Universidade Eduardo Mondlane

Faculdade de Ciências

Departamento de Matemática e Informática

TRABALHO DE LICENCIATURA

ESTUDO DE UM SISTEMA DE INFORMAÇÃO DE GESTÃO DE STOCKS UTILIZANDO UMA BASE DE DADOS RELACIONAL

CASO DE ESTUDO

SOCIMAVEL

(STI)

AUTOR: DOMINGOS LIMA

SUPERVISOR: [illegible]

IT

UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

Trabalho de Licenciatura

Modelo Conceptual do Sistema de Informação de Gestão de Stock usando Extreme Programming

Caso de Estudo

SOICO Televisão

(STV)

Autor: Botomo Ngongo Michel

Supervisor: dr. Higínio Lúcio Nunes

Maputo, Setembro de 2008

Dedicatória

À Jeová Deus, pela bondade imerecida, pelo privilégio de conhecê-lo e servi-lo, pela força que me concedeu, possibilitando a realização deste trabalho.

Agradecimentos

À minha irmã Botomo Basambi Sophie, que desde cedo comportou-se como minha mãe, proporcionando-me o apoio espiritual, emocional e carnal.

Aos meus queridos irmãos e irmãs espirituais de diversas congregações pelas quais passei, nomeadamente Yolo Ouest, Ngaba Centre, Polana Cimento, Choupal, Mahotas Sul, Malanga e Mavalane Leste.

Aos professores da Universidade Eduardo Mondlane, Faculdade de Ciências e Departamento de Matemática e Informática, pela contribuição científica ao longo do curso.

À Dra. Teresa Nogueira, pela confiança que depositou em mim.

Aos Professores Dr. Emílio Mosse e Leopoldo Nhapossa pelo apoio e força que me deram no período mais crítico da minha formação.

Ao meu supervisor dr. Higínio Lúcio Nunes pela dedicação, orientação, compreensão, criticas e ensinamentos metodológicos por ele transmitidos ao longo deste trabalho.

Agradeço ainda aos meus colegas e amigos Celso Nhapulo, Pedro Franice Muchanga, Ermínio Pita Jasse, por tudo.

Aos prestimosos colegas Adérito Cossa e Abdul Dulá pelas criticas e dicas de programação em ambiente Visual Basic, à turma de informática de 2004 pelo companheirismo e suporte. Extensivamente, agradeço aos funcionários do DMI, Sr. Augusto e Dona Zulmira pelo seu contributo para a minha formação.

A todos aqueles que directa ou indirectamente contribuíram para a minha formação, muito obrigado.

Declaração de honra

"Declaro por minha honra, que este trabalho é resultado da minha investigação e foi realizado somente para ser submetido como Trabalho de Licenciatura em Informática na Universidade Eduardo Mondlane."

Botomo Ngongo Michel

(Botomo Ngongo Michel)

Maputo, Setembro 2008

Resumo

As tecnologias de informação no ambiente de desenvolvimento de software passam por um processo de mudanças. Os projectos de software tornam-se cada vez mais complexos, sujeitos a constantes alterações e com prazos cada vez mais restritos para atender as demandas de mercado, garantindo a qualidade de produto e a satisfação do cliente. Neste cenário, os métodos tradicionais e ágeis são analisados criticamente numa busca do que é mais eficaz no projecto do desenvolvimento de software.

O Extreme Programming é uma metodologia ágil, voltada para projectos cujos requisitos mudam com frequência. Ele é baseado em valores e práticas que visam o desenvolvimento ágil de software, garantindo desta forma a qualidade e entrega atempada de software produzido.

Procurou-se aplicar esta metodologia no desenvolvimento de um Modelo Conceptual de Sistema de Informação de Gestão de Stock do armazém principal da SOICO Televisão. O processo de pedido e processamento de requisições têm criado muitos transtornos aos funcionários, levando muito tempo para verem seus pedidos despachados. O controlo manual de Stock da parte da DAP tem retardado a tomada de decisão e está muito sujeito a erros.

É neste contexto que se propõe este Sistema de Gestão de Stock, de forma que os funcionários possam efectuar seus pedidos a partir dos seus postos de trabalhos e obtenham resposta enquanto vão desempenhando suas tarefas diárias.

Para desenhar o modelo do sistema, foi usado a Linguagem de modelação unificada (*Unifying Modeling Language* - UML) como forma de acatar a recomendação da metodologia de desenvolvimento adoptada para este sistema, *extreme programming*, o Sql Server 2000 como Sistema de Gestão de Base de Dados e o Visual Basic.Net para o desenvolvimento do protótipo no ambiente Visual Studio 2008, usando as regras e práticas sugeridas por *eXtreme Programming*.

Epígrafe

"Melhor ser consertado pelas críticas, do que ser destruído pelos elogios "
(Norman Vincent Peale)

"Há homens que lutam um dia e são bons.
Há outros que lutam um ano e são melhores.
Há os que lutam muitos anos e são muito bons.
Porém, há os que lutam toda a vida.
Esses são os imprescindíveis."

(Bertolt Brecht).

Glossário

Palavra	Significado
Estória	Breve descrição da funcionalidade do Sistema
Feedback	Reacção ou Retorno.
Release	Conjunto de funcionalidade bem definidas e que representam algum valor para o cliente
Refactoring	Técnica de Reestruturar o código pouco perceptível, mantendo a funcionalidade do sistema.
Visual Basic .net	Linguagem de programação totalmente orientada a objectos, criada pela Microsoft e distribuída com o Visual Studio.NET
Extreme Programming	Metodologia ágil, voltada para projectos cujos requisitos mudem com frequência, utiliza desenvolvimento orientado a objectos, equipas de até 12 desenvolvedores e desenvolvimento incremental

Símbolos e Acrónimos

Sigla	Descrição
SOICO	*Sociedade Independente de Comunicação*
STV	*Soico Televisão*
XP	*Extreme Programming*
GStock	*Gestão de Stock*
DAP	*Departamento de Administração e Património*
UML	*Unified Modeling Language*

Índice

DEDICATÓRIA I

AGRADECIMENTOS II

DECLARAÇÃO DE HONRA III

RESUMO IV

EPÍGRAFE V

GLOSSÁRIO VI

SÍMBOLOS E ACRÓNIMOS VII

1. INTRODUÇÃO 4

1.1 DEFINIÇÃO DO PROBLEMA 5
1.2 OBJECTIVOS 6
1.2.1 Geral 6
1.2.2 Específicos 6
1.3 METODOLOGIA 6

2. REVISÃO BIBLIOGRÁFICA 9

2.1 ORIENTAÇÃO A OBJECTOS 9
2.2 METODOLOGIAS DE DESENVOLVIMENTO 12
2.2.1 EXTREME PROGRAMMING: VISÃO GERAL 13
2.2.2 VALOR DA XP 14
2.2.2.1 Feedback 14
2.2.2.2 Comunicação 14
2.2.2.3 Simplicidade 15
2.2.2.4 Coragem 16
2.2.3 PRÁTICAS DA XP 17
2.2.3.1 Cliente Disponível ou presente 17
2.2.3.2 Planeamento 17
2.2.3.3 Stand up meeting 18
2.2.3.4 Programação em Par 19
2.2.3.5 Refactoring 20
2.2.3.6 Desenvolvimento guiado por testes 20
2.2.3.7 Código colectivo 21
2.2.3.8 Padrões de desenvolvimento 21
2.2.3.9 Design simples 22
2.2.3.10 Metáfora 22
2.2.3.11 Ritmo sustentável 23
2.2.3.12 Integração contínua 23
2.2.3.13 Releases curtos 23
2.2.4 EQUIPA XP 24
2.2.4.1 Gestor de projecto 24
2.2.4.2 Coach 24

2.2.4.3 Analista de Teste 24
2.2.4.4 Redactor técnico 25
2.2.4.5 Desenvolvedor 25

3. LOGISTICA NAS ORGANIZAÇÕES **26**

3.1 IMPORTÂNCIA DA LOGÍSTICA 27
3.1.1. Administração de Stock 27
3.1.2. Entrada e Processamento de Pedidos 29
3.2. SISTEMAS DE INFORMAÇÃO NA LOGÍSTICA 30
3.3. O FUTURO DA LOGÍSTICA 32

4. SISTEMA DE GESTÃO DE STOCK **33**

4.1 DESCRIÇÃO DO SISTEMA ACTUAL 33
4.1.1. Cenário do Sistema Actual 33
4.1.2. Constrangimentos 35
4.2 DESCRIÇÃO DO SISTEMA PROPOSTO 35
4.2.1. Cenário básico do Sistema Proposto 36
4.2.2. Actores do Sistema 38

5. MODELAÇÃO DO SISTEMA **39**

5.1 MODELAÇÃO VISUAL DO SISTEMA 39
5.1.1 Diagramas de Use Cases 40
5.1.2 Diagrama de Classes 45
5.1.3 Diagrama de Actividades 46
5.1.4 Diagramas de Interacção 47
5.1.5 Diagramas de Estados 49
5.1.6 Diagramas de Componentes 50

6. SEGURANÇA DO SISTEMA **52**

6.1 SEGURANÇA DA INFORMAÇÃO 52
6.2 CONCEITOS DE SEGURANÇA 53
6.3 MECANISMOS DE SEGURANÇA IMPLEMENTADA PARA O SISTEMA DE GESTÃO DE STOCK 54

7. DISCUSSÃO **55**

7.1 ESTUDO COMPARATIVO ENTRE METODOLOGIAS ÁGEIS E METODOLOGIAS TRADICIONAIS 55
7.2 IMPLEMENTAÇÃO DA XP NESTE TRABALHO 57
7.3 IMPACTO DA IMPLEMENTAÇÃO DA GSTOCK NA STV 57
7.4 PROBLEMAS ENFRENTADOS NA IMPLEMENTAÇÃO DA GSTOCK 58

8. CONCLUSÕES E RECOMENDAÇÕES **59**

8.1 CONCLUSÕES 59
8.2. RECOMENDAÇÕES 60

9. REFERÊNCIAS **61**

10. ANEXOS **63**

10.1. FOLHA DE REQUISIÇÃO 63
10.2. FOLHA DE REQUISIÇÃO PREENCHIDA 64
10.3. FICHA DE CONTROLO DE STOCK 65
10.4. DESCRIÇÃO DE USES CASES 66
10.5. CODIGO FONTE DE ALGUNS MÉTODOS 70
10.6. MANUAL DE UTILIZADOR 76

Índice de Tabelas e Figuras

Figura 1. Papel da logística nas organizações (adaptado de Peixoto, 2004)..... 27
Figura 2. Cenário do sistema actual..... 35
Figura 3. Cenário do sistema proposto..... 38
Figura 4. Diagrama de caso de uso..... 43
Figura 5. Diagrama de Classe..... 46
Figura 6. Diagrama de Actividade do processo efectuar a requisição..... 47
Figura 7. o diagrama de sequencia do caso de uso efectuar a requisição..... 48
Figura 8. o diagrama de sequencia do caso de uso processar a requisição..... 49
Figura 9. O diagrama de estados da requisição..... 50
Figura 10. O diagrama de componentes para o caso de Estudo..... 51

Tabela 1. Actores do Sistema..... 39
Tabela 2. Casos de Uso e Actores..... 42
Tabela 3. Decrição do use case Processar a requisição..... 45
Tabela 4. Decrição do use case Efectuar a requisição..... 45
Tabela 5. Descrição de Componentes..... 52
Tabela 6. Estudo comparativo das metedologias Ágeis e Tradicionais..... 57

1. Introdução

A informação assume hoje um papel importante como elemento essencial na sobrevivência e desenvolvimento num contexto de grande dinamismo e competitividade. Sendo a informação um conjunto de dados, colocados num contexto útil e de grande significado, quando fornecidos atempadamente e de forma adequada a um determinado propósito, proporcionam orientações, instruções e conhecimento ao seu receptor, ficando este mais habilitado para desenvolver determinada actividade ou tomar decisões (Varajão, 1998).

Um sistema de informação é um recurso vital para a vida duma organização. Ele ajuda a empresa na tomada de decisão e na realização de suas actividades bem como a alcançar os objectivos por ela traçados. "A informação é sobretudo o recurso estratégico da organização, superior a qualquer outro factor de produção, pois facilita a combinação e a utilização de outros factores produtivos" (Serano, António, et all: 2004).

Com o evoluir das organizações, estas passaram a lidar com grande volume de informação, geralmente em folhas de papel. De acordo com Loureiro (2002), a necessidade de armazenar selectiva, rápida, eficaz e categoricamente a informação em máquinas (bandas magnéticas, dispositivos físicos e de memória virtual) surgiu, numa primeira fase, como urgência de substituição, no que diz respeito à inviabilidade de registarmos os nossos dados em folhas de papel.

A STV (SOICO[1] Televisão) é uma televisão privada que, desde 2001, transmite o seu sinal por difusão em Moçambique, tendo actualmente se expandido para 7 das 10 províncias do País. Esta emissora televisiva tem a sua sede localizada na rua Timor Leste nº 108 na baixa da cidade de Maputo. Para cumprir com seus objectivos como televisão, ela dispõe de um armazém central que serve para guardar vários equipamentos: electrónicos, eléctricos, de escritórios entre outros. O fluxo de pedidos de requisições condiciona exponencialmente a eficiência dos processos de armazenamento e do despacho de autorização dos pedidos efectuados pelos funcionários, dificultando assim o controlo de stock.

[1] SOICO: Sociedade Independente de Comunicação

A Gestão do stock é um factor preponderante que obriga as empresas a investir em soluções informáticas. Nenhuma tecnologia permite uma resolução perfeita de todos os problemas, mas pode ajudar a reduzir significativamente os erros humanos e a aumentar a produtividade de todos os processos de uma empresa.

Um bom sistema de gestão de stock permite um melhor controlo do mesmo, reduz os erros e optimiza os processos e a utilização dos espaços. Com uma concorrência cada vez mais intensa, esse ganho de eficiência pode ser vital no sector da distribuição, tendo em conta a necessidade de reduzir ao máximo o tempo gasto no processo de pedido e autorização de requisições.

Para resolver o problema da gestão de stock dos produtos que se encontram no armazém principal da STV, usou-se a metodologia extreme programming que recomenda o uso do UML para a modelação do sistema. A linguagem de programação usada foi o visual basic.net no ambiente de desenvolvimento visual studio.net 2008. O Sistema de Gestão de Base de Dados foi o SQL Server 2000. Os relatórios do sistema foram produzidos com recursos à Crystal Reports.

1.1 Definição do problema

Depois de analisar detalhadamente o cenário diário vivido que retrata o controlo de Stock no Armazém principal da televisão, identificou-se os seguintes problemas:

- Tempo de resposta para o processamento da requisição é prolongado: O processo de pedido e processamento da requisição leva muito tempo porque envolve muitas pessoas, desde o chefe hierárquico directo do requisitante até o fiel do armazém;
- Gasto de papel: muitas folhas de requisições, que não ajudam muito na hora de produção de relatórios, são preenchidas diariamente;
- Dificuldades enormes em produzir relatórios: O controlo de entradas e saídas é totalmente manual, o que torna a produção de relatórios (inventários de Stock, relatórios de Consumo por departamento e por produto) uma dor de cabeça, para além de interferir na análise de dados para tomada de decisão;
- Controlo de Stock: O controlo de Stock é uma tarefa penosa porque implica fazer inventários semanais das quantidades existentes no armazém, confrontando-as com as fichas manuais de controlo de Stock de cada produto (anexo 10.3).

1.2 Objectivos

Dados os problema expostos, foram definidos os seguintes objectivos para o presente trabalho.

1.2.1 Geral

Desenvolver o modelo conceptual e implementar um Sistema de informação de Gestão de Stock para STV

1.2.2 Específicos

- Analisar o sistema de informação de controlo de Stock em uso;
- Estudo da metodologia *Extreme Programing;*
- Desenvolver o modelo conceptual para a gestão de stock usando *Extreme Programing*;
- Desenhar o protótipo do Sistema de Gestão de Stock para STV, aplicando a metodologia *Extreme Programming*

1.3 Metodologia

Para atingir os objectivos acima mencionados foram realizadas as seguintes actividades associadas a cada objectivo específico:

Analisar o sistema de informação de controlo de Stock em uso.

Durante um período de três meses (Outubro 2007 – Janeiro 2008), o autor deste trabalho realizou um estágio na instituição já anteriormente citada, ao fim do qual conseguiu-se identificar os problemas mencionados na secção 1.1.

Para analisar o sistema de informação de controlo de stock:

- Foram observadas algumas sessões diárias, a partir da entrada de novos equipamentos, recepção da ficha de controlo de entradas, preenchimento da mesma e principalmente o procedimento de pedido e despacho das requisições efectuado pelos funcionários da STV.

- Foram realizadas entrevistas informais ao Sr. Ramiro Santos, Director de Administração e Parimómio que é responsável pelo processamento das requisições. Foram de igual ajuda os fieis de armazém, Sr João Vasco, Sr Altenor e Sr Delfim Octávio.

Para estudar a metodologia Extreme Programming

Para estudar a metodologia extreme programming foi feita a pesquisa bibliográfica, isto é a consulta de livros e páginas de Internet.

Para desenvolver o modelo conceptual

Para desenhar o modelo conceptual do sistema, usou-se a Linguagem de modelação unificada (*Unifying Modeling Language* - UML) como forma de acatar a recomendação da metodologia de desenvolvimento adoptada para este sistema, *extreme programming*. A UML utiliza conceitos simples e uma notação padrão para especificar, construir, visualizar e documentar os sistemas de informação, permitindo um fácil desenvolvimento do sistema. De acordo com Nunes & O'Neill (2001), "o facto de a UML utilizar um conjunto de símbolos padrão, faz com que ela funcione como um meio de comunicação entre os diversos elementos envolvidos no processo". Por outro lado, esta linguagem pode ser usada para documentar o sistema ao longo do desenvolvimento. A ferramenta de modelação usada foi Judy Professional.

Para desenhar o protótipo do Sistema para a Gestão de Stock da STV

Foi usado o Sql Server 2000 como Sistema de Gestão de Base de Dados e o Visual Basic.Net como a linguagem para o desenvolvimento do protótipo no ambiente Visual Studio 2008 usando as regras e práticas sugeridas pela metodologia XP^2. Os relatórios do sistema foram extraídos com base na ferramenta *CrystalReport*.

Visual Basic.NET é uma linguagem de programação totalmente orientada a objectos e com suporte total a UML, criada pela Microsoft e distribuída com o Visual Studio .NET

(Versão seguinte ao Visual Basic 6.0), embora hoje já haja o Visual Basic 2008 (wikipedia, 2008).

O Visual Basic.NET é um produto extremamente diferente do seu antecessor (Visual Basic 6.0), não podendo ser considerada uma versão seguinte. Não apenas a maneira de programar foi alterada, mas todo o conceito de orientação à objectos trouxe poder para a linguagem.

O Microsoft Visual Studio 2008 cumpre a visão da Microsoft de aplicações clientes permitindo que os desenvolvedores criem com muita rapidez e com a mais alta qualidade e riqueza. O Visual Studio 2008, junta ferramentas com as quais as organizações sentirão maior facilidade em capturar e analisar informações, o que significa melhor tomada de decisões de negócios. O Visual Studio 2008 se baseia em três pilares para proporcionar melhor experiência para os programadores: Melhorias na produtividade do desenvolvedor; Gestão do ciclo de vida do software e utilização das mais recentes tecnologias.

[2] XP: Extreme Programming

2. Revisão Bibliográfica

Este capítulo tem como objectivo fazer uma incursão pela bibliografia relevante e que ajudará a compreender melhor o enfoque deste trabalho.

2.1 Orientação a objectos

A orientação a objectos, também conhecida como Programação Orientada a Objectos (POO) ou ainda em inglês Object-Oriented Programming (OOP) é um paradigma de análise, projecto e programação de sistemas de software baseado na composição e interacção entre diversas unidades de software chamadas de objectos.

A análise e projecto orientados a objectos têm como meta identificar o melhor conjunto de objectos para descrever um sistema de software. O funcionamento deste sistema se dá através do relacionamento e troca de mensagens entre estes objectos.

Na programação orientada a objectos, implementa-se um conjunto de classes que definem os objectos presentes no sistema de software. Cada classe determina o comportamento (definidos nos métodos) e estados possíveis (atributos) de seus objectos, assim como o relacionamento com outros objectos.

Conceitos fundamentais

Classe

Representa um conjunto de objectos com características afins. Uma classe define o comportamento dos objectos, através de métodos, e quais estados ele é capaz de manter, através de atributos. Exemplo de classe: Estudante.

Objecto

É uma instância de uma classe. Um objecto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objectos. Exemplo de objectos da classe Estudante: João, José, Maria.

Atributos

São características de um objecto. Basicamente a estrutura de dados que vai representar a classe.

Exemplos

Estudante: nome, endereço, telefone, nível. Livro: autor, editora, ano. Por sua vez, os atributos possuem valores. Por exemplo, o atributo cor pode conter o valor azul.

Métodos

Definem as habilidades dos objectos. A acção só ocorre quando o método é invocado através do objecto. Dentro do programa, a utilização de um método deve afectar apenas um objecto em particular.

Mensagem

É uma chamada a um objecto para invocar um de seus métodos, activando um comportamento descrito por sua classe. Também pode ser direccionada directamente a uma classe (através de uma invocação a um método estático).

Sobrecarga

É a utilização do mesmo nome para símbolos ou métodos com operações ou funcionalidades distintas. Geralmente diferencia-se os métodos pela sua assinatura.

Herança (ou generalização)

É o mecanismo pelo qual uma classe (subclasse) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e estados possíveis (atributos). Há Herança múltipla quando uma subclasse possui mais de uma super-classe. Essa relação é normalmente chamada de relação "é um". Um exemplo de herança: Mamífero é super-classe de Humano. Ou seja, um Humano é um mamífero.

Associação

É o mecanismo pelo qual um objecto utiliza os recursos de outro. Pode tratar-se de uma associação simples "usa um" ou de um acoplamento "parte de". Por exemplo: Um humano usa um telefone. A tecla "1" é parte de um telefone.

Encapsulamento

Consiste na separação de aspectos internos e externos de um objecto. Este mecanismo é utilizado amplamente para impedir o acesso directo ao estado de um objecto (seus atributos), disponibilizando externamente apenas os métodos que alteram estes estados. Exemplo: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões e os sinais de tom).

Abstracção

É a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objectos, uma classe é uma abstracção de entidades existentes no domínio do sistema de software.

Polimorfismo

É o princípio pelo qual duas ou mais classes derivadas de uma mesma super classe podem invocar métodos que têm a mesma assinatura (lista de parâmetros e retorno) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objecto do tipo da super classe. A decisão sobre qual o método que deve ser seleccionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução, através do mecanismo de ligação tardia. No caso de polimorfismo, é necessário que os métodos tenham exactamente a mesma identificação, sendo utilizado o mecanismo de redefinição de métodos. Esse mecanismo de redefinição não deve ser confundido com o mecanismo de sobrecarga de métodos.

Interface

É um contrato entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface.

2.2 Metodologias de Desenvolvimento

A muito tempo a indústria de software vem passando por grandes transformações e novos desafios, entre eles desenvolver softwares com qualidade, no menor tempo possível e que atendam as necessidades dos clientes.

Com estes novos desafios a indústria de software passou a dar valor a algumas áreas da informática, como a engenharia de software e qualidade de software, com intuito de atender as exigências do mercado.

A indústria começou a utilizar metodologias de desenvolvimento de software, adoptou métricas e padrões para alcançar níveis aceitáveis de qualidade, prever custos e prazos em seus projectos. Porém ainda são poucos os projectos que conseguem obter pleno sucesso em seu desenvolvimento, onde prazo e orçamento estabelecidos e as necessidades do cliente sejam realmente atendidos (Larman, 2003).

As metodologias utilizadas nos projectos pesquisados eram as mais variadas, podemos citar modelo em cascata, modelo iterativo. Neste trabalho será utilizado o termo desenvolvimento tradicional para os projectos que se utilizem do modelo em cascata e todos os outros que se baseiam nele.

Analisando os motivos para a baixa taxa de sucesso dos projectos desenvolvidos com modelos tradicionais, cita-se os principais motivos e bastante comuns entre eles:

- Tempo elevado entre cada fase do projecto, não acompanhando as mudanças de requisitos do projecto;
- Falta de conhecimento por parte do cliente da sua real necessidade, dando margem às especulações dos desenvolvedores;
- Forte linearidade no desenvolvimento do projecto;

Focando nas fragilidades do modelo tradicional, surgiram metodologias para desenvolvimento ágil de software.

Temos algumas características destas metodologias:

- Foco nas pessoas, não no processo, evitando especulações dos desenvolvedores;
- Atender as reais necessidades do cliente;
- Ausência de linearidade no desenvolvimento do projecto;
- O cliente aprender a suas reais necessidades durante o projecto e repassar estas novas necessidades a equipa de desenvolvimento;

Uma destas metodologias de desenvolvimento ágil é o *Extreme Programming,* metodologia que prima a qualidade do software desenvolvido, que atenda as reais necessidades do cliente e seja entregue dentro do prazo definido (Beck e Andress, 2004). Esta metodologia será detalhada a seguir.

2.2.1 Extreme Programming: Visão geral

XP é uma metodologia ágil para desenvolvimento de software, com qualidade e que atenda as necessidades do cliente. Alguns praticantes definem a XP como a prática e a perseguição da mais clara simplicidade, aplicado ao desenvolvimento de software.

Uma metodologia voltada para projectos cujos requisitos mudem com frequência, utilizem desenvolvimento orientado a objectos, equipas de até 12 desenvolvedores e desenvolvimento incremental.

A XP busca o máximo de valor a cada dia de trabalho da equipa para o seu cliente. Em um curto espaço de tempo o cliente terá um produto que possa ser utilizado, podendo aprender com o mesmo e reavaliar se o que foi desenvolvido é realmente o desejado.

Por ser uma metodologia recente, a XP sofre mudanças em suas concepções e, portanto, é comum encontrar variações. A adaptação ao ambiente de desenvolvimento deve ser levada em conta, se um valor trouxer mais prejuízos do que benefícios é necessário reavaliar a utilização desta metodologia.

A XP é organizada em torno de um conjunto de práticas e valores que actuam perfeitamente para assegurar um alto retorno do investimento efectuado pelo cliente. A seguir serão apresentados os valores e em seguida as práticas (Beck e Andress, 2004).

2.2.2 Valor da XP

Os valores são as directrizes da XP. Eles definirão as atitudes das equipas e as principais prioridades da metodologia.

2.2.2.1 Feedback[3]

O cliente aprende com o sistema que utiliza e com este aprendizado consegue reavaliar o produto recebido, com isso pode alimentar a equipa de desenvolvimento com as suas reais necessidades. Com o *feedback*, o cliente conduz o desenvolvimento do seu produto, estabelece prioridades e informa aquilo que é realmente importante.

Analogamente, há o *feedback* dado pelo desenvolvedor ao cliente, onde o mesmo aponta riscos, estimativas e alternativas de *design*. Este retorno é o aprendizado do desenvolvedor sobre o que o cliente deseja.

Com este valor, os tempos entre as fases do projecto são extremamente pequenos, o cliente está todo tempo em contacto com a equipa de desenvolvimento, muito diferente dos modelos tradicionais, onde o cliente entrava em contacto com a equipa um bom tempo depois do último *feedback* dado.

Um ponto que muitas empresas de *software* falham é não dar valor ao que o cliente realmente deseja, utilizam cegamente metodologias e acabam esquecendo o real propósito de um software: facilitar o trabalho de pessoas.

Com isto, muitos sistemas acabam dificultando as tarefas das pessoas e como defesa as empresas alegam ter um produto genérico e que atenda as normas legais.

2.2.2.2 Comunicação

Para que o *feedback* entre cliente e desenvolvedor possa ser efectuado com sucesso é necessário ter uma boa comunicação entre eles. A XP prega que esta comunicação ocorra da forma mais directa e eficaz possível, oferecendo agilidade aos assuntos tratados. Recomenda-se o contacto directo (face-a-face) entre cliente e desenvolvedor, para evitar

[3] Reacção ou retorno

qualquer tipo de especulação ou mal entendido entre as partes e para que possíveis dúvidas possam ser resolvidas imediatamente.

Além de sanar as dúvidas no desenvolvimento, o cliente deverá estar disponível para a equipa, ou mesmo presente no ambiente de trabalho da empresa. Isto fará com que o cliente entenda o sistema e enriquecerá os relacionamentos pessoais, criando um elo de parceria e confiança mútua.

Algumas equipas não se adaptam bem a este valor. Este problema deve ser trabalhado em conjunto com a equipa. Enquanto não se acostumarem a falar e a trocar idéias com seus companheiros o sucesso da metodologia estará comprometido (Ron Jeffries, et all, 2000).

2.2.2.3 Simplicidade

Para que o cliente possa aprender durante o projecto e consiga dar o *feedback* necessário à equipa, não basta apenas uma boa comunicação, é necessário que os desenvolvedores implementem da forma mais simples possível o que o cliente deseja.

A lei é: faça a coisa mais simples que pode funcionar. Com esta filosofia, o cliente terá a funcionalidade rapidamente e da forma desejada, dando um *feedback* instantaneamente evitando especulações. O desenvolvedor deve implementar apenas o necessário para que o cliente tenha seu pedido atendido.

Ser simples não é um acto de desespero, é um acto de consciência e absoluta precisão. Muitas pessoas confundem simplicidade e facilidade. O mais simples nem sempre é o mais fácil e também não é escrever menos código. Simplicidade significa codificar o necessário para que um requisito seja atendido e entregue ao cliente.

Evita-se suposições, o futuro é incerto e por causa disso não necessita atenção. Os requisitos evoluem gradativamente em conjunto com o sistema e a arquitectura do projecto. Algumas vezes, o que é necessário hoje será descartado amanhã, e outras vezes o que seria necessário num futuro próximo nunca será utilizado.

2.2.2.4 Coragem

Por ser um processo de desenvolvimento novo e baseado em diversas premissas que contrariam o modelo tradicional, a XP exige que os desenvolvedores tenham coragem para:

- Desenvolver software de forma incremental;
- Manter o sistema simples;
- Permitir que o cliente defina prioridades;
- Fazer desenvolvedores trabalharem em pares:
- Investir tempo em testes automatizados;
- Estimar estórias na presença do cliente;
- Expor código a todos os membros da equipa;
- Integrar o sistema diversas vezes ao dia;
- Adoptar ritmo sustentável de desenvolvimento;
- Abrir mão de documentos que servem como defesa;
- Propor contratos de escopo variável;
- Propor a adopção de um processo novo.
- Assumir em relação ao cliente possíveis atrasos e problemas de implementação;
- Colocar desenvolvedores e clientes frente a frente;
- Implantar uma nova versão do sistema no cliente semanalmente;
- Apostar em seus colaboradores aumentando suas responsabilidades;
- Modelar e documentar apenas quando for de extrema necessidade.

2.2.3 Práticas da XP

Como o nome já diz, as práticas são um conjunto de actividades que deverão ser seguidas pelas equipas que desejam utilizar a XP.

2.2.3.1 Cliente Disponível ou presente

A XP trabalha com uma visão diferente do modelo tradicional em relação ao cliente. A XP sugere que o cliente esteja no dia-a-dia do projecto, acompanhando os passos dos desenvolvedores, onde a sua ausência representa sérios riscos ao projecto.

As funcionalidades do sistema são descritas brevemente em estórias em conjunto com os testes conceptuais e serão estes os indicadores para uma boa implementação. No momento que os desenvolvedores irão implementar as estórias nada mais eficazes do que dialogar com o cliente para entender a estória, fazendo-se necessária a presença do cliente no ambiente de desenvolvimento.

Ao terminar uma estória, com a presença do cliente, a mesma poderá ser validada rapidamente e a equipa receber o feedback necessário sobre a funcionalidade, criando ciclos rápidos e precisos.

2.2.3.2 Planeamento

A XP utiliza o planeamento para assegurar que a equipa de desenvolvimento esteja trabalhando naquilo que gere o máximo de valor para o cliente. Este planeamento é feito várias vezes durante o projecto. É o momento onde o cliente solicita as funcionalidades desejadas através de estórias, onde a equipa estima o custo de cada estória e planeja as releases e as iterações.

Todas as funcionalidades do sistema são descritas em estórias, pequenos cartões em que o cliente deve descrever o que deseja com suas palavras e da forma mais simples possível. Lembrando que a simplicidade também deve ser respeitada pelo cliente.

Após a definição das estórias é necessário estimar o tempo das mesmas para que o cliente priorize o que deve ser implementado. A XP utiliza uma unidade chamada ponto, que refere-se a um dia de trabalho ideal do desenvolvedor, onde o mesmo não precisaria atender telefonemas, participar nas reuniões, ou seja, estaria preocupado apenas com a estória em questão.

Muitas vezes algumas estórias consomem semanas de trabalho, oferecendo uma certa dificuldade de serem estimadas. A XP recomenda que estas estórias sejam quebradas em tarefas menores e que as mesmas não utilizem mais que alguns pontos de um desenvolvedor; recomenda-se quatro pontos no máximo.

Em cada estória é escrita a quantidade de pontos estimada pelo desenvolvedor, a XP recomenda que as estimativas sejam efectuadas em equipa e se possível com a presença do cliente para que durante a estimativa eventuais dúvidas sejam sanadas.

A XP tem por objectivo gerar valor para o cliente ao longo do projecto, por isso software é desenvolvido de forma incremental, onde a cada entrega o cliente possa utilizar o sistema e obter benefícios do mesmo. Estas entregas, a XP denomina como sendo releases, pequenos intervalos de tempo, máximo dois meses, onde funcionalidades que gerem valor ao cliente sejam entregues.

A divisão dos releases é efectuada no início do projecto, geralmente com tamanhos fixos e pequenos. Após esta divisão o cliente define as estórias que farão parte dos releases e tenta-se evitar um planeamento muito longo, pois na entrega de cada release o cliente aprenderá com o sistema e possivelmente irá alterar as estórias para o próximo release.

Mesmo os releases sendo efectuados em curto espaço de tempo, continua representando um tempo muito longo para o feedback do cliente. Por esta razão os releases são divididos em espaços menores, chamados de iterações.

Uma iteração contêm um conjunto de estórias a serem implementadas, com duração entre uma a três semanas, onde ao final da mesma o cliente possa validar as implementações efectuadas e fornecer o feedback necessário à equipa.

2.2.3.3 Stand up meeting

O dia de trabalho de uma equipa XP sempre começa com um stand up meeting. É uma reunião rápida pela manhã, com aproximadamente 20 minutos de duração e onde seus integrantes permaneçam preferencialmente em pé.

Segundo estudos uma reunião em pé é mais rápida, evita bate-papos paralelos e faz os integrantes irem directamente ao assunto. Mais uma vez, ágil e simples.

A reunião tem por objectivo actualizar a equipa sobre o que foi implementado no dia anterior e trocar experiências das dificuldades enfrentadas. Neste momento também são

decididas as estórias que serão implementadas no dia e em conjunto definir os responsáveis por cada uma delas.

2.2.3.4 Programação em Par

A XP exige que todo e qualquer código implementado no projecto seja efectuado em dupla, chamada de programação em par, é uma técnica onde dois desenvolvedores trabalham no mesmo problema, ao mesmo tempo e no mesmo computador. Um deles é responsável pela digitação (condutor) e outro acompanhando o trabalho do parceiro (navegador).

Esta prática agrega diversas técnicas de programação, enquanto o condutor codifica o problema o navegador permanentemente revisa o código digitado, onde pequenos erros de programação que demoraria algumas horas para serem depurados, facilmente são percebidos pelo navegador da dupla.

Um dos grandes benefícios da programação em par é a troca de experiências e idéias entre os desenvolvedores. A solução para um problema geralmente é a soma das idéias da dupla, tornando uma solução e codificação muito mais simples e eficaz.

É com esta prática que a XP uniformiza o nível dos desenvolvedores da equipa, através da troca de experiências.

Após alguns meses trabalhando em duplas todos os desenvolvedores conhecerão todas rotinas do sistema, prontos para qualquer modificação.

Uma grande preocupação em relação a esta prática é a questão da produtividade dos desenvolvedores. Porém, é um erro pensar que somente uma pessoa estará codificando enquanto o outro apenas observa.

O membro que não está codificando não apenas observa, mas também troca idéias, gera soluções e evita praticamente todos erros de codificação além de cobrar padrões de desenvolvimento da equipa.

Estudos indicam que a produtividade de uma equipa que utiliza *pair programming* e de equipas que tenham desenvolvedores sozinhos é praticamente a mesma, porém a qualidade do código gera facilidade de manutenção e outros ganhos a médio e longo prazo.

2.2.3.5 Refactoring

Um desenvolvedor ao deparar com um código mal escrito ou pouco legível mas que esteja funcionando, nos modelos tradicionais de desenvolvimento, dificilmente efectuaria alterações neste código, mesmo que tivesse que implementar novas funcionalidades.

A XP prega que todo desenvolvedor ao encontrar um código duplicado, pouco legível, mal codificado, sem padronização, lento, com código legado ou uso incorrecto de outras implementações, tem por obrigação alterar este código deixando-o mais legível e simples, porém esta alteração não pode mudar o comportamento do código em questão.

Esta prática anda de mãos dadas com o código colectivo, já que todo desenvolvedor tem a possibilidade de melhorar qualquer código do sistema.

A padronização oferece facilidades aos desenvolvedores no momento de implementar novas funcionalidades ou efectuar qualquer tipo de manutenção, uma vez que o código se encontra simples e claro.

Uma questão importante é que a prática de *refactoring* está apoiada pelos testes automatizados, pois facilmente o desenvolvedor terá um *feedback* se a alteração por ele efectuada irá gerar qualquer tipo de comportamento anormal no sistema, sofrendo o aprendizado sobre a alteração por ele efectuada (Martin Fowler, 2001).

2.2.3.6 Desenvolvimento guiado por testes

Esta actividade é considerada extremamente chata e dispendiosa por muitos desenvolvedores na modelagem tradicional, porém para os desenvolvedores de uma equipa XP esta actividade deve ser encarada com extrema naturalidade. Todo código implementado deve coexistir com um teste automatizado, assim garantindo o pleno funcionamento da nova funcionalidade.

É com base nestes testes automatizados que qualquer desenvolvedor terá coragem para alterar uma parte do código que não tenha sido implementada por ele, já que executando os testes automatizados poderá verificar instantaneamente se a modificação efectuada alterou o comportamento do sistema.

Com a implementação de testes o desenvolvedor poderá amadurecer o entendimento sobre o problema que irá solucionar, já que os testes são codificados antes da nova implementação.

Na XP existem dois tipos de testes, os testes de unidade e de aceitação. O teste de unidade tem por objectivo verificar se os resultados gerados por cada classe estão correctos, já o teste de aceitação tem por objectivo verificar se a interacção entre as classes que implementam uma funcionalidade (estória) atendem a necessidade de forma correcta. Os testes de aceitação são descritos pelo cliente e implementados pelos desenvolvedores, facilitando ainda mais a interacção entre as partes.

2.2.3.7 Código colectivo

No modelo tradicional de desenvolvimento, é comum dividir o projecto em partes e colocar responsáveis por cada uma destas partes. Porém apenas uma pessoa torna-se conhecedora daquela parte.

Este tipo de divisão traz sérios problemas ao projecto, uma vez que se aquela parte necessitar inúmeras alterações, apenas uma pessoa estará capacitada para alterá-la. Com estas inúmeras alterações, esta pessoa pode se tornar um gargalo no projecto.

A XP trava uma batalha contra este tipo de divisão, já que não existe uma pessoa responsável por uma parte do código. Cada desenvolvedor tem acesso a qualquer parte do sistema e tem liberdade para alterá-la a qualquer momento e sem qualquer tipo de aviso.

Esta prática tem como consequência um código revisado por diversas pessoas e caso algo não esteja claro, com certeza será alterado por alguma pessoa (*refactoring)* para que o mesmo torne-se legível.

2.2.3.8 Padrões de desenvolvimento

Um dos valores da XP é a comunicação, e o código também é uma forma da equipa se comunicar. Uma forma clara de se comunicar através do código, é manter um padrão no projecto para que qualquer um possa rapidamente entender o código lido.

A XP recomenda a adopção de um padrão desde o início do projecto, preferencialmente padrões utilizados pela comunidade da linguagem de desenvolvimento.

2.2.3.9 Design simples

Nota-se que todas as práticas da XP focam que o maior valor possível seja gerado para o cliente, para tal premissa ser verdadeira a XP prega um *design* do sistema da forma mais simples possível para que atenda a necessidade do cliente.

Umas das premissas do desenvolvimento tradicional é que o custo de uma alteração no sistema cresce exponencialmente ao longo do projecto, com isto tenta-se criar soluções genéricas preparando o sistema para futuras alterações.

Este tipo de pensamento dá margens para especulações e implementações que na maioria dos casos não terá utilidade para o cliente. A XP parte do princípio que o custo de uma alteração tende a crescer lentamente e se estabilizar ao longo do projecto, esta premissa é dita em função dos avanços nas linguagens e práticas de programação, novos ambientes e ferramentas de desenvolvimento, utilização de orientação a objectos no desenvolvimento e em conjunto com estes novos avanços existe o fruto das outras práticas XP, deixando o código simples, legível e passível de alteração a qualquer momento.

2.2.3.10 Metáfora

Muitas vezes pessoas tentam explicar um assunto ou problema a outras pessoas por um período sem obter o êxito necessário na explicação dada, simplesmente as outras pessoas não conseguem entender a mensagem que está se tentando repassar.

Ao criar comparações e analogias com o assunto que está em questão as pessoas passarão a entender deste assunto de uma forma muito mais rápida e possivelmente não a esquecerão mais. Este tipo de artifício é chamado de metáfora na XP, e deve ser utilizado com intensidade durante o projecto, uma vez que facilita a comunicação e fixação dos assuntos entre as partes.

Esta prática anda em conjunto com o ritmo sustentável, já que para o desenvolvedor criar boas metáforas exige criatividade e desenvolvimento de idéias, o que torna necessário uma boa condição física e bem-estar por parte do desenvolvedor.

2.2.3.11 Ritmo sustentável

Uma grande problema nos projectos de software é a falta de tempo para encerrar o mesmo, e uma das técnicas mais adoptadas para compensar a falta de tempo é submeter os desenvolvedores (humanos) a trabalharem até mais tarde e muitas vezes sacrificarem seus finais de semana.

Nos primeiros momentos este tipo de atitude tem efeitos positivos, porém passado alguns dias o rendimento da equipa cai drasticamente, dando margens a erros pela falta de concentração no desenvolvimento, justamente pelo cansaço físico do desenvolvedor.

A XP proíbe que os desenvolvedores trabalhem até mais tarde. A XP sugere um ritmo sustentável de 40 horas semanais, respeitando assim a individualidade e o físico de cada desenvolvedor. Desta forma a equipa estará sempre concentrada e muito menos propensa a pequenas falhas na implementação.

2.2.3.12 Integração contínua

No desenvolvimento tradicional geralmente as equipas são organizadas de modo que uma parte (módulo) fique sob responsabilidade de um desenvolvedor, cabe a esta pessoa efectuar testes e codificação que dizem respeito a sua parte. Esta estratégia reduz a complexidade e as preocupações de um desenvolvedor.

Interfaces de integração são convencionadas para que futuramente estas partes possam se comunicar, este tipo de desenvolvimento está propenso a sérios erros de integração, uma vez que os períodos em que as partes são integradas e testadas são extremamente longos.

A XP propõe uma integração contínua, se possível deve ser efectuada diversas vezes ao dia para que toda a equipa tenha conhecimento do código recém desenvolvido. Com esta prática o feedback sobre a alteração efectuada será retornado em um menor espaço de tempo.

2.2.3.13 Releases curtos

No modelo tradicional o projecto é dividido em fases, de um modo que o cliente começará a ter benefício com o sistema muito próximo de o mesmo estar finalizado. A maior parte do investimento do projecto é feita antes mesmo de estar concluído, sem ter gerado qualquer tipo de valor económico ao cliente.

A XP recomenda que pequenos investimentos sejam efectuados de forma gradual e que busque grandes recompensas o mais rapidamente possível. Com a prática de releases curtos, a XP pretende dar o máximo de valor económico ao cliente em um curto espaço de tempo.

Release é um conjunto de funcionalidades bem definidas e que representam algum valor para o cliente. Um projecto XP pode ter um ou mais releases no seu ciclo de desenvolvimento.

2.2.4 Equipa XP

Em uma equipa de XP existem papéis a serem desempenhados por um ou mais desenvolvedores. Estes papéis serão listados a seguir.

2.2.4.1 Gestor de projecto

Pessoa responsável pelos assuntos administrativos do projecto, mantendo um forte relacionamento com o cliente para que o mesmo participe das actividades do desenvolvimento.

O gestor do projecto é responsável por filtrar assuntos não relevantes aos desenvolvedores e aspectos que só devam ser implementados em iterações futuras.

Para um bom exercício de gestor de projecto é necessário que a pessoa conheça e acredite nos valores e práticas da XP para que possa cobrar isto da sua equipa.

2.2.4.2 Coach

Pessoa responsável pelas questões técnicas do projecto, recomenda-se que esta pessoa seja a com maior conhecimento do processo de desenvolvimento, dos valores e práticas da XP, é de responsabilidade do *coach* verificar o comportamento da equipa frente o processo XP, sinalizando os eventuais erros cometidos pela equipa.

2.2.4.3 Analista de Teste

Pessoa responsável em garantir a qualidade do sistema através dos testes escritos. Ele deve ajudar o cliente a escrever os casos de testes e no final de cada iteração verificar se o software atende todos os casos de testes.

Recomenda-se que esta pessoa não seja um desenvolvedor, para evitar uma visão tendenciosa já que não conhece o código desenvolvido. O analista de teste deve ter uma visão muito parecida com a do cliente e em muitos projectos esta pessoa acaba exercendo o papel de redactor técnico.

2.2.4.4 Redactor técnico

Pessoa responsável em documentar o sistema, evitando um forte trabalho dos desenvolvedores neste tipo de actividade, permitindo uma maior dedicação ao trabalho de codificação.

Esta pessoa deve estar em plena sintonia com os desenvolvedores e cliente para que a documentação reflicta o código escrito e as regras de negócio atendidas pelo sistema.

2.2.4.5 Desenvolvedor

Pessoa responsável em analisar, projectar e codificar o sistema. Na XP não existe diferença entre analista, projectista e programador uma vez que em vários momentos do projecto o desenvolvedor estará exercendo alguma destas actividades.

Naturalmente existe níveis distintos de desenvolvedores dentro de uma equipa, mas com as práticas da XP, como pair programming, a tendência é a equipa se tornar uniforme em suas habilidades.

3. Logistica nas organizações

Neste capítulo abordar-se-á alguns aspectos ligados à logística nas organizações, visto que a gestão de stock faz parte da logística.

Apesar do estudo da logística empresarial ter ganho notoriedade na última década, as actividades chave da logística, como transporte, *stock* e pedidos, já são exercidas desde os primórdios do comércio de mercadorias. De acordo com Peixoto (2004), a grande contribuição da logística empresarial moderna está no encadeamento de actividades, na filosofia integrativa e na análise sistémica.

Logística empresarial tem como objectivo prover o cliente com os níveis de serviços desejados. A meta de nível de serviço logístico é providenciar bens ou serviços correctos, no lugar certo, no tempo exacto e na condição desejada ao menor custo possível, isto é, transportes, manutenção de *stocks*, processamento de pedido e de várias actividades de apoio adicionais. A figura abaixo mostra o papel da logística na empresa e a *interface* com o sector comercial e de produções/operações.

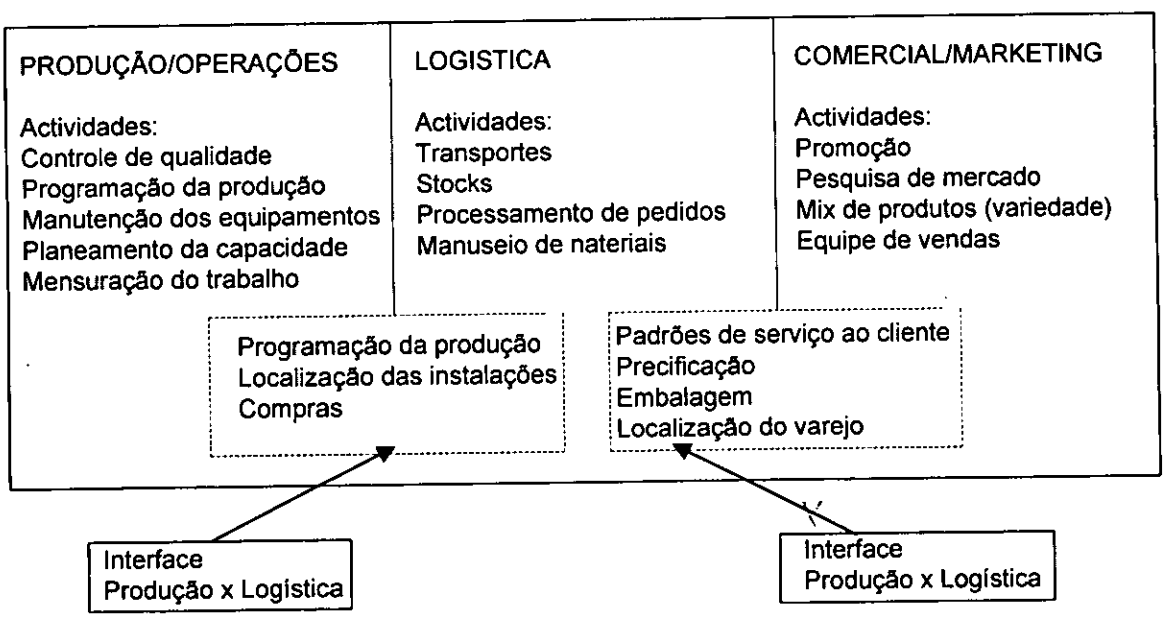


Figura 1. Papel da logística nas organizações (adaptado de Peixoto, 2004)

A administração de materiais e a distribuição física integram-se para formar o que se chama hoje de logística empresarial. Muitas companhias desenvolveram novos organogramas para melhor tratar das actividades de suprimento e distribuição, frequentemente dando *status* de alta administração para a função, ao lado de marketing e produção. O tempo da logística empresarial está chegando e uma nova ordem das coisas está começando.

Podemos definir logística como sendo "a arte e a ciência para abastecer, produzir e distribuir material e produtos no lugar adequado, nas quantidades correctas e nas datas necessárias" (Dicionário de Logística, 2006).

3.1 Importância da Logística

Basicamente a importância da logística para a indústria/empresa moderna está consubstanciada nos seguintes aspectos:

- Valor adicionado ao produto ou serviço;
- Custos referentes às actividades logísticas;
- Globalização e internacionalização dos mercados;
- Diversidade, personalização dos produtos e entrega rápida;
- Aplicação dos conceitos e praticas da logística fora da indústria.

3.1.1. Administração de Stock

Os *stocks* se encaixam no composto de actividades logísticas. O inventário consome grandes somas de capital, que poderiam ser usadas em outros projectos da empresa. Também ele é necessário para manter o nível de serviço ao cliente, assim como a operação eficiente das actividades de produção e distribuição. A boa gestão de *stocks* é essencial.

O enfoque apropriado para controlar os níveis de *stocks* deveria ser cuidadosamente desenvolvido a partir do padrão particular de demanda que cada produto apresenta. Apesar de muitas destas técnicas exigirem certos conhecimentos avançados de estatística

e programação matemática, os casos mais importantes das técnicas de empurrar ou puxar *stocks* foram descritos em nível básico. Eles servem como métodos fundamentais para gerar procedimentos mais complicados de gestão. O enfoque *just-in-time* que minimiza a necessidade de *stocks* de produtos acabados.

A administração de *stocks* tem como tarefa minimizar o investimento em inventário ao mesmo tempo que providencia os níveis de disponibilidade almejados. Este é o problema de encontrar o balanço óptimo dos custos de aquisição, manutenção de *stocks* e faltas. Tanto os métodos teóricos como práticos para controlo de inventário têm esta finalidade.

No esforço de garantir o fluxo de produtos numa organização, os profissionais de logística podem ter necessidades de coordenar seus esforços com actividades que não estão totalmente sob seu controlo. A aquisição, suprimento, obtenção, ou qualquer que seja o título dado, é basicamente uma função de compras. A programação da produção é tradicionalmente uma responsabilidade da manufactura ou operação.

Apesar disso, a administração destas áreas pode ter impacto significativo nos objectivos logísticos e estão mais e mais tornando-se parte das responsabilidades do pessoal de logística. Por isso, estes devem preocupar-se com aqueles aspectos da aquisição e da programação da produção que podem afectar o fluxo de materiais.

A aquisição e a programação da produção são duas actividades que afectam substancialmente o fluxo de materiais de e para a organização. Elas podem não ser responsabilidade total do pessoal de logística, mas existem sobreposições suficientes para que elas sejam tratadas pelo menos como actividades de *interface* e, no máximo, como parte da organização logística. De qualquer forma, a área de logística deveria estar envolvida no processo de decisão dessas funções, que afecta a programação e o volume do fluxo de bens.

A aquisição é responsável por obter os materiais produtivos necessários (assim como materiais para revenda) a preços justos, em quantidades especificadas e entregá-los no local e instante certos. É essencialmente uma função de compras. Entretanto, algumas

decisões importantes, como selecção de fornecedores, política de preços e fazer ou comprar, estabelecem os custos e a importância do suprimento físico na organização.

A inclusão ou não da programação da produção e da aquisição no composto de actividades logísticas ainda é questão de debate.

Assim, comunicações lentas e imprecisas podem custar muito caro para a organização, pois consumidores irados transformam-se em vendas perdidas, os *stocks* tornam-se excessivos, o transporte fica imprevisível e a programação da produção pode gerar preparações desnecessárias e custosas. Processamento rápido e exacto dos pedidos minimiza o tempo de resposta ao cliente e suaviza o comportamento do fluxo de mercadorias pelo sistema logístico.

As características essenciais da entrada e processamento de pedidos são:

1) A natureza da entrada e processamento dos pedidos;

2) As actividades básicas do sistema de entrada de pedidos;

3) Os enfoques alternativos para a entrada e processamento de pedidos;

4) Os procedimentos operacionais do sistema de entrada de pedidos.

3.1.2. Entrada e Processamento de Pedidos

O terceiro elemento-chave das actividades logísticas primárias é a entrada e processamento de pedidos. É uma actividade importante, pois sua duração faz parte do tempo de ciclo total, que é elemento-chave do nível de serviço oferecido aos clientes. Sua velocidade e precisão são itens importantes para a administração desta função. O fluxo de informações de pedidos é factor a ser considerado no projecto e operação do sistema logístico.

O processamento de pedidos subdivide-se em tarefas como a entrada de pedidos, tratamento, verificação de crédito, relatórios de andamento e cobrança (Peixoto, 2004). Nem todos estes processos afectam a duração do tempo de ciclo. O bom projecto do

sistema minimiza o número destas tarefas a serem completadas em sequência, de forma a abreviar o tempo do ciclo máximo possível.

3.2. Sistemas de Informação na Logística

O projecto e a operação do sistema de entrada e processamento de pedidos têm sido afectados pela moderna tecnologia mecânica e electrónica. A decisão de optar-se pela automação deve ser cuidadosamente avaliada em função do volume de ordens a serem processadas e da flexibilidade necessária. Devido ao elevado volume de informação que a STVtem que processar, torna-se útil a automação das actividades relacionadas com a gestão da informação.

"As informações não podem ser melhores que os dados que as geraram" é uma frase frequentemente citada sobre a qualidade da informação que alimenta o processo decisório. Reconheceu-se há muito tempo que o desempenho do planeamento e controlo em termos de gestão depende da quantidade, forma e precisão das informações disponíveis. Até alguns anos atrás, os dados na organização eram classificados, recuperados e manipulados manualmente.

Com a introdução e disseminação dos computadores nos negócios, o manuseio da informação ficou bem mais formalizado. Elaborados SI são hoje lugar-comum.

Conforme Homewebbing (2006), o sistema de informações logísticas (SIL) é um subsistema do sistema de informação para gestão (SIG), que providencia a informação especificamente necessária para a administração logística.

Boa informação é um ingrediente vital no planeamento, operação e controlo de sistemas logísticos. Com a crescente popularidade dos computadores na comunidade de negócios, eles transformaram -se nos principais guardiãs e manipuladores de boa parte do sistema de informações operacionais de uma organização. As actividades de armazenagem de dados, classificação, manipulação e análise são designadas aos SIG.

A estrutura composta por pessoas, equipamentos, métodos e controlo dirigidos para problemas específicos de fluxo de materiais é chamada de sistema de informações logísticas (Homewebbing, 2006).

O sistema gera informação que é utilizada por grande variedade de pessoas na organização, desde a alta administração até o pessoal operacional, sendo empregado para planear, operar e controlar o sistema logístico. O sistema em si tem três partes básicas: 1) a entrada, 2) o processamento e 3) a saída.

Ter excelente planeamento para disponibilizar produtos e serviços para os clientes não garante que os objectivos logísticos serão cumpridos. Estes planos devem ser colocados em acção e seus desempenhos devem ser continuamente monitorados. Assim, é responsabilidade da operação do sistema logístico definir a estrutura interna na empresa, que deverá controlar o fluxo de bens e serviços e planejar as actividades logísticas.

A organização e o controlo são duas actividades chaves em logística. A organização trata especificamente da estruturação dos relacionamentos entre os profissionais da empresa, de maneira a administrar as actividades logísticas eficazmente. O projecto organizacional define quem tem autoridade e responsabilidade pelo planeamento e controlo dos custos e do nível de serviços logísticos.

O controlo da logística recebe constante atenção por parte da administração, pois um ambiente constantemente mutável e eventos imprevistos a todo momento desviam as actividades logísticas dos seus níveis de desempenho planejados. Pode-se acabar não atingindo os objectivos da companhia.

O controlo da actividade de gerir envolve:

1) A definição de metas e padrões de desempenho;

2) A medida do desempenho;

3) A tomada de acções correctivas.

As ferramentas básicas do controlo são os diversos relatórios e auditorias que medem o desempenho. O uso de computadores para auxiliar a administração nas tarefas rotineiras de controlo do sistema está-se disseminando cada vez mais.

3.3. O Futuro da Logística

O futuro da logística é mesmo brilhante. As tendências económicas mostram que os custos para movimentação de bens e distribuição de serviços devem crescer proporcionalmente às outras actividades, tais como manufacturas e *marketing*. O aumento nos custos de combustível, a implantação de melhorias de produtividade e a questão ecológica vai contribuir para o prestígio da logística. A maior importância dos assuntos logísticos vai atrair maior atenção por parte da administração (Homewebbing, 2006).

A inflação, o consumismo e a ecologia são forças que actuam para prestigiar a logística. Entretanto, o carácter de suas operações não vai ser muito diferente do actual. O uso de tecnologias novas e revolucionárias não será o foco de atenção dos administradores para solução de seus problemas logísticos nas duas próximas décadas. Pelo contrário, a administração cautelosa vai procurar extensões da tecnologia existente. O uso de computadores deverá expandir-se substancialmente.

Concluímos que a logística empresarial também chamada de distribuição física, administração de transportes e administração de materiais tem como actividades típicas, entre outras: transportes, gestão de *stocks*, processamento de pedidos, compras, armazenagem, manuseio de materiais, embalagem e programação de produção.

A administração de actividades logísticas em um ambiente organizacional está voltada àquelas actividades necessárias para deixar produtos e serviços disponíveis aos clientes no momento, local e forma desejados. A ênfase situa-se nos problemas da empresa produtiva, pois a maioria das oportunidades e da pesquisa está nesta área.
A logística empresarial estuda como a administração pode prover melhor nível de rentabilidade nos serviços de distribuição aos clientes e consumidores, através de planeamento, organização e controlo efectivos para as actividades de movimentação e armazenagem que visam facilitar o fluxo de produtos. A logística é um assunto vital.

4. *Sistema de Gestão de Stock*

Este capítulo coloca como foco de análise, o sistema de gestão de stock da STV, os problemas actuais e a solução proposta para a resolução dos mesmos.

4.1 Descrição do Sistema Actual

A seguir temos a descrição do sistema actual

4.1.1. Cenário do Sistema Actual

Quando os funcionários da STV precisam de algum material que se encontra no armazém, preenchem a folha de requisição (ver anexo 10.1) onde constam: o material solicitado e sua respectiva quantidade, o departamento beneficiário da requisição e o nome do requisitante. A seguir, o requisitante encaminha o pedido ao seu chefe hierárquico imediato, que assina a requisição como forma de autorizar a solicitação no armazém principal.

O requisitante dirige-se ao responsável no Departamento de Administração e Património que, por não ter a informação das quantidades existentes no stock a tempo real, consulta o fiel do armazém que, usando as fichas manuais que contêm a informação de saídas e entradas de cada material guardado no armazém, escreve na folha de requisição o inventário referente a cada item que figura na requisição. O requisitante, com a folha de requisição nas mãos, dirige-se novamente ao director de Administração e Património a fim de apresentar a informação fornecida pelo fiel do armazém.

Avaliando as quantidades fornecidas, o director de Administração e Património toma a decisão no sentido de despachar a requisição. De referir que ele pode cancelar, rectificar ou autorizar a requisição feita. Depois da decisão tomada, o requisitante passa novamente pelo armazém, munido da requisição assinada (autorizada) e o fiel de armazém procede à entrega de cada item que figura na requisição; ao mesmo tempo dá-se tempo de actualizar o stock sob pena de esquecer, dificultando assim o trabalho. Este processo pode levar mais ou menos dez a quinze minutos quando todos os envolvidos estão presentes. Como forma de comprovar o levantamento do material, após o levantamento do mesmo, o requisitante assina no livro dando a informação sobre o material e a data em que foi levantado.

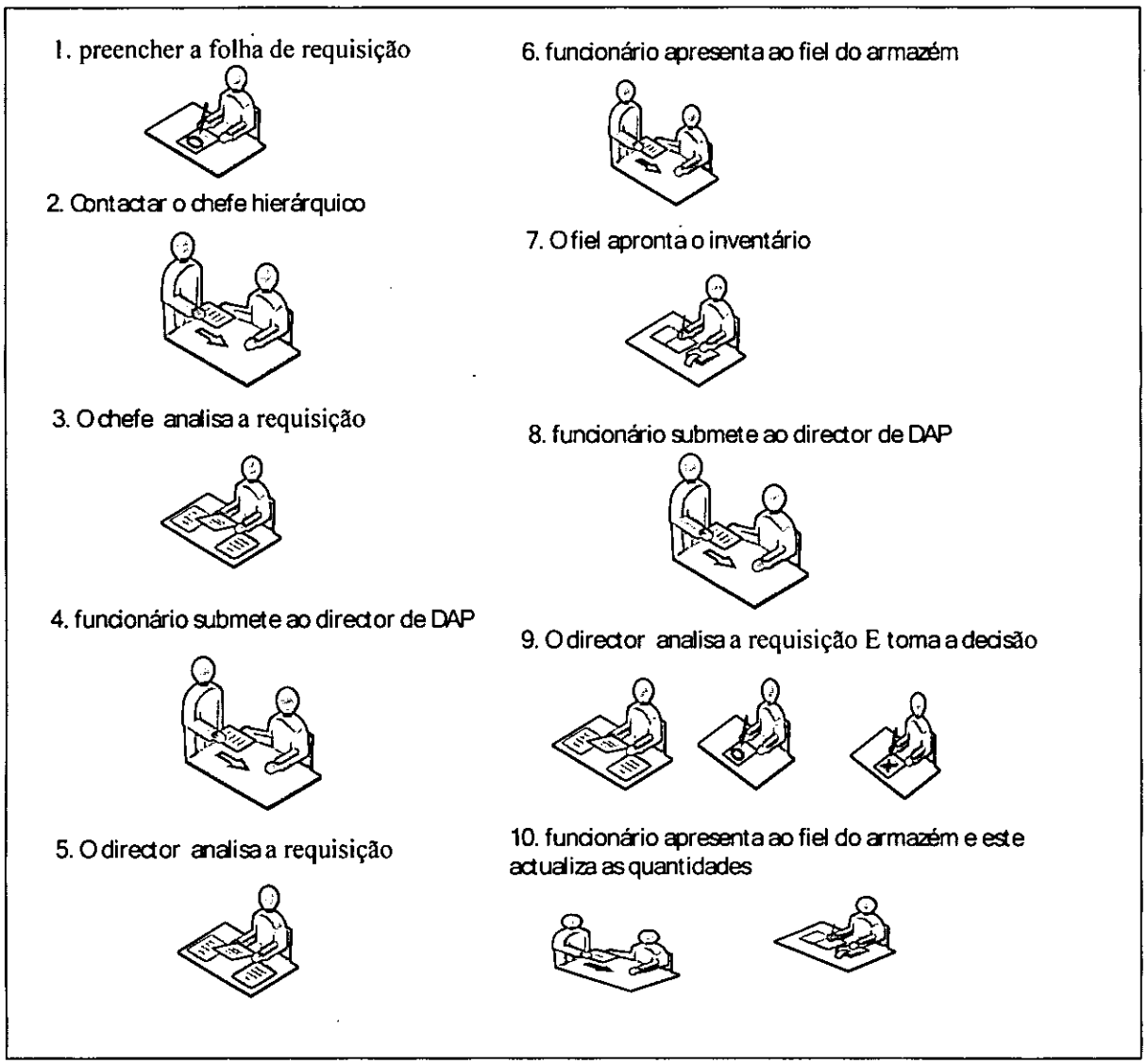


Figura 2. Cenário do sistema actual

4.1.2. Constrangimentos

O sistema actual apresenta os constrangimentos seguintes:

- A deslocação obrigatória do requisitante para efectuar a requisição
- Falta de informação confiável da parte do director de Administração e Património que confia apenas no relatório do fiel do armazém
- Muita burocracia

4.2 Descrição do Sistema Proposto

Um sistema de gestão de stock do armazém baseia-se fundamentalmente numa base de dados estruturada, onde todos os artigos e localizações estão referenciados e respectivamente associados. Quando é dada a entrada de um item no armazém (computador, livros, envelopes...), os seus dados são registados na base de dados.

O artigo fica então imediatamente disponível no sistema e pode ser, em seguida, arrumado numa zona devidamente identificada. Todos os processos internos (arrumação, reposicionamento, inventários, etc.) são controlados a partir da aplicação.

Através deste sistema também é possível emitir alertas quando os stocks atingem certos valores, como por exemplo o de ruptura, permitindo o abastecimento do armazém a tempo e horas. As várias ferramentas de análise, como os relatórios de estatísticas de fluxos em armazém proporcionam aos responsáveis logísticos uma melhor gestão dos bens.

A informatização de todos estes processos logísticos dá aos gestores a oportunidade de introduzirem novas regras em organizações resistentes à mudança. Quando se opta pelo abandono dos métodos de trabalho arcaicos, para dar lugar aos mais eficazes, os gestores passam a ter mais tempo para cuidar de tarefas que contribuem para a evolução da empresa, aproveitando assim a potencialidade que as tecnologias de informação e comunicação oferecem.

O Sistema de Gestão de Stock do Armazém da STV terá as seguintes funcionalidades:

- Realizar com simplicidade a gestão dos fluxos de entrada e saída de material;
- Garantir agilidade e segurança às rotinas operacionais diárias da organização;

- Emitir relatórios que possibilitam a análise minuciosa das operações por parte da administração;
- Direccionar os relatórios para o monitor do computador, impressora, ou arquivos, nos formatos texto, Microsoft Word e Microsoft Excel, o que permite exportação de dados para outras aplicações.
- Simplificar o processo de pedido e autorização de material no armazém
- Dar continuidade ao processo de colocação de etiqueta identificadora, servindo como número de património para o efeito de auditoria.

4.2.1. Cenário básico do Sistema Proposto

O sistema funcionará de seguinte maneira:

1. Qualquer funcionário previamente cadastrado, a partir de seu local de trabalho, terá acesso ao sistema mediante a autenticação, isto é, username e password.
2. Dependendo dos privilégios atribuídos pelo administrador de sistema, o funcionário submeterá o seu pedido e logo a seguir um email será enviado para o Responsável designado (director de administração e património), que irá autorizar ou cancelar o pedido efectuado. O Responsável terá acesso à lista de pedidos pendentes e irá processá-los. Se um pedido for despachado, o requerente será informado do caso via email, de uma forma automática e este dirigir-se-á ao armazém para levantar a sua encomenda, caso seja autorizado. Visto que o sistema irá funcionar na rede local com uma base de dados centralizado, o fiel de armazém terá acesso a autorização de pedido a tempo real e bem antes que o requerente se apresente, este irá aprontá-lo. Na devolução, o processo é simplificado, pois o registo de produto levantado é armazenado na base de dados o que permitirá que se controle mais facilmente quem está com que produto.
3. Qualquer funcionário terá acesso ao histórico de seus pedidos assim como ao estado em que cada um se encontra. (Pendente, Autorizado, Cancelado ou Em Uso)

4. O Sistema faz a auditoria interna quanto a todas as operações efectuadas por qualquer que seja o funcionário.

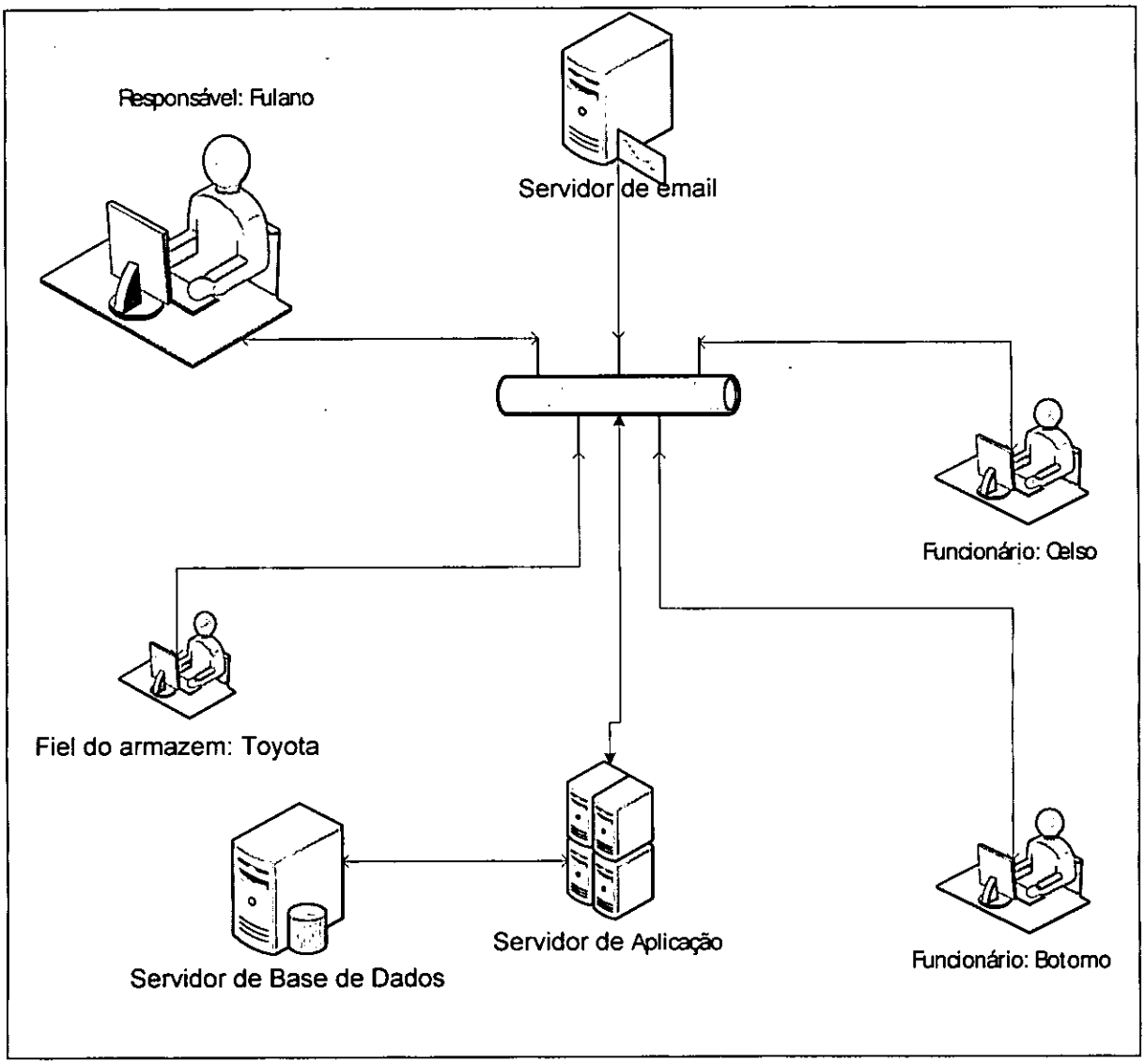


Figura 3. Cenário do sistema proposto

4.2.2. Actores do Sistema

Actores	Descrição
Administrador do Sistema	Cuidar do cadastro de usuários e atribuições de privilégios, atribuição de username e password
Funcionário	Qualquer funcionário potencial que eventualmente precise de utilizar o sistema. Este pode efectuar a requisição, ver a lista de suas requisições e mudar seu password.
Fiel do Armazém	É um usuário normal que tem ainda o privilégio de cadastrar os fornecedores, categorias de produtos (Informática, Eléctrica, etc.), subcategoria (Impressora, PC, Toalha, etc.), lançar as entradas, efectuar o levantamento e devolução de material autorizado
Responsável	É a pessoa que processa as requisições, consulta e analisa o inventário de stock, faz a auditoria de sistema e consulta os relatórios que irão ajudar na tomada de decisão
Usuário	Qualquer utilizador do sistema

Tabela 1. Actores do Sistema

5. Modelação do Sistema

O caso de estudo dará mais ênfase as fases de análise de requisitos (concepção), análise (elaboração) e desenho (construção), já que as principais abstracções do modelo do sistema se encontram nestas fases de desenvolvimento. Desenvolver-se-á uma modelação em UML, para se criar uma base de dados de suporte a um sistema de gestão de Stock na STV.

O desenvolvimento do sistema proposto seguiu uma abordagem Orientada à Objectos, deste modo, optou pelo uso da UML – *Unified Modeling Language,* como recomendado pela metodologia utilizada neste trabalho, pois pela abrangência e simplicidade dos conceitos utilizados, a UML facilita o desenvolvimento de um sistema de informação, permitindo a integração de aspectos de natureza organizacional que constituem o negócio e os elementos de natureza tecnológica, que irão constituir o sistema informático, ajudando a dominar a complexidade das regras de negócio e definir os processos e fluxos informativos (Nunes e O'Neill, 2001).

Ainda nesse contexto, conforme argumentam Silva e Videira, a UML, alarga o âmbito das aplicações alvo comparativamente a outros métodos existentes designadamente porque permite, por exemplo, a modelação de sistemas concorrentes, distribuídos para a *Web*, SIG, etc., uma vez que a sua ênfase é a definição de uma linguagem de modelação standard. A UML é independente das linguagens de programação, das ferramentas CASE bem como dos processos de desenvolvimento.

Um princípio básico no esforço de definição da UML é a simplicidade, a coerência e a introdução de novos elementos.

Em seguida, descreve-se a apresentação de alguns diagramas que representam a fase de análise do sistema, ilustrando a modelação visual por detrás do sistema informático.

5.1 Modelação Visual do Sistema

Por influência dos métodos desenvolvidos na primeira metade da década de 1990, o UML inclui os diagramas de casos de utilização (use *cases*) que permitem a especificação de requisitos funcionais segundo uma aproximação focada primordialmente nos utilizadores do sistema. A modelação de requisitos funcionais (e mesmo o próprio desenvolvimento do *software*) através de especificação de casos de utilização é

actualmente considerada como uma abordagem extremamente adequada, quer por facilitar a comunicação entre a equipa de projecto e os clientes/utilizadores quer ainda por promover a comunicação, gestão e condução no desenvolvimento do próprio projecto. Porém, há alguns aspectos importantes na especificação de requisitos, que têm a ver com a sua apresentação, organização e nível de detalhe.

5.1.1 Diagramas de Use Cases

Um diagrama de casos de utilização ilustra um conjunto de casos de utilização, de actores e suas relações. As suas aplicações comuns são para:

- Modelar o contexto de um sistema. Neste caso, a ênfase encontra-se na identificação da fronteira do sistema, dos seus actores e no significado das suas funções.
- Modelar os requisitos do sistema.

5.1.1.1 Casos de Uso e Actores

Casos de Uso	Actor
Cadastro de departamentos	Administrador do Sistema
Cadastro de Funcionários	Administrador do Sistema
Cadastro de Utilizadores do Sistema	Administrador do Sistema
Cadastro de Categoria de Produtos	Fiel do Armazém
Cadastro de Subcategorias	Fiel do Armazém
Cadastro de Produtos	Fiel do Armazém
Cadastro de Fornecedores	Fiel do Armazém
Lançamento de Entradas de Produtos	Fiel do Armazém
Efectuar a requisição	Funcionário
Processamento da requisição	Responsável
Levantamento da requisição	Fiel do Armazém
Devolução de Material	Fiel do Armazém
Consulta do inventário de Stock	Responsável
Consulta das entradas efectuadas	Fiel do Armazém e Responsável
Consulta das requisições efectuadas	Todos
Consulta do histórico das requisições	Responsável
Consulta do relatório de Consumo	Responsável
Alteração da Senha	Todos
Efectuar login	Todos

Tabela 2. Casos de Uso e Actores

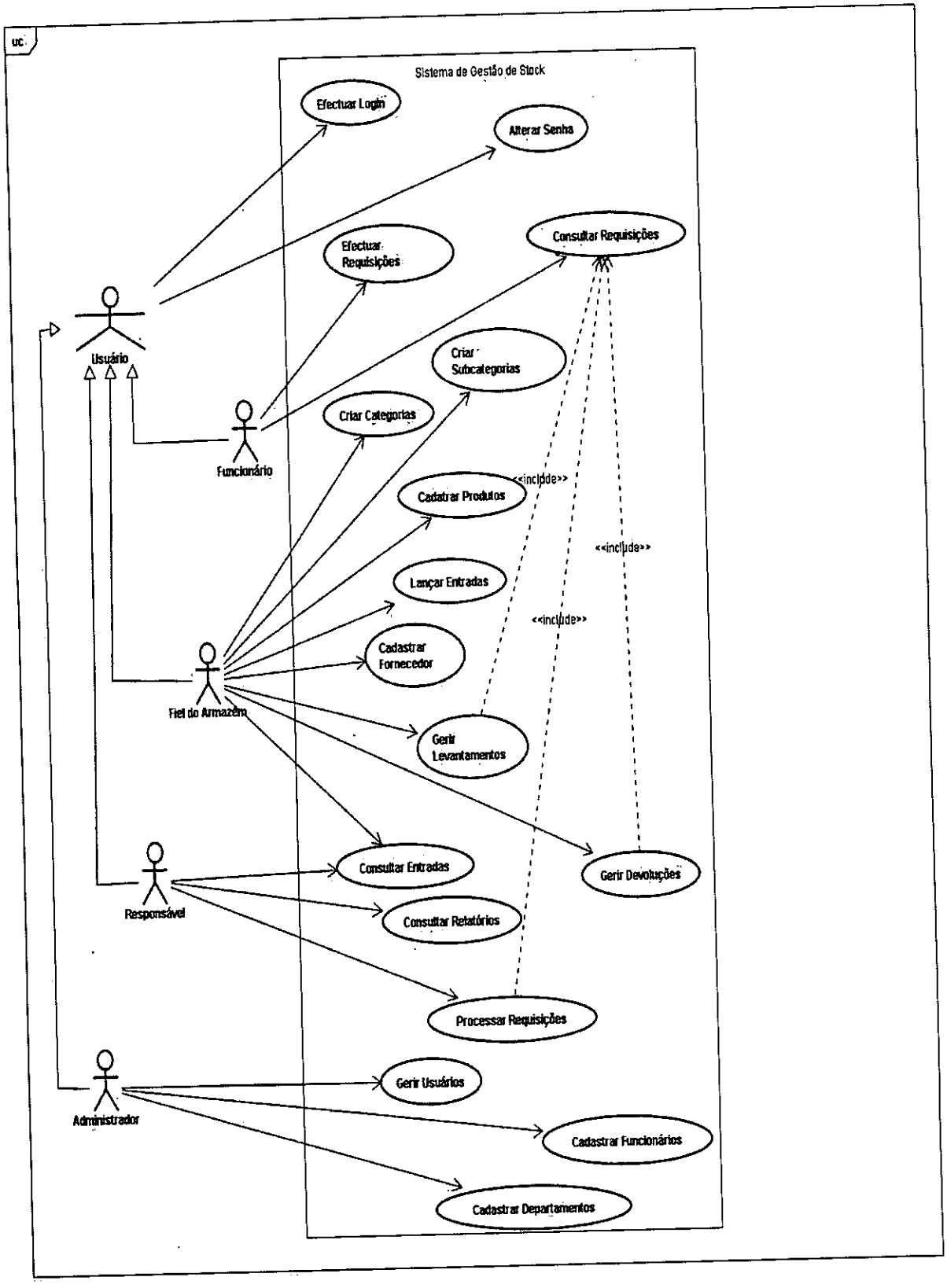


Figura 4. Diagrama de caso de uso

5.1.1.2 Descrição dos Use Cases

Cada um dos use cases identificados deve ser detalhado ou descrito em termos de cenários de utilização. Esses cenários são possíveis caminhos seguidos dentro do *use case*, de forma a fornecer ao actor uma resposta (Shneider e Winters, 1999).

Nome	***Processar a requisição (US001)***
Pré-condição	O responsável é um utilizador válido do sistema
Actor	Responsável
Descrição	1. O use case começa quando o responsável clica no menu movimento, submenu processar a requisição. 2. Automaticamente, o sistema apresenta a lista de requisições pendentes organizadas pelas datas de submissão 3. O responsável selecciona a requisição que deseja processar 4. O sistema apresenta a lista de produtos que constam na requisição, as quantidades correspondentes. Para ajudar o responsável a tomar decisão, o sistema coloca ao lado de cada produto o stock actual do produto e a observação (Ruptura, Insuficiência, Esgotado) 5. O responsável pode rectificar as quantidades. 6. O responsável processa a requisição por clicar no botão autorizada ou cancelada. 7. O sistema manda a mensagem de confirmação da decisão tomada. a. Se a decisão for autorizada, o sistema elimina na requisição, os produtos esgotados e exige que se ajuste as quantidades no caso de ineficiência, senão a requisição permanece pendente. b. Se a decisão for cancelada, o sistema cancela a requisição. 8. Automaticamente, o sistema manda um email ao funcionário requisitante informando-o da decisão tomada.

Pós-condição	

Tabela 3. Descrição do use case Processar a requisição

Nome	***Efectuar requisição (UC002)***
Pré-condição	O requisitante é um utilizador válido do sistema
Actor	Funcionário
Descrição	1. O use case começa quando o funcionário clica no menu movimento, submenu efectuar a requisição. 2. Automaticamente, o sistema identifica o requisitante através do username e password 3. O funcionário selecciona o beneficiário da requisição caso não seja seu próprio departamento, pois é o beneficiário seleccionado por defeito pelo sistema 4. O funcionário tem acesso à lista de produtos 5. O funcionário filtra a lista pela lista de categoria de produtos ou pelos iniciais do produto 6. O funcionário adiciona os produtos à requisição 7. De cada vez que um produto é adicionado, o sistema verifica se este produto já faz parte da requisição. 8. O funcionário submete a sua requisição. 9. Automaticamente, o sistema manda o email para o responsável que processa as requisições tendo a informação da lista de produtos solicitados e do nome do requisitante 10. O sistema confirma a submissão da requisição e permite que o funcionário faça mais uma requisição caso ele queira.
Pós-condição	

Tabela 4. Descrição do use case Efectuar a requisição

5.1.2 Diagrama de Classes

Pela análise verifica-se que temos catorze (14) classes que se relacionam segundo o diagrama apresentado na figura abaixo. Este mostra a visão lógica e a estrutura estática do sistema. Mas antes de prosseguir, apresentamos as classes com os seus respectivos atributos e operações.

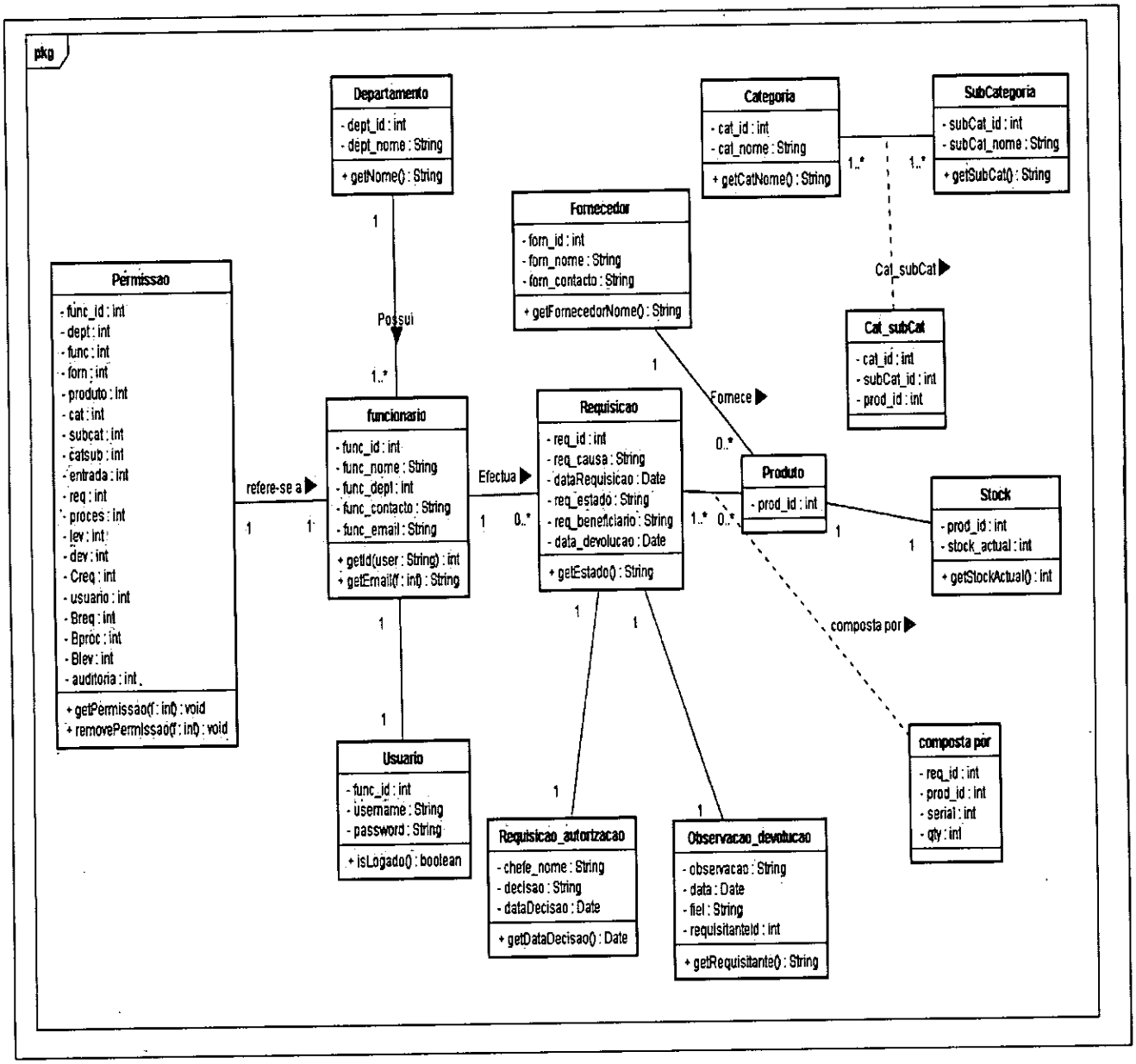


Figura 5. Diagrama de Classe

5.1.3 Diagrama de Actividades

O diagrama de actividade constitui um elemento de modelação simples, mas eficaz para descrever fluxo de trabalho numa organização, a sequência de actividades de um sistema ou para detalhar as operações de uma classe, incluindo comportamentos que possuam processamento paralelo.

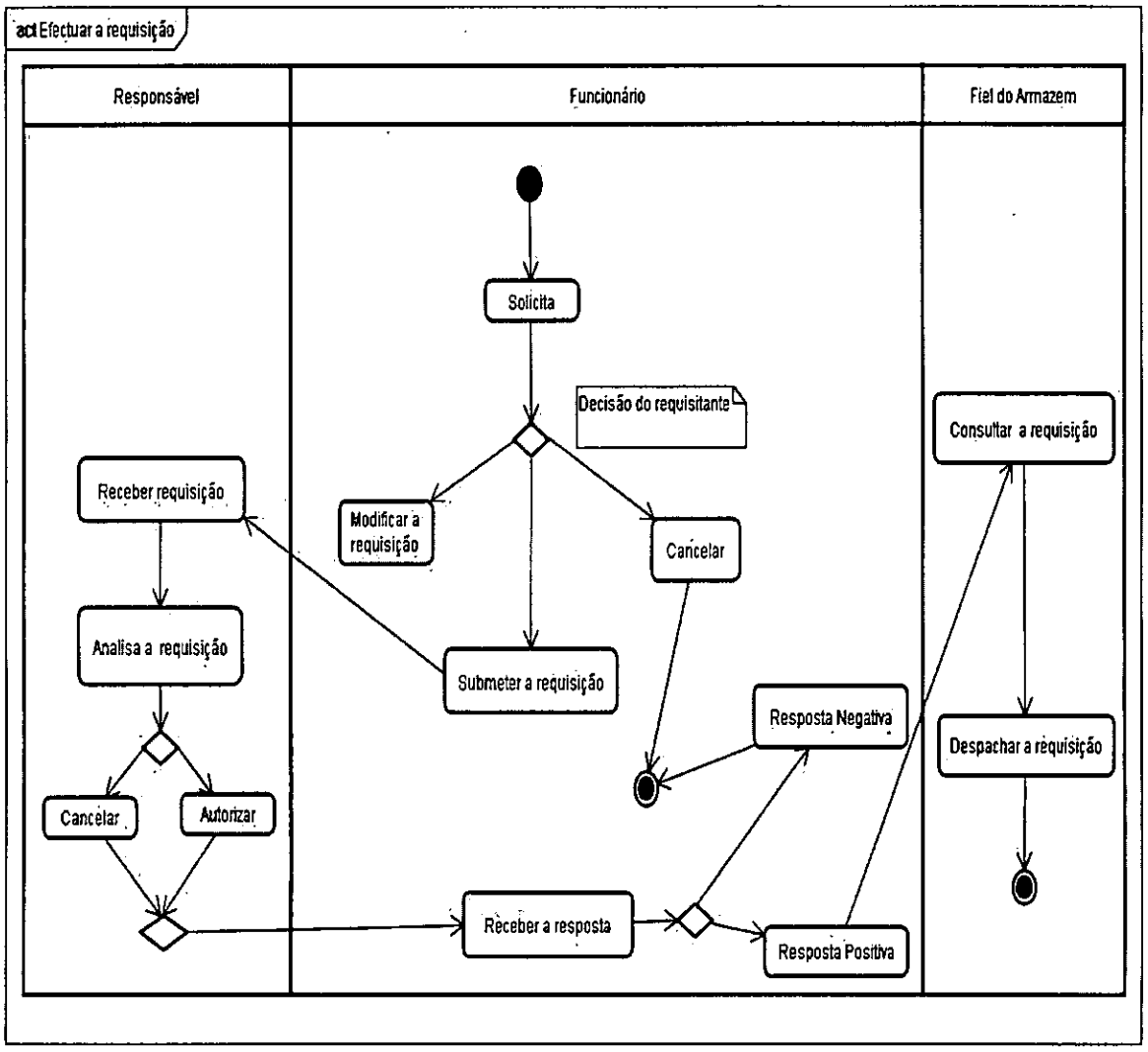


Figura 6. Diagrama de Actividade do processo efectuar a requisição.

5.1.4 Diagramas de Interacção

Diagrama de interacção é uma designação genérica que, em UML, se aplica a diagrama de sequência e colaboração.

5.1.4.1 Diagrama de Sequência

O diagrama de sequência é um diagrama de interacção que realça a ordem cronológica das mensagens entre objectos.

As figuras a seguir apresentam o diagrama de sequência do caso de uso efectuar e processar a requisição respectivamente.

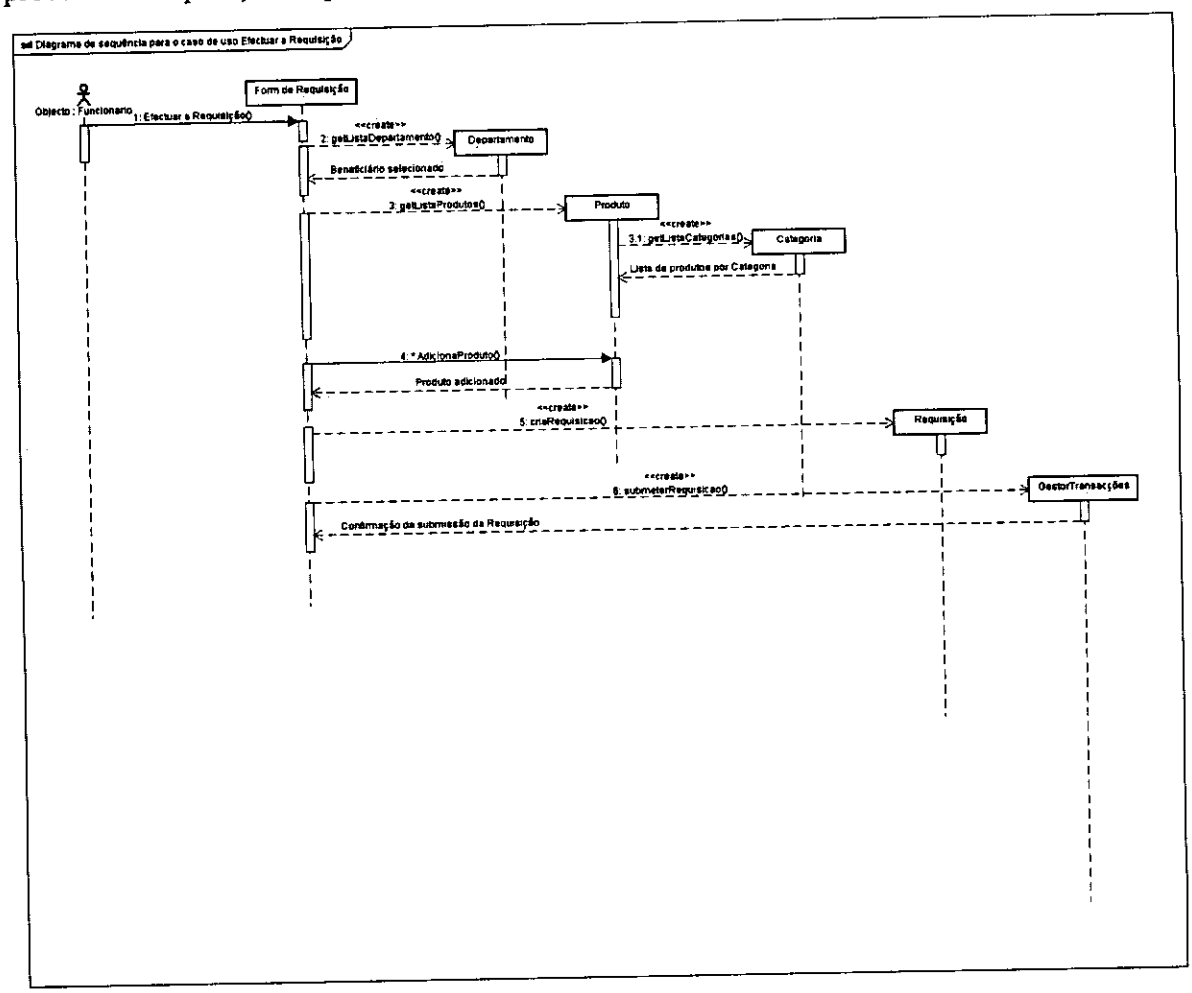


Figura 7. Diagrama de sequência do caso de uso efectuar a requisição.

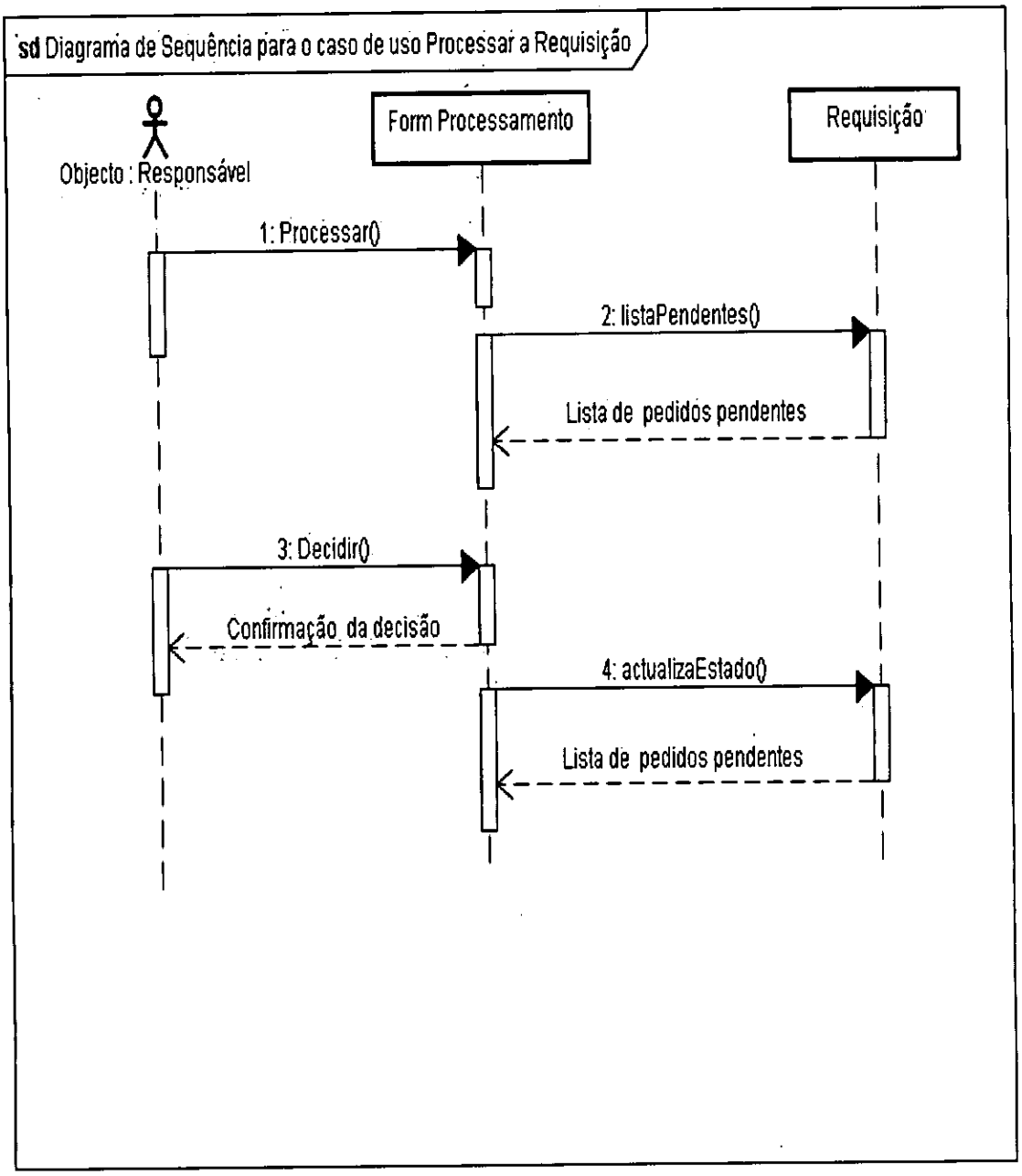


Figura 8. Diagrama de sequência do caso de uso processar a requisição.

5.1.4.2 Diagrama de Colaboração

O diagrama de colaboração privilegia a organização estrutural dos objectos que recebem e enviam mensagens. Este diagrama demonstra apenas a interacção entre os objectos.

5.1.5 Diagramas de Estados

O diagrama de estados é utilizado para descrever o comportamento de um objecto. Um estado representa uma situação estável de um objecto que se prolonga durante um intervalo de tempo, durante o qual os atributos não sofrem qualquer alteração de valor, nem o objecto sofre estímulos externos.

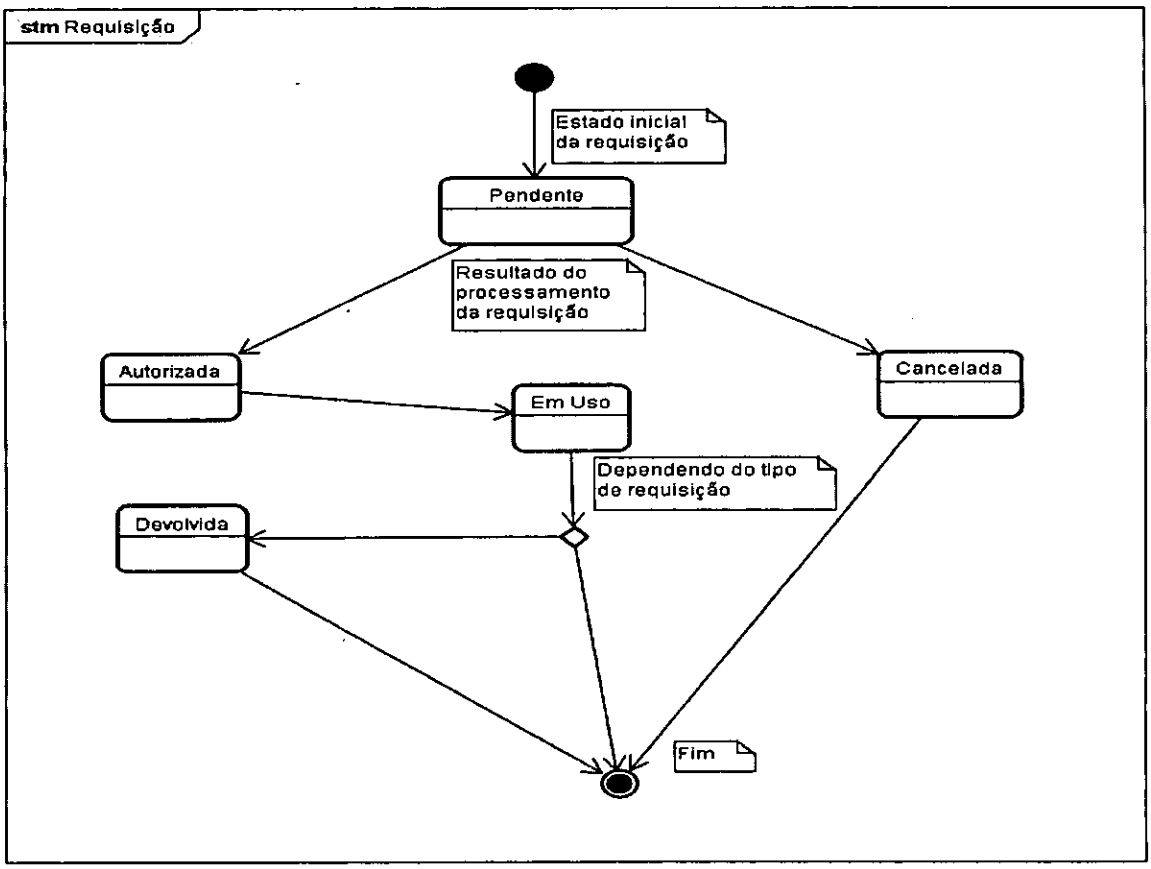


Figura 9. O diagrama de estados da requisição.

5.1.6 Diagramas de Componentes

Um diagrama de componentes mostra um conjunto de componentes e suas relações. Para o caso em estudo podemos ter o seguinte diagrama:

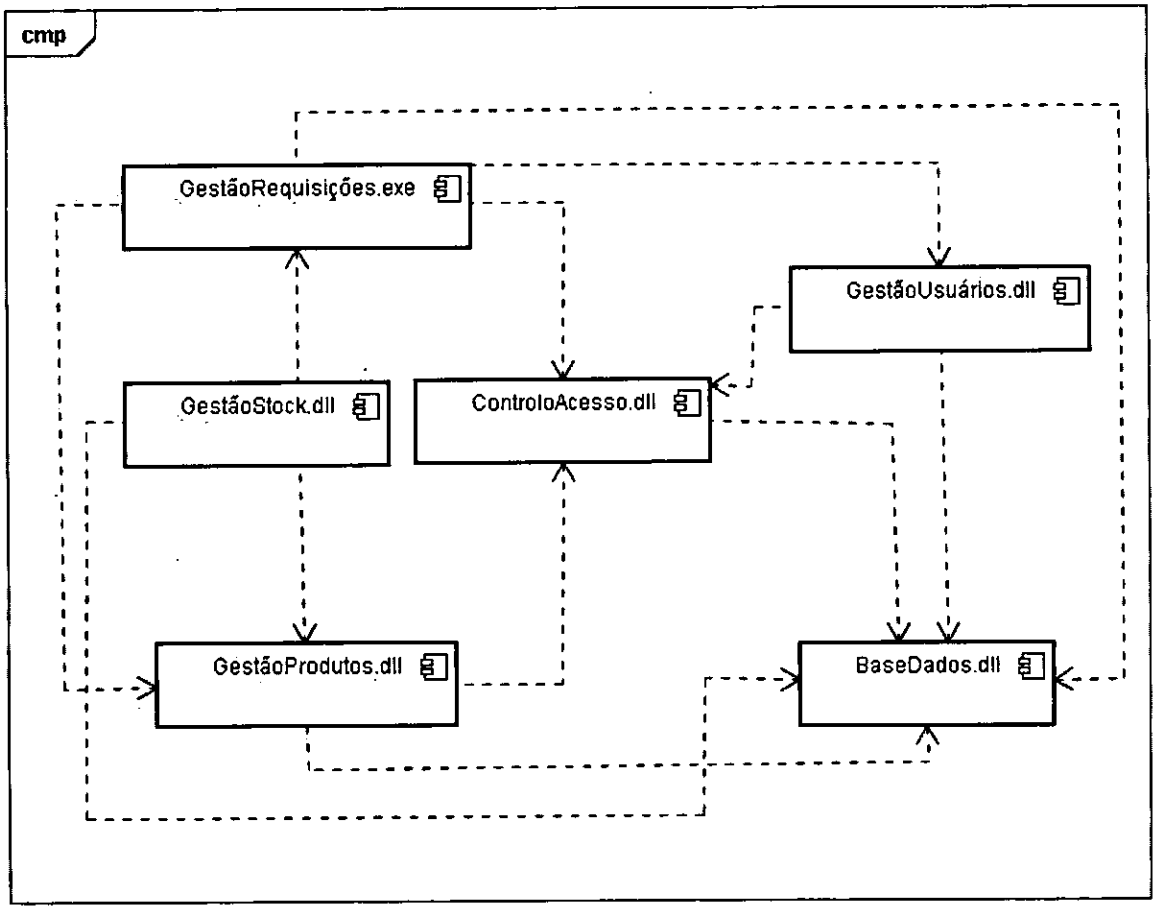


Figura 10. Diagrama de componentes para o caso de Estudo

5.1.6.1 Descrição dos Componentes

Componente	Descrição
ControloAcesso.dll	Responsável por conter as regras e a política de acesso às operações e objectos de sistema. Depende apenas do componente *BaseDados.dll* para guardar a sua informação
GestãoRequisições.exe	Responsável por todas operações relacionadas com as requisições (efectuar, processar, levantar, devolver). Depende do componente *ControloAcesso.dll* para verificar se o usuário tem permissões para executar as operações. Também precisa dos outros componentes (GestãoProdutos.dll, BaseDados.dll, GestãoUsuários.dll), pois deve guardar informações sobre os produtos requisitados e os requisitantes.
GestãoProdutos.dll	Encarregue por todas as operações relativas à gestão de produtos (classificação, entradas, etc.). Depende do componente *ControloAcesso.dll* para verificar se o usuário tem permissões para executar as operações. Também Depende do componente *BaseDados.dll* para guardar a sua informação.
GestãoStock.dll	Encarregue por todas as operações relativas à gestão de stock dos produtos. Depende dos componentes *GestãoProdutos.dll e GestãoRequisições.exe.* Depende também do componente *BaseDados.dll* para guardar a sua informação.
GestãoUsuários.dll	Responsável por gerir os utilizadores do sistema e atribuir privilégios.

Tabela 5. Descrição de Componentes

6. Segurança do sistema

Nos últimos anos, os sistemas de rede têm crescido consideravelmente em tamanho, complexidade e susceptibilidade à ataques. Ao mesmo tempo, o conhecimento, as ferramentas e as técnicas disponíveis aos agressores têm crescido com a mesma rapidez ou até mais rápido (Symantec, 2004).

Infelizmente, as técnicas de defesa não têm crescido tão rápido. As tecnologias actuais vêm apresentando limitações, sendo necessárias soluções inovadoras para lidar com o nível das ameaças actuais.

Visto que o modelo proposto funciona num ambiente em rede, e mais de um usuário tem acesso ao mesmo, ele não está isento de vulnerabilidades e de malfeitores. Assim, torna-se necessário a implementação de medidas de Segurança da Informação, tal como o uso de chaves de acesso ao sistema, garantindo o acesso restrito da informação.

6.1 Segurança da informação

Segurança de Informação está relacionada com a protecção existente ou necessária sobre dados que possuem valor para alguém ou para uma organização. Possui aspectos básicos como confidencialidade, integridade e disponibilidade da informação que nos ajuda a entender as necessidades de sua protecção e que não se aplica ou está restrita a sistemas computacionais, nem a informações electrónicas ou qualquer outra forma mecânica de armazenamento. Ela se aplica a todos os aspectos de protecção e armazenamento de informações e dados, em qualquer forma.

6.2 Conceitos de segurança

A Segurança da Informação refere-se à protecção existente sobre as informações de uma determinada empresa ou pessoa, isto é, aplica-se tanto as informações corporativas quanto as pessoais.

Entende-se por informação todo e qualquer conteúdo ou dado que tenha valor para alguma organização ou pessoa. Ela pode estar guardada para uso restrito ou exposta ao público para consulta ou aquisição.

Podem ser estabelecidas métricas (com o uso ou não de ferramentas) para a definição do nível de segurança existente e, com isto, serem estabelecidas as bases para análise da melhoria ou piora da situação de segurança existente.

A segurança de uma determinada informação pode ser afectada por factores comportamentais e de uso de quem se utiliza dela, pelo ambiente ou infra-estrutura que a cerca ou por pessoas mal intencionadas que tem o objectivo de furtar, destruir ou modificar a informação.

A tríade CIA (*Confidentiality, Integrity and Availability*) - Confidencialidade, Integridade e Disponibilidade - representa as principais propriedades que, anualmente, orientam a análise, o planeamento e a implementação da segurança para um determinado grupo de informações que se deseja proteger.

Outras propriedades estão sendo apresentadas (legitimidade e autenticidade) na medida em que o uso de transacções comerciais em todo o mundo, através de redes electrónicas (públicas ou privadas) se desenvolve.

Os conceitos básicos podem ser explicados conforme abaixo:

- *Confidencialidade* - propriedade que limita o acesso a informação tão somente às entidades legítimas, ou seja, àquelas autorizadas pelo proprietário da informação.
- *Integridade* - propriedade que garante que a informação manipulada mantenha todas as características originais estabelecidas pelo proprietário da informação,

incluindo controlo de mudanças e garantia do seu ciclo de vida (nascimento, manutenção e destruição).

- *Disponibilidade* - propriedade que garante que a informação esteja sempre disponível para o uso legítimo, ou seja, por aqueles usuários autorizados pelo proprietário da informação.

O nível de segurança desejado, pode se consubstanciar em uma "*política de segurança*" que é seguida pela organização ou pessoa, para garantir que uma vez estabelecidos os princípios, aquele nível desejado seja perseguido e mantido.

Para a montagem desta política, tem que se ter em conta:

- Os riscos associados à falta de segurança,
- Os benefícios,
- Os custos de implementação dos mecanismos.

6.3 Mecanismos de segurança implementada para o sistema de gestão de Stock

Para garantir a segurança do sistema tomou-se as seguintes medidas:

- Todos os utilizadores do sistema têm username e password que os identificam com a possibilidade de alteração do password sempre que assim o desejarem.

- Os usuários têm privilégios diferentes que se reflectem logo que efectuam o login na aplicação pois nem todas as funcionalidades são vistas por todos.

- Todas as operações efectuadas no sistema assim como as horas são registadas, facilitando deste modo a auditoria do sistema

7. Discussão

7.1 Estudo comparativo entre metodologias ágeis e metodologias Tradicionais

Nos últimos tempos, a indústria de software vem passando por grandes transformações e novos desafios, dentre os quais podemos mencionar: desenvolver softwares com qualidade, no menor tempo possível e que atendam as necessidades dos clientes (Sommerville, 2003).

Com estes novos desafios, a indústria de software passou a dar valor a algumas áreas da informática, como a de engenharia de software e qualidade de software, com intuito de atender as exigências do mercado.

Tendo este alvo a atingir, a indústria começou a utilizar metodologias de desenvolvimento de software, adoptou métricas e padrões para alcançar níveis aceitáveis de qualidade, prever custos e prazos em seus projectos. Porém, ainda são poucos os projectos que conseguem obter pleno sucesso em seu desenvolvimento, onde prazo, orçamento estabelecidos e as necessidades do cliente sejam realmente atendidos.

As metodologias utilizadas nos projectos são várias mas a título de exemplo, podemos mencionar o modelo em cascata, e modelo iterativo. Analisando os motivos para a baixa taxa de sucesso dos projectos desenvolvidos com modelos tradicionais, temos como principais motivos os seguintes:

- Tempo elevado entre cada fase do projecto, não acompanhando as mudanças de requisitos do projecto;
- Falta de conhecimento por parte do cliente da sua real necessidade, dando margem às especulações dos desenvolvedores;
- Forte linearidade no desenvolvimento do projecto;

Enfocando nas fragilidades do modelo tradicional, surgiram metodologias para desenvolvimento ágil de software.

Algumas características destas metodologias são:

- Enfoque nas pessoas, não no processo, evitando especulações dos desenvolvedores;
- Atendimento das reais necessidades do cliente;
- Ausência de linearidade no desenvolvimento do projecto;
- Aprendizagem, por parte do cliente das suas reais necessidades durante o projecto e o repassar destas novas necessidades à equipa de desenvolvimento;

A tabela a seguir compara a XP com metodologias tradicionais

Extreme Programming	***Metodologias Tradicionais***
Enfoque nas pessoas	Enfoque nos processos
Menos tempo com documentação e mais com a implementação	Maior tempo gasto na documentação
Flexível à mudança dos requisitos	Aplicadas apenas em situações em que os requisitos do software são estáveis e requisitos futuros são previsíveis.
Entregas constantes de partes operacionais do software	O cliente deve esperar muito para ver o funcionamento do software
Participação total do cliente durante o desenvolvimento do software	O cliente participa nas fases iniciais de requisitos e das validações dos produtos.
Comunicação informal	Comunicação formal

Tabela 6. Estudo comparativo das metodologias Ágeis e Tradicionais

7.2 Implementação da XP neste trabalho

Os quatros valores (feedback, comunicação, simplicidade e coragem) recomendados por esta metodologia foram praticados, pois o cliente (a STV) conduziu o desenvolvimento do seu produto, estabeleceu prioridades e informou aquilo que era realmente importante embora não fosse da iniciativa dele porque como já mencionado, esta aplicação foi proposta no âmbito do estágio que o candidato realizou na instituição em causa.

As linhas de comunicação foram mantidas abertas durante todo processo e a coragem foi necessária para propor ao cliente a mudança de alguns procedimentos usados há vários anos na organização e, neste contexto, muitas versões do sistema foram apresentadas com muita simplicidade. Todavia, algumas práticas sugeridas por XP não foram implementadas. Por exemplo, sendo o único analista de sistema e programador envolvido no projecto, não era possível falar da programação a dois, código colectivo, trabalho em equipa. Mesmo assim, o código de aplicação foi mantido simples, fácil de entender e suficientemente comentado para o uso posterior no ambiente de desenvolvimento em equipa.

7.3 Impacto da implementação da GStock na STV

A implementação da Gstock na STV teve um impacto positivo de modo que tanto os funcionários assim como os gestores têm tirado benefícios notáveis. Reduziu-se significativamente o tempo de resposta no atendimento dos pedidos efectuados, há menos burocracia, e o controlo eficiente do stock e bem como uso de papel no processo de pedido de requisição ficou para a história.

Para além do que anteriormente referimos, os gestores facilmente fazem a análise de gastos e conseguem responder às necessidades da empresa na parte de abastecimento dos produtos frequentemente utilizados, fazendo uso dos relatórios produzidos pelo sistema.

7.4 Problemas enfrentados na implementação da GStock

A ordem da implementação da GStock foi muito demorada. Tendo chegado quatro meses depois da apresentação da aplicação na sala de reuniões da STV, questionou-se até que ponto a aplicação responderia as expectativas.

Alguns factores que contribuíram para a demora foram:

- Alocação de um computador no armazém.
- Receio de substituição do poder de decisão de Responsável pela aplicação por não perceber exactamente o que significaria a assinatura de autorização da requisição nesta nova abordagem.
- Resistência à mudança.

Outro factor foi a própria infrastrutura tecnológica que a Televisão dispõe (refiro-me à qualidade da rede local). A rede sofre várias oscilações e a GSTOCK, por depender dela para o seu bom funcionamento acaba por sofrer as consequências. Em resultado desta situação, muitas vezes os utilizadores se queixavam por que a aplicação falhava quando tentassem aceder a máquina servidora da Base de Dados.

Para contornar este problema, procurou-se reduzir o número de acessos à base de dados central, criando uma base de dados local (incorporada no acto de instalação) que ao arrancar a aplicação, é sincronizada com a base de dados central através de uma rotina (anexo 10.5). Importa referir que esta base de dados local em Access, contém dados que não sofrem alterações constantes, como a lista de departamentos e a de produtos. Até então, esta solução resultou na estabilidade da aplicação, enquanto se resolve a questão da melhoria da rede local.

8. Conclusões e recomendações

8.1 Conclusões

Chegou-se às seguintes conclusões:

1. Para desenvolver um sistema de informação para uma organização, os envolvidos devem perceber de antemão o motivo da sua implementação e que benefícios terão, caso decidam optar pelas tecnologias de informação e comunicação.
2. A XP é uma metodologia ágil para desenvolvimento de software, com qualidade e que atenda as necessidades do cliente.
3. A Aplicação da metodologia extreme programming mostrou-se benéfica e ponctual uma vez que os requisitos do sistema mudavam constantemente e exigia-se uma participação activa do cliente durante todo processo de desenvolvimento.
4. O plano de manutenção do sistema deixou muito a desejar porque a metodologia usada neste trabalho não dá muita ênfase à esta fase de ciclo de desenvolvimento de software. Como já fez-se referência, não foi possível implementar na totalidade as práticas e regras sugeridas por extreme programming, principalmente a programação a dois, o que não permitiu o entendimento do código usado na programação se estendesse a outros.
5. A UML mostrou-se benéfica para especificação, construção, visualização e documentação do sistema de informação para gestão de stock da STV, permitindo um fácil desenvolvimento do mesmo.
6. O Sql Server 2000 é um Sistema de Gestão de Base de Dados relacional, robusto, confiável. O Visual Basic.Net permitiu a programação puramente orientada à objectos. O *Crystal Report* mostrou-se capaz e eficaz na produção de relatórios que envolvem dados guardados em várias tabelas.

8.2. Recomendações

De modo a resolver os actuais problemas de gestão do stock da STV, com recurso aos resultados do presente trabalho recomenda-se que:

- A STV possua uma rede local consistente e estável, para permitir a continuidade da utilização deste sistema de uma forma persistente e estável.
- A STV Adquira um data base server que possa armazenar os dados e crie a política de backup por que neste momento toda informação referente às requisições estão armazenadas na base de dados, numa máquina que não consegue responder eficientemente aos pedidos frequentes dos utilizadores.
- A Aplicação seja adaptada a fim de resolver a questão de controlo de património da empresa (todos bens adquiridos) e o registo do valor monetário dos produtos que entram no armazém para avaliação de custo em função de tempo no caso de abate, por exemplo.

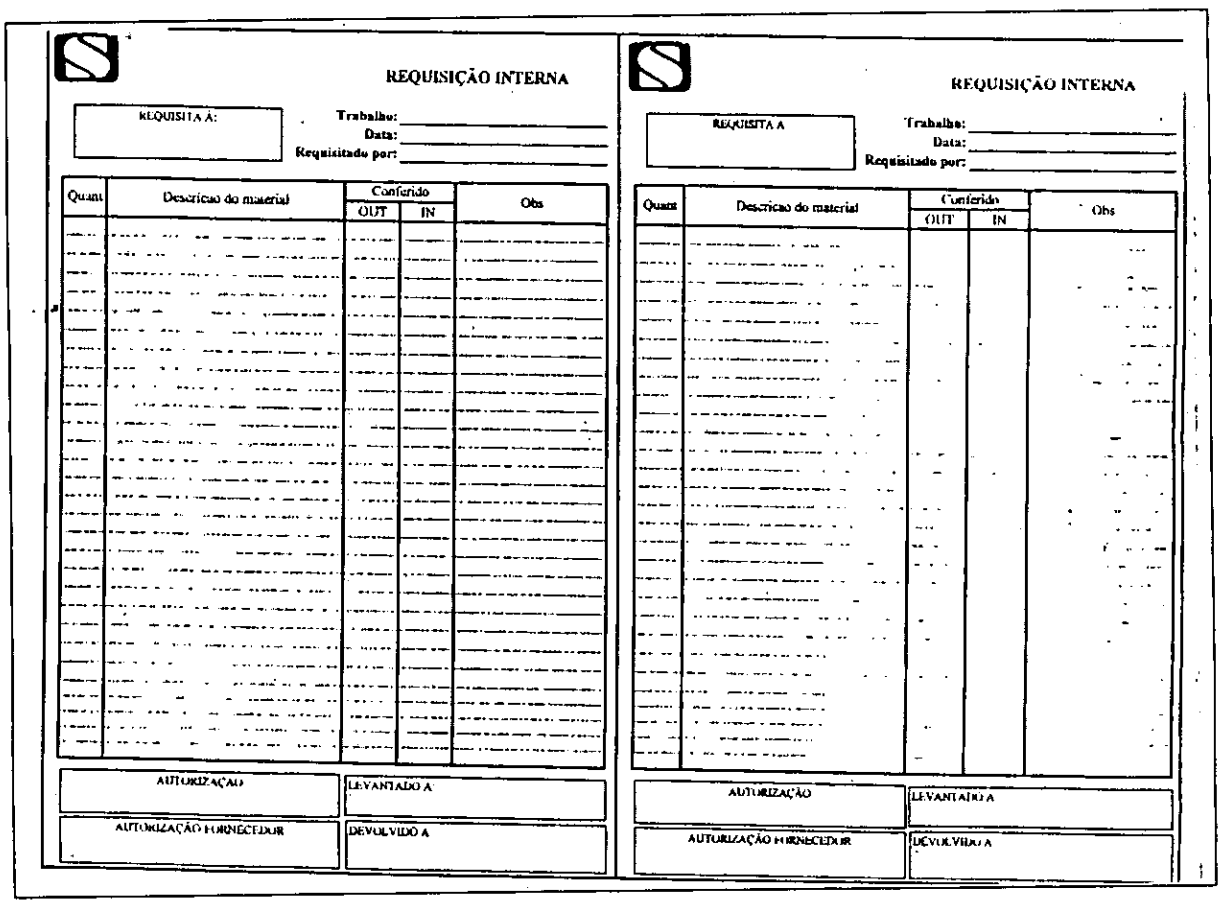
9. Referências

- Camarão, P.C.B. (1994). Glossário de Informática, Segunda edição revista e ampliada, Rio de Janeiro, LTC Livros Técnicos e Científicos.
- Carneiro, C. (2002). Introdução à Segurança de Sistemas de Informação, Lisboa, FCA Editora de Informática.
- CARTER,ANTÓN CARTER, Ryan A. ANTÓN, Aldo Dagnino. WILLIAMS, Laurie. Envolving Beyond Requirements Creep : A Risk-Based Evolutionary Prototyping Model. IEEE – 2001
- Damas, L.(2005). SQL, Lisboa, Sexta Edição actualizada e aumentada, FCA Editora de Informática.
- Eriksson, H. e M. Penker (2000). Business Modeling with UML, New York, Wiley Computer Publishing.
- FREITAS, Ricardo Wragg. Prototipagem de Software. FEUP – Universidade do Porto – Faculdade de Engenharia. 2002
- FRITZKE JR., Udo. Elementos da Engenharia de Software. PUC Minas – Poços de Caldas.
- GORDON,BIEMAN GORDON, Scott. BIEMAN, James M. Rapid Prototyping: Lessons Learned. IEEE Software – 1995
- http://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objecto acessado aos 26 de Agosto de 2008 pelas 15:13
- http://pt.wikipedia.org/wiki/Visual_Studio_.NET acessado aos 26 de Agosto de 2008 pelas 10:07
- Kent Beck, Cynthia Andres. Extreme Programming Explained: Embrace Change, Second Edition. Addison Wesley Professional.2004
- Larman, C. (2002). Applying UML And Patterns An Introduction To Object-Oriented Analysis And Design And The Unified Process, Second Edition. New York, Prentice Hall PTR.
- Larman, C. e V. Basili (June, 2003), "Iterative and Incremental Development: A Brief History"

- Laudon, C. E Laudon, J. (2000). Management Information Systems, 6th edition, Prentice Hall, New Jersey.
- Macome, E (1995). Introdução a Metodologia de Investigação, Maputo, Universidade Eduardo Mondlane.
- Martin Fowler (2001): *Refactoring: Improving the Design of Existing Code*, Addison-Wesley,
- Miguel, A. (2003). Gestão de Projectos de Software, Lisboa, FCA Editora de Informática.
- MOLINARI, Leonardo. Testes de Software. Editora Érica. 2003
- NOGUEIRA,LUQI NOGUEIRA, Juan Carlos. LUQI. BHATTACHARYA, Swapan. A Risk Assessment Model for Software Prototyping Project. IEEE – 2000
- Nunes, M. e O´Neill (2001). Fundamental de UML. Lisboa, FCA Editora de Informática.
- PAULA, FILHO, Wilson de Pádua Paula. Engenharia de Software. Editora LTC. 2003
- Pereira, J.L. (1998). Tecnologia de Base de Dados, Lisboa, Segunda Edição actualizada e aumentada, FCA Editora de Informática.
- PRESSMAN, Roger S. Software Engineering – A practitioner's Approach. Mc Graw Hill. 5ª Edição. 2002
- Serrano et al (2004). Gestão de Sistemas e Tecnologias de Informação, Lisboa, FCA
- SOMMERVILLE, Ian. Engenharia de Software. Editora Makron Books. 2003
- YOURDON, Edward. Administrando o Ciclo de Vida do Sistema. Editora Campus. 1992

10. Anexos

10.1. Folha de requisição



REQUISIÇÃO INTERNA

REQUISITA À: | Trabalho: | Data: | Requisitado por:

Quant	Descrição do material	Conferido OUT	Conferido IN	Obs

AUTORIZAÇÃO | LEVANTADO A

AUTORIZAÇÃO FORNECEDOR | DEVOLVIDO A

REQUISIÇÃO INTERNA

REQUISITA A | Trabalho: | Data: | Requisitado por:

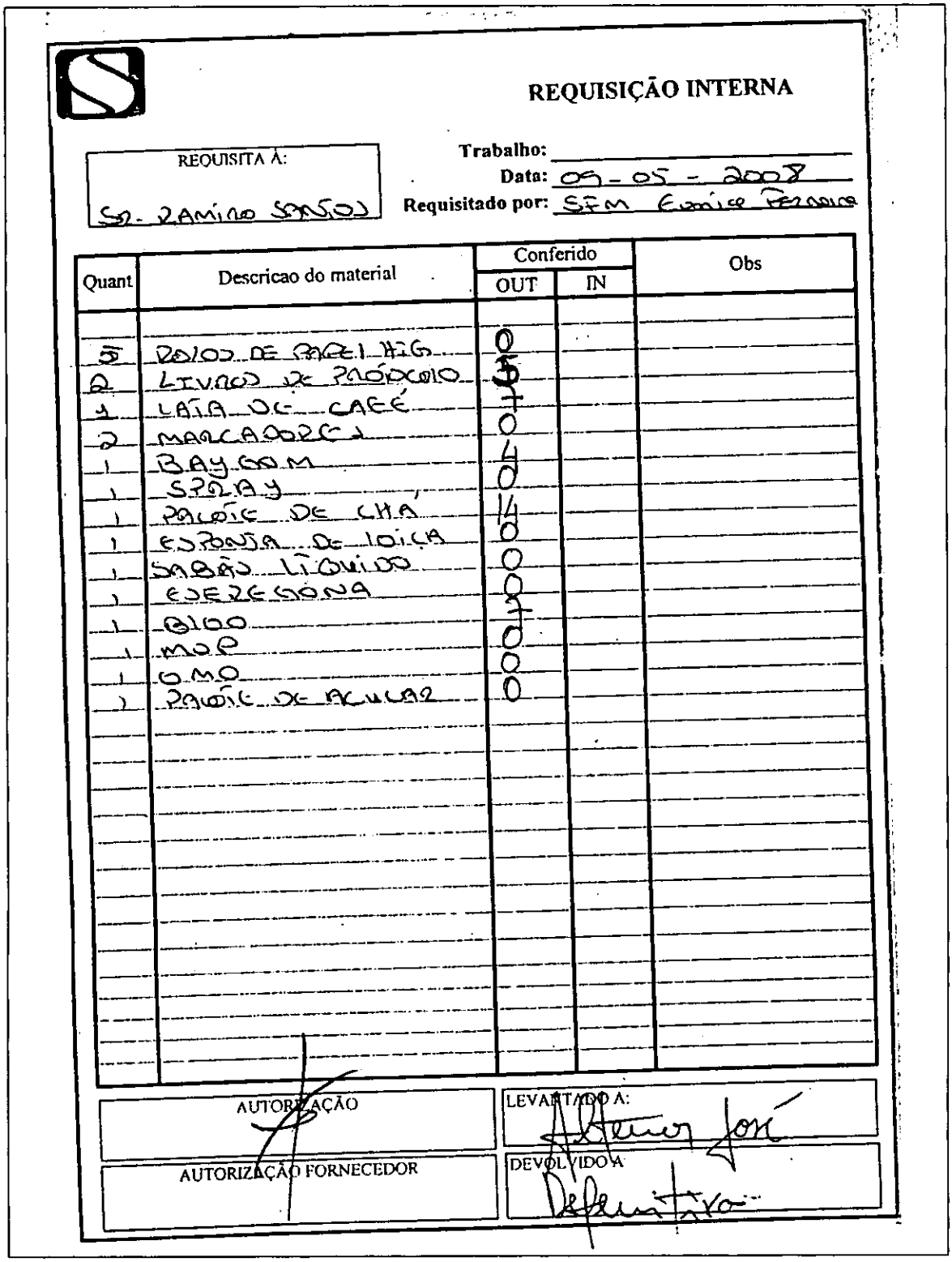
Quant	Descrição do material	Conferido OUT	Conferido IN	Obs

AUTORIZAÇÃO | LEVANTADO A

AUTORIZAÇÃO FORNECEDOR | DEVOLVIDO A

Figura 10. Folha de Requisição

10.2. Folha de requisição preenchida



REQUISIÇÃO INTERNA

REQUISITA À: Sr. RAMIRO SANSO

Trabalho:
Data: 09 - 05 - 2008
Requisitado por: SFM Eunice Ferreira

Quant	Descricao do material	Conferido OUT	Conferido IN	Obs
5	ROLO DE PAPEL HIG.	0		
2	LIVRO DE PAÓDIO	0		
1	LATA DE CAFÉ	0		
2	MARCADORES	0		
1	BAYGAM	0		
1	SPRAY	0		
1	PACOTE DE CHA	0		
1	ESPONJA DE LOUÇA	0		
1	SABÃO LÍQUIDO	0		
1	ESFREGONA	0		
1	BLOO	0		
1	MOP	0		
1	OMO	0		
1	PACOTE DE AÇUCAR	0		

AUTORIZAÇÃO

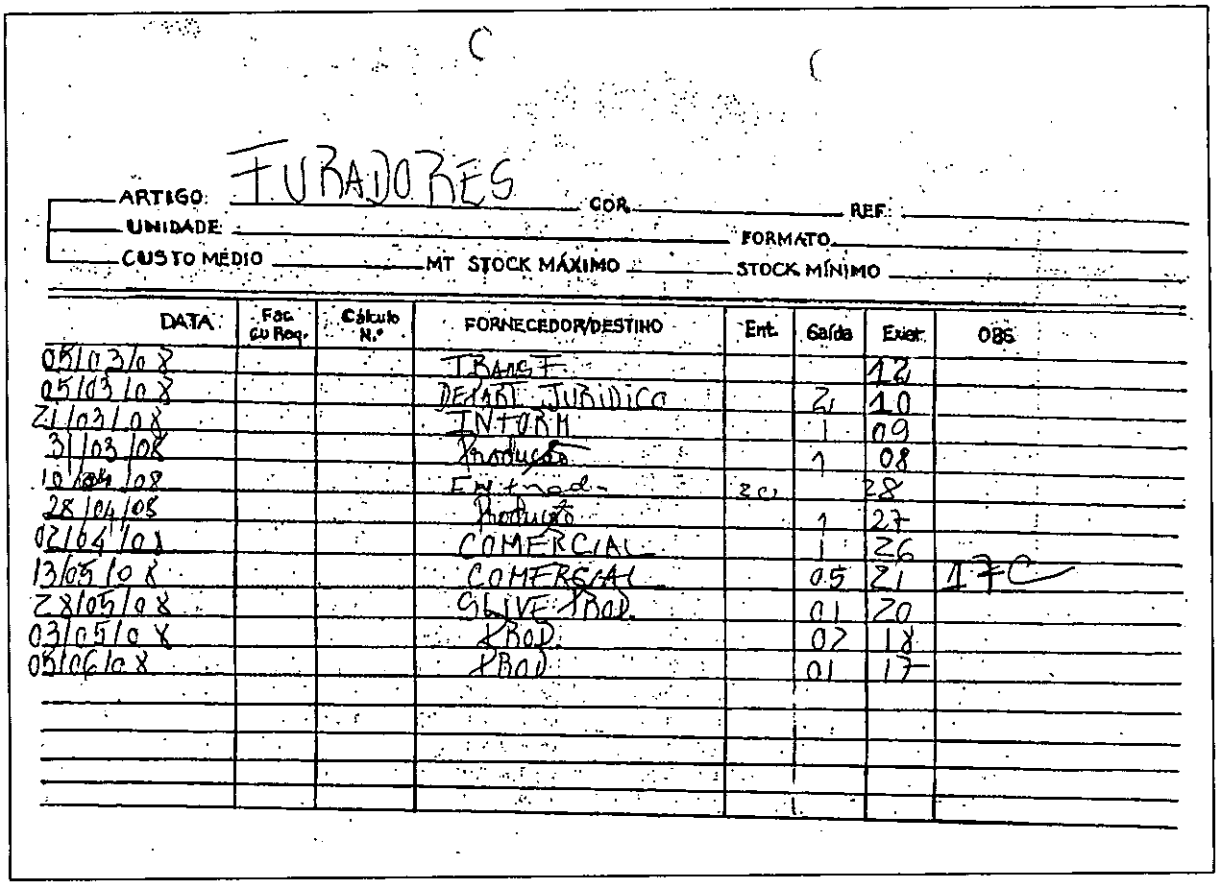
LEVANTADO A:

AUTORIZAÇÃO FORNECEDOR

DEVOLVIDO A:

Figura 11. Folha de Requisição preenchida

10.3. Ficha de Controlo de Stock



ARTIGO: FURADORES COR: REF:
UNIDADE: FORMATO:
CUSTO MÉDIO: MT STOCK MÁXIMO: STOCK MÍNIMO:

DATA	Fac ou Req.	Cálculo N.º	FORNECEDOR/DESTINO	Ent.	Saída	Exist.	OBS
05/03/08			IBANEF			12	
05/03/08			DEPART. JURÍDICO		2	10	
21/03/08			INTERN		1	09	
31/03/08			Produção		1	08	
10/04/08			Entrad.	20		28	
28/04/08			Produção		1	27	
02/04/08			COMERCIAL		1	26	
13/05/08			COMERCIAL		05	21	IFC
28/05/08			OLIVE PROD.		01	20	
03/05/08			PROD.		02	18	
05/06/08			PROD		01	17	

Figura 12. Ficha de Controlo de Stock de Furador

10.4. Descrição de Uses Cases

Nome	***Cadastro de departamentos (UC003)***
Pré-condição	O administrador do Sistema é um utilizador válido do sistema
Actor	Administrador do Sistema
Descrição	1. O use case começa quando o administrador do Sistema clica no menu cadastrar, submenu departamento. 2. O administrador do Sistema preenche o nome do departamento e adiciona sucessivamente na lista clicando no botão adicionar. 3. O administrador por fim clica no botão incluir para adicionar os departamentos que se encontram na lista. 4. O sistema manda uma mensagem de confirmação .
Pós-condição	

Nome	***Cadastro de Funcionários (UC004)***
Pré-condição	O administrador do Sistema é um utilizador válido do sistema
Actor	Administrador do Sistema
Descrição	1. O use case começa quando o administrador do Sistema clica no menu cadastrar, Funcionário. 2. O administrador do Sistema preenche o nome e apelido do funcionário separados pelo espaço. 3. O sistema automaticamente cria o email do funcionário seguindo o formato da empresa, por exemplo se preencher Botomo Ngongo, o sistema irá criar o email botomo.ngongo@soico.co.mz 4. O administrador do Sistema clica no botão procurar para lançar o departamento em que pertence o funcionário. 5. O sistema abre a lista de departamentos previamente cadastrados e o administrador do sistema selecciona o departamento dando duplo clique. 6. Por fim o administrador clica no botão gravar e o sistema responde por uma mensagem de confirmação
Pós-condição	

Nome	***Cadastro de Utilizadores do Sistema (UC005)***
Pré-condição	O administrador do Sistema é um utilizador válido do sistema
Actor	Administrador do Sistema
Descrição	1. O use case começa quando o administrador do Sistema clica no menu configuração, sub menu cadastro de utilizadores. 2. O administrador do Sistema clica no botão Go para seleccionar o funcionário. 3. O sistema automaticamente verifica se trata-se de um funcionário já utilizador de sistema. Se não for o administrador o selecciona. 4. O sistema automaticamente propõe o username e password 5. O administrador cria preenche o username e password 6. O sistema verifica se já existe um usuário com o username dado. Caso não o administrador atribui privilégios. 7. Por fim o administrador clica no botão adicionar e o sistema responde por uma mensagem de confirmação.
Pós-condição	

Nome	***Cadastro de Categoria de Produtos (UC006)***
Pré-condição	O Fiel do Armazém é um utilizador válido do sistema
Actor	Fiel do Armazém
Descrição	1. O use case começa quando o administrador do Sistema clica no menu configuração , sub menu cadastro de utilizadores . 2. O administrador do Sistema clica no botão Go para seleccionar o funcionário. 3. O sistema automaticamente verifica se trata-se de um funcionário já utilizador de sistema. Se não for o administrador o selecciona. 4. O sistema automaticamente propõe o username e password 5. O administrador cria preenche o username e password 6. O sistema verifica se já existe um usuário com o username dado. Caso não o administrador atribui privilégios. 7. Por fim o administrador clica no botão adicionar e o sistema responde por uma mensagem de confirmação .
Pós-condição	

10.5. Codigo fonte de alguns métodos

```
Public Class Sincronizador
    Public Sub IncluirProduto()
        Dim rs As ADODB.Recordset = New ADODB.Recordset
        With rs
            .Open("select * from produto", c,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            Dim rsA As ADODB.Recordset = New ADODB.Recordset
            rsA.Open("select * from produto", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            While Not rs.EOF
                rsA.AddNew()
                rsA.Fields("prod_id").Value =
rs.Fields("prod_id").Value.ToString
                rsA.Update()
                rs.MoveNext()
            End While
            rsA.Close()
            .Close()
        End With
    End Sub
    Public Sub deleteProduto()
        Dim rs As ADODB.Recordset = New ADODB.Recordset
        With rs
            .Open("delete * from produto", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
        End With
    End Sub
```

```
    Public Sub incluirCategoria()

        Dim rsCat As ADODB.Recordset = New ADODB.Recordset

        With rsCat
            .Open("select * from categoria", c,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            Dim rsA As ADODB.Recordset = New ADODB.Recordset
            rsA.Open("select * from categoria", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            While Not rsCat.EOF
                rsA.AddNew()
                rsA.Fields("cat_id").Value =
rsCat.Fields("cat_id").Value.ToString
                rsA.Fields("cat_nome").Value =
rsCat.Fields("cat_nome").Value.ToString
                rsA.Update()
                rsCat.MoveNext()
            End While
            rsA.Close()
            .Close()
        End With
    End Sub

    Public Sub deleteCategoria()
        Dim rs As ADODB.Recordset = New ADODB.Recordset
        With rs
            .Open("delete * from categoria", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
        End With
    End Sub
```

```
    Public Sub incluirDepartamento()

        Dim rsCat As ADODB.Recordset = New ADODB.Recordset

        With rsCat
            .Open("select * from departamento", c,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            Dim rsA As ADODB.Recordset = New ADODB.Recordset
            rsA.Open("select * from departamento", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            While Not rsCat.EOF
                rsA.AddNew()
                rsA.Fields("dept_id").Value =
rsCat.Fields("dept_id").Value.ToString
                rsA.Fields("dept_nome").Value =
rsCat.Fields("dept_nome").Value.ToString
                rsA.Update()
                rsCat.MoveNext()
            End While
            rsA.Close()
            .Close()
        End With
    End Sub

    Public Sub deleteDepartamento()
        Dim rs As ADODB.Recordset = New ADODB.Recordset
        With rs
            .Open("delete * from departamento", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
        End With
    End Sub
```

```
    Public Sub incluirSubCategoria()

        Dim rsC As ADODB.Recordset = New ADODB.Recordset
        With rsC
            .Open("select * from subcategoria", c,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            Dim rsA As ADODB.Recordset = New ADODB.Recordset
            rsA.Open("select * from subcategoria", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            While Not rsC.EOF
                rsA.AddNew()
                rsA.Fields("subcat_id").Value =
rsC.Fields("subcat_id").Value.ToString
                rsA.Fields("subcat_nome").Value =
rsC.Fields("subcat_nome").Value.ToString
                rsA.Fields("prod_stock_min").Value =
rsC.Fields("prod_stock_min").Value.ToString
                rsA.Fields("prod_tipo").Value =
rsC.Fields("prod_tipo").Value.ToString
                rsA.Fields("serial").Value =
rsC.Fields("serial").Value.ToString
                rsA.Update()
                rsC.MoveNext()
            End While
            rsA.Close()
            .Close()
        End With
    End Sub
```

```
    Public Sub deleteSubcategoria()
        Dim rs As ADODB.Recordset = New ADODB.Recordset
        With rs
            .Open("delete * from subcategoria", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
        End With
    End Sub


    Public Sub associar()
        Dim rsCat As ADODB.Recordset = New ADODB.Recordset
        With rsCat
            .Open("select * from cat_subcat", c,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            Dim rsA As ADODB.Recordset = New ADODB.Recordset
            rsA.Open("select * from cat_subcat", ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
            While Not rsCat.EOF
                rsA.AddNew()
                rsA.Fields("cat_id").Value =
rsCat.Fields("cat_id").Value.ToString
                rsA.Fields("subcat_id").Value =
rsCat.Fields("subcat_id").Value.ToString
                rsA.Fields("prod_id").Value =
rsCat.Fields("prod_id").Value.ToString
                rsA.Update()
                rsCat.MoveNext()
            End While
            rsA.Close()
            .Close()
        End With
    End Sub
```

```
    Public Sub deleteAssociacao()
        Dim rs As ADODB.Recordset = New ADODB.Recordset
        With rs
            .Open("delete       *       from       cat_subcat",       ca,
ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockPessimistic)
        End With
    End Sub

    Public Sub sincroniza()
        Me.Refresh()
        Me.IncluirProduto()
        Me.incluirDepartamento()
        Me.incluirSubCategoria()
        Me.associar()
        Me.incluirCategoria()
    End Sub

    Public Sub Refresh()
        Me.deleteProduto()
        Me.deleteDepartamento()
        Me.deleteSubcategoria()
        Me.deleteAssociacao()
        Me.deleteCategoria()
    End Sub
End Class
```

10.6. Manual de Utilizador

Sociedade Independente de Comunicação

SOICO Televisão

(STV)

Manual de Utilizador de Gestão de Stock

Departamento Técnico

Sector de Interactividade e Desenvolvimento de Sistema

1. Introdução

A implementação de um sistema de informação é uma fase muito importante na engenharia de Software. Nesta fase o sistema desenvolvido é submetido a testes para averiguar se ele está pronto a satisfazer os requisitos delineados na fase de análise.

A GSTOCK foi concebida para facilitar o processo de pedido de material de armazém, controlando assim a situação de stock no tempo real, a fim de produzir relatórios quantitativos que possam ajudar os gestores a tomarem decisões de uma forma rápida e concisa, tendo os dados rapidamente processados e sujeitos a análise.

2. Tipos de Utilizador

A *GSTOCK* terá quatro tipos de utilizador.

1. **Administrador do Sistema**: Este vai cuidar do cadastro de usuários e atribuições de privilégios, atribuição de username e password
2. **Usuário Normal**: Qualquer funcionário potencial que possa precisar de utilizar o sistema. Este pode efectuar a requisição, ver a lista de suas requisições e mudar seu password
3. **Fiel do Armazém**: É um usuário normal que tem ainda o privilegio de cadastrar os fornecedores, categorias de produtos (Informática, Electrica, etc.), subcategoria (Impressora, PC, Toalha, etc.), lançar as entradas, efectuar o levantamento e devolução de material autorizado
4. **Responsável**: É a pessoa que processa as requisições, consulta e analisa o inventário de stock, faz a auditoria de sistema

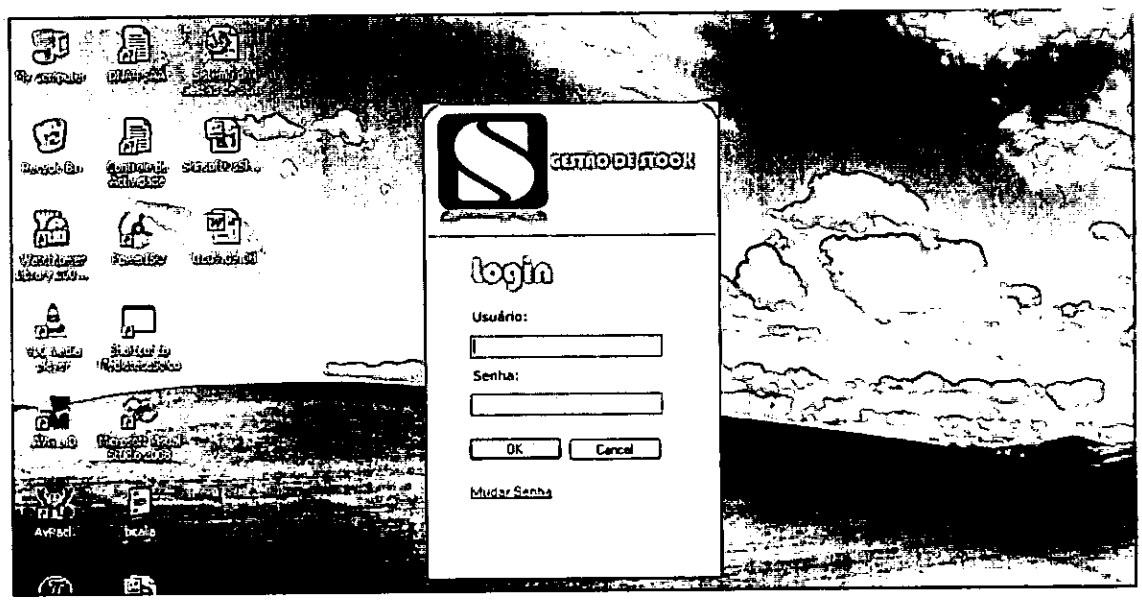
3. GSTOCK: Passo a Passo

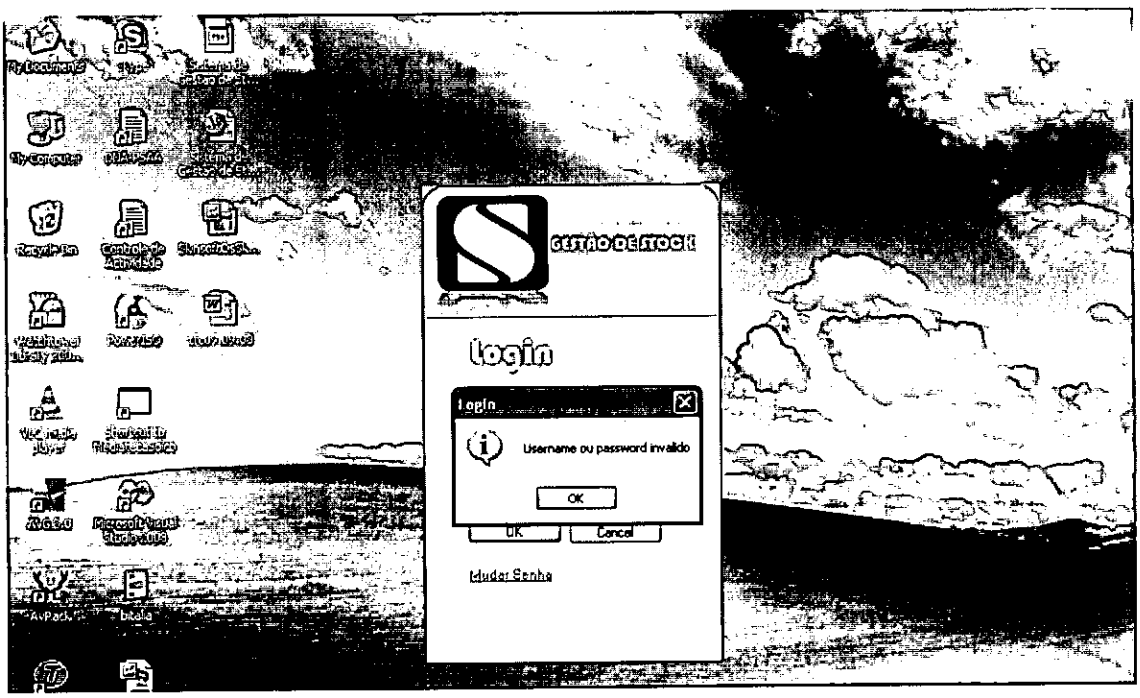
Vamos começar por ilustrar como efectuar uma requisição.

Tudo começa por clicar no ícone GSTOCK que aparece no espaço de trabalho.

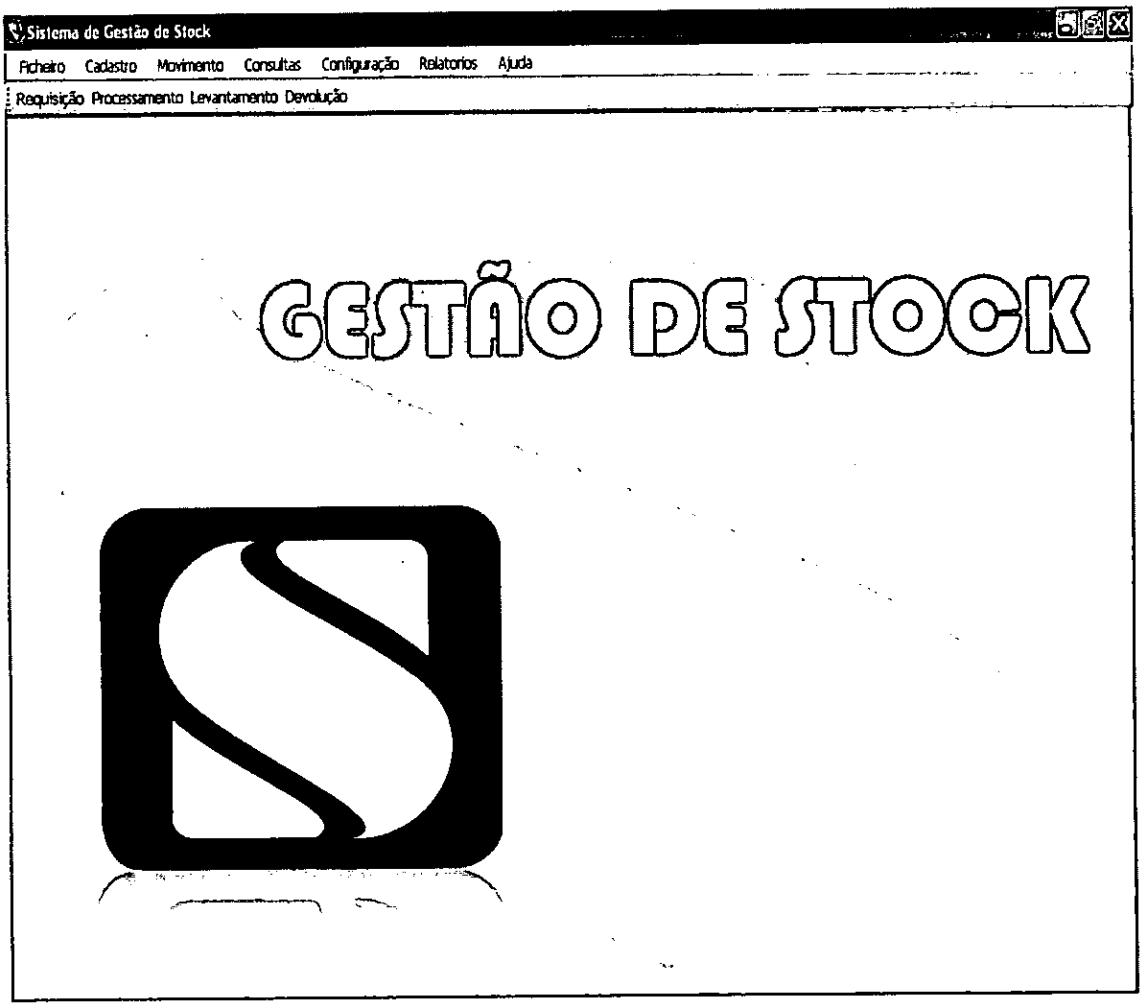
O sistema espera que o usuário introduza o username e password correctos.

Caso o usuário erre na introdução desses dados, uma mensagem de erro aparece informando que os dados não são válidos.



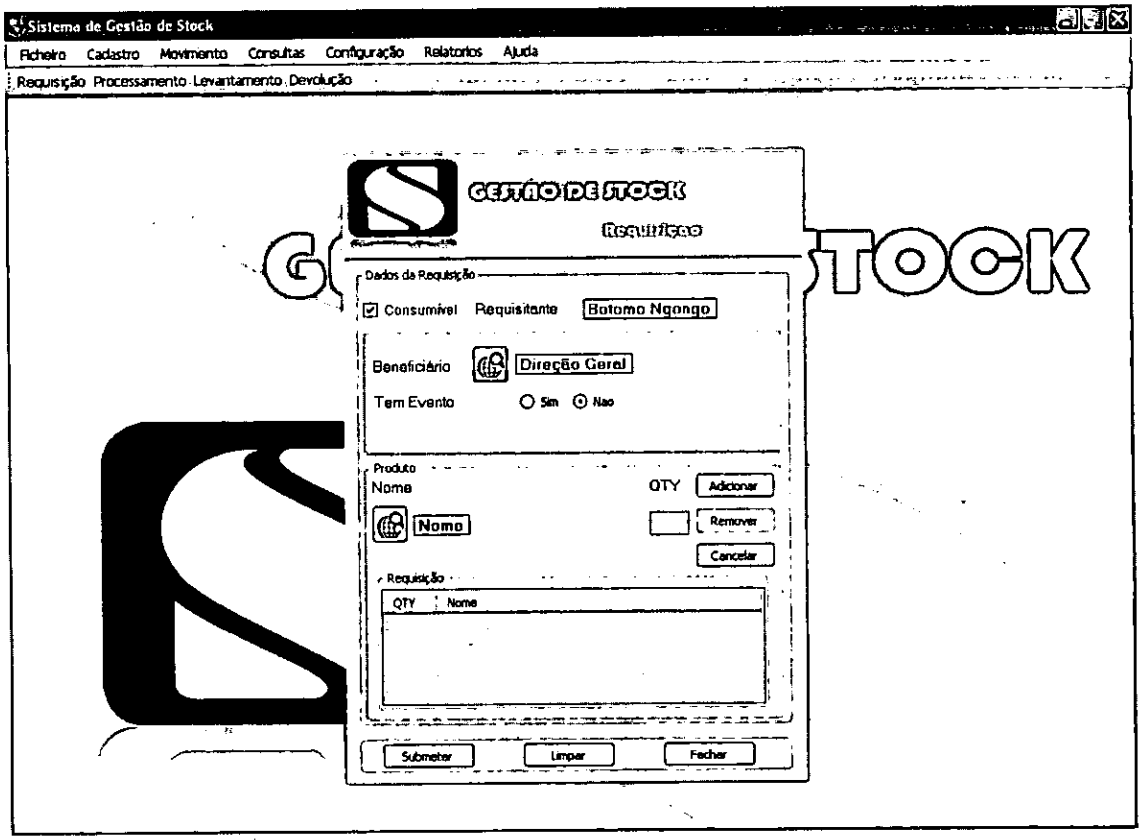

GESTÃO DE STOCK
login
Login
Username ou password invalido
OK
OK
Cancel
Mudar Senha

Temos o Menu Principal



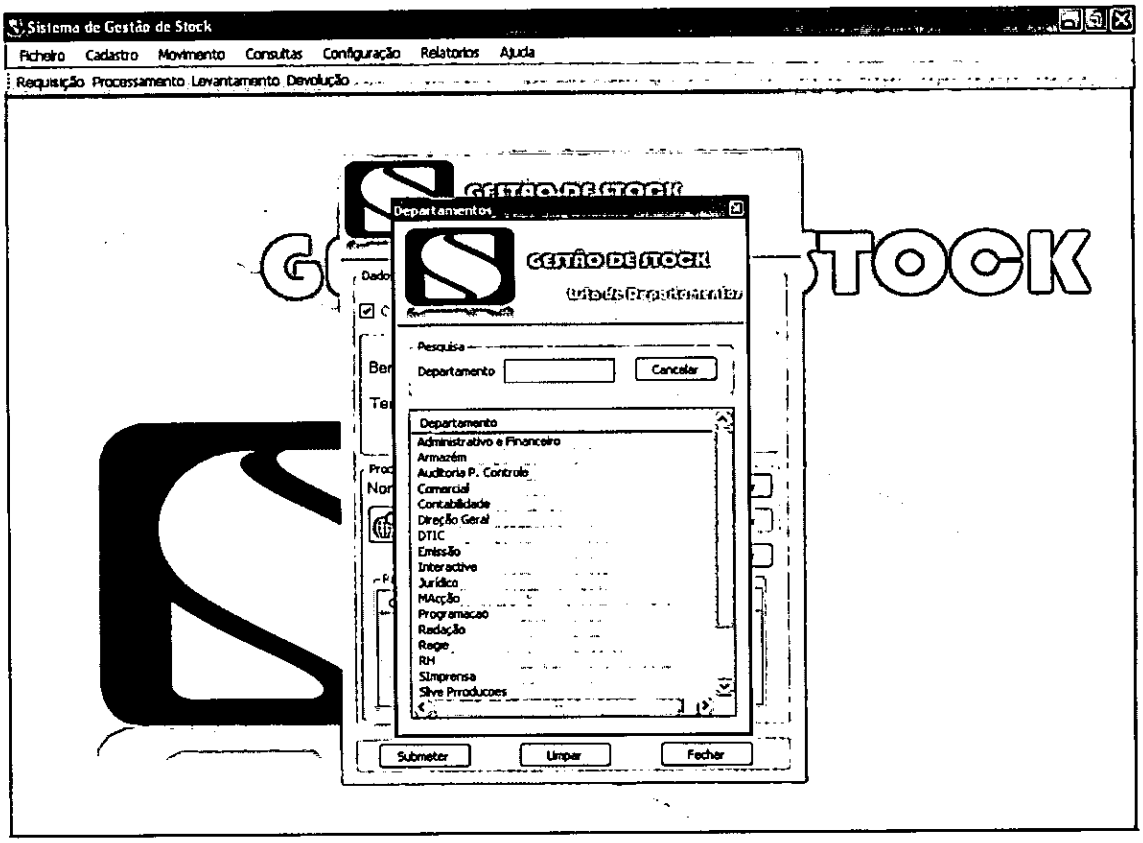
- Para efectuar a requisição

1. Clica na barra de ferramenta onde vem **Requisição,** ou entra no menu **Movimento** e no sub menu **Efectuar Requisição.**

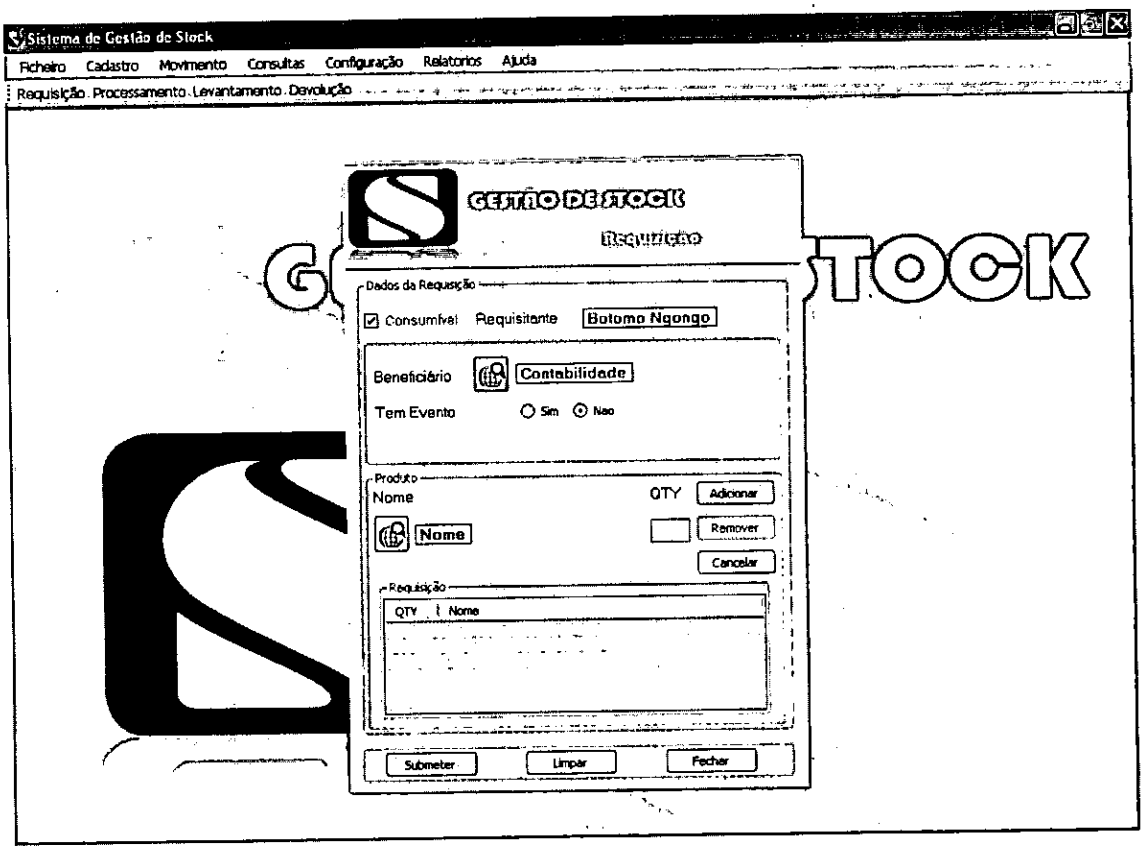


2. Dependendo do tipo de produto que pretende requisitar (consumível ou não), coloque o tic no checkbox com etiqueta **Consumível**

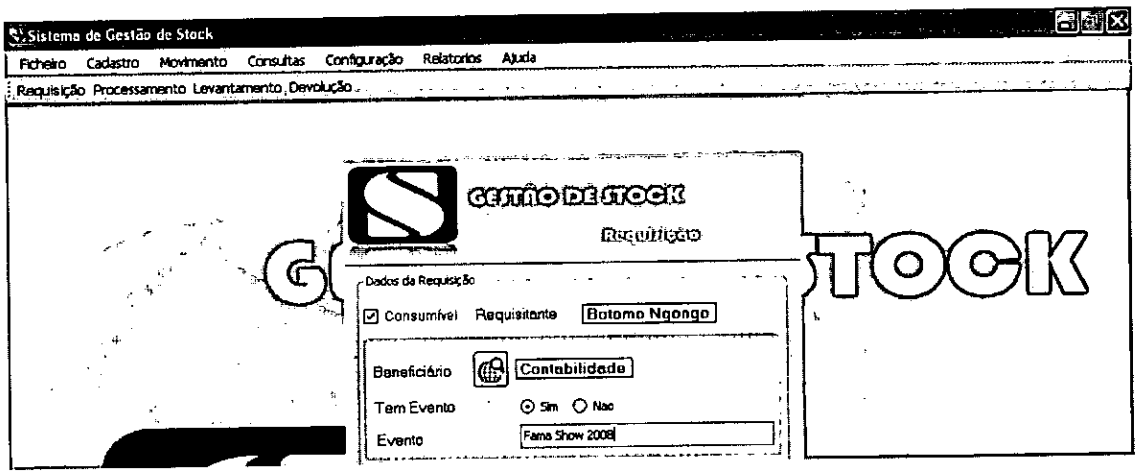
3. Escolha o beneficiário clicando no botão ao lado de etiqueta **Requisita a**



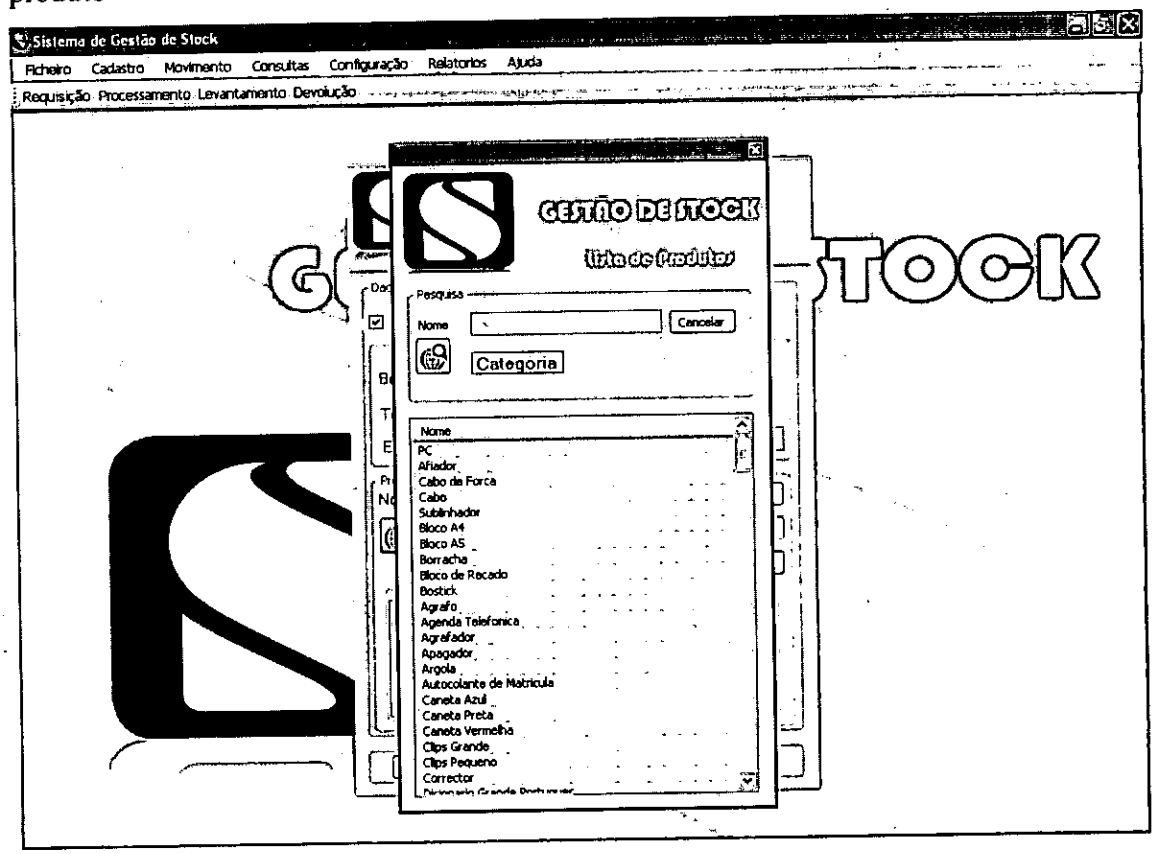
4. Selecciona o beneficiário na lista apresentada dos departamentos dando double click no departamento



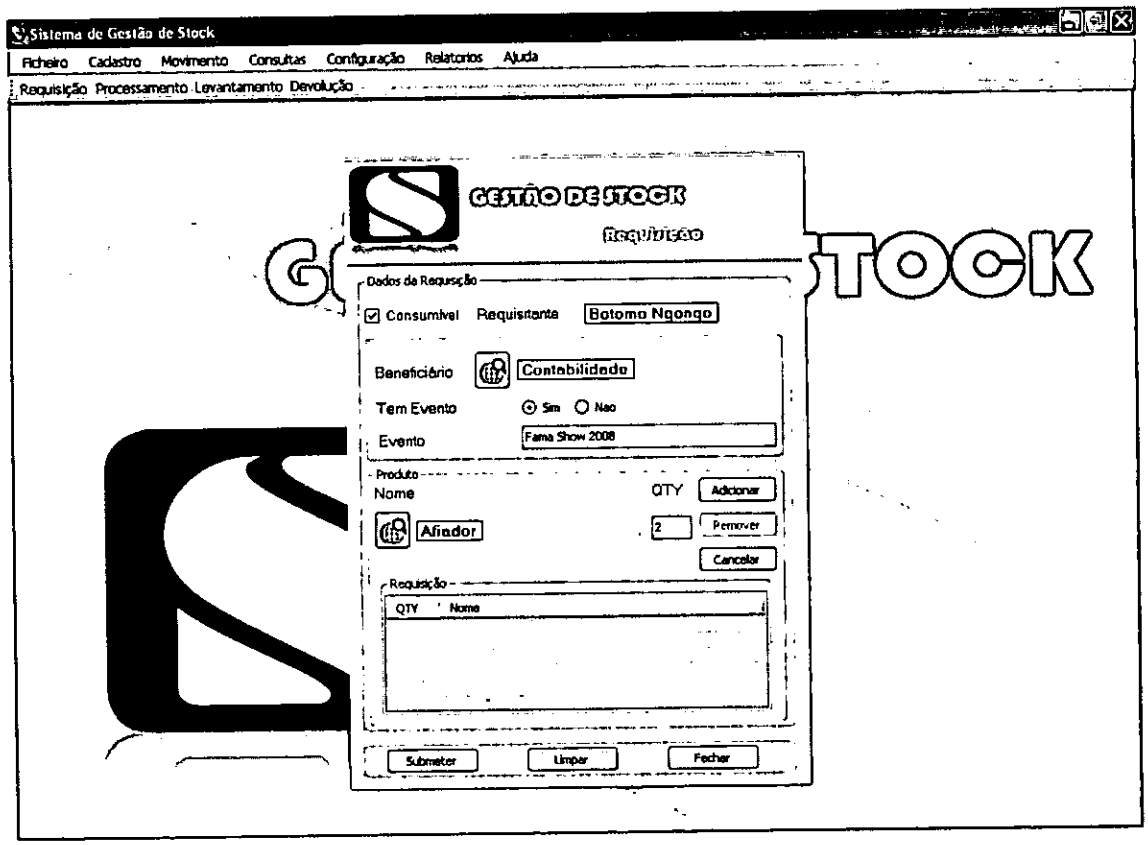
5. Por defeito a requisição não tem evento, caso tenha escolha a opção **Sim** e preencha a caixa de texto

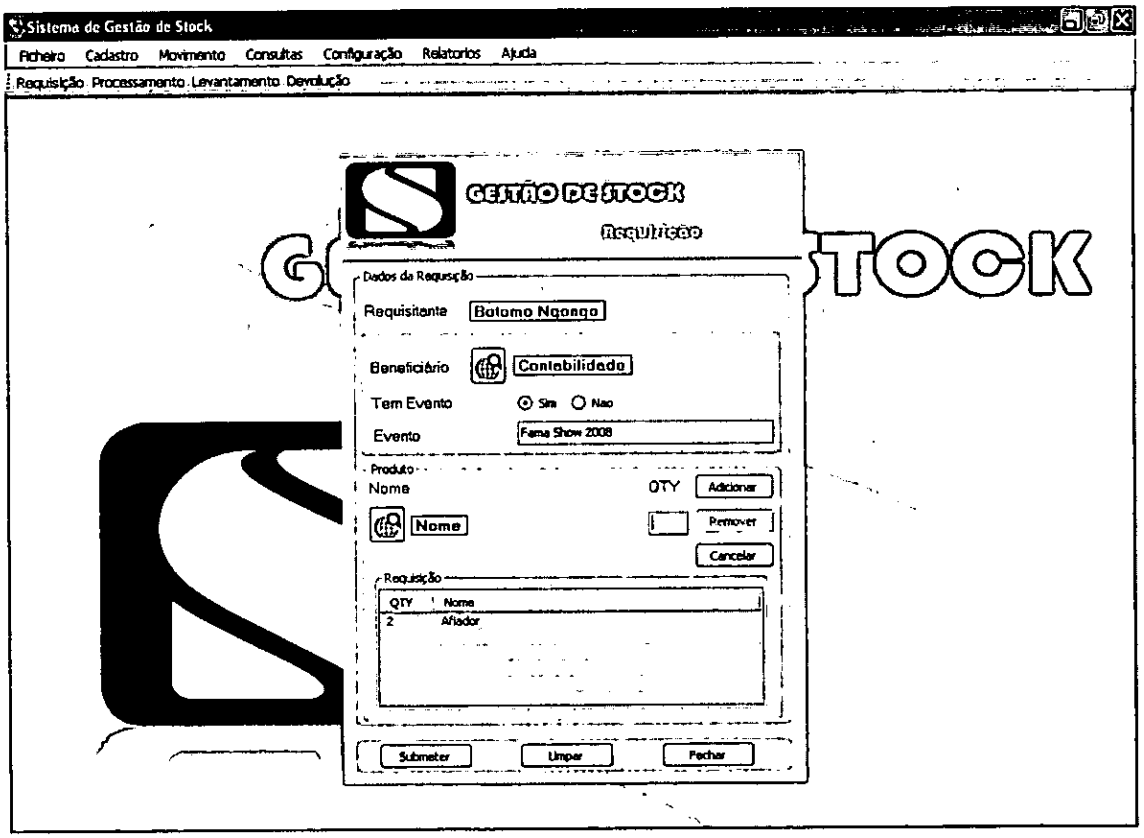


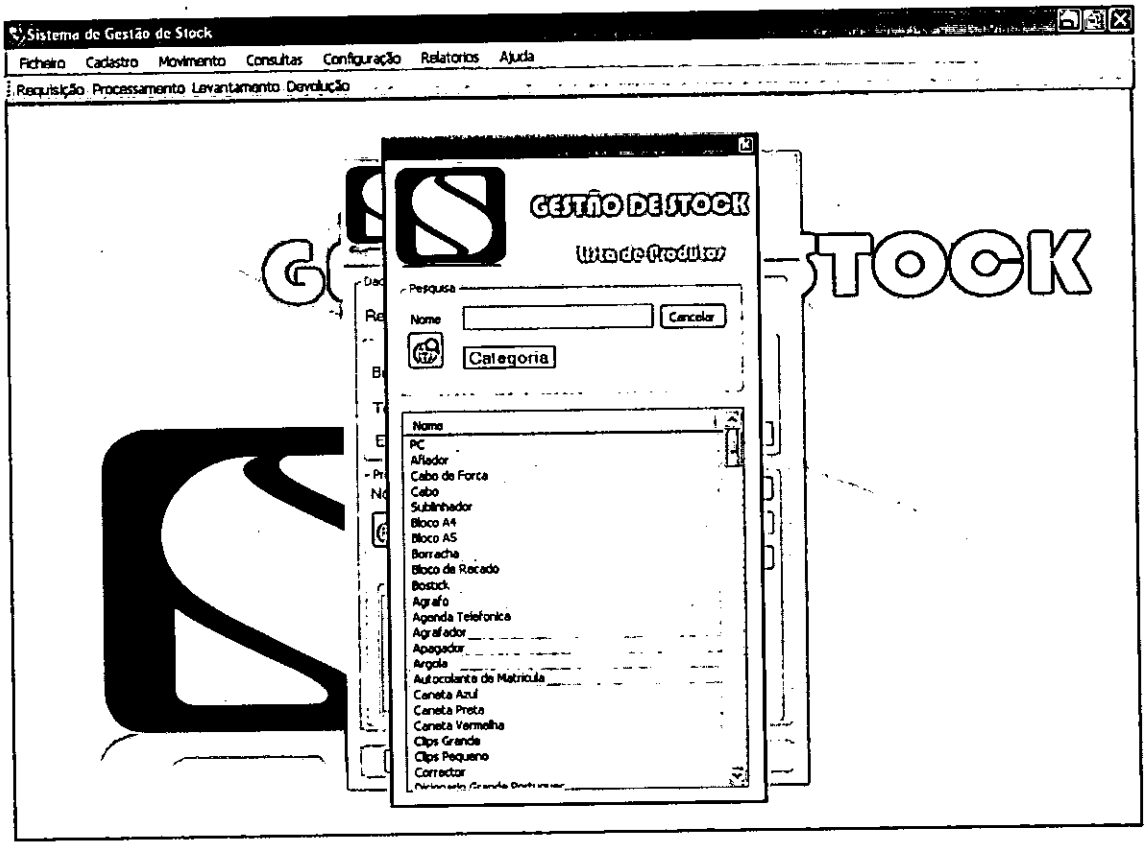
6. Clica no botão ao lado do **Nome** na secção de produto para escolher o produto

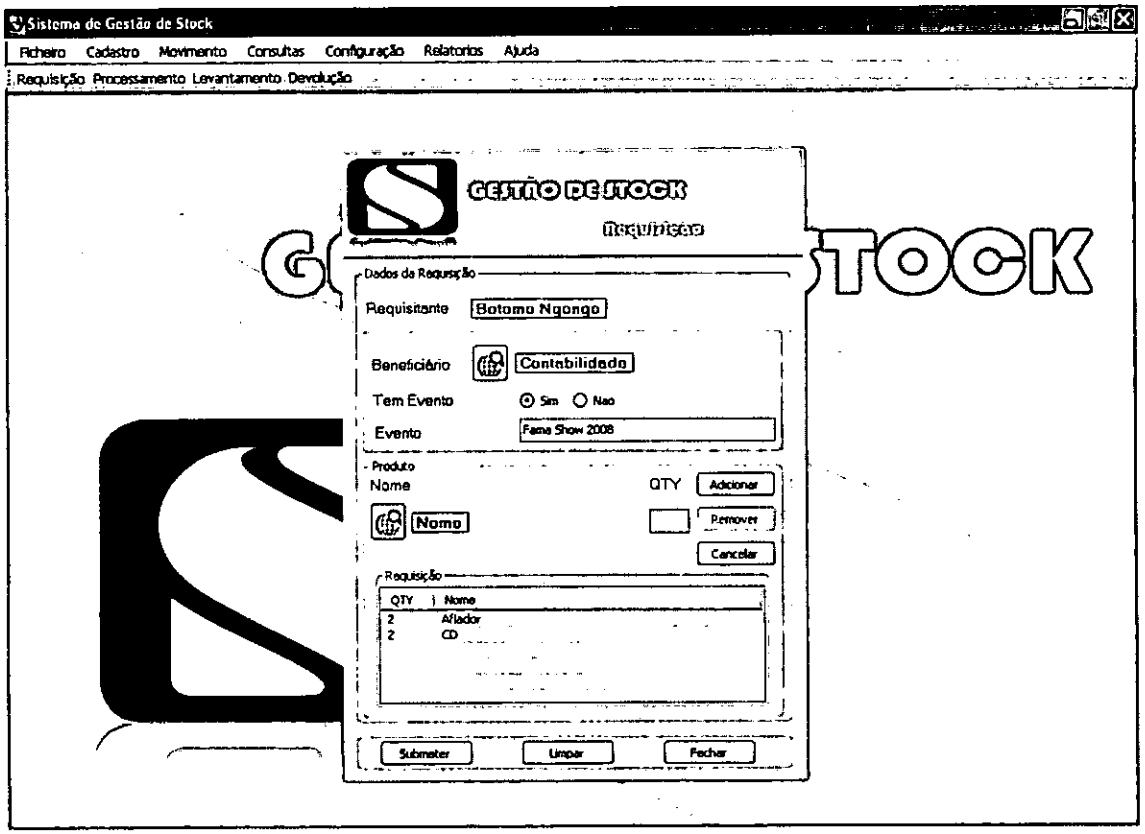


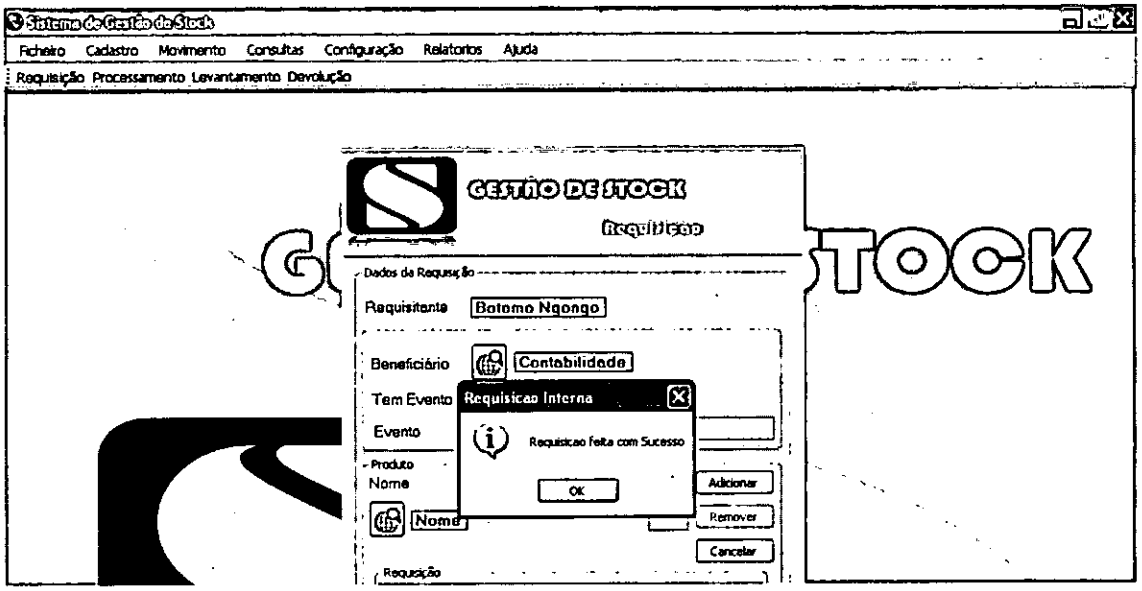
7. Selecciona o produto dando double click no mesmo dentro da lista. Na lista dos produtos, pode-se filtrar o produto colocando as primeiras letras do mesmo ou pode-se filtrar por categoria para facilitar a selecção de produto.
8. Coloca a quantidade desejada e clica em **Adicionar.** Poderá repetir o processo tantas vezes quanto precisar de produtos. Poderá rectificar a quantidade de produto já adicionado, poderá elimina-lo na lista por respectivamente dar double click no item na lista ou por simplesmente selecciona-lo e a seguir clica em **Rectificar** ou **Remover**.
9. Clica em **Submeter** para finalizar o processo

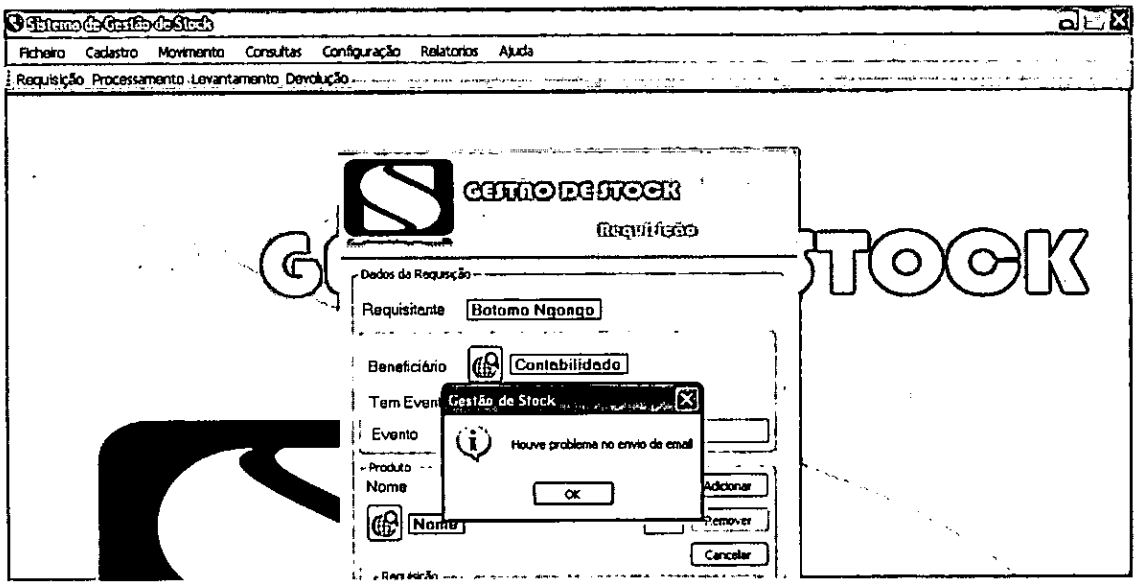

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Consumivel
Requisitante
Botomo Ngongo
Beneficiário
Contabilidade
Tem Evento
Sim
Nao
Evento
Fama Show 2008
Produto
Nome
QTY
Adicionar
Afiador
2
Remover
Cancelar
Requisição
QTY
Nome
Submeter
Limpar
Fechar

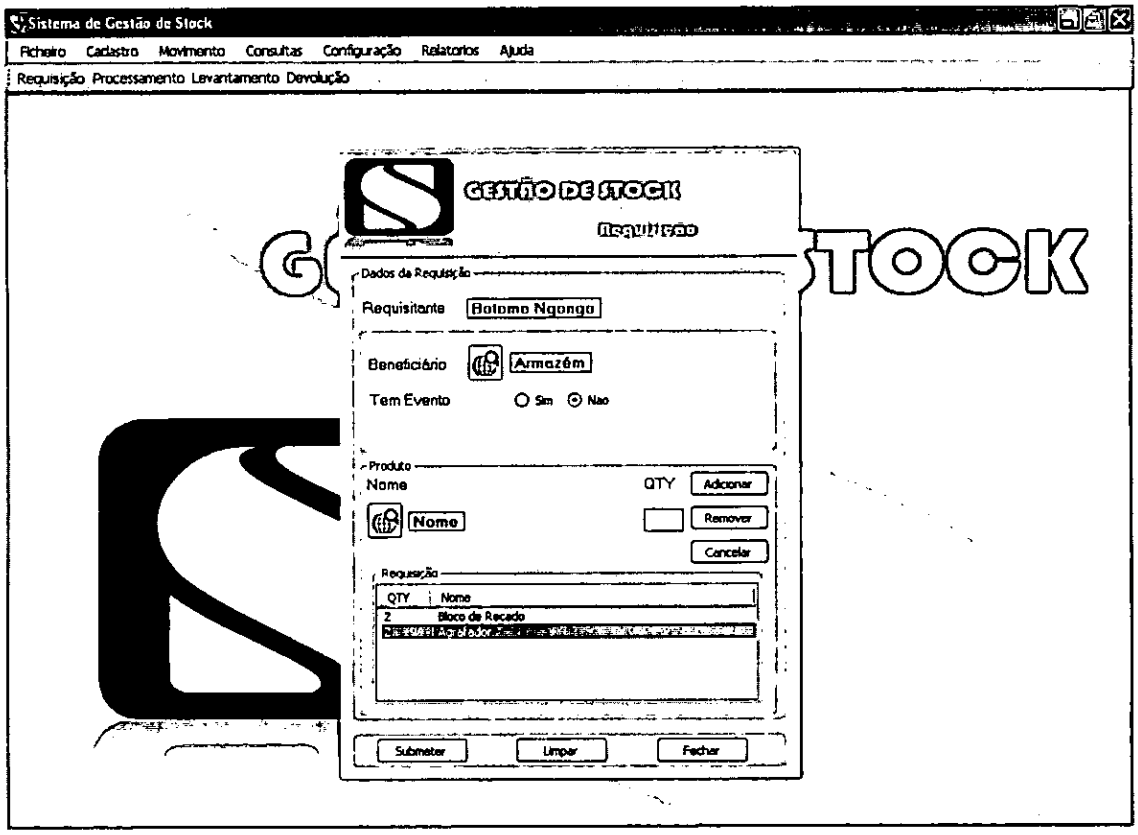

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Contabilidade
Tem Evento
Sim
Nao
Evento
Fama Show 2008
Produto
Nome
QTY
Adicionar
Nome
Remover
Cancelar
Requisição
QTY
Nome
2
Afiador
Submeter
Limpar
Fechar

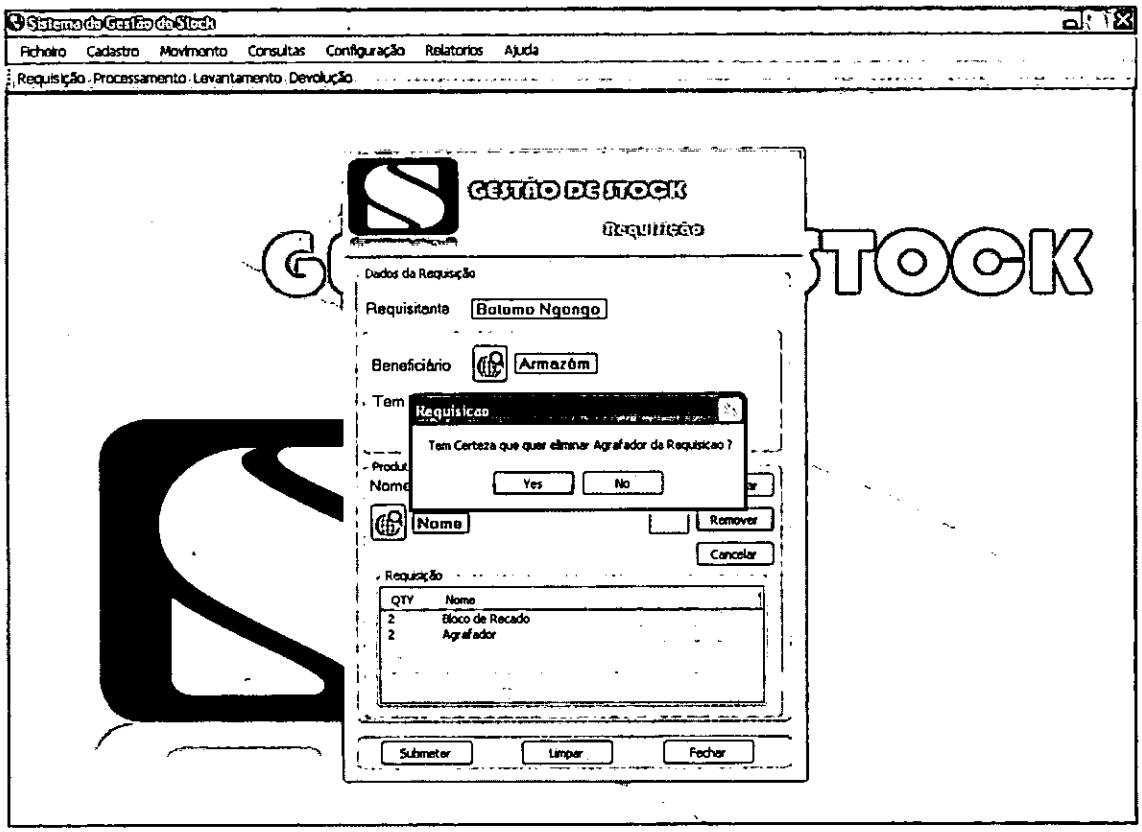

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Pesquisa
Nome
Cancelar
Categoria
Nome
PC
Afiador
Cabo de Força
Cabo
Sublinhador
Bloco A4
Bloco A5
Borracha
Bloco de Recado
Bostick
Agrafo
Agenda Telefonica
Agrafador
Apagador
Argola
Autocolante de Matricula
Caneta Azul
Caneta Preta
Caneta Vermelha
Clips Grande
Clips Pequeno
Corrector

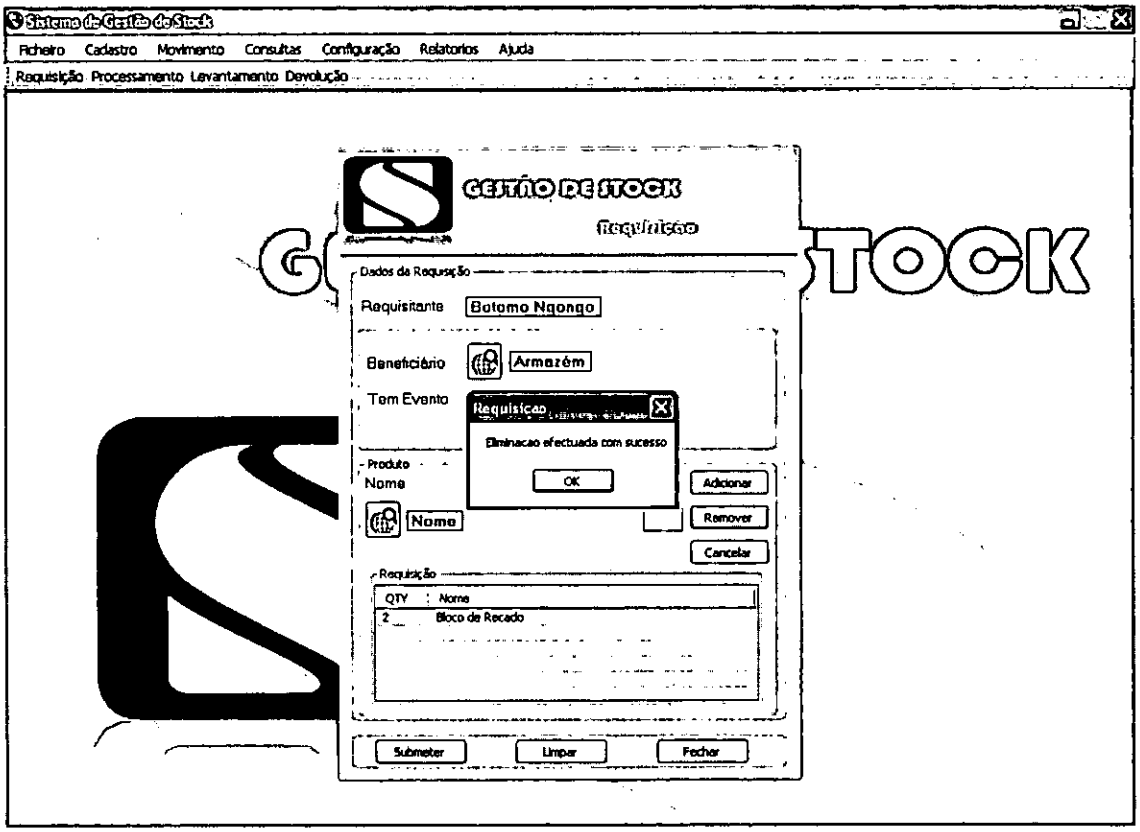

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisicao
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Contabilidade
Tem Evento
Sim
Nao
Evento
Fama Show 2008
Produto
Nome
QTY
Adicionar
Nomo
Remover
Cancelar
Requisição
QTY
Nome
2
Afiador
2
CD
Submeter
Limpar
Fechar

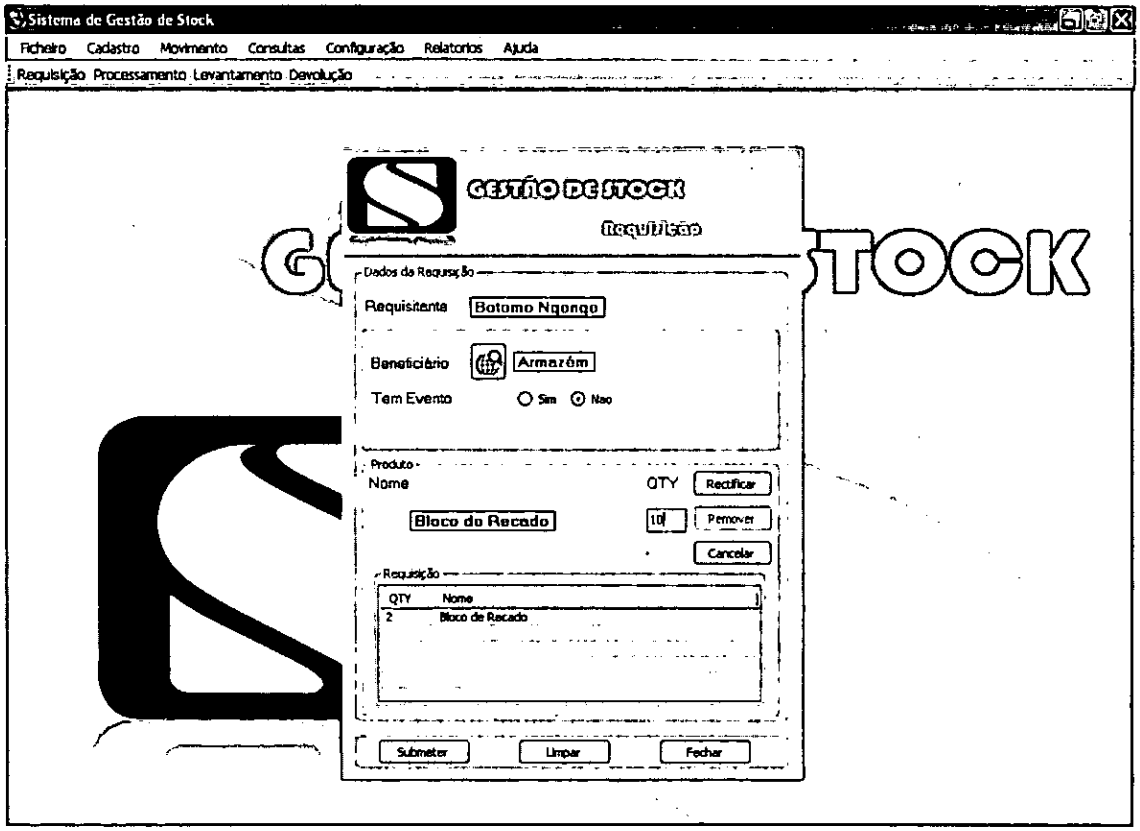

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisicao
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Contabilidade
Tem Evento
Evento
Requisicao Interna
Requisicao feita com Sucesso
OK
Produto
Nome
Nomo
Adicionar
Remover
Cancelar
Requisição

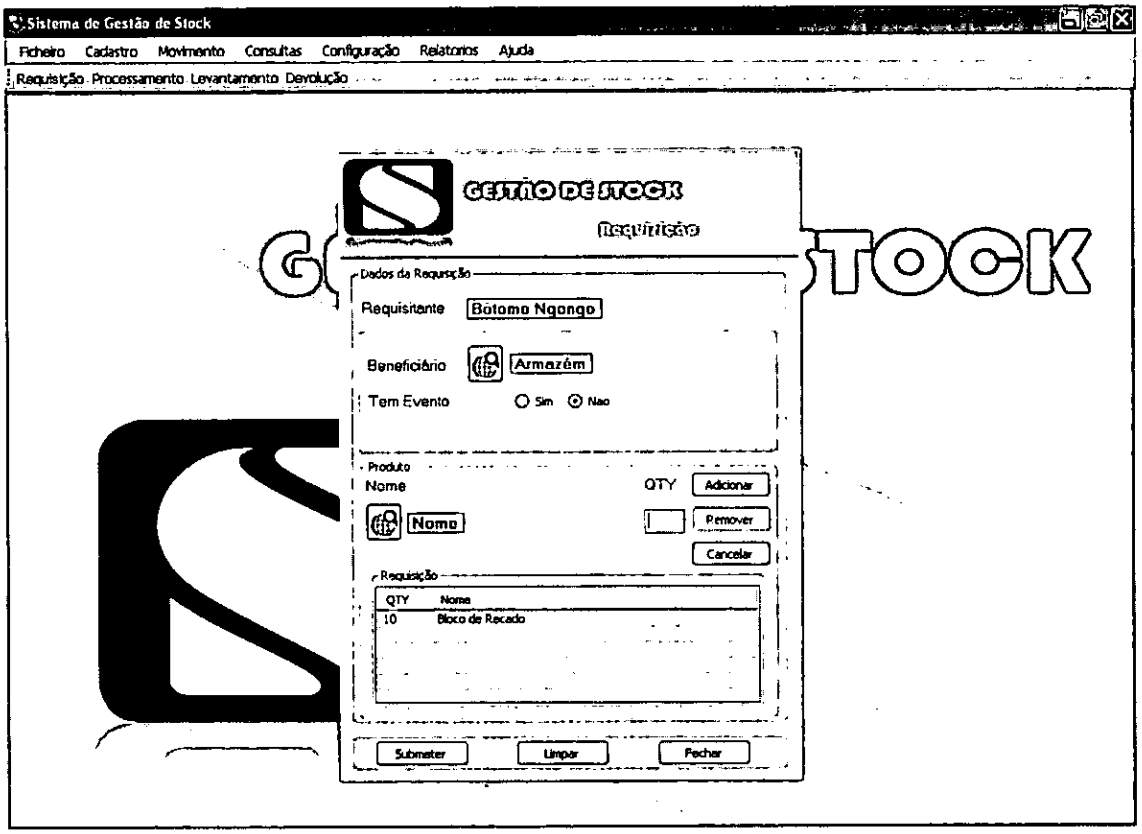

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Contabilidade
Tem Evento
Evento
Produto
Nome
Nome
Adicionar
Remover
Cancelar
Gestão de Stock
Houve problema no envio de email
OK

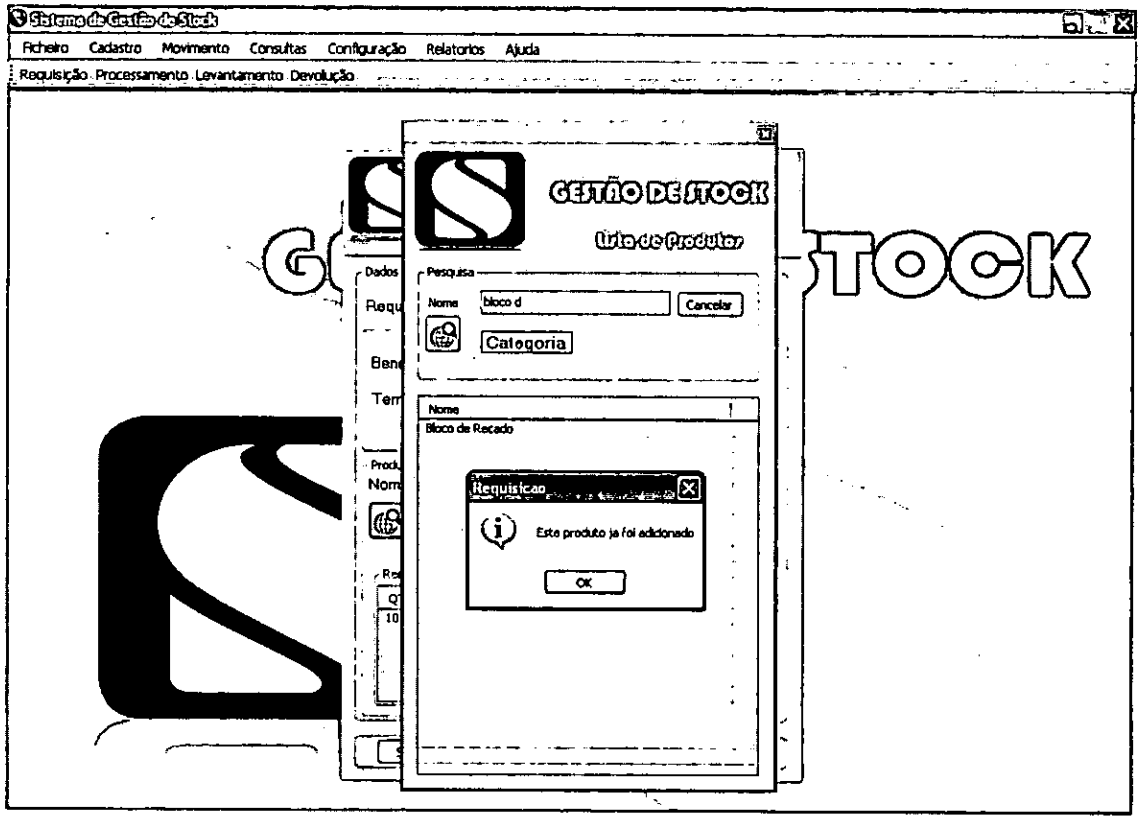

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Armazém
Tem Evento
Sim
Nao
Produto
Nome
QTY
Adicionar
Nome
Remover
Cancelar
Requisição
QTY
Nome
2
Bloco de Recado
Submeter
Limpar
Fechar


Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Armazém
Tem
Requisicao
Tem Certeza que quer eliminar Agrafador da Requisicao ?
Yes
No
Nome
Remover
Cancelar
Requisição
QTY
Nome
2
Bloco de Recado
2
Agrafador
Submeter
Limpar
Fechar


Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Armazém
Tem Evento
Requisicao
Eliminacao efectuada com sucesso
OK
Produto
Nome
Nome
Adicionar
Remover
Cancelar
Requisição
QTY
Nome
2
Bloco de Recado
Submeter
Limpar
Fechar

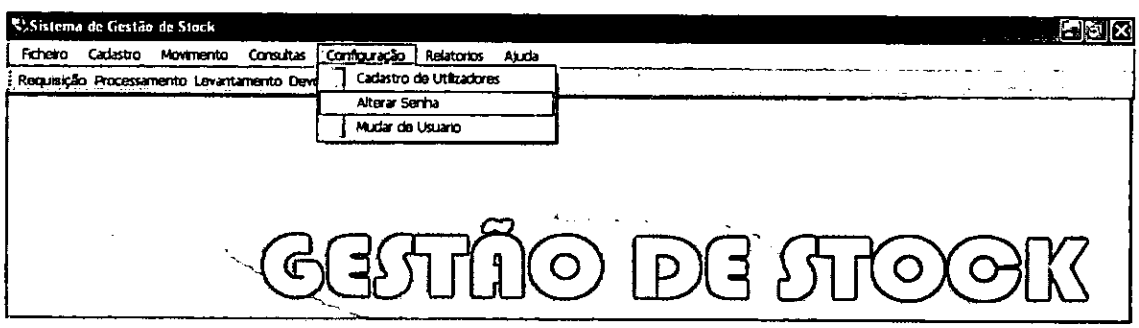

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Armazém
Tem Evento
Sim
Nao
Produto
Nome
QTY
Rectificar
Bloco do Recado
10
Remover
Cancelar
Requisição
QTY
Nome
2
Bloco de Recado
Submeter
Limpar
Fechar

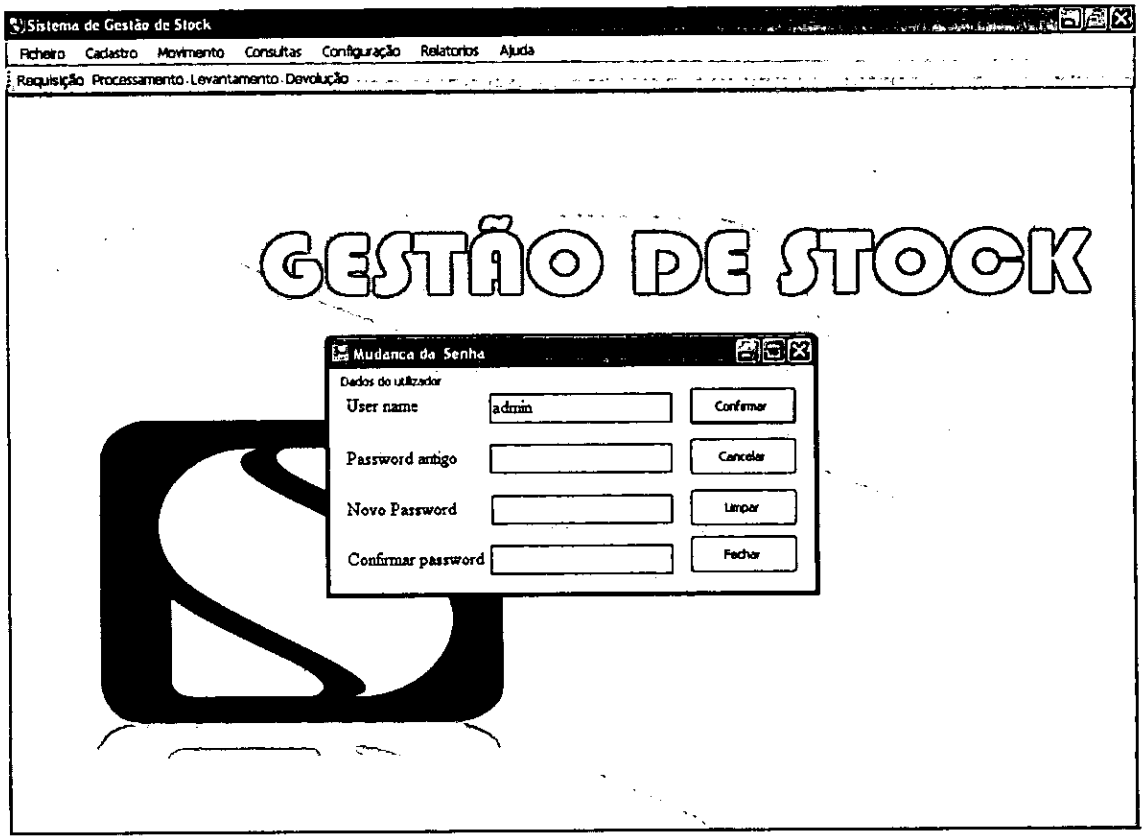

Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Requisição
Dados da Requisição
Requisitante
Botomo Ngongo
Beneficiário
Armazém
Tem Evento
Sim
Nao
Produto
Nome
QTY
Adicionar
Nome
Remover
Cancelar
Requisição
QTY
Nome
10
Bloco de Recado
Submeter
Limpar
Fechar



Para alterar a senha

1. Clica em **Configuração/Alterar Senha**
2. Confirma os dados e clica em **Confirmar**




Sistema de Gestão de Stock
Ficheiro Cadastro Movimento Consultas Configuração Relatorios Ajuda
Requisição Processamento Levantamento Devolução
GESTÃO DE STOCK
Mudanca da Senha
Dados do utilizador
User name
admin
Confirmar
Password antigo
Cancelar
Novo Password
Limpar
Confirmar password
Fechar

Alguns relatórios do sistema

Relatório de Consumo de CD

Departamento de Administração e Património

SOICO

Produto: CD **Total Gasto:** 67

Período: De 01-Jul-2008 À 05-Aug-2008

Requisitante	Beneficiário	Data	Qty
Arlesia Siqauque	Contabilidade	17-Jul-2008	12
Botomo Ngongo	Direção Geral	13-Jul-2008	2
Botomo Ngongo	Direção Geral	18-Jul-2008	2
Carlos Noronha	STV Producao	01-Jul-2008	5
Elisio Maxlhuza	Redação	09-Jul-2008	3
Fatima Uaide	Redação	28-Jul-2008	1
Nela Duarte	MAcção	31-Jul-2008	5
Neusa Poeira	STV Producao	16-Jul-2008	32
Reginaldo Mondlane	Slive Prroducoes	04-Aug-2008	5

Relatório de Stock

Departamento de Administração e Património

Inventário de Stock Data : 10-Sep-2008 Armazém AOPT

Ordem	Designação	Quantidade
1	Papel Higienico	14
2	OMO	7
3	Esfregona	1
4	Handy Andy	24
5	Palha de Aco	9
6	Guardanapo Humido	2
7	Guardanapo Seco	1
8	Vassora	3
9	Lava Vidro	5
10	Pano de Po	8
11	Pano de Loica	0
12	Sabao Liquido para as Maos	3
13	Sabao Liquido para loica	0
14	Sabao	35
15	Recarga Blue	8
16	Po Vim	4
17	Baygon	4
18	Air Freshener	1
19	Bombas para WC	3
20	Escova de Pia	5
21	Esponja para lavar Loica	0
22	Desigordurante	3
23	Serra Liquida Cobra	6

Para sair

1. Clica em **Ficheiro**
2. **Sair**