a programação ficarão mais claros. Refatoramento ajuda a clarear a estrutura do código e, por isso, a identificação de erros fica mais fácil.

1.2.2.3 Aumenta a qualidade e, enfim, a produtividade

Como fechamento desta fase de análise das vantagens e benefícios trazidos com o uso do refatoramento, é fácil concluir que esta é uma técnica muito importante na qualidade do código produzido. Somando-se o que fora visto acima, melhoras no projeto, mesmo depois do software pronto, mais facilidade na compreensão, mais fácil encontrar erros, etc, é possível afirmar que a técnica realmente ajuda na melhora do código produzido e ajuda a combater a Entropia do Software. Também é fácil perceber que, com um código mais compreensível e fácil de manter, teremos atualizações e correções de erros implementadas mais rapidamente e assim, um ganho na produtividade das pessoas envolvidas[Fowler, 2000].

1.2.3 Como refatorar?

1.2.3.1 Requisitos para o refatoramento

Existem duas regras essenciais que devem ser seguidas para que o refatoramento possa ser corretamente aplicado. É fortemente recomendado que nenhum destes seja desprezado.

- Não adicione funcionalidade e realize refatoramento ao mesmo tempo, são duas etapas completamente distintas;
- Tenha testes automáticos.

Estas regras são fundamentais para que se obtenham os resultados esperados com a aplicação de técnicas de refatoramento. Refatoramento é aplicado sobre código que funciona, ao adicionar funcionalidade não se tem código funcionando ainda, consequentemente não se têm testes. A adição de funcionalidades é uma etapa completamente distinta do refatoramento, que deve ser feita antes ou depois ou ainda antes e depois, mas jamais durante[Fowler, 2000]. A razão básica é que não será possível certificar-se que as mudanças efetuadas pelo refatoramento foram eficazes.

Para se mexer em código funcionando, deve se ter um meio para certificar-se de que ele continua funcionando, deve-se certificar que nenhum erro foi introduzido. Para isto, são necessários alguns testes que realizem esta verificação automaticamente. Pois quando a menor mudança for realizada no software, os testes indicarão se tudo continua funcionando de acordo com o esperado.

1.2.4 Quando refatorar?

Outro aspecto importante da técnica de refatoramento é quando ela deve ser utilizada e aplicada. Para responder a esta pergunta, deve-se responder a outra: quando um código deve ser modificado para melhorar sua legibilidade?[Fowler, 2000] Muito embora sejam várias e provavelmente válidas as respostas a esta pergunta, lista-se abaixo algumas situações interessantes para o uso de refatoramento.

1.2.4.1 Quando parece difícil adicionar uma nova funcionalidade

Sempre que é necessário adicionar uma nova característica a um software e esta adição leva a repetição de código em várias partes do sistema, deve-se refatorar, ou reorganizar, o código existente antes de implementar a nova funcionalidade[Fowler, 2000].

É certo que mais trabalho será preciso para organizar esta parte do sistema que se encontra espalhada, porém a adição desta nova função e de outras, que por ventura venham a ser necessárias, será mais rápida e fácil. Além disto, diminui-se o risco de introdução de erros no sistema.

1.2.4.2 Quando não é possível compreender o código

Para realizar alguma modificação ou atualização em um software, é preciso compreendê-lo para que a alteração não quebre a estrutura do código ou introduza comportamentos inesperados. Utilizar refatoramento para compreender um trecho código é uma prática muito eficaz, pois qualquer modificação indevida será apresentada nos testes efetuados após cada refatoramento[Fowler, 2000].

1.2.4.3 Após alguma alteração ou produção de um novo trecho de código

Na fase de programação dificilmente o programador está preocupado em quem irá manter o código que ele produz. A sua preocupação maior é ter sua tarefa completada em tempo hábil e de acordo com os cronogramas. Este mesmo programador não irá pensar na legibilidade daquilo que está produzindo. Somado a isto, tem-se o fato de que é muito raro que um programador acerte a melhor e mais legível forma de resolver determinado problema na primeira tentativa. Por estas razões, é fácil concluir que é de fundamental importância que ao produzir um novo trecho de código ou ao alterar um código existente, o programador seja levado também a realizar a fase de refatoramento do código. É mais fácil enxergar problemas de projeto ou de estrutura depois que o software está pronto e funcionando[Opdyke, 1999].

1.2.5 Refatoramento em Larga Escala

As técnicas de refatoramento apresentadas até hoje são pequenas mudanças, muito pontuais, que devem ser aplicadas em partes do código. Estas mudanças são úteis, mas não resolvem o problema de software de médio a grande porte, que teve seu desenvolvimento

espalhado durante anos por diversos programadores. Não existem técnicas bem planejadas e organizadas de como refatorar um software como um todo.

Como separa a interface dos objetos de negócio? Como separar aquilo que é dependente de sistema operacional do que não é? Estes são exemplos de mudanças que exigiriam a reorganização do software como um todo e que poderiam gerar técnicas de refatoramento. Porém ainda não existem refatoramentos neste nível.

1.2.6 Formato

Um refatoramento é composto pelas seguintes partes:

- Um **nome**, esta parte é importante para facilitar a comunicação entre as pessoas que discutam o tema[Fowler, 2000].
- Um pequeno **resumo** descrevendo a situação na qual este refatoramento é necessário e o que ele faz[Fowler, 2000].
- A **motivação** descreve as razões que fazem este refatoramento útil e necessário e as circunstâncias sob as quais a sua aplicação é desencorajada.
- O **mecanismo** descreve os passos que devem ser seguidos para que o refatoramento possa ser aplicado de forma mais segura.
- Os **exemplos** apresentam um uso simples deste refatoramento com o objetivo de mostrar como ele funciona.

1.2.7 Exemplo

O objetivo aqui é fornecer uma visão prática da técnica apresentada. Abaixo se apresenta o refatoramento *Extract Method* tal como é apresentado no catálogo produzido por Martin Fowler. Este é um exemplo simples e não tem a pretensão de ilustrar toda as vantagens que podem ser obtidas com o uso constante da técnica.

Extract Method

Você tem um fragmento de código que pode ser agrupado.

Torne o fragmento um método separado e dê um nome que reflita o seu propósito.

```
void printOwing (double sal){
        printBanner();  //imprime detalhes
        System.out.println ("Nome:" + _nome);
        System.out.println("Salário:" + sal);
}
```

O trecho acima, após a aplicação do refatoramento se torna o seguinte:

```
void printOwing (double sal){
        printBanner();
        printDetails(sal);
}
 void printDetails(double salario){
        System.out.println ("Nome:" + _nome);
        System.out.println("Salário:" + salario);
}
```

Motivação

Este é um dos refatoramentos mais comumente utilizados. Este refatoramento deve ser utilizado quando um método é muito longo e são necessários muitos comentários para explicar o que ele faz[Fowler, 2000].

Mecanismo

Crie um novo método

Copie o código que deve compor o novo método

Procure, neste código, referências a variáveis que são locais ao escopo do método de origem.

Se algumas dessas variáveis são temporárias e apenas utilizadas no código do novo método, elas devem se tornar variáveis locais ao novo método.

Caso alguma dessas variáveis é modificada dentro do código em questão e utilizada fora dele, faça o novo método retornar este valor. Caso mais de uma variável se encaixe nesta situação, utilize o refatoramento *Split Temporary Variable* e tente novamente.

Passe como parâmetro para o novo método as variáveis locai que são lidas no código extraído.

Compile quando todas as variáveis locais estiverem resolvidas de acordo com o descrito acima.

Substitua o trecho de código na origem pela invocação do novo método

Compile teste

Exemplo

Na maioria dos casos, a aplicação deste refatoramento não é problemática[Fowler, 2000], de modo que um exemplo mais extenso será omitido nesta apresentação.

2 Desenvolvimento

2.1 Atividades Desenvolvidas

A seguir serão descritas, passo a passo, todas as atividades realizadas no processo de elaboração do relatório de refatoramento do software Panificador. Uma vez elaborado este relatório, a ByteCom Sistemas terá a posse de um documento com todas as instruções de como efetuar melhorias no código fonte do sistema citado, além de indicações de pontos críticos que devem ser alterados com maior urgência. Os passos para a elaboração deste documento foram:

- **Passo 1:** Inicialmente foram escolhidas, através de alguns critérios simples de seleção, algumas Units do projeto para que estas fossem refatoradas. Os critérios adotados para esta seleção foram os seguintes:

  1. Units relativas à partes do sistema de uso crítico, como as units correspondentes à parte de frente de loja (atendimento ao cliente);
  2. Units cuja última data de alteração era muito antiga. Em geral, estas units apresentavam código menos organizado, padronizado e legível;
  3. Units que, após analisadas, pudessem servir como modelo para o refatoramento de outras Units do projeto que não seriam analisadas. Muitas Units de cadastro, por exemplo, possuem características bem semelhantes;

- **Passo 2:** Para cada uma das Units selecionadas foi feita uma análise inicial, durante a qual foram listadas, textualmente, deficiências encontradas no código fonte. Para cada problema encontrado foram armazenadas também a linha ou grupo de linhas de código onde os problemas ocorrem. Isto possibilitará que a correção dos problemas seja feita com maior facilidade, pois será possível saber exatamente que problemas foram encontrados e onde os mesmos ocorrem;

- **Passo 3:** Após criada a listagem descrita no passo 2, foi feita uma classificação de cada erro encontrado por grupo de erro. Ao final foram encontrados 9 (nove)

grupos de erros que englobavam todos os erros encontrados nas Units analisadas.

- **Passo 4:** Uma vez de posse da listagem de erros e dos grupos de erros bem definidos foi criada uma uma matriz de Units X Grupos de erros. O principal objetivo desta matriz é fornecer uma visualização prática e rápida do estado de cada Unit, uma vez que toda informação gerada até então era completamente textual.

- **Passo 5:** Finalmente, para cada grupo de erros encontrado foi criado um refatoramento com a estrutura proposta por Martin Fowler em seu livro "Improving the Design of Existing Code". Tal estrutura encontra-se descrita a seguir:

  - **Nome**: parte muito importante pois identifica unicamente um refatoramento e já passa para os membros da equipe de desenvolvimento indicações do erro encontrado e do procedimento necessário para corrigi-lo;
  - **Resumo:** descreve rapidamente para que tipo de problema o refatoramento é indicado e como o mesmo deve ser aplicado;
  - **Motivação:** descreve as razões que fazem este refatoramento útil e necessário e as circunstâncias sob as quais a sua aplicação é desencorajada;
  - **Mecanismo:** descreve, detalhadamente, os passos que devem ser seguidos para que a aplicação do refatoramento possa ser feita de forma segura e o mais simplificada possível. O mecanismo é a parte mais importante de qualquer refatoramento, pois descreve como realizar as mudanças no código fonte;
  - **Exemplos:** apresentam casos simples de uso do refatoramento com o objetivo de mostrar, na prática, como ele funciona e quais as vantagens obtidas com sua aplicação;

Os documentos gerados em cada um dos passos descritos encontram-se em anexo.

Após os 5 (cinco) passos citados acima, foi concluída a elaboração do relatório de refatoramento. O relatório é composto de todos os documentos gerados em cada passo e servirá como guia para que a equipe de desenvolvimento possa efetuar as modificações sugeridas no código fonte. Além disso, a lista de refatoramentos gerada no passo 5 serve como um guia de boas práticas de programação. Um dos objetivos fundamentais é fazer com que os erros encontrados no código fonte durante o refatoramento não se repitam mais dali por diante, quando código fonte novo for gerado. O ideal é que o próximo refatoramento realizado encontre erros diferentes dos encontrados nesta primeira aplicação do refatoramento. Assim, estes novos refatoramentos gerados serão integrados ao guia de boas práticas de programação. A cada repetição deste processo a empresa atingirá um nível de maturidade mais alto no que diz respeito à técnicas e práticas de programação, e isto fará com que o código fonte gerado tenha cada vez mais qualidade.

A aplicação periódica do refatoramento traz para uma empresa de desenvolvimento de software os seguintes resultados:

- Preservação dos projetos de software;
- Ajuda a encontrar erros no código fonte;
- Aumenta a produtividade da equipe de desenvolvimento;
- Aumenta o tempo de vida útil do sistema;

3 Conclusão

O estágio supervisionado é uma atividade bastante importante e necessária para um aluno concluínte de um curso universitário. É importante porque o aluno tem a oportunidade de colocar em prática os seus conhecimentos adquiridos durante sua vida acadêmica e com isso ganhar experiência profissional. E é necessário porque o aluno tem a possibilidade de conhecer suas deficiências e suas qualidades profissionais, tentando, assim, aperfeiçoar-se profissionalmente.

Por se tratar de uma prática não muito difundida, principalmente no meio acadêmico, encontrei várias dificuldades na confecção do relatório de refatoramento. Este assunto, embora bastante atual e importante, não fora abordado por nenhuma das disciplinas do curso. Devido a isto, fui obrigado a começar a estudar o assunto durante o período do estágio supervisionado. Outra dificuldade enfrentada foi a escassez de literatura acerca do assunto. Apenas um livro especificamente voltado para o refatoramento foi consultado durante o estágio. Embora este livro tenha servido como base de referência, foram consultadas diversas páginas na Internet que tratavam do assunto.

Gostei muito de realizar este trabalho. Embora tenha sido uma experiência extremamente nova, consegui observar muitas vantagens que podem ser obtidas com a prática do refatoramento. Creio que o assunto deveria ser abordado nas disciplinas do curso de Ciência da Computação, sobretudo nas disciplinas relativas à Engenharia de Software. Após a realização deste trabalho, tenho a certeza de ter alcançado o maior dos objetivos de um estágio supervisionado: o crescimento profissional. Hoje vejo o refatoramento como uma prática fundamental no desenvolvimento de software, e tenho planos de realizá-lo com freqüência nos projetos em que eu estiver engajado daqui por diante.

4 Referências Bibliográficas

[Fowler, 1999] Refactoring - Improving The Design of Existing Code, Martin Fowler, Editora Booch Jacobson Rumbaugh

Diversas páginas na Intenet

Anexo 2A

- **UnitAdiantamento** (149 linhas de código - Ultima Atualização: 11/2000)

1. Componentes ainda com nomenclatura original;
2. Linhas 85 à 91 deveriam ter tratamento preventivo de erros (tratamento de exceções);
3. Linhas 109 à 129 deveriam ter tratamento preventivo de erros (tratamento de exceções);
4. Falta comentário no código;
5. Linhas 109 à 129: trecho de código poderia estar separado em um procedimento de impressão (p. ex. ImprimeRecibo);

- **UnitComanda**. (829 linhas de código - Ultima Atualização: 03/2000)

1. Linha13: definição de tipo com nome que descaracteriza o mesmo;
2. Componentes ainda com nomenclatura original;
3. Funções e procedimentos de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
4. Linhas 215 à 229 deveriam ter tratamento preventivo de erros (tratamento de exceções);
5. Linhas 153 à 163 deveriam ter tratamento preventivo de erros (tratamento de exceções);
6. Linha de comando 173 em lugar inapropriado – depois do close do form;
7. Linha 238: variável local definida com nome pouco significativo ou pouco auto-explicativo;
8. Linha 257: expressão de condição do IF não cobre todos os casos que deveria cobrir;
9. Linha 291: variáveis locais definidas com nomes pouco significativos ou pouco auto-explicativos;
10. Linhas 307 à 317: trecho de código está confuso e deveria estar separado em um outro procedimento (VerificaLimiteQuantidade);
11. Linha 288: procedimento muito grande e com boa parte do código se repetindo. Código repetido deveria estar em um procedimento separado;
12. Linhas 288 à 370 deveriam ter tratamento preventivo de erros (tratamento de exceções);
13. Linhas 480 à 603: procedimentos para manipulação da lista encadeada deveriam estar em uma Unit à parte, pois são usados em outras Units onde também estão se repetindo. Reaproveitamento de código pode ser muito melhorado;
14. Faltam comentários no código de uma maneira geral;
15. Linhas 688 e 763 deveriam ser substituídas por procedimentos com nomes que tornassem o entendimento do código mais simplificado;

- **UnitAlteracaoPrecos** (393 linhas de código - Ultima Atualização: 09/2000)

1. Funções e procedimentos de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
2. Linha 93: procedimento com boa parte do código se repetindo. Código repetido deveria estar em um procedimento separado;
3. Linhas 157 à 161 deveriam ter tratamento preventivo de erros (tratamento de exceções);
4. Linhas 190 à 200 deveriam ter tratamento preventivo de erros (tratamento de exceções);
5. Linhas 209 à 219 deveriam ter tratamento preventivo de erros (tratamento de exceções);
6. Linhas 228 à 237 deveriam ter tratamento preventivo de erros (tratamento de exceções);
7. Linhas 245 à 253 deveriam ter tratamento preventivo de erros (tratamento de exceções);

8. Linha 285: variável local definida com nome pouco significativo ou pouco auto-explicativo;
9. Linha 283: procedimento com número de linhas excessivo.
10. Falta comentários no código de uma maneira geral. Foram colocados apenas indicações do autor e da data. Pequenos trechos possuem comentários.

- **UnitAlterarSenha** (105 linhas de código - Ultima Atualização: indefinido)

1. Linhas 67 à 70 deveriam ter tratamento preventivo de erros (tratamento de exceções);
2. Falta comentários no código de uma maneira geral;

- **UnitBaixaCheque** (116 linhas de código - Ultima Atualização: indefinido)

1. Componentes ainda com nomenclatura original;
2. Falta comentários no código de uma maneira geral;

- **UnitBEstProducao** (648 linhas de código - Ultima Atualização: 04/2000)

1. Componentes ainda com nomenclatura original;
2. Código confuso, ou seja, de difícil compreensão;
3. Linha 202: procedimento com número de linhas excessivo;
4. Linhas 236 à 241: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
5. Unit possui excesso de comentários que, de certa forma, dificultam o entendimento do código. Alguns comentários são redundantes;
6. Utiliza código muito antigo feito por Ronaldo e por Luciano. Código mal feito utilizando passagens de parâmetros confusos.
7. Linha 340: procedimento "AtribuiBoolean" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
8. Linha 372 e 540: Utilização de números soltos no código ao invés da definição e uso de constantes;
9. Linhas 565 à 588: Três IF's soltos deveriam estar em um ninho de IF's. A melhor opção na realidade seria um case, pois verdadeiro para um deles exclui os demais. Testes estão sendo feitos desnecessariamente implicando em perda de desempenho.
10. Falta comentários no código de uma maneira geral;

- **UnitCadCliente** (371 linhas de código - Ultima Atualização: 01/2001)

1. Componentes ainda com nomenclatura original;
2. Funções e procedimentos de uso específico na Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
3. Falta comentários em alguns trechos do código;

- **UnitCadDependentes** (107 linhas de código - Ultima Atualização: 03/2000)

1. Componentes ainda com nomenclatura original;
2. Linhas 58 à 60 deveriam ter tratamento preventivo de erros (tratamento de exceções);
3. Falta comentários no código de uma maneira geral;

- **UnitCadEncomenda** (977 linhas de código - Ultima Atualização: 07/2000)

1. Componentes ainda com nomenclatura original;
2. Linha13: definição de tipo com nome que descaracteriza o mesmo;
3. Funções e procedimentos de uso específico na Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
4. Linha 225: procedimento com número excessivo de linhas de comando;
5. Linhas 235 e 236: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
6. Linhas 251 à 255: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
7. Linhas 301 à 303: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
8. Linhas 317 e 318 deveriam ter tratamento preventivo de erros (tratamento de exceções);
9. Linha 292: procedimento com número excessivo de linhas de comando;
10. Linha 360: variável local definida com nome pouco significativo ou pouco auto-explicativo;
11. Linha 393: procedimento com número excessivo de linhas de comando;
12. Linha 396: variáveis locais definidas com nomes pouco significativos ou pouco auto-explicativos;
13. Linhas 405 à 415: trecho de código está confuso e deveria estar separado em um outro procedimento (VerificaLimiteQuantidade);
14. Linha 393: procedimento que merecia maior preocupação em relação ao tratamento preventivo de erros (tratamento de exceções);
15. Linha 487: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
16. Linhas 534 e 536 deveriam ter tratamento preventivo de erros (tratamento de exceções);

17. Linha 544: trecho de código confuso. Boa parte do código poderia estar em procedimentos separados para possibilitar o melhor entendimento do código;
18. Linha 582 deveria ter tratamento preventivo de erros (tratamento de exceções);
19. Lihas 632 à 755: procedimentos para manipulação da lista encadeada deveriam estar em uma Unit à parte, pois são usados em outras Units onde também estão se repetindo. Reaproveitamento de código pode ser muito melhorado;
20. Linha 759 deveria ter tratamento preventivo de erros (tratamento de exceções);
21. Linha 778: nomenclatura original de componentes sendo usada no código, o que dificulta o entendimento do código fonte;
22. Linha 791 deveria ser substituída por procedimento com nome que tornasse o entendimento do código mais simplificado;
23. Linha 797: procedimento deveria ter tratamento preventivo de erros (tratamento de exceções);
24. Linha 825: procedimento deveria ter tratamento preventivo de erros (tratamento de exceções);
25. Linha 845: procedimento deveria ter tratamento preventivo de erros (tratamento de exceções);
26. Rotinas de impressão merecem tratamento de erros genérico;
27. Faltam comentários no código de uma maneira geral;

- **UnitCadFornecedor** (317 linhas de código - Ultima Atualização: 03/2000)

1. Componentes ainda com nomenclatura original;
2. Falta comentários em alguns trechos do código;
3. Linhas 193 e 194 deveriam ter tratamento preventivo de erros (tratamento de exceções);
4. Linhas 218 e 219 deveriam ter tratamento preventivo de erros (tratamento de exceções);
5. Linha 220: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
6. Linha 230: procedimento com código confuso e com linhas de código em excesso;

- **UnitVendas** (1459 linhas de código - Ultima Atualização: 12/2000)

1. Linha 35: definição de tipo com nome que descaracteriza o mesmo;
2. Componentes ainda com nomenclatura original;
3. Funções e procedimentos de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
4. Linha 296: variável local definida com nome pouco significativo ou pouco auto-explicativo;
5. Linha 318: expressão de condição do IF não cobre todos os casos que deveria cobrir;
6. Linha 437: variável local definida com nome pouco significativo ou pouco auto-explicativo;
7. Linhas 461 à 472: trecho de código confuso. Poderia estar separado em outro procedimento para facilitar o entendimento do código;

8. Linha 485: Utilização de número solto no código ao invés da definição e uso de constante;
9. Linhas 591 à 610: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
10. Linha 573: procedimento com número de linhas de código excessivo;
11. Linha 640: procedimento com número de linhas de código excessivo;
12. Linhas 653 à 672: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
13. Linhas 674 à 693: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
14. Linha 703: procedimento com número de linhas de código excessivo;
15. Linhas 707 à 748 deveriam ter tratamento preventivo de erros (tratamento de exceções);
16. Linhas 719 à 721: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
17. Linhas 727 à 729: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
18. Linhas 731 à 745: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
19. Linha 897: procedimento com número de linhas de código excessivo;
20. Linhas 920 à 933: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
21. Linhas 946 à 949: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
22. Linha 963: procedimento poderia ser mais genérico para que pudesse ser uasdo em outros pontos do código. Sugestão : receber um array de Queries para fechar ao invés de fechar Queries fixas;
23. Linhas 979 à 1104: procedimentos para manipulação da lista encadeada deveriam estar em uma Unit à parte, pois são usados em outras Units onde também estão se repetindo. Reaproveitamento de código pode ser muito melhorado;
24. Linhas 1197 à 1215: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
25. Linha 1261: procedimento ProcessaEncomenda precisa ser atualizado assim como foi feito no procedimento ProcessaComanda (linha 1267);
26. Faltam comentários no código de uma maneira geral;

- **UnitCadCargo** (159 linhas de código - Ultima Atualização: 11/2000)

1. Componentes ainda com nomenclatura original;
2. Linhas 136 e 137 deveriam ter tratamento preventivo de erros (tratamento de exceções);
3. Faltam comentários de uma maneira geral e existem comentários desatualizados no código;

- **UnitCadCheques** (186 linhas de código - Ultima Atualização: indefinido)

1. Componentes ainda com nomenclatura original;
2. Procedimento de uso específico da Unit definido na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linha 89: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
4. Linha 172 deveria ter tratamento preventivo de erros (tratamento de exceções);
5. Faltam comentários no código de uma maneira geral;

- **UnitCadFuncionario** (437 linhas de código - Ultima Atualização: 01/2001)

1. Componentes ainda com nomenclatura original;
2. Linha 294 deveria ter tratamento preventivo de erros (tratamento de exceções);
3. Nomenclatura original de componentes sendo utilizada no código;
4. Linha 331: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;

- **UnitCadItensForn** (200 linhas de código - Ultima Atualização: 03/2000)

1. Componentes ainda com nomenclatura original;
2. Linha 75 deveria ter tratamento preventivo de erros (tratamento de exceções);
3. Linha 115: expressão de condição do IF não cobre todos os casos que deveria cobrir;
4. Linha 155: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
5. Faltam comentários no código de uma maneira geral;

- **UnitCadPanificadora** (209 linhas de código - Ultima Atualização: 05/2000)

1. Componentes ainda com nomenclatura original;
2. Linhas 114 e 187 deveriam ter tratamento preventivo de erros (tratamento de exceções);
3. Faltam comentários no código de uma maneira geral;
4. Nomenclatura original de componentes sendo usada no código fonte;

- **UnitCadProduto** (556 linhas de código - Ultima Atualização: 02/2001)

1. Componentes ainda com nomenclatura original;
2. Funções, procedimentos e variáveis de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linha 134: função com número de linhas de código excessivo;

4. Linha 231: procedimento com número de linhas de código excessivo;
5. Linhas 246 à 262: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
6. Linhas 298 à 300 aparentam ter erro de lógica, pois da forma como o código se apresenta o mesmo é executado diversas vezes. Uma só vez já resolveria o problema. Não está causando erros no uso mas certamente o está tronando mais lento;
7. Linha 361: procedimento com número de linhas de código excessivo;
8. Linhas 366, 368, 381 e 382 deveriam ter tratamento preventivo de erros (tratamento de exceções);
9. Linhas 364 à 376: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
10. Linhas 378 à 402: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
11. Linha 481 deveria ter tratamento preventivo de erros (tratamento de exceções);
12. Faltam comentários no código de uma maneira geral;

- **UnitCaixa** (274 linhas de código - Ultima Atualização: 12/2000)

1. Componentes ainda com nomenclatura original;
2. Funções e procedimentos de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linhas 123, 149, 154, 191, 223 e 239 deveriam ter tratamento preventivo de erros (tratamento de exceções);
4. Faltam comentários em alguns trechos do código fonte;

- **UnitConfigTerminal** (172 linhas de código - Ultima Atualização: indefinido)

1. Componentes ainda com nomenclatura original;
2. Linha 61 deveria ter tratamento preventivo de erros (tratamento de exceções);
3. Faltam comentário no código de uma maneira geral;

- **UnitCons_Cargo** (251 linhas de código - Ultima Atualização: 03/2000)

1. Componentes ainda com nomenclatura original;
2. Linha 191 deveria ter tratamento preventivo de erros (tratamento de exceções);
3. Faltam comentários no código de uma maneira geral;

- **UnitCons_Cliente** (457 linhas de código - Ultima Atualização: 01/2001)

1. Componentes ainda com nomenclatura original;
2. Linhas 318, 356 e 375 deveriam ter tratamento preventivo de erros (tratamento de exceções);
3. Faltam comentários no código de uma maneira geral;

- **UnitCons_EntradaNF** (318 linhas de código - Ultima Atualização: 02/2000)

1. Componentes ainda com nomenclatura original;
2. Funções e procedimentos de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linha 90: procedimento com número de linhas de código excessivo;
4. Linha 132 deveria ter tratamento preventivo de erros (tratamento de exceções);
5. Linhas 106 à 131: trecho de código poderia estar separado em um outro procedimento para facilitar o entendimento do código;
6. Linha 244 deveria ter tratamento preventivo de erros (tratamento de exceções);
7. Faltam comentários em boa parte do código fonte;

- **UnitCons_Fornec** (350 linhas de código - Ultima Atualização: 01/2000)

1. Componentes ainda com nomenclatura original;
2. Procedimento de uso específico da Unit definido na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linha 220 deveria ter tratamento preventivo de erros (tratamento de exceções);
4. Faltam comentários em alguns trechos do código fonte;

- **UnitCons_Func** (501 linhas de código - Ultima Atualização: 01/2001)

1. Componentes ainda com nomenclatura original;
2. Procedimentos de uso específico da Unit definidos na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linha 163: procedimento com número de linhas de código excessivo;
4. Linha 197 deveria ter tratamento preventivo de erros (tratamento de exceções);
5. Linha 321: nomenclatura original de componentes sendo usada no código, o que dificulta o entendimento do código fonte;
6. Linha 339 deveria ter tratamento preventivo de erros (tratamento de exceções);
7. Linha 369 deveria ter tratamento preventivo de erros (tratamento de exceções);
8. Linha 386 deveria ter tratamento preventivo de erros (tratamento de exceções);
9. Linha 410 deveria ter tratamento preventivo de erros (tratamento de exceções);
10. Linha 436 deveria ter tratamento preventivo de erros (tratamento de exceções);
11. Linha 457 deveria ter tratamento preventivo de erros (tratamento de exceções);
12. Linha 487 deveria ter tratamento preventivo de erros (tratamento de exceções);
13. Linhas 334 à 340: trecho de código deveria estar em um procedimento separado pois é utilizado em diversos outros pontos do código. Repetição desnecessária de código fonte. Repetição do código ocorre nas linhas: 364, 381, 405 e 431;
14. Faltam comentários no código de uma maneira geral;

- **UnitCons_Genero** (241 linhas de código - Ultima Atualização: 03/2000)

1. Componentes ainda com nomenclatura original;
2. Função de uso específico da Unit definida na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linha 224 deveria ter tratamento preventivo de erros (tratamento de exceções);
4. Faltam comentários no código de uma maneira geral;

- **UnitConsReceita** (571 linhas de código - Ultima Atualização: 03/2000)

1. Componentes ainda com nomenclatura original;
2. Linha 160: procedimento com número de linhas de código excessivo;
3. Linhas 180 à 213: trecho de código deveria estar em um procedimento separado;
4. Linha 214 deveria ter tratamento preventivo de erros (tratamento de exceções);
5. Linha 425: uso de números soltos no código ao invés de definição de constantes;
6. Linha 455: procedimento deveria ter tratamento preventivo de erros (tratamento de exceções);
7. Faltam comentários em alguns trechos de código fonte;

- **UnitReceituario** (597 linhas de código - Ultima Atualização: 08/2000)

1. Componentes ainda com nomenclatura original;
2. Linha 136 deveria ter tratamento preventivo de erros (tratamento de exceções);
3. Linhas 321 e 323: uso de números soltos no código ao invés de definição de constantes;
4. Linha 362: procedimento com passagem de parâmetro confuso. Código muito antigo que deve ser refeito. Código confuso;
5. Linha 383: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
6. Linha 579: uso de números soltos no código ao invés de definição de constantes;
7. Faltam comentários em alguns trechos do código;

- **UnitRegProducao** (670 linhas de código - Ultima Atualização: 01/2000)

1. Componentes ainda com nomenclatura original;
2. Linha 358: uso de números soltos no código ao invés de definição de constantes;
3. Linhas 390 e 392: uso de números soltos no código ao invés de definição de constantes;
4. Linha 468 deveria ter tratamento preventivo de erros (tratamento de exceções);
5. Linha 485: procedimento usado tem nome que não possui relação com a operação que se deseja realizar;
6. Linha 510: uso de números soltos no código ao invés de definição de constantes;
7. Linha 526: procedimento com número de linhas de código excessivo;

8. Linhas 549 e 553 à 560: procedimento "inserecampo" perde o sentido de existir no momento em que substitui uma única linha de código e não melhora o entendimento do código como um todo;
9. Linha 576: procedimento com número de linhas de código excessivo;
10. Linha 576: uso no procedimento de números soltos no código ao invés de definição de constantes;
11. Linhas 613 à 636: trecho de código deveria estar em um procedimento separado para facilitar o entendimento do código fonte;

- **UnitConsProducao** (383 linhas de código - Ultima Atualização: 08/2000)

1. Componentes ainda com nomenclatura original;
2. Procedimento de uso específico da Unit definido na área "public" da mesma. "Encapsulamento de dados e funções";
3. Linhas 119 à 129: trecho de código deveria estar em um procedimento separado para facilitar o entendimento do código fonte;
4. Linha 256: procedimento com número de linhas de código excessivo;
5. Linhas 273 à 294: trecho de código deveria estar em um procedimento separado para facilitar o entendimento do código fonte;
6. Linha 304 deveria ter tratamento preventivo de erros (tratamento de exceções);
7. Uso de números soltos no código ao invés de definição de constantes;
8. Faltam comentários no código de uma maneira geral;

- **UnitFuncces** (378 linhas de código - Ultima Atualização: 09/2000)

1. Linha 64: procedimento com código confuso;
2. Linha 174: procedimento com código mal organizado e confuso;
3. Linha 353: procedimento com código confuso;
4. Unit contém algumas funções que não são mais utilizadas. Código desnecessário;
5. Faltam comentários em alguns trechos de código;

- **UnitDB** (879 linhas de código - Ultima Atualização: 10/2000)

1. Linha 284: procedimento deveria ter tratamento preventivo de erros (tratamento de exceções);
2. Linha 336: função deveria ter tratamento preventivo de erros (tratamento de exceções);
3. Linha 418: procedimento com número de linhas de código excessivo;
4. Linha 418: procedimento com código mal organizado e confuso;
5. Linha 823: procedimento deveria ter tratamento preventivo de erros (tratamento de exceções);
6. Unit contém algumas funções que não são mais utilizadas. Código desnecessário;
7. Faltam comentários em alguns trechos de código;

- **UnitErro** (89 linhas de código - Ultima Atualização: 03/1999)

- **UnitImpressoraFiscal** (27 linhas de código - Ultima Atualização: 09/2000)

- **UnitImpFiscBematech** (262 linhas de código - Ultima Atualização: 09/2000)

  1. Faltam comentários no código de maneira geral;

- **UnitImpMatComum** (230 linhas de código - Ultima Atualização: 08/2000)

  1. Faltam comentários no código de maneira geral;

- **UnitOpRelCaixa** (230 linhas de código - Ultima Atualização: 08/2000)

  1. Componentes ainda com nomenclatura original;
  2. Linha 164 deveria ter tratamento preventivo de erros (tratamento de exceções);
  3. Linha 198 deveria ter tratamento preventivo de erros (tratamento de exceções);
  4. Linha 287: procedimento com número de linhas de código excessivo;
  5. Linhas 291 à 312: trecho de código deveria estar em um procedimento separado para facilitar o entendimento do código fonte;
  6. Linhas 316 à 337: trecho de código deveria estar em um procedimento separado para facilitar o entendimento do código fonte;
  7. Faltam comentários no código de maneira geral;

Grupos de erros:

- **Grupo 1**: Falta de comentários em alguns trechos do código;
- **Grupo 2**: Falta de comentários no código de uma maneira geral;
- **Grupo 3**: Componentes ainda com nomenclatura original;
- **Grupo 4**: Código mal organizado e/ou código confuso,
- **Grupo 5**: Falta definição de constantes, variáveis e/ou tipos definidos pelo usuário com nomes pouco significativos (pouco auto-explicativos);
- **Grupo 6**: Repetição desnecessária de código, procedimentos com número excessivo de linhas de código, código deveria estar em um procedimento separado, código desnecessário;
- **Grupo 7**: Falha no encapsulamento de dados e funções;
- **Grupo 8**: Falta de tratamento preventivo de erros (tratamento de exceções);
- **Grupo 9**: Uso de instruções inadequadas;

Anexo 2B

	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5	Grupo 6	Grupo 7	Grupo 8	Grupo 9
UnitAdiantamento 149 - 11/2000									
UnitComanda 829 - 03/2000									
UnitAlteracaoPrecos 393 - 09/2000									
UnitAlterarSenha 105 - indefinida									
UnitBaixaCheque 116 - indefinida									
UnitBEstProducao 648 - 04/2000									
UnitCadCliente 371 - 01/2001									
UnitCadDependentes 107 - 03/2000									
UnitCadEncomenda 977 - 07/2000									

Grupos de erros:
Grupo 1: Falta de comentários em alguns trechos do código;
Grupo 2: Falta de comentários no código de uma maneira geral;
Grupo 3: Componentes ainda com nomenclatura original;
Grupo 4: Código mal organizado e/ou código confuso,
Grupo 5: Falta definição de constantes, variáveis e/ou tipos definidos pelo usuário com nomes pouco significativos (pouco auto-explicativos);
Grupo 6: Repetição desnecessária de código, procedimentos com número excessivo de linhas de código, código deveria estar em um procedimento separado, código desnecessário;
Grupo 7: Falha no encapsulamento de dados e funções;
Grupo 8: Falta de tratamento preventivo de erros (tratamento de exceções);
Grupo 9: Uso de instruções inadequadas;

	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5	Grupo 6	Grupo 7	Grupo 8	Grupo 9
UnitCadFornecedor 317 - 03/2000	■		■	■				■	
UnitVendas 1459 - 12/2000		■	■	■	■	■	■	■	■
UnitCadCargo 159 - 11/2000		■	■					■	
UnitCadCheques 186 - indefinida		■	■				■	■	
UnitCadFuncionario 437 - 01/2001			■	■				■	
UnitCadItensForn 200 - 03/2000		■	■					■	■
UnitCadPanificadora 209 - 05/2000		■	■	■				■	
UnitCadProduto 556 - 02/2001		■	■			■	■	■	■
UnitCaixa 274 - 12/2000	■		■				■	■	

Grupos de erros:
Grupo 1: Falta de comentários em alguns trechos do código;
Grupo 2: Falta de comentários no código de uma maneira geral;
Grupo 3: Componentes ainda com nomenclatura original;
Grupo 4: Código mal organizado e/ou código confuso,
Grupo 5: Falta definição de constantes, variáveis e/ou tipos definidos pelo usuário com nomes pouco significativos (pouco auto-explicativos);
Grupo 6: Repetição desnecessária de código, procedimentos com número excessivo de linhas de código, código deveria estar em um procedimento separado, código desnecessário;
Grupo 7: Falha no encapsulamento de dados e funções;
Grupo 8: Falta de tratamento preventivo de erros (tratamento de exceções);
Grupo 9: Uso de instruções inadequadas;

	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5	Grupo 6	Grupo 7	Grupo 8	Grupo 9
UnitConfigTerminal 172 - indefinida		■	■					■	
UnitCons_Cargo 251 - 03/2000		■	■					■	
UnitCons_Cliente 457 - 01/2001		■	■					■	
UnitCons_EntradaNF 318 - 02/2000	■		■			■	■	■	
UnitCons_Fornec 350 - 01/2000	■		■				■	■	
UnitCons_Func 501 - 01/2001		■	■	■		■	■	■	
UnitCons_Genero 241 - 03/2000		■	■				■	■	
UnitConsReceita 571 - 03/2000	■		■		■	■		■	
UnitReceituario 597 - 08/2000	■		■	■	■			■	

Grupos de erros:
Grupo 1: Falta de comentários em alguns trechos do código;
Grupo 2: Falta de comentários no código de uma maneira geral;
Grupo 3: Componentes ainda com nomenclatura original;
Grupo 4: Código mal organizado e/ou código confuso,
Grupo 5: Falta definição de constantes, variáveis e/ou tipos definidos pelo usuário com nomes pouco significativos (pouco auto-explicativos);
Grupo 6: Repetição desnecessária de código, procedimentos com número excessivo de linhas de código, código deveria estar em um procedimento separado, código desnecessário;
Grupo 7: Falha no encapsulamento de dados e funções;
Grupo 8: Falta de tratamento preventivo de erros (tratamento de exceções);
Grupo 9: Uso de instruções inadequadas;

	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5	Grupo 6	Grupo 7	Grupo 8	Grupo 9
UnitRegProducao 670 - 01/2000									
UnitConsProducao 383 - 08/2000									
UnitFuncoes 378 - 09/2000									
UnitDB 879 - 10/2000									
UnitErro 89 - 03/1999									
UnitImpressoraFiscal 27 - 09/2000									
UnitImpFiscBematech 262 - 09/2000									
UnitImpMatComum 230 - 08/2000									
UnitOpRelCaixa 230 - 08/2000									

Grupos de erros:
Grupo 1: Falta de comentários em alguns trechos do código;
Grupo 2: Falta de comentários no código de uma maneira geral;
Grupo 3: Componentes ainda com nomenclatura original;
Grupo 4: Código mal organizado e/ou código confuso,
Grupo 5: Falta definição de constantes, variáveis e/ou tipos definidos pelo usuário com nomes pouco significativos (pouco auto-explicativos);
Grupo 6: Repetição desnecessária de código, procedimentos com número excessivo de linhas de código, código deveria estar em um procedimento separado, código desnecessário;
Grupo 7: Falha no encapsulamento de dados e funções;
Grupo 8: Falta de tratamento preventivo de erros (tratamento de exceções);
Grupo 9: Uso de instruções inadequadas;

Anexo 2C

Grupos de erros e respectivos refatoramentos:

- **Grupos 1 e 2**: Falta de comentários em alguns trechos do código, falta de comentários no código de uma maneira geral;

  - **Nome:** Comentando Código Fonte;

  - **Resumo:** Este refatoramento deve ser usado para prover o código de comentários de dois tipos básicos:
    - comentários junto ao cabeçalho de funções e procedimentos para identificar qual o objetivo dos mesmos, o seu status (concluído, em teste, precisa ser melhorado etc), a data da última alteração e o nome do programador responsável;
    - comentários no corpo dos procedimentos e/ou funções explicando o que está sendo feito em uma determinada linha de código ou em um determinado trecho de código;

  - **Motivação:** Apesar de não se tratar de alterações no código fonte em si, este refatoramento contribui de forma muito preciosa para que o código existente possa ser entendido e até mesmo alterado por qualquer programador, sendo este seu maior benefício. Além disso, o comentário inicial, colocado imediatamente antes do todo e qualquer procedimento e/ou função, ajuda na compreensão geral da funcionalidade do mesmo, como também fornece informações importantes para a equipe de desenvolvimento como o status, a ultima data de alteração e quem foi o responsável pela criação ou pela ultima alteração daquele trecho de código.

  - **Mecanismo:** Para cada Unit do projeto faça:

    1. Para todo e qualquer procedimento insira um comentário imediatamente antes de seu início com o seguinte formato:
    - Descrição: descrição sucinta de qual o objetivo geral do procedimento/função;
    - Status: situação em que se encontra o procedimento/função. Deve informar se o mesmo está concluído, se está em testes, se precisa ser melhorado etc;

- Data da última alteração: data em que o procedimento foi criado ou alterado pela última vez;
- Programador responsável: membro da equipe da programadores que criou o procedimento/função ou que o alterou pela última vez;

Caso este comentário já exista procure atualizá-lo da melhor maneira possível observando se as informações ali contidas condizem com a realidade do código do procedimento/função;

2. Analise o código de cada um dos procedimentos/funções da unit e identifique linhas de código que merecem a atribuição de comentários para tornar seu entendimento mais fácil. Siga as regras abaixo:

- Em hipótese alguma altere código fonte durante este refatoramento;
- Caso o comentário já exista revise-o e faça as alterações que julgar necessárias;
- Faça comentários simples e objetivos, tanto quanto for possível;
- Evite comentar linhas de código de entendimento trivial;
- Procure colocar os comentários sempre ao lado da linha de código comentada. O excesso de linhas de comentário entre linhas de código pode quebrar a seqüência do código fonte e prejudicar seu entendimento;
- Para comentar um bloco de código (laços, cases, ninhos de IF's etc) coloque o devido comentário na linha imediatamente anterior ao início do mesmo deixando uma linha de separação entre o início do comentário e a linha de código anterior ao bloco e outra linha de separação entre o fim do bloco e a linha de código imediatamente posterior;

3. Revise o trabalho realizado procurando por falhas nos comentários;
4. Salve e feche a Unit comentada;

- **Exemplo 1:**

  - Procedimento antes do refatoramento *Comentando Código Fonte*:

```
procedure SalarDadosBD(BancoDeDados: TIBDadaBase; Tabelas: array of TIBDataSet; Transacao : TIBTransaction);
begin
    try
        BancoDeDados.ApplyUpdates(Tabelas);
        Transacao.Commit;
    except
        Transacao.Rollback;
        raise;
     end;
end;
```

  - Procedimento após o refatoramento *Comentando Código Fonte*:

```
// Descrição: Salva no banco de dados todas as alterações feitas nas tabelas passadas no vetor de tabelas
// Status: Concluído
// Data da Última Alteração: 13/06/2000
// Responsável: Rodrigo F. de Albuquerque
procedure SalarDadosBD(BancoDeDados: TIBDadaBase; Tabelas: array of TIBDataSet; Transacao : TIBTransaction);
begin
    try
        BancoDeDados.ApplyUpdates(Tabelas); //Para cada tabela do Array salva as alterações no Banco de Dados
        Transacao.Commit; // Confirma definitivamente as alterações no Banco de Dados
    except
        Transacao.Rollback; // Em caso de erro, desfaz todas as alterações. Volta o BancoDeDados ao estado anterior
        raise;
     end; // try
end;
```

- **Exemplo 2:**

  - Função antes do refatoramento *Comentando Código Fonte*:

```
function TfrmLimpaBancoDados.RetornaDataFinal(IndiceMes, IndiceAno: Integer): String;
var
  DataFinal : String;
begin
  DataFinal := IntToStr(IndiceMes + 1);
  if (IndiceMes + 1) < 10 then
    DataFinal := '0' + DataFinal;

  if cmbMes.ItemIndex in [0,2,4,6,7,9,11] then
    DataFinal := DataFinal + '/31/'
  else
  if cmbMes.ItemIndex in [3,5,8,10] then
    DataFinal := DataFinal + '/30/'
  else
    DataFinal := DataFinal + '/29/';

  DataFinal := DataFinal + IntToStr(IndiceAno + 2000);
  RetornaDataFinal := DataFinal;
end;
```

- Função após o refatoramento *Comentando Código Fonte*:

```
// Descrição: Retorna um string referente ao último dia do mês escolhido pelo usuário no formato mm/dd/yyyy
// Status: precisa ser melhorado. Falta fazer tratamento de fevereiro para anos bissextos
// Última Alteração: 10/02/2001
// Responsável: Rodrigo F. de Albuquerque
function TfrmLimpaBancoDados.RetornaDataFinal(IndiceMes, IndiceAno: Integer): String;
var
  DataFinal : String;
begin
  DataFinal := IntToStr(IndiceMes + 1);

  if (IndiceMes + 1) < 10 then  // caso mês escolhido seja menor que 10 (outubro) concatena 0 na frente. Ex: 9 = 09
    DataFinal := '0' + DataFinal;

  // Verifica o mês escolhido e concatena no String o último dia do mês. As possibilidades são: 28, 29 30 e 31
  if cmbMes.ItemIndex in [0,2,4,6,7,9,11] then
    DataFinal := DataFinal + '/31/'
  else
  if cmbMes.ItemIndex in [3,5,8,10] then
    DataFinal := DataFinal + '/30/'
  else
    DataFinal := DataFinal + '/29/';

  DataFinal := DataFinal + IntToStr(IndiceAno + 2000); // Finalmente concatena no final do String o ano selecionado

  RetornaDataFinal := DataFinal;  // Comando que faz a função retornar o String encontrado

end;
```

➢ **Grupo 3**: Componentes ainda com nomenclatura original;

- **Nome:** Nomeando Componentes Adequadamente;

- **Resumo:** Este refatoramento serve para garantir que todos os componentes utilizados em uma aplicação qualquer terão sua nomenclatura original alterada de forma que o novo nome atribuído a cada componente possa identificá-lo com maior facilidade.

- **Motivação:** Apesar de muito simples este refatoramento ajuda a minimizar o tempo necessário para o entendimento do código fonte, sobretudo em trechos de código fonte onde constantemente são feitas referências a componentes. Este refatoramento auxilia também, em boa parte dos casos, o programador a efetuar mudanças na própria interface com o usuário, uma vez que um componente mencionado no código fonte pode ser facilmente localizado no formulário onde o mesmo se encontra.

- **Mecanismo:** Para cada Formulário do projeto faça:

1. Verifique o nome atribuído a cada um dos componentes que estão no formulário. Se o componente ainda estiver com a nomenclatura original nomeie-o de forma que o mesmo possa ser facilmente localizado no formulário apenas pelo nome. Para os casos em que o componente já não possui a nomenclatura original, verifique se o nome atribuído ao mesmo torna fácil a sua localização no formulário e caso isso não acorra altere novamente a propriedade "Name" do mesmo. Em qualquer um dos casos o nome atribuído ao componente deverá seguir a "**Nomenclatura Padrão de Componentes, Constantes e Variáveis**" da ByteCom Sistemas.

2. Analise o código fonte da Unit correspondente ao Formulário que acaba de ser alterado. Verifique se no código fonte são feitas referências a algum componente cujo nome foi alterado. Caso isso ocorra, substitua no código fonte o antigo nome do componente pelo nome que acaba de ser atribuído ao mesmo.

3. Verifique se a Unit que está sendo trabalhada é usada por alguma outra Unit do projeto. Caso isto ocorra, verifique se algum componente cujo nome foi alterado é referenciado no código fonte das Units que usam a Unit do Formulário atual. Caso isso ocorra, substitua no código fonte o antigo nome do componente pelo nome que acaba de ser atribuído ao mesmo.

4. Salve as alterações feitas, compile e teste;

- **Exemplo:**
  - Trecho de código antes do refatoramento *Nomeando Componentes Adequadamente*:

```
...
case ComboBox1.ItemIndex of
        1: label1.Caption := "Descrição";
        2: label1.Caption := "Código";
        3: label1.Caption := "Código de Barras";
        4: label1.Caption := "Tipo";
end;
...
```

  - Trecho de código após o refatoramento *Nomeando Componentes Adequadamente*:

```
...
case cbxOpcaoConsulta.ItemIndex of
        1: lblCampoPesquisa.Caption := "Descrição";
        2: lblCampoPesquisa.Caption := "Código";
        3: lblCampoPesquisa.Caption := "Código de Barras";
        4: lblCampoPesquisa.Caption := "Tipo";
end;
...
```

➢ **Grupo 4**: Código mal organizado e/ou código confuso;

- **Nome:** Organizando Código Fonte;

- **Resumo:** Este refatoramento diz respeito basicamente à forma como o código fonte está fisicamente organizado. O principal objetivo é arrumar o código de forma a facilitar o seu entendimento por outra pessoa que venha a ter contato com o mesmo. Neste refatoramento só devemos alterar a disposição do código, mantendo o código fonte em si inalterado;

- **Motivação:** Este refatoramento ajuda a minimizar o tempo necessário para o entendimento do código fonte, sem que para isso tenhamos que alterar o código fonte em si. Como conseqüência direta, este refatoramento tende a abreviar o tempo necessário para que sejam feitas futuras alterações no código fonte e até mesmo para a introdução de código fonte novo para atender a novas funcionalidades que possam surgir;

- **Mecanismo:** Para cada Unit do projeto faça:

1. Verifique os seguintes aspectos do código fonte efetuando alterações quando julgar necessário:
   - Padrão de tabulação;
   - Espaçamento entre blocos de código;
   - Quebra de linhas de código muito extensas;
   - Agrupamento de instruções semelhantes, caso a ordem das mesmas não provoque alteração no resultado gerado pelo código fonte;

2. Revise o trabalho realizado;

3. Salve as alterações feitas, compile e teste;

- **Exemplo:**

  - Procedimento antes do refatoramento *Organizando Código Fonte*:

```
procedure TfrmVendas.IniciaNovaCompra;
var ano, mes, dia : word;
begin
DMPan.IBPanTran.Active := TRUE;
DMPan.IBPanTran.Commit;
qryGeral.Open;
btnPagamento.Enebled := FALSE;
qryEstoque.close;
bDesconto := FALSE;
btnCancelar.Enabled := FALSE;
qryMVEstoque.Open;
grdProduto.SetFocus;
preparaqueryInserirIB(qryMVCaixa,'',nil,nil);
btnDebitoEmConta.Enabled := FALSE;
with qryMVCaixa do
begin
FieldByName('COD_CAIXA').asInteger := iCodCaixa;
FieldByName('VALOR').asFloat:= 0.00;
FieldByName('COD_FUNCX').asInteger := frmPrincipalDlg.Security1.UserID;
end;
DecodeDate(qryMVCaixa.FieldByName('DATA').asDateTime, ano, mes, dia);
ExcluiTodos;  //limpa a lista que contém os produtos da lista e suas respectivas quantidades.
with frmDesconto do
begin
Desconto := 0;
Total := 0;
TotalComDesconto := 0;
end;
if bImpressora then
begin
if not Impressor.ImpFiscalOK then
begin
showmessage(' Ocorreu algum erro de comunicação com a Impressora Fiscal.'+#13+'  Para evitar problemas durante
a venda, desative a mesma');
bImpressora := FALSE;  //DesativaImpressoraFiscal
end
else
begin
Impressor.AbreCupomFiscal;  //Comando para cupom fical
iNumItem := 0;
lbxItens.Clear;
end;
end;
edtCodProd.SetFocus;
end;
```

- Procedimento após o refatoramento *Organizando Código Fonte*:

```
procedure TfrmVendas.IniciaNovaCompra;
var
        ano, mes, dia : word;
begin

DMPan.IBPanTran.Active := TRUE;
DMPan.IBPanTran.Commit;

qryGeral.Open;
qryEstoque.close;
qryMVEstoque.Open;
grdProduto.SetFocus;
preparaqueryInserirIB(qryMVCaixa,'',nil,nil);

bDesconto := FALSE;
btnPagamento.Enebled := FALSE;
btnCancelar.Enabled := FALSE;
btnDebitoEmConta.Enabled := FALSE;

with qryMVCaixa do
begin
        FieldByName('COD_CAIXA').asInteger := iCodCaixa;
        FieldByName('VALOR').asFloat:= 0.00;
        FieldByName('COD_FUNCX').asInteger := frmPrincipalDlg.Security1.UserID;
end;

DecodeDate(qryMVCaixa.FieldByName('DATA').asDateTime, ano, mes, dia);

ExcluiTodos;   //limpa a lista que contém os produtos da lista e suas respectivas quantidades.

with frmDesconto do
begin
        Desconto := 0;
        Total := 0;
        TotalComDesconto := 0;
end;

if bImpressora then
begin
        if not Impressor.ImpFiscalOK then
        begin
                showmessage(' Ocorreu algum erro de comunicação com a Impressora Fiscal.'+#13+
                                ' Para evitar problemas durante a venda, desative a mesma');
                bImpressora := FALSE;  //DesativaImpressoraFiscal
        end
        else
        begin
                Impressor.AbreCupomFiscal;  //Comando para cupom fical
                iNumItem := 0;
                lbxItens.Clear;
        end;
end;

edtCodProd.SetFocus;

end;
```

➢ **Grupo 5**: Falta definição de constantes. Variáveis e/ou tipos definidos pelo usuário com nomes pouco significativos (pouco auto-explicativos);

- **Nome:** Definição adequada de constantes e variáveis;

- **Resumo:** Este refatoramento serve para garantir que todas as constantes e variáveis definidas em uma aplicação qualquer tenham seus nomes definidos de forma que os mesmos possam, de imediato, farnecer ao programador informações sobre o tipo de dado da constante/variável e para que fim ela é usada.

- **Motivação:** Apesar de muito simples este refatoramento ajuda a minimizar o tempo necessário para o entendimento do código fonte, sobretudo em trechos de código onde se faz o uso demasiado de variáveis e/ou constantes. Este refatoramento auxilia também, em boa parte dos casos, o programador a efetuar mudanças no código fonte, seja por alteração ou adição de funcionalidades, aumentando assim a produtividade da equipe de desenvolvimento.

- **Mecanismo:** Para cada Unit do projeto faça:

1. Verifique o nome atribuído a cada uma das constantes e variáveis que são usadas no código fonte da Unit. Se a variável e/ou constante tiver um nome que não seja compatível com o tipo de dado representado ou com a função da mesma na Unit, atribua-a um novo nome. O nome atribuído à variável/constante sempre deverá seguir a "**Nomenclatura Padrão de Componentes, Constantes e Variáveis**" da ByteCom Sistemas. Caso haja números soltos no código, troque-os pela definição de constantes seguindo o exposto acima.

2. Analise o código fonte da Unit que acaba de ser alterada. Verifique todo o código fonte à procura de instruções onde o nome da variável/constante que acaba de ser alterado ainda é o nome antigo. Caso isso ocorra, substitua no código fonte o antigo nome da variável/constante pelo nome que acaba de ser atribuído à mesma.

3. Verifique se a Unit que está sendo trabalhada é usada por alguma outra Unit do projeto. Caso isto ocorra, verifique todo o código fonte desta nova Unit à procura de instruções onde o nome da variável/constante que acaba de ser alterado na Unit original ainda é o nome antigo. Caso isso ocorra, substitua no código fonte o antigo nome da variável/constante pelo nome que acaba de ser atribuído à mesma.

4. Salve as alterações feitas, compile e teste;

- **Exemplo:**

  - Trecho de código antes do refatoramento *Definição adequada de constantes e variáveis:*

```
Type
 Apont = ^ListaDupla;
 ListaDupla = record
  proximo: Apont;
  codigo: integer;
  quantidade : real;
 end;

 ...

end;

...

Var
  L : Apont;

...


function TfrmComanda.ElementoExistente(cod: integer): boolean;
var
  aux: Apont;
  flag: boolean;
begin
  aux:= L;
  flag:= FALSE;
  if (aux = nil) then
    flag:= FALSE
  else
  begin
    while ((aux^.codigo <> cod) and (aux^.proximo <> nil)) do
      aux:= aux^.proximo;
    if (aux^.codigo = cod) then
      flag := TRUE;
  end;
  ElementoExistente := flag;
end;

...
```

```
function TfrmComanda.TamanhoMaximoLista(L: Apont): boolean;
var
      i: integer;
begin
   if L <> nil then
   begin
      i := 1;
      while L^.proximo <> nil do
      begin
         L:= L^.proximo;
         i := i + 1;
      end;
      if i >= 50 then
         TamanhoMaximoLista := TRUE
      else
         TamnhoMaximoLista := FALSE;
   end
   else
      TamanhoMaximoLista := FALSE;

end;
```

- Trecho de código após o refatoramento *Definição adequada de constantes e variáveis:*

```
Type
 ApontadorDeLista = ^ListaEncadeada;
 ListaEncadeada = record
   pProximo: ApontadorDeLista;
   iCodigoProduto: integer;
   rQuantidade : real;
 end;

 ...

end;

...

Var
   pListaProdutos : ApontadorDeLista;

Const
   iTAMANHO_MAXIMO_LISTA : integer = 50;

...

function TfrmComanda.ElementoExistente(iCodProduto: integer): boolean;
var
   pApontadorAuxiliar: ApontadorDeLista;
   bEncontrou: boolean;
begin
   pApontadorAuxiliar:= pListaProdutos;
   bEncontrou:= FALSE;
   if (pApontadorAuxiliar = nil) then
      bEncontrou:= FALSE
   else
   begin
      while ((pApontadorAuxiliar ^.iCodigoProduto <> iCodProduto) and (pApontadorAuxiliar ^.pProximo <> nil)) do
         pApontadorAuxiliar := pApontadorAuxiliar ^.pProximo;
      if  (pApontadorAuxiliar ^.iCodigoProduto = CodProduto) then
         bEncontrou := TRUE;
   end;
   ElementoExistente := bEncontrou;
end;
```

```
...

function TfrmComanda.TamanhoMaximoLista(pListaProdutosAuxiliar: ApontadorDeLista): boolean;
var
      iContadorDeNodos: integer;
begin
   if pListaProdutosAuxiliar <> nil then
   begin
      i := 1;
      while pListaProdutosAuxiliar ^.pProximo <> nil do
      begin
         pListaProdutosAuxiliar:= pListaProdutosAuxiliar ^.pProximo;
         iContadorDeNodos := iContadorDeNodos + 1;
      end;
      if iContadorDeNodos >= iTAMANHO_MAXIMO_LISTA then
         TamanhoMaximoLista := TRUE
      else
         TamnhoMaximoLista := FALSE;
   end
   else
      TamanhoMaximoLista := FALSE;

end;
```

- **Grupo 6**: Repetição desnecessária de código, procedimentos com número excessivo de linhas de código, código deveria estar em um procedimento separado, código desnecessário;

  - **Nome:** Extraindo Método;

  - **Resumo:** Você tem um fragmento de código que pode ser agrupado. Torne o fragmento um método separado e dê um nome que reflita o seu propósito.

  - **Motivação:** Este refatoramento, além de tornar o código fonte muito mais legível e flexível, faz com que o reaproveitamento de código fonte seja incrementado. Isto porque trechos de código estarão sempre agrupados em métodos cujo nome já exprime qual o seu propósito, trazendo como benefícios o aumento da reusabilidade de código fonte, a diminuição do número médio de linhas de código por método e uma conseqüente quebra da complexidade do código fonte existente.

  - **Mecanismo:**

    1. Crie um novo método;
    2. Copie o código que deve compor o novo método;
    3. Procure, neste código, referências a variáveis que são locais ao escopo do método de origem;
    4. Se algumas dessas variáveis são temporárias e apenas utilizadas no código do novo método, elas devem se tornar variáveis locais ao novo método;
    5. Caso alguma dessas variáveis é modificada dentro do código em questão e utilizada fora dele, faça o novo método retornar este valor. Caso mais de uma variável se encaixe nesta situação, utilize a passagem de parâmetros por variável para as variáveis em questão
    6. Passe como parâmetro para o novo método as variáveis locai que são lidas no código extraído;
    7. Compile quando todas as variáveis locais estiverem resolvidas de acordo com o descrito acima;
    8. Substitua o trecho de código na origem pela invocação do novo método;
    9. Compile teste;

- **Exemplo:**

  - Método antes do refatoramento *Extraindo Método*:

```
procedure TfrmAdiantamento.btnSalvarClick(Sender: TObject);
var
  Arquivo : TextFile;
  i : integer;
begin

  if CamposOk then
    begin
      qryRetiradaFuncionario.FieldByName('COD_USUARIO').asInteger := frmPrincipalDlg.Security1.UserID;
      qryRetiradaFuncionario.FieldByName('DATA_ADIANTAMENTO').asDateTime := dtpDataAdiantamento.DateTime;
      NossoPost([qryRetiradaFuncionario]);
      AplicarDadosQuery([qryRetiradaFuncionario]);
      showmessage('Procedimento de cadastro de adiantamento foi bem sucedido.');

      if (Mensagem('Deseja imprimir o comprovante ?','Pergunta', MB_YESNO + MB_ICONQUESTION +
MB_DEFBUTTON2)=IDYES) then
      begin
        AssignFile(Arquivo,'LPT1');
        Rewrite(Arquivo);
        WriteLn(Arquivo, '========================================');
        WriteLn(Arquivo, 'RECIBO DE ADIANTAMENTO DE SALARIO');
        WriteLn(Arquivo, '----------------------------------------');
        WriteLn(Arquivo, 'Eu, '+qryFunc.FieldByName('NOME').asString+',');
        WriteLn(Arquivo, 'declaro que recebi em '+DateToStr(dtpDataAdiantamento.Date));
        WriteLn(Arquivo, 'a quantia de R$ '+
            FloatToStrF(qryRetiradaFuncionario.FieldByName('VALOR_ADIANTAMENTO').asFloat,
            ffNumber, 10, 2));
        qryGeral.Open;
        WriteLn(Arquivo, 'de '+qryGeral.FieldByName('PANIFICADORA').asString);
        qryGeral.Close;
        WriteLn(Arquivo, 'referente a adiantamento salarial.');
        WriteLn(Arquivo, ' ');
        WriteLn(Arquivo, 'Ass.:_____');
        WriteLn(Arquivo, '                                    ');
        WriteLn(Arquivo, '========================================');
        for i := 1 to 16 do
          WriteLn(Arquivo, '                                    ');
        CloseFile(Arquivo);
      end;

      // ---------------------------------------------------------------------
      PreparaQueryInserir(qryRetiradaFuncionario,'COD_ADIANTAMENTO',nil,nil);
      cbxFuncionario.SetFocus;
      edtValor.Text := '0,00';
    end;
end;
```

- Método após o refatoramento *Extraindo Método*:

```
procedure TfrmAdiantamento.ImprimeComprovanteAdiantamento;
var
  Arquivo : TextFile;
  i : integer;
begin

    AssignFile(Arquivo,'LPT1');
    Rewrite(Arquivo);
    WriteLn(Arquivo, '========================================');
    WriteLn(Arquivo, 'RECIBO DE ADIANTAMENTO DE SALARIO');
    WriteLn(Arquivo, '----------------------------------------');
    WriteLn(Arquivo, 'Eu, '+qryFunc.FieldByName('NOME').asString+',');
    WriteLn(Arquivo, 'declaro que recebi em '+DateToStr(dtpDataAdiantamento.Date));
    WriteLn(Arquivo, 'a quantia de R$ '+
                FloatToStrF(qryRetiradaFuncionario.FieldByName('VALOR_ADIANTAMENTO').asFloat,
                ffNumber, 10, 2));
    qryGeral.Open;
    WriteLn(Arquivo, 'de '+qryGeral.FieldByName('PANIFICADORA').asString);
    qryGeral.Close;
    WriteLn(Arquivo, 'referente a adiantamento salarial.');
    WriteLn(Arquivo, ' ');
    WriteLn(Arquivo, 'Ass.:_____');
    WriteLn(Arquivo, '                                    ');
    WriteLn(Arquivo, '========================================');
    for i := 1 to 16 do
          WriteLn(Arquivo, '                                    ');
    CloseFile(Arquivo);

end;


procedure TfrmAdiantamento.btnSalvarClick(Sender: TObject);
begin
  if CamposOk then
    begin
      qryRetiradaFuncionario.FieldByName('COD_USUARIO').asInteger := frmPrincipalDlg.Security1.UserID;
      qryRetiradaFuncionario.FieldByName('DATA_ADIANTAMENTO').asDateTime := dtpDataAdiantamento.DateTime;
      NossoPost([qryRetiradaFuncionario]);
      AplicarDadosQuery([qryRetiradaFuncionario]);
      showmessage('Procedimento de cadastro de adiantamento foi bem sucedido.');

      if (Mensagem('Deseja imprimir o comprovante ?','Pergunta', MB_YESNO + MB_ICONQUESTION +
MB_DEFBUTTON2)=IDYES) then
      begin
          ImprimeComprovanteAdiantamento;
      end;

      PreparaQueryInserir(qryRetiradaFuncionario,'COD_ADIANTAMENTO',nil,nil);
      cbxFuncionario.SetFocus;
      edtValor.Text := '0,00';
    end;
end;
```

➢ **Grupo 7**: Falha no encapsulamento de dados e funções;

- **Nome:** Encapsulando Dados, Funções e Procedimentos;

- **Resumo:** Este refatoramento deve ser usado basicamente para prover segurança aos dados, funções e procedimentos de uma determinada classe do projeto.

- **Motivação:** Um dos principais recursos oferecidos pelas linguagens orientadas a objeto consiste no encapsulamento de dados, funções e procedimentos. O uso correto deste recurso, além de facilitar a legibilidade do código fonte, faz com que se possa prover segurança aos dados, funções e procedimentos usados por uma determinada classe. Em alguns casos, não muito raros, esta segurança é condição fundamental para o correto e confiável funcionamento do aplicativo.

- **Mecanismo:** Para cada Unit do projeto faça:

  1. Para cada classe definida na Unit verifique se existe a seção Public. Se existir verifique se existem outras Units que fazem uso desta Unit;
  2. Caso existam, verifique nas outras Units se existem referências a cada variável / constante / função / procedimento encontrado na seção Public da Unit que está sendo analisada.
  3. Remova cada um dos elementos para os quais não se encontraram referências nas Units que utilizam a Unit atual da seção Public para a seção Private. Caso sejam encontradas referências nas outras Units, deixe que o elemento referenciado permaneça com sua declaração na seção Public da Unit que está sendo analisada;
  4. Salve as alterações feitas, compile e teste;

- **Exemplo:**

  - Devido ao fato de não apresentar resultados que possam ser demonstrados textualmente, não serão exibidos exemplos para este refatoramento;

➢ **Grupo 8**: Falta de tratamento preventivo de erros (tratamento de exceções);

- **Nome:** Tratando Exceções;

- **Resumo:** Este refatoramento deve ser usado para prover um aplicativo de tratamento preventivo de erros. Deve ser usado sempre que exista alguma possibilidade de ocorrência de erro em uma instrução ou conjunto de instruções, se a linguagem de programação usada disponibilizar este recurso;

- **Motivação:** Um dos aspectos mais admiráveis em um software de qualidade é a sua robustez, isto é, a capacidade de recuperar-se de erros graves sem que para isso seja necessário reiniciar a aplicação ou até mesmo a máquina. Esta é uma das principais vantagens oferecidas pelo tratamento exceções. Além disso, um bom tratamento de exceções ajuda também pelo fato de fornecer ao usuário mensagens mais precisas acerca do erro ocorrido. Isto facilita a comunicação entre o usuário e os técnicos de suporte da empresa de software.

- **Mecanismo:** Para cada Unit do projeto faça:

1. Procure por linhas de comando ou conjuntos de linhas de comando que possam estar sujeitas a algum tipo de erro grave. Para isto, você deverá pensar em todas as causas possíveis de erros para o trecho de código em questão;
2. Para cada trecho de código selecionado aplique o tratamento de erros oferecido pela linguagem de programação que esteja sendo usada. Procure cobrir todas as possibilidades de código do erro possíveis, sempre do mais específico para o mais genérico. Isto facilitará o envio ao usuário de uma mensagem de erro o mais clara possível. Tenha sempre entre as possibilidades de erros o erro mais genérico que possa ocorrer para o caso de o erro ocorrido não ser nenhum dos mais específicos que você tentou tratar. Se for o caso, juntamente com este passo aplique o refatoramento *Extraindo Método.*
3. Salve as alterações feitas, compile e teste;

- **Exemplo:**

  - Método antes do refatoramento *Tratando Exceções*:

```
Procedure AplicarDadosQuery(Tabelas: array of TQuery);
var
  iCont:Integer;
  db : TDataBase;
begin
  db := TDataBase.Create(nil);
  db := Session.OpenDatabase('Panificador');
  db.StartTransaction ;
  try
    Tabelas[0].ApplyUpdates;                    // poderia ocorrer erro grave sem tratamento
    for iCont := 1 to high(tabelas) do
      Tabelas[iCont].ApplyUpdates;              // poderia ocorrer erro grave sem tratamento
    db.Commit; {on success, commit the changes}
    Tabelas[0].CommitUpdates;
    for iCont := 1 to high(tabelas) do
      Tabelas[iCont].CommitUpdates;
  finally
    Session.CloseDatabase(db);
  end;
end;
```

  - Método após o refatoramento *Tratando Exceções*:

```
Procedure AplicarDadosQuery(Tabelas: array of TQuery);
var
  iCont:Integer;
  db : TDataBase;
begin
  db := TDataBase.Create(nil);
  db := Session.OpenDatabase('Panificador');
  db.StartTransaction ;
  try
    try
      Tabelas[0].ApplyUpdates;
      for iCont := 1 to high(tabelas) do
        Tabelas[iCont].ApplyUpdates;
      db.Commit; {on success, commit the changes}
    except
      db.Rollback; {on failure, undo the changes}
      case errors[i].errorCode of
        9729 : raise exception.Create('9729 - Item duplicado.');
        9732 : raise exception.Create('9732 - Existe pelo menos um campo obrigatório que não foi preenchido.');
        9733 : raise exception.Create('9733 - Registro possui ligação com outra tabela, logo, NÃO pode ser excluído.');
        13059: raise exception.Create('13059 - Erro SQL genérico');
      end;
    end;
    Tabelas[0].CommitUpdates;
    for iCont := 1 to high(tabelas) do
      Tabelas[iCont].CommitUpdates;
  finally
    Session.CloseDatabase(db);
  end;
end;
```

- **Grupo 9**: Uso de instruções inadequadas;

  - **Nome:** Usando Instruções Adequadamente;

  - **Resumo:** Este refatoramento está basicamente relacionado com a legibilidade do código fonte e com o desempenho do software. Devemos usá-lo quando houve a escolha errada entre duas ou mais instruções que podem efetuar a mesma operação, porém de maneiras distintas;

  - **Motivação:** Este refatoramento ajuda a melhorar a legibilidade do código fonte e consequentemente aumenta a produtividade da equipe de desenvolvimento. Em alguns casos pode influenciar também na velocidade de execução do software, melhorando o seu desempenho e exigindo menor velocidade de processamento do hardware;

  - **Mecanismo:** Para cada Unit do projeto faça:

1. Procure por instruções que possam ser substituídas por outras semelhantes porém mais indicadas para o contexto. Em geral este erro ocorre quando se fazem escolhas dentro dos seguintes grupos de instruções:

   - Ifs, If then else, Case;
   - While, Repeat, For;

2. Para cada instrução encontrada onde se verifique que a substituição é adequada, efetue todas as alterações que julgar necessárias para fazer a substituição. Caso contrário, não altere o código fonte;
3. Salve as alterações feitas, compile e teste;

- **Exemplo:**

  - Método antes do refatoramento *Usando Instruções Adequadamente*:

```
procedure TfrmBEstProducao.edtCodProducExit(Sender: TObject);
begin
 if TRIM(edtCodproduc.Text)<>'' then
 begin
  SQLProducao(StrToInt(edtCodProduc.Text),TRUE);
  // Se a tabela estivar vazio o que Indica que o Codigo não exsite
  if qryProducao.IsEmpty then
    mensagem('Produção não existe','Atenção',MB_OK + MB_ICONWARNING)
  else
  begin
   if qryProducao.fieldbyname('SITUACAO').asInteger = PRODUCAO_PLANEJADA then
   begin
    PreencherSQLIngrediente(qryProducao.fieldbyname('Cod_Receita').AsInteger,
            qryproducao.fieldbyname('NumB_UNI').asFloat,
            qryproducao.fieldbyname('Num_UNI_PLANJ').asFloat);
    btnPesquisar.enabled:=false;// desabilita botão de Pesquisa
    edtCodProduc.enabled:=False;// Desabilita o edit da pesquisa
    pnlDados.Enabled := TRUE;
    btnSalvar.Enabled := TRUE;
    btnSalvar.SetFocus;// coloca o focu no botão de confirrmar
   end;
   if qryProducao.fieldbyname('SITUACAO').asInteger = PRODUCAO_CONCLUIDA then
     MensagemErro('Produção já foi executada em: '+qryProducao.fieldbyname('Data_prod').asString,'Atenção');
   if qryProducao.fieldbyname('SITUACAO').asInteger = PRODUCAO_BAIXA_ESTOQUE then
    showmessage('Já foi dada baixa no estoque para esta produção.');

  end;
 end;
end;
```

  - Método após o refatoramento *Usando Instruções Adequadamente*: (Solução intermediária)

```
procedure TfrmBEstProducao.edtCodProducExit(Sender: TObject);
begin
 if TRIM(edtCodproduc.Text)<>'' then
 begin
  SQLProducao(StrToInt(edtCodProduc.Text),TRUE);
  // Se a tabela estivar vazio o que Indica que o Codigo não exsite
  if qryProducao.IsEmpty then
    mensagem('Produção não existe','Atenção',MB_OK + MB_ICONWARNING)
  else
  begin
   if qryProducao.fieldbyname('SITUACAO').asInteger = PRODUCAO_PLANEJADA then
   begin
    PreencherSQLIngrediente(qryProducao.fieldbyname('Cod_Receita').AsInteger,
            qryproducao.fieldbyname('NumB_UNI').asFloat,
            qryproducao.fieldbyname('Num_UNI_PLANJ').asFloat);
    btnPesquisar.enabled:=false;// desabilita botão de Pesquisa
    edtCodProduc.enabled:=False;// Desabilita o edit da pesquisa
    pnlDados.Enabled := TRUE;
    btnSalvar.Enabled := TRUE;
    btnSalvar.SetFocus;// coloca o focu no botão de confirrmar
   end
   else
   if qryProducao.fieldbyname('SITUACAO').asInteger = PRODUCAO_CONCLUIDA then
     MensagemErro('Produção já foi executada em: '+qryProducao.fieldbyname('Data_prod').asString,'Atenção')
   else
   if qryProducao.fieldbyname('SITUACAO').asInteger = PRODUCAO_BAIXA_ESTOQUE then
    showmessage('Já foi dada baixa no estoque para esta produção.');
  end;
 end;
end;
```

- Método após o refatoramento *Usando Instruções Adequadamente*: (Solução Ideal)

```
procedure TfrmBEstProducao.edtCodProducExit(Sender: TObject);
begin
 if TRIM(edtCodproduc.Text)<>'' then
 begin
  SQLProducao(StrToInt(edtCodProduc.Text),TRUE);
  // Se a tabela estivar vazio o que Indica que o Codigo não exsite
  if qryProducao.IsEmpty then
    mensagem('Produção não existe','Atenção',MB_OK + MB_ICONWARNING)
  else
  Case qryProducao.fieldbyname('SITUACAO').asInteger  of
      PRODUCAO_PLANEJADA :
        begin
              PreencherSQLIngrediente(qryProducao.fieldbyname('Cod_Receita').AsInteger,
                    qryproducao.fieldbyname('NumB_UNI').asFloat,
                    qryproducao.fieldbyname('Num_UNI_PLANJ').asFloat);
              btnPesquisar.enabled:=false;// desabilita botão de Pesquisa
              edtCodProduc.enabled:=False;// Desabilita o edit da pesquisa
              pnlDados.Enabled := TRUE;
              btnSalvar.Enabled := TRUE;
              btnSalvar.SetFocus;// coloca o focu no botão de confirrmar
        end
      PRODUCAO_CONCLUIDA :
              MensagemErro('Produção já foi executada em: '+qryProducao.fieldbyname('Data_prod').asString,'Atenção')
      PRODUCAO_BAIXA_ESTOQUE :
              showmessage('Já foi dada baixa no estoque para esta produção.');
  end;
 end;
end;
```

Anexo 2D

"Nomenclatura Padrão de Componentes, Constantes e Variáveis"

Abaixo serão listadas nomenclaturas básicas para alguns componentes e tipos de dados. É de extrema importância que estas nomenclaturas sejam obedecidas pois o seu uso facilita o entendimento do código fonte, especialmente quando lidamos com o trabalho em equipe.

Paleta de componentes Standard :

- MainMenu : mpr
- PopupMenu : pup
- Label : lbl
- Edit : edt
- Memo : mem
- Button : btn
- CheckBox : cbx
- RadioButton : rbt
- ListBox : lbx
- ComboBox : cmb
- ScrollBar : sbr
- GroupBox : gbx
- RadioGroup : rgp
- Panel : pnl
- ActionList : acl

Paleta de componentes Additional :

- BitBtn : btn
- SpeedButton : sbt
- MaskEdit : med
- StringGrid : sgd
- DrawGrid : dgd
- Image : img
- Shape : shp
- Bevel : bvl
- ScrollBox : sbx
- CheckListBox : clb
- Splitter : spt
- StaticText : stx
- ControlBar : cbr
- Chart : cht

Paleta de componentes Win32 :

- TabControl : tcl
- PageControl : pcl
- ImageList : iml
- RichEdit : ret
- TrackBar : tbr
- ProgressBar : pbr
- UpDown : upd
- HotKey : hky

- Animate : amt
- DateTimePicker : dtp
- MonthCalendar : mcr
- TreeView : tvw
- ListView : lvw
- HeaderControl : hcl
- StatusBar : stb
- ToolBar : tbr
- CoolBar : cbr
- PageScroller : psc

Paleta de componentes Data Access :

- DataSource : dsc
- Table : tbl
- Query : qry
- StoredProc : spc
- DataBase : dtb
- Session : ssn
- BatchMove : btm
- UpdateSQL : udt
- NestedTable : ntb

Paleta de componentes Data Controls :

- DBGrid : grd
- DBNavigator : ngt
- DBText : txt
- DBEdit : edt
- DBMemo : mem
- DBImage : img
- DBListBox : lbx
- DBComboBox : cmb
- DBCheckBox : cbx
- DBRadioGroup : rgp
- DBLookupListBox : llb
- DBLookupComboBox : lcb
- DBRichEdit : ret
- DBCtrlGrid : cgd
- DBChart : cht

Exemplo do uso :

Suponha que queremos usar um **label** que identifique um **Edit** ande será digitado o CGC de uma empresa num **form** de cadastro de fornecedores. As nomenclaturas corretas para o caso descrito seriam :

lblCGC	**para o Label**
edtCGC	**para o Edit**

Obs : Para o Form usaremos a nomenclatura frm. No caso acima teríamos então o form de nome frmCadFornecedor.

Padrão para nomenclatura de tipos de dados - Constantes e Variáveis:

- **Shortint : si**
- **Integer : i**
- **Longint : li**
- **Byte : bt**
- **Word : w**
- **Real : r**
- **Single : sg**
- **Double : dbl**
- **Extended : ex**
- **Comp : cp**
- **Currency : cr**
- **Char : c**
- **String : s**
- **Boolean : b**
- **Variant : vr**
- **Arrays : a**
- **Set : st**
- **Record : Rcd**

Exemplo do uso :

Suponha que queremos declarar variáveis para armazenar o código, o salário e o nome de um determinado funcionário. Uma opção para as declarações destas variáveis seria :

```
btCodigo : Byte ;
sgSalario : Single ;
sNome : String ;
```

Obs.: É necessário conhecer a faixa de valores para cada tipo de dado. PESQUISE !!!

Conclusão Geral

O desenvolvimento das atividades relativas ao estágio supervisionado tanto foram de grande valia para o aperfeiçoamento e aplicabilidade dos conhecimentos adquiridos durante as disciplinas do curso de Ciência da Computação como para o aprendizado de assuntos que não foram contemplados no decorrer do mesmo. O objetivo maior da disciplina de estágio supervisionado fora alcançado, pois o aluno pôde aplicar seus conhecimentos de forma efetiva no desenvolvimento de uma ferramenta demandada por um mercado específico e que certamente será comercializada, dando cunho profissional ao trabalho realizado. Além disso, as atividades relativas ao refatoramento de software trouxeram conhecimentos novos e também o enriquecimento profissional, uma vez que o refatoramento fora aplicado em um software que já vem sendo comercializado pela ByteCom Sistemas Ltda., empresa onde o estágio foi realizado.

As atividades desenvolvidas durante o estágio supervisionado geraram para a ByteCom Sistemas os seguintes benefícios: disponibilização da ferramenta de integração do software Panificador no mercado, contribuindo para o aumento de faturamento e de número de clientes da empresa, e melhorias obtidas no código fonte do Panificador após a aplicação do procedimento descrito no relatório de refatoramento. Além disso, espera-se que, após este estágio, a ByteCom Sistemas passe a adotar a prática do refatoramento com maior freqüência, ficando esta "cultura de aplicação do refatoramento de software" como outro benefício direto gerado pelo estágio supervisionado.

Agradecimentos

Aos meus pais, que me apoiaram e confiaram em mim durante toda minha vida

A todos que fazem o Núcleo SOFTEX GENESIS de Campina Grande - POLIGENE

Aos professores do DSC, sem os quais não seria possível obter uma formação acadêmica de qualidade

A todos os colegas de curso

ANEXO GERAL I

DECLARAÇÃO

Declaramos para os devidos fins que o Sr. RODRIGO FIGUEIREDO DE ALBUQUERQUE, solteiro, residente na rua Treze de Maio, 233 – Edf. Morada Nobre – apt 101, Bairro Centro, Campina Grande – Paraíba, cumpriu estágio supervisionado nesta empresa, na função de Analista, durante os meses de Dezembro 2000 à Abril de 2001, onde desenvolveu as atividades relatadas no Plano de Estágio.

Campina Grande, 15 de Maio de 2001

BYTECOM SISTEMAS LTDA.
Robert Kalley Cavalcanti de Menezes
Supervisor Técnico

ANEXO GERAL II

PLANO DE ESTÁGIO

1 Ambiente do Estágio

1.1 Dados da Empresa

Razão Social: ByteCom Sistemas Ltda.
CNPJ: 03.688.196/0001-47
Endereço: Rua Treze de Maio 294 sala 301- Centro, Campina Grande, Paraíba
Tele/Fax: (83)310-1438
E-mail: bytecom@uol.com.br
URL: http://www.bytecom.com.br

1.2 Nicho de mercado

A ByteCom Sistemas é uma empresa encubada no CENTRO SOFTEX GENESIS de Campina Grande - POLIGENE que atual na informatização de indústrias/empresa do setor de Alimentação. Os principais clientes da ByteCom Sistemas são panificadoras, confeitarias, restaurantes, fast foods e pizzarias.

1.3 Estado da Informática na Empresa

A empresa atualmente utiliza a estrutura do laboratório de prototipagem de software do Centro Softex Genesis de Campina Grande – Poligene. No total são 08 (oito) computadores IBM com processadores Pentium Celeron 333 MHz, com 32 e 64 MB de memória RAM e um Servidor Netfinity 3500 também da IBM. Todos os computadores possuem acesso à Internet 24 horas por dia .

A equipe de desenvolvimento é formada basicamente por alunos da graduação do curso de Ciências da Computação e do curso de Desenho Industrial da UFPB, Campus II. O supervisor é Robert Kalley Menezes, coordenador do Centro Softex Genesis de Campina Grande – Poligene e professor do Departamento de Sistemas e Computação – DSC.

1.4 Dados do(a) Estagiário(a)

Nome : Rodrigo Figueiredo de Albuquerque
Matrícula : g9711130-8
Endereço : R. Treze de Maio, 294, Apt 301 – Centro, Campina Grande –PB
Telefone : (83)322-3376
E-mail : sergipano@zipmail.com.br

2 Supervisão

2.1 Supervisor técnico

Nome: Roberto Kalley Cavalcanti de Menezes
Naturalidade: Campina Grande - PB
E-mail: kmenezes@dsc.ufpb.br
Endereço: Av. Manoel Alves de Oliveira, 1095 Cidade Universitária
58.100-000 - Campina Grande - Pb

2.2 *Supervisor acadêmico*

Nome: Francilene Procópio Garcia
Naturalidade: Campina Grande - PB
E-mail: garcia@dsc.ufpb.br
Endereço: Av. Aprígio Veloso, 882 Cidade Universitária
58.109-970 - Campina Grande - Pb

3 Resumo do Problema Objeto do Estágio

O principal produto da ByteCom Sistemas, o Panificador, precisa ser refatorado visando atingir uma melhoria de desempenho e também o atendimento de novos requisitos funcionais. Tais requisitos dizem respeito a um controle multi-loja que não pode ser feito com a atual versão do software e que está sendo demandado pelo mercado-alvo da empresa.

4 Proposta de Solução

4.1 *Objetivo*

Refatorar a atual versão do Panificador visando uma melhoria de desempenho e desenvolver uma ferramenta que usada em conjunto com o Panificador possa prover um controle multi-loja satisfatório, atendento assim aos novos requisitos funcionais e não funcionais demandados pelo mercado-alvo da ByteCom Sistemas.

5 Atividades a Serem Desenvolvidas e Cronograma

5.1 *Atividades*

Descrição da Atividade	Duração
1. Refatoramento do sistema atual com ênfase no atendimento de novos requisitos e melhoria de desempenho	60 horas
2. Análise de requisitos da ferramenta de integração multi-loja e ferramentas auxiliares (responsáveis pelo envio/recebimento de informações entre as lojas da rede de lojas)	60 horas
3. Modelagem de dados e funções da ferramenta de integração multi-loja e ferramentas auxiliares (responsáveis pelo envio/recebimento de informações entre as lojas da rede de lojas)	40 horas
4. Implementação da primeira versão das ferramentas auxiliares (responsáveis pelo envio/recebimento de informações entre as lojas da rede de lojas)	80 horas
5. Implementação da primeira versão da ferramenta de integração multi-loja	120 horas
6. Testes da ferramenta de integração multi-loja e ferramentas auxiliares (responsáveis pelo envio/recebimento de informações entre as lojas da rede de lojas)	40 horas
7. Relatório final de resultados	40 horas

Total = 440 horas

5.2 Cronograma de Atividades Proposto

Cronograma de Atividades							
	Duração						
Atividade	0-60	60-120	120-160	160-240	240-320	360-400	400-440
1							
2							
3							
4							
5							
6							
7							

5.3 Atividades Desenvolvidas e Tempo de Duração

Atividade 1 - 130 Horas
Atividade 2 - 75 Horas
Atividade 3 - 80 Horas
Atividade 4 - 45 Horas
Atividade 5 - 60 Horas
Atividade 6 - Não foi realizada
Atividade 7 - 47 Horas

Campina Grande, 15 de Maio de 2001

Supervisor Técnico

Supervisor Acadêmico

Coordenador da disciplina
Estágio Supervisionado no DSC/UFPB.