CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO
FABIANO HOEFLING MELO

SIGMETAL: Sistema de Gerenciamento de Estoque de Matéria Prima para Metalúrgicas usando Arduino

LAGES
2014

FABIANO HOEFLING MELO

SIGMETAL: Sistema de Gerenciamento de Estoque de Matéria Prima para Metalúrgicas usando Arduino

Projeto apresentado à Banca Examinadora do Trabalho de Conclusão do Curso de Ciência da Computação para análise e aprovação

LAGES
2014

FABIANO HOEFLING MELO

SIGMETAL: Sistema de Gerenciamento de Estoque de Matéria Prima para Metalúrgicas usando Arduino

Trabalho de Conclusão do Curso de Ciência da Computação apresentado ao Centro Universitário UNIFACVEST como parte dos Requisitos para obtenção do título de bacharel em Ciência da Computação.
Prof.MSc. Márcio José Sembay

Lages, SC ____/____/2014. Nota_____ _____

_____

_____

_____

LAGES
2014

AGRADECIMENTOS

Agradeço a toda minha família que sempre me apoiou, a minha mãe que sempre me incentivou a estudar muito, ao meu pai que sempre diz que se não sou o maior devo ser o melhor, a minha irmã, que sempre me auxilia em todos os momentos.

Deus, por sempre me dar forças e por preparar um caminho para alcançar os meus objetivos.

Aos professores que transmitiram seus conhecimentos e ofereceram sugestões para o desenvolvimento deste trabalho, sempre ajudando em dificuldades encontradas.

SUMÁRIO

LISTA DE ABREVIATURAS

AWT – Abstract Windowing Toolkit
CASE – Computer Aided Software Engineering
CLP – Controlador Lógico Programável
DAO – Data Access Object
EAI – Enterprise Application Integration
ERP – Enterprise Resource Planning
GPS – Global Positioning System
GUI – Graphical User Interface
HD – Hard Disk
HTML – Hyper Text Markup Language
IDE – Integrated Development Environment
Java SE - Java Standard Edition
JDK – Java Development Kit
JPA – Java Persistence API
JPQL – Java Persistence Query Language
JRE – Java Runtime Environment
JVM – Java Virtual Machine
LCD – Liquid Crystal Display
MVC – Model-View-Controller
POJO – Plain Old Java Object
POO – Programação Orientada a Objetos
SGBDR – Sistema Gerenciador de Banco de Dados Relacional
SGDB – Sistema Gerenciador de Banco de Dados
SI – Sistemas de Informação
SKU's – Stock Keeping Unitss
SQL – Structured Query Language
TCC – Trabalho de Conclusão de Curso
TI – Tecnologia da Informação
WEB – World Wide Web

LISTA DE FIGURAS

RESUMO

O SIGMETAL é um sistema desktop de gerenciamento de estoque com o objetivo de automatizar o processo de estocagem, venda de produtos e controle da quantidade de matéria prima na entrada e contagem da mesma na saída, através da automatização do processo com arduino. Para isso foi feito um estudo de caso de uma empresa que atua no ramo da metalurgia, onde foi analisado o controle de estoque interno da mesma para a implementação de um sistema de controle adequado. Essa microempresa metalúrgica atende a produção e venda de peças. Para controle de estoque utilizam planilhas e arquivos de texto, sendo todo o processo feito manualmente. A metodologia utilizada foi à pesquisa bibliográfica descritiva, estudo de caso, método de abordagem indutivo. O trabalho está estruturado em tópicos e subtópicos abordando sobre controles internos, estruturas, ambientes, atividades, controles de estoques, automação, desde conceitos até detalhamento do sistema de estoque. O sistema foi desenvolvido em JAVA pela ferramenta Netbeans associada com JPA, utilizando o banco de dados MySQL para o armazenamento das informações, através da biblioteca Hibernate, que realiza a abstração deste banco de dados, no código fonte. Como resultado, destaca-se a possibilidade de comparação e a facilidade da análise de dados. Através da implantação de estratégias e ajustes, o controle interno pode contribuir com a contenção de perdas, redução de riscos, desvios, fraudes, furtos e conluio entre o quadro de colaboradores, clientes e fornecedores.

Palavras chaves: Controle de Estoque, Java SE, JPA, Arduino.

ABSTRACT

The SIGMETAL is a desktop system for inventory management in order to automate the process of storing, selling products and controlling the amount of raw material at the inlet and outlet on the same count, by automating the process with arduino. For this a case study of a company that operates in the field of metallurgy, where it was analyzed the internal control of the same stock for the implementation of an appropriate control system was made. This micro metal meets the production and sale of parts. For inventory control using spreadsheets and text files, with the whole process done manually. The methodology used was the literature descriptive case study method of inductive approach. The work is structured in topics and subtopics on addressing internal controls, structures, environments, activities, inventory controls, automation, from concepts to details of the inventory system. The system was developed in JAVA Netbeans tool associated with the JPA, using the MySQL database for storing information via the Hibernate library that performs the abstraction of this database, the source code. As a result, there is the possibility to compare the ease of data analysis. Through the implementation of strategies and appropriate internal control may contribute to the containment of losses, risk reduction, diversion, fraud, theft and collusion between the number of employees, customers and suppliers.

Keywords: Inventory Control, Java SE, JPA, Arduino.

I. INTRODUÇÃO

As empresas comerciais no seu processo de venda e as industriais no seu processo de produção, período de compra, transformação e venda, utilizam o processo de comando de estoque para compor a relação entre essas etapas.

Os estoques tem um desempenho essencial na versatilidade funcional da empresa, em determinado instante do procedimento formado por essas etapas. Verificam entradas e saídas entre as duas fases dos procedimentos de venda e de produção, dessa forma reduzem as consequências de falhas de planejamento e as variações imprevisíveis de oferta e procura ao mesmo tempo em que afastam ou reduzem as interações das diversas partes da organização empresarial.

Esse sistema aborda a seguinte questão: como a empresa pode reduzir seu custo através do gerenciamento de estoque e a informatização e automação do seu processo industrial.

O objetivo deste trabalho é a apresentação de um Sistema de Gerenciamento de Estoque de Matéria Prima para Metalúrgicas usando Arduino, que auxilie o controle de projetos e tarefas a serem realizadas por um usuário ou grupo de pessoas. A aplicação tem diversos módulos, tais como: cadastros, movimentos, utilitários, controle e ajuda.

Para o desenvolvimento do sistema utiliza-se a linguagem de programação Java, e na persistência dos dados é utilizado a *Java Persistence API* (JPA), que permite persistir os dados em bancos de dados diversos através da utilização de objetos em Java sem que seja necessária a utilização da *Structured Query Language* (SQL) para persistência das informações. O banco de dados utilizados para armazenar os dados persistidos pela JPA é o Mysql.

Este trabalho justifica-se por ser o controle de estoque um sistema que busca solucionar o problema dos gastos e ausência dos produtos, apresenta alternativas viáveis e métodos eficientes de gestão de estoque para as empresas metalúrgicas, com foco na satisfação do cliente.

1.1 Justificativa

O software de controle de estoque usando automação com Arduino, no processo de entrada e saída fornecerá, um relatório detalhado do que a empresa tem e do que a empresa precisará comprar para suprir as necessidades de seus atuais e novos clientes. Um gerenciamento como este, é muito importante pela eficiência para a procura na hora da venda e redução de despesas evitando a compra de produtos já existentes no estoque. Como a empresa está crescendo cada vez mais no mercado de vendas de produtos metalúrgicos, venda direta a clientes e venda as revendedoras, fez se a necessidade de um sistema informatizado de controle de estoque.

Atualmente o mercado passa por intensas transformações tecnológicas, que contribuem para a melhoria dos processos da gestão empresarial, através de informações necessárias, suscetível de uma maior precisão. Estas tecnologias permitem ao administrador informações precisas para um planejamento eficiente na tomada de decisões. Para alcançar um desempenho ideal de um sistema de controle de estoque em determinada empresa, é necessário que sejam transparentes as instruções apresentadas pela função compras, com o intuito de que as atitudes referentes às tarefas de verificação das condições dos estoques e a extensão dos lotes de reabastecimento e dos estoques de segurança sejam as mais adequadas para cada caso.

Sendo assim, desenvolver um conjunto de regras para as decisões relacionadas ao fornecimento e gerenciamento dos suprimentos de uma empresa, é indispensável para uma melhor aplicação de recursos financeiros, maior fluxo na produtividade e como resultado uma estabilidade maior do processo rentável.

É possível atingir impactos sobre os lucros da corporação, mediante de redução de gastos na operação de compra de materiais, superior aos alcançados com finalizações similares em outros setores de custos e vendas (BALLOU, 2006).

Muitas empresas pequenas não acumulam grandes gastos e investimentos, devido a pouca renda com que sobrevivem. Várias vezes, sem que constatem, estas empresas começam a ter recursos cada vez maiores em estoque. Na maior parte devido ao precário fluxo de informações resultante de uma direção ruim na área de compra, estoque, produção e vendas.

Assim, procurou-se propor com este sistema, utilizando a automação com arduino no controle de entrada de matéria prima e saída do produto final, vantagens importantes na manutenção de estoques da empresa, pesquisando de acordo com a perspectiva do gasto e do benefício, para que as condições de atividades ao consumidor sejam consideradas e que os

níveis de estoques não sejam mantidos em altos níveis, com altos custos de manutenção, absoluto equilíbrio e agilidade em abastecer a falta de algum item ou produto do estoque.

1.2 Objetivo do Trabalho

1.2.1 Objetivo Geral

Desenvolvimento de um sistema Desktop de gerenciamento de estoque de Matéria Prima para Metalúrgicas usando Arduino, através da criação de um protótipo de software e hardware para o acompanhamento da produção, estocagem e controle da entrada e saída de matéria prima.

1.2.2 Objetivos Específicos

Os objetivos específicos consistem no desenvolvimento de um sistema de gerenciamento de estoque de Matéria Prima, usando Arduino e algumas atividades comerciais para Metalúrgicas, através da otimização do processo de controle, na entrada da matéria prima, o qual realiza a pesagem da mesma, por meio de um sensor de pressão acoplado ao arduino e ligado ao sistema, evitando assim falhas no processo de armazenagem, indicando corretamente a diferença se houver entre a quantidade do produto descrito na nota fiscal e o existente após a pesagem. Também, verifica a saída do produto final, por meio de um cálculo onde é subtraído o peso de cada unidade produzida do peso total, mostrando assim a quantidade que deve ser apresentada ao sistema, após a contagem com a utilização de um sensor magnético, o qual é acoplado no sistema da mesma forma que o processo descrito acima, portanto é possível determinar se a quantidade de produtos estipulados confere através da comparação dos parâmetros indicados pelo sistema.

Exemplificar a implementação do sistema, como resultado do estudo das tecnologias utilizadas, destacando-se a linguagem de programação Java, o ambiente de desenvolvimento NetBeans, os framework Swing, Hibernate e JPA e por fim a utilização do sistema embarcado arduino, na automação da entrada de matéria prima e na saída do produto final.

1.3 Metodologia

1.3.1 Estudo de Caso

O estudo de caso para o desenvolvimento do sistema foi realizado em uma empresa de pequeno porte que atua no ramo metalúrgico, produzindo peças automotivas. Para levantar as necessidades dos usuários foi preciso realizar diversas entrevistas com os futuros usuários do sistema, além disso, catalogar os documentos existentes na empresa que são utilizados de forma natural.

Atualmente essa empresa tem a maior parte de seu controle manual, utilizando formulários pré - impressos feitos em gráficas ou comprados em livrarias. Algumas coisas são controladas no computador através de planilhas e arquivos de texto.

Telas e relatórios devem ser impressos para compor as necessidades.

O trabalho foi dividido em quatro etapas:

Planejamento, correspondendo a uma identificação geral das necessidades, identificação e seleção de alternativas e definição de plano do trabalho.

Análise, que incluiu a identificação detalhada das funcionalidades do sistema (Levantamento de Requisitos) e a respectiva descrição (Especificação do Sistema) de modo a que os mesmos requisitos possam ser validados pelos utilizadores finais do sistema.

Desenho, onde foi realizada a definição detalhada da arquitetura global da solução (módulos, tabelas, interface, máquinas, etc.).

Desenvolvimento, tarefa na qual é realizada a programação dos diversos componentes do sistema.

Testes (ou Integração), em que o sistema no seu global é verificado com o objetivo de obter a aceitação do utilizador.

Instalação, tarefa onde são executadas as atividades relacionadas com a disponibilização do sistema para os seus utilizadores finais, e que normalmente é designada por entrada do sistema em produção.

Finalmente a Manutenção, o momento que corresponde ao tempo de vida útil do sistema e durante o qual serão efetuadas todas as alterações posteriores à entrada em funcionamento do produto.

1.3.2 Estudo Bibliográfico

Para a realização deste trabalho foram pesquisados livros e artigos periódicos sobre controle de estoque, sistema de informação, engenharia de software e programação orientada a objetos (POO). Conceitos estes empregados no desenvolvimento do sistema Sigmetal, ainda

no mesmo sucedeu o estudo da linguagem de programação Java plugada a alguns frameworks que constituindo uma plataforma de desenvolvimento, visando à agilidade de produção do mesmo.

Na automação da entrada e saída de produtos do controle de estoque, foram analisados conteúdos referentes à automação industrial, sobre IDE Arduino e sua linguagem de programação, sensores e eletrônica.

1.3.3 Cronograma

O seguinte cronograma foi utilizado para o desenvolvimento deste trabalho.

Atividades Realizadas	Jun	Jul	Ago	Set	Out	Nov	Dez
Pesquisa							
Desenvolvimento e apresentação da pré-proposta							
Modelagem do sistema							
Implementação do software							
Fase de testes							
Entrega e defesa do TCC à banca avaliadora							
Correções							
Entrega e defesa do TCC à banca avaliadora							

Tabela 1 - Cronograma do TCC 2

Fonte: Autoria própria.

1.4 Estrutura do Trabalho

Para o desenvolvimento deste trabalho foram cumpridas as seguintes etapas: pesquisa de material bibliográfico, revisão bibliográfica, modelagem do sistema, implementação do software e testes.

Na primeira fase foi realizado um levantamento bibliográfico, coletando os dados e informações necessárias para o início do trabalho. Na revisão bibliográfica fundamentou-se teoricamente e proporcionou embasamento técnico, justificando os tópicos abordados no TCC e o padrão de processo adotado para o desenvolvimento do sistema. A modelagem do sistema foi realizada com auxílio de ferramenta CASE, principalmente o JUDE/Community 5.5.1, já

na parte de modelagem das tabelas do banco de dados foi utilizada a ferramenta Sybase Power Designer 15.3. Para a modelagem foram feitas pesquisas em livros, periódicos, páginas da Web, trabalhos de conclusões de curso, entre outras modalidades.

Para implementação e testes do sistema foi utilizado a tecnologia Java, JDK 7.0, JRE 7 e uma JVM. A IDE utilizada para a codificação foi o NetBeans. Para documentação do projeto utilizou-se o Javadoc, uma ferramenta que utiliza os comentários do código para montar um arquivo HTML com a documentação do mesmo.

Já na parte de automação o ambiente de desenvolvimento Arduino (baseada em processamento), e as suas codificações embarcadas no microcontrolador Atmega328 com a programação *Wiring* que é baseada na linguagem C/C++.

Hardware utilizado:

- Notebook para implementação e documentação;
- Processador Intel Core™ i7, 1.73GHz;
- 4 GB de memória RAM;
- HD 640 GB;
- Placa arduino-Uno-Rev3;
- Placa Protoboard.

II. FUNDAMENTAÇÃO TEÓRICA

2.1 Controle de Estoque

A manutenção de estoque é essencial para a empresa, uma vez que ela gerência as perdas, transvios e aperfeiçoa os valores para fins de avaliação, assim como, verifica o excessivo investimento, que afeta o capital de giro.

Segundo Slack et al. (1997, p.381), a definição de estoque seria: "O estoque é definido como a acumulação armazenada de recursos materiais em um sistema de transformação".

Para Garcia et al (2006, p.9),"gestão de estoques é um conceito que está presente em praticamente todo o tipo de empresas, assim como na vida cotidiana das pessoas".

Para Corrêa (1998), a gestão atual e do que esta por vir, tem o estoque como elemento gerencial crucial. Esses estoques possibilitam autonomia às fases dos processos de transição entre as quais se encontram. Durante o procedimento de modificação entre duas fases, interrupções de uma fase não interferem na outra devido ao fato de elas estarem mais independentes entre si em consequência de maiores de estoques.

Existem alguns tipos diferentes de estoques e seu controle. São eles:

Estoque de Produção / Processo - São estoques formados no decorrer da produção ou de processamento do produto, entre o período de fabricação e o deslocamento adequado deste produto até o paradeiro final.

Estoque de Organização - São empregados para sustentar o processo de produção ou abastecimento atuando constantemente sem pausas.

Estoque Regular / Cíclico - São fundamentais para atender aos pedidos nos espaços de tempo entre os reabastecimentos frequentes, dependendo da proporção dos lotes de produções, das restrições do local de estocagem, do gasto da manutenção do estoque.

Estoque de Segurança - São as unidades a mais, alimentadas manualmente acessíveis em um local de estocagem, para anteceder a situação em que a procura ultrapassa as expectativas ou apesar de que parte do lote tenha sido recusada na inspeção de qualidade.

Estoque Sazonal ou de Antecipação - São os estoques formados para abastecer condições de fornecimento quando a capacidade de produção e a procura estão instáveis. Quando estes intervalos são antecipados, os estoques podem ser adiantados para assegurar o abastecimento.

Estoque em Trânsito - São os estoques que estão em trânsito entre o lugar de armazenamento ou de fabricação. A capacidade de estoque em trânsito será maior de acordo com a velocidade de locomoção e distância.

Estoque Virtual (EV) - São as unidades que já foram encomendadas ao setor de compras e este já o concretizou, entretanto não alcançaram ao local de depósito ou ainda permanecem em procedimento de liberação pelo controle de qualidade.

Estoque Obsoleto - São os lotes que foram estragados ou reprovados na linha de produção, que estão deteriorando ou tem sua validade vencida.

Os estoques excedentes constituem gastos funcionais e possibilidade de recursos bloqueados, ao contrário registros de estoques baixos podem acarretar prejuízos econômicos e em virtude à falta de produtos, ocasiona perdas excessivas, dentro de um círculo empresarial.

Com a ampliação do número de SKU's (*Stock Keeping Units*s), o crescimento da distinção de produtos, bem como a concorrência mundial, tem prejudicado mais essa função de obter o melhor momento neste trade-off.

> Os gerenciamentos de estoques mostram os lucros alcançados pela empresa no decorrer das atividades financeiras, fixando suas atividades direcionadas na utilização de ferramentas administrativas fundamentadas em métodos que possibilitem a análise ordenada das técnicas aplicadas para atingir os objetivos pretendidos. (VIANA, 2002, p. 107).

As vantagens apresentadas por Ballou (1993) em relação à administração adequada dos estoques são: o aperfeiçoamento das tarefas de assistência ao cliente; atuação com segurança em oposição à alta de preços e imprevistos; conseguem ajustar a economia de escala nas compras; agem como abafador entre os pedidos e o abastecimento. Antes das compras a economia de escala destaca atenção da verificação dos registros de estoques. Entretanto se a empresa mantém uma quantidade elevada de estoques e não efetua esta verificação antecipadamente, gastos elevados no suporte destes estoques protegerão as economias formadas pelas compras de lotes maiores.

Para Dias (1993) os modelos de estoques identificados em uma fábrica são:

- Matérias-primas: Qualquer firma de alguma maneira tem um estoque de matéria-prima. Representam toda a matéria que é inserida ao artigo final. Seu gasto é correspondente à capacidade de fabricação, são elementos fundamentais para produção.

- Materiais em processo: Dispor de um estoque em excessivo volume desses elementos provoca maiores despesas para a empresa. Com o objetivo de que isso não ocorra obriga-se antecipar o rodízio do estoque. Normalmente esses materiais encontram-se inacabados, mas assumem diferentes qualidades no término do seguimento rentável. São elementos empregados nos sistemas de produção das mercadorias.

- Produtos acabados: São os produtos já fabricados, apesar de não vendidos. Entretanto em estabelecimentos que já dispõe de pedidos dessas mercadorias o estoque é baixo, porém em certas situações os itens são produzidos antes de realizar sua saída, isso acaba sendo estabelecido por meio de estimativas de vendas, por causa do procedimento e pelo capital investido.

Através do controle de estoques é possível calcular, a quantidade essencial que deve ser comprada na próxima requisição ao fornecedor. É um setor fundamental em uma empresa, grande, média ou pequena.

Atualmente o controle de estoque é realizado por meio de softwares, estes softwares são chamados de: Programas para controle de estoque, Sistema Integrado de Gestão ou Programas para Automação comercial.

Para Oliveira (1999) os sistemas computadorizados de gestão de estoque diminuem sincronizadamente as aplicações e os gastos com o suporte de estoques, assessoram a empresa a conceder serviço de alta qualidade a os clientes. Os sistemas de controle de estoque organizam informações que se revelam nas alterações nos produtos em estoque.

Bertaglia (2003) afirma que oposto do ponto de vista ultrapassado de operador que efetua e monitora os pedidos, hoje em dia o associado consciente pela aquisição passa a ter função de analista de abastecimento e agenciador. Com o emprego da TI – Tecnologia da informação, a atenção do administrador consiste em analisar a capacidade adequada a ser comprada naquela ocasião para a empresa, o controle das encomendas é efetuado de modo mais simples, eficiente e preciso.

Tecnologia da informação, a atenção do administrador consiste em analisar a capacidade adequada a ser comprada naquela ocasião para a empresa, o controle das encomendas é efetuado de modo mais simples, eficiente e preciso.

A seguir uma relação com as mais importantes vantagens de informatizar seu controle de estoque:

Controlar a entrada e saída de mercadorias.

Saber a quantidade de cada produto em estoque.

Saber a localização física do produto no estoque.

Ser avisado quando o saldo de algum produto estiver baixo.

Emitir relatório de produtos encalhados.

Emitir relatórios de curva ABC de produtos mais vendidos.

Evitar e detectar furtos ou fraudes dentro da empresa.

Mais agilidade no atendimento ao cliente.

De acordo com Martins (2002), os produtos em estoque são ordenados decrescentemente na posição de destaque, através da análise ABC, que é uma das ferramentas mais comum para se examinarem estoques. Essa verificação apoia-se na pesquisa, em custo financeiro ou quantidade, em determinado período de tempo do consumo.

Diversas organizações por receio que muitos produtos venham a se ausentar na sua linha de manufatura ou no estoque da sede de fornecimento, implicando deste modo no fornecimento de itens ao cliente, armazenam vários produtos em estoque.

É essencial classificar os produtos conforme a sua influência proporcional no estoque, diminuindo custo, sem prejudicar o grau de atendimento sustentando uma gestão melhor.

Através da ferramenta ABC, que ordena o estoque e consiste no raciocínio do diagrama de Pareto concebido pelo economista italiano Vilfredo Pareto, que é uma ferramenta antiga, no entanto muito eficiente. Portanto, possibilita diferentes níveis de domínio com fundamento na atenção referente ao produto.

Características da classificação ABC dos produtos:

Classe A: São os artigos de grande valia em virtude da sua influência econômica, fundamentais em estoques de elevada prioridade, ponto de cuidado do administrador de matérias-primas.

Classe B: Compreendem os artigos que até então são apontados financeiramente como fundamentais, depois dos artigos de categoria A, adquirem atenção moderada.

Classe C: Sua ausência impossibilita a continuação do processo, portanto não deve ser desconsiderado sua influencia. Porém o método estipula que seu encontro econômico não seja trágico, o que assegura pouco empenho.

Hoje em dia encontra-se uma série ampla de softwares no mercado, com a finalidade de realizar o controle de estoques e vários deles além de inspecionar o estoque, possibilita ainda inspecionar demais extensões da organização como o financeiro, faturamento e todas as operações que fazem parte do seu cotidiano.

Segundo Francischini (2002), os problemas mais preocupantes no processo de restituição de estoque acontecem em três circunstâncias cruciais: crescimento inesperado de

pedidos, crescimento não esperado de pedidos do produto em estoque podem acontecer por diversas circunstâncias, como, a vinda de um pedido considerável do item final para certo cliente, do aumento da fabricação para depósito do item final, ofertas, etc.

Atraso no procedimento da requisição de compra – problemas na estrutura de informação do almoxarifado ou do setor de compras podem acontecer no atraso considerável no envio da encomenda;

Demora no abastecimento pelo fornecedor – o fornecedor em decorrência de limitações no seu processo de fabricação, movimentação ou submissão da licença alfandegária, nem sempre tem possibilidades de atingir seus prazos de fornecimento. Deste modo no caso de alguma coisa desviar-se do ideado, o modo mais simples de abordar esse problema é deixar na coordenação dos clientes, após ter sido dimensionado um estoque mínimo ou estoque de segurança.

Estoque de segurança corresponde às mercadorias mantidas no depósito para qualquer imprevisto de ausência de mercadorias ou demora no fornecimento pela empresa, evitando a falta de produtos para os clientes.

> A melhor forma é adotar um sistema de segurança que supra toda e qualquer variação do sistema; porém, isso implicará custo elevadíssimo e que a empresa poderá não suportar. Então, a solução é determinar um estoque de segurança que possa aperfeiçoar os recursos disponíveis e minimizar os custos envolvidos. (POZO, 2004, p.66).

Vasconcelos (2003), aponta o estoque alto como um dos "pecados mortais" do comércio. "É um vício do período de inflação, que não foi perdido, mas atualmente se esperar para comprar, é possível que se consiga um preço menor". O exagero de estoque e a mistura desordenada, por ausência de um estudo detalhado, ocasionando na venda de produtos inexistentes no estoque e os existentes não consegue vender. Hoje, esperar para comprar mais perto da data da venda, pode-se conseguir pagar um preço menor. Na atual busca constatou-se a incapacidade de alguns gestores, em aperfeiçoar a movimentação de mercadorias nas empresas, ou seja, no gerenciamento de materiais.

O controle de estoque nas organizações age como se fossem o “pulmão” para atingir rendimentos maiores, o gestor que não aderir o regime de gestão, especialmente, a manutenção dos estoques de produtos, passará por extremas limitações nas conquistas de soluções importantes para a empresa.

2.2 Sistema de Informação

O sistema de informação tem como matéria prima a informação. Pode ser marcado como uma união de componentes interligados que distribui, resgata ou processa informação.

Em um enfoque mais simples Melo (2006, p. 30) observa que sistema de informação é "todo e qualquer sistema que tem informações como entrada visando gerar informações de saída".

Sistema de informação, segundo O´Brien (2013), é apontado como um conjunto de pessoas, informações, ações ou bens materiais em geral, combinados. Estes fatores relacionam-se entre si para conferir informação e transmiti-la de maneira adequada em função dos propósitos de uma empresa.

Segundo Cury (2000, p.116) "a organização é um sistema planejado de esforço cooperativo no qual cada participante tem um papel definido a desempenhar e deveres e tarefas a executar".

Os principais componentes de um sistema são:

a) A explicação das ideias, dos clientes como ao da linguagem, o escopo é a destinação da criação da estrutura;

b) Os acessos da linguagem, o qual seu funcionamento diferencia as energias que viabilizam ao sistema o equipamento, a força e os dados para o procedimento, estabelecendo assim as saídas;

c) O método de alteração do sistema, o qual altera a entrada em uma solução;

d) As saídas do sistema assemelham-se as soluções do método de alteração e devem ser adequados com as finalidades do sistema;

e) Os controles e as avaliações do sistema têm como objetivo conferir se as saídas estão adequadas com as finalidades;

f) A retroalimentação é a inclusão de uma saída com aspecto de dados.

O ato de se distribuir computadores em todo o estabelecimento (Bibliotecas, Centro de Documentação, Instituições, etc.), conectando-os em redes e colocando sistemas aplicativos, não irá ordenar a mesma, no que concerne aos sistemas de informação computadorizados.

Segundo Martins e Alt (2001), estes sistemas precisam estar alinhados aos propósitos planejados pela empresa. A decisão sobre os melhores sistemas, não é apenas os melhores que se apresentam no mercado, mas sim, os que mais se adaptam as necessidades das empresas. Segundo o autor, as tecnologias são apenas uma ferramenta que devem ser utilizadas para fins de melhoria no nível de planejamento e controle.

As organizações necessitam de uma análise estratégica de seus dados, das suas informações e de sua tecnologia da informação para que se tornem mais capazes devido às exigências e mudanças frequentes da sociedade da informação e a imposição que as instituições se tornem cada vez mais preparadas, fazem com que elas também se transformem.

> De acordo com Rezende o planejamento estratégico é fundamental para a sobrevivência das organizações privadas ou públicas que estão preocupadas com sua inteligência organizacional. Ele deve ser elaborado, implementado e avaliado a partir de estudos e pesquisas que relatam seu conceito, importância, benefícios, e resultados os quais devem ser discutidos, adaptados, sedimentados e aceitos por todos na organização, por meio de uma metodologia adequada (Rezende, 2008b).

Segundo BATISTA (2004, p.39), "o objetivo de usar sistemas de informação é a criação de um ambiente empresarial em que as informações sejam confiáveis e possam fluir na estrutura organizacional".

Para Rezende (2011), na introdução da elaboração de um projeto de sistemas de informação e da informática, é necessária a seleção de um critério eficiente, acertado e rápido. Além do procedimento organizacional, a análise dos sistemas de informação, dos sistemas de tecnologia e da informática deve ser posto em atuação alguns métodos como: compreensão por todos na empresa; execução e documentação; apoiado na criatividade, aprimoramento e integração; na demarcação dos papéis e compromissos; cooperação dos comprometidos no empreendimento; determinação, crescimento e correção do estudo. Devem-se admitir, até então, os fatos lógicos e adequação, reunidos para a busca da eficácia do projeto, os quais sejam observados quando da execução e análise.

BATISTA (2004), aponta que os sistemas são divididos em: sistema de categoria estratégica, de conhecimento, tático e operacional. Os dados concebidos pelos sistemas de categoria estratégica são usados no conceito do planejamento estratégico da empresa, isto é, obtenção de resolução. Os sistemas de nível tático são empregados no comando de análises funcionais, descreve as estratégias ou objetivos a serem realizadas. Os sistemas de conhecimento compreendem a transferência de instruções e conhecimento entre as repartições. Os sistemas de nível operacional são usados para o andamento das obrigações cotidianas da instituição, como modelo: sistema de compra e venda.

Para Laudon e Laudon (2004), há diversos modelos de sistemas, desse modo como existem diversos interesses, detalhes e categorias no meio de uma instituição, para ele qualquer sistema isolado conseguirá disponibilizar e proporcionar todos os conhecimentos que a instituição necessita. Determinado sistema de informação usado pelas instituições é o de controle de estoque, nele são compreendidos todos os dados do objeto, combinação de

identificação, especificação, número de quantidades atuais e venda dos artigos. Aplica-se da mesma forma o ponto de estoque mínimo, avisando da obrigação de realizar a recolocação, para que seja impedida a ausência do artigo em estoque. Esse sistema até então gera atas com todos os dados de entradas, saídas e devoluções de cada artigo.

Para Wanke (2000) a Tecnologia da Informação TI acarreta grande chance de acerto de vendas e maior assistência dos graus de estoques proporcionando oportunidades para que os próprios consigam ser diminuídos.

Para Gordon & Gordon (2006, p.6) "A tecnológica da informação (TI) inclui, hardware, software, sistemas de gerenciamento de banco de dados e tecnologias de comunicação de dados".

É de essencial relevância que a instituição ajude na procura da produtividade e não apenas da eficácia, ainda que tenha constatação da carência de eficácia na atividade de TI, assim como um elevado conhecimento prático. Desta forma, a procura da eficiência da aplicação de TI, autoriza que a instituição deva ter benefícios concorrentes. Os resultados decisivos mostram a direção de um plano de aplicação em TI o qual solicita o trabalho do administrador da menor empresa, para o reconhecimento dos métodos administrativos. Como recurso em sistemas, foram recomendados sistemas incorporados – ERP direcionados para a existência destas empresas e o uso da internet como mecanismos MSN e Skype, para instituir ligações entre provedor e usuários.

Segundo Correa (1998), as empresas adotaram amplamente os sistemas integrados de gestão ou ERPs (*EnterpriseResource Planning*. Sendo acessível apenas a organizações de amplo porte, devido ao seu custo neste período. No decorrer dessa década, o mercado das grandes corporações estabeleceram suas preferências sobre os sistemas a serem obtidos e firmados, limitando as probabilidades de operações comerciais dos provedores de ERPs nesse segmento empresarial, saturando assim o mercado das grandes empresas.

A introdução de sistemas ERPs – para as corporações que adotam pela sua aplicação, como ocorre com qualquer ferramenta resulta de tal maneira benefícios e perdas. Os principais benefícios da aplicação de sistemas constituídos de gerenciamento são: automação das funções, como resultado da exclusão de interfaces manuais, economia de tempo, diminuição de gastos, expansão da produção, evolução da qualidade de dados acessíveis na instituição, diminuição nos limites de tempo de resposta ao negócio. Entre as perdas estão: o aumento de gastos e a submissão entre as seções (um setor pode ser afetado se um deles não abastecer o programa, até haja a transferência de dados) e a impossibilidade de um sistema ERP de converter a organização em uma instituição globalizada.

O êxito de uma instituição pode derivar de um sistema de informação ERP eficaz, o que pode acarretar um impacto significativo nas táticas institucionais da empresa.

Esse impacto consegue favorecer determinado elemento que se relaciona com os SI's, a empresa, os clientes e ou utilizadores.

Essas novas ideias migraram o enfoque da "administração de informações" para a "administração dos procedimentos e usuários". Nessas circunstâncias, a carência de formação dos divergentes espaços, procedimentos, programas, suporte de informações e todos os outros ativos de conhecimento preencheram os pensamentos dos gestores de TI. Aparecem com grande dinamismo as definições de EAI (Enterprise Application Integration) e as amplas e caras maneiras de linguagem de middleware.

Contudo, uma transformação até então superior acontecia, acrescentando mais alterações à difícil equação da inclusão de sistemas: a Internet. A alternativa (ou carência) de conceder parte dos conhecimentos institucionais a clientes ou sistemas que ultrapassavam os limites institucionais, estabeleceu que o modelo variasse mais uma vez. A satisfação das funções cliente-servidor aparentava estar com os dias computados. Atuais paradigmas de permissão aos dados foram gerados, todos constituídos em operações rápidas, através das atuais circunstâncias da web. Ainda, atuais sistemas foram incorporados ao sumo, entre eles, sistemas mais aprimorados de segurança, inúmeros programas para intranet e extranet, etc.

2.3 Automação Industrial

A expressão automação está rigorosamente relacionada a atividades que não resultam da interferência humana, ou seja, são realizadas através de comando automático.

Ainda segundo Rosário (2005), a automação industrial pode ser vista como uma ciência constituinte de três campos: a informática pelo software que irá comandar totalmente o processo, a eletrônica responsável pelo hardware, à mecânica na condição de artifícios mecânicos (atuadores).

Nestas circunstâncias, para realizar projetos neste campo requer uma ampla série de informações, estabelecendo uma equipe bastante extensa e variada de projetistas, ou então uma atividade em grupo devidamente organizada com características de união de disciplinas.

A vantagem de utilizar sistemas que envolvem diretamente a informatização de acordo com Moraes e Lauro (2007, p.12), "é a possibilidade à expansão utilizando recursos de fácil acesso; nesse contexto, são de extraordinária importância os controladores lógicos programáveis (CLP)".

Segundo o autor Silveira e Santos (2009), automação industrial tem como propósito de aumentar a capacidade e produtividade da instituição, com o emprego de softwares ou instrumentos precisos para instrumentalizar um equipamento ou um núcleo de fabricação. Esta frase é usada entre as fábricas, uma vez que insere do mesmo modo, nos equipamentos de grande capacidade ou até mesmo pequena, automatizados, para confeccionar os respectivos procedimentos.

A fim de garantir, assim, a perfeição e um padrão dos artigos e assegurando qualidade estável dos produtos elaborados, através da redução de trabalhos manuais, com a implantação destes equipamentos trazendo vantagens aos contribuintes. Desta forma, a automação industrial possibilita ao indivíduo a elaboração de um projeto em tarefas que requeriam capacidade lógica, como na situação de domínio e reparo de falhas no decorrer dos procedimentos de produção, proporcionando, assim, melhores possibilidades de estabilidade, com a elevação da fabricação com pequenos gastos.

2.4 Metalúrgicas

A metalurgia e uma área da engenharia que está associada com a produção de metais e ligas metálicas, que são capazes de se apresentar ao longo da aplicação do metal como um constituinte ou componente de uma infraestrutura ou ferramenta, com o aspecto e com as características adequadas a sua aplicação técnica.

Para Dieter (1981, p.1), “A metalurgia mecânica não é uma matéria que pode ser estudada isoladamente. Na realidade, é uma combinação de diversas disciplinas e diferentes abordagens ao problema da interpretação da resposta dos metais a forças”.

Visa respostas, quando são empregadas forças ou cargas aos metais, circunstância onde temos de conhecer valores limites que podem ser suportados sem que ocorram falhas na estrutura.

A metalurgia é uma tecnologia atual, que pode ser representada como um campo de uma ciência mais ampla, chamada de Ciência e Tecnologia dos Materiais.

Segundo D.callister Junior e Rethwisch (2012, p.2), “Ciência dos Materiais engloba a análise das relações que existem entre as estruturas e as propriedades dos materiais. Enquanto Engenharia dos Materiais está relacionada com o projeto, ou na engenharia da estrutura de um material analisando o arranjo de seus componentes internos”.

Ao longo dos tempos, os metais mais explorados eram o cobre, prata e ouro os quais eram extraídos no estado bruto e pelo fato de serem maleáveis permitiam por martelagem serem utilizados como ferramentas, ornamentos e outros objetos simples.

O ferro meteórico ("ferro do céu") foi o metal primordial a ser utilizado pelo ser humano. A chegada dos primeiros materiais metálicos, em ouro ou cobre, estabeleceu o término da era neolítica e o surgimento da idade do Cobre, à qual se seguiram a Idade do Bronze e por fim a Idade do Ferro.

O estudo da metalurgia possibilita o seu desmembramento em três amplas áreas: metalurgia mecânica, metalurgia química e metalurgia física.

A metalurgia química está vinculada com as técnicas de extração dos metais e com as características químicas dos metais e ligas, a partir dos seus minérios.

A metalurgia física, vincula as formações atômicas das etapas atuais com as características físicas, elabora a análise da formação e da estrutura dos metais e ligas.

Enfim, a metalurgia mecânica, também chamada por metalomecânica, realiza a pesquisa para alcançar a conformação pretendida para as matérias-primas metálicas. É nesta área que se analisam os métodos de deformação plástica: forjadura, laminagem, extrusão, trefilagem, estampagem, entre outros. São também pesquisados os métodos de maquinagem e de soldadura.

2.5 Engenharia de Softwares

Engenharia de software é uma zona da ciência da computação direcionada a caracterização, evolução e conservação de sistemas de software empregando conhecimentos e técnicas de coordenação de planos, planejando disciplina, eficiência e qualidade.

Sommerville (2007), afirma que Engenharia de Software baseia-se no uso de uma aproximação sistemática, ordenada e avaliada corretamente de acordo com as convicções da ciência da computação. Ademais, ela abrange o conhecimento e a procura por aproximações para o andamento de ações ligadas a extensões de software.

Recentemente é normal que instituições de criação de software, efetuem a documentação inteiramente de seus sistemas, tendo como objetivo, portanto modificações, aperfeiçoamentos e otimizações futuras, desta forma, conseguimos explicar determinada atitude, como uma uniformização, visando a intenção de atender a instituição contratante e dessa forma facilitar a rotina dos desenvolvedores.

A falta de uniformização consegue conceber uns resultados nem um pouco promissores para os incluídos na concepção, com o avanço do gasto do software, que consegue acarretar a renúncia da proposta por parte da instituição empregadora, avaliação de tempo de elaboração inábil, acarretando assim perdas para a organização convencionada.

Guedes (2011), sugere que todo software tem de ser modelado, visto que há uma enorme desigualdade entre edificar uma residência sem planta e um edifício. É necessário a elaboração de um projeto e documentação. Por mais fácil que seja um software é sempre necessário ajusta-lo, porque os sistemas de informação normalmente dispõe a característica de crescer.

Pressman (2011), estabelece que possam existir muitos modelos de etapas de existência. Não existe uma configuração exclusiva e referência para a solução das barreiras de se elaborar um aplicativo, o que se encontram é diversos exemplos, instruções e recursos que envolvem todas as etapas de elaboração, que unidos e devidamente empregados constituíram a engenharia de software.

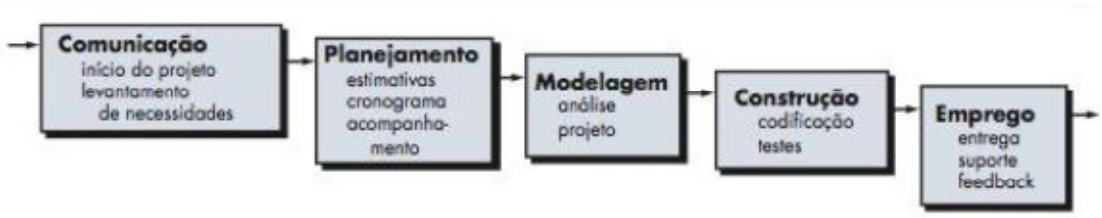

Figura 1 - Cascata. (Fonte: Engenharia de Software, PRESSMAN p. 60).

Ciclo de vida modelo cascata (Clássico), (ilustrado na Figura 1), o padrão concebido por Royce em 1970, igualmente denominado como tratamento ‘top-down’, tem como fundamental particularidade o seguimento de operações no qual qualquer etapa decorre plenamente e seus serviços são observados como acesso para uma recente etapa. A modelação em cascata é o mais tradicional paradigma de crescimento aplicado para aperfeiçoar um programa, e o ciclo de vida é uma metodologia em série formada por programação, verificação, plano, crescimento, avaliação e correção.

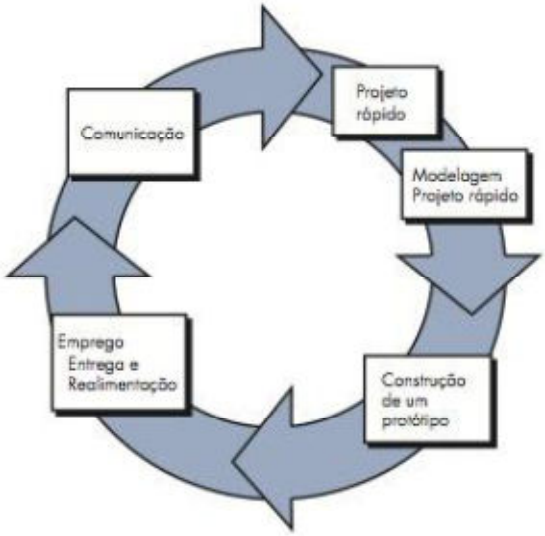

Figura 2 - Prototipação. (Fonte: Engenharia de Software, PRESSMAN p. 63).

A prototipação, que é o arquétipo usado neste trabalho (ilustrado na Figura 2), é um padrão usado cada vez que há somente uma concepção inicial de que o software deve realizar. Assim, o analista ou programador não tem noção de como serão as atividades, sugerindo desta forma três prováveis maneiras:

a) um protótipo escrito em papel, com uma concepção melhor desenvolvida como solução aos conceitos primitivos;

b) um protótipo em formato digital, mostrando limitadas utilidades possíveis e um subconjunto de particularidades;

c) um software já efetivo que pode ter uma fração ou totalidade das atividades fundamentais previstas, no entanto com diversas peculiaridades que irão se adequar pelo atual desenvolvedor para satisfazer o usuário.

A prototipação mostra uma etapa de crescimento em que o analista reuni e aprimora as exigências, prepara um plano eficaz e cria um modelo. Logo após o usuário analisa o modelo, e mais uma vez o modelo é aprimorado podendo retornar para a exibição do software ou prosseguir na engenharia. No momento que o software deixa a condição de modelo, ele adquire o número das versões (em termos atuais, sai da versão Alfa ou Beta).

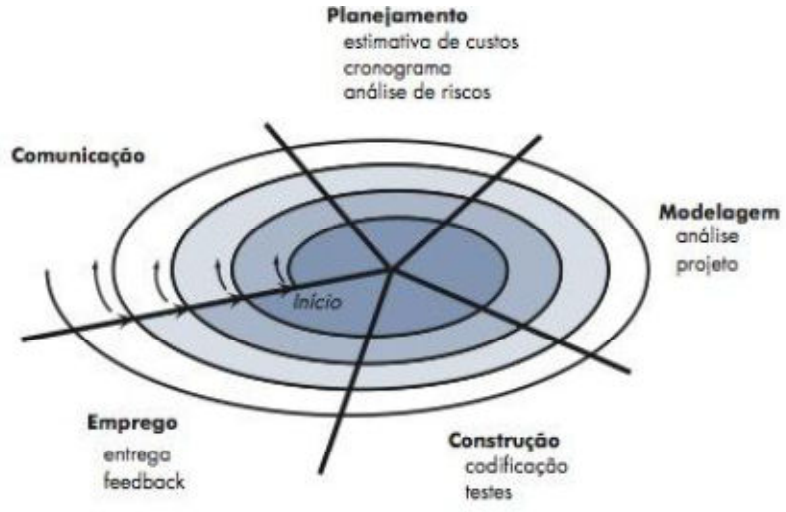

Figura 3 – Modelo Espiral. (Fonte: Engenharia de Software, PRESSMAN p. 65).

O modelo espiral agregou o ciclo de vida clássico com a prototipação (ilustrado na Figura 3), transferindo um modelo atual de verificação. Ele apresenta quatro consideráveis operações: Planejamento, diagnóstico de ameaças, criação e análise pelo usuário. Ou melhor, nesta circunstância o usuário qualifica o aplicativo apenas depois de o próprio ter sido analisado pela engenharia, e as avaliações iniciais são executadas pelos programadores. Este é um exemplo utilizado em amplos sistemas, na qual o conhecimento da equipe de programação é amplo e a análise do usuário não resulta em importantes mudanças do aplicativo.

A Prototipação é uma etapa de existência para o crescimento de um aplicativo. Uma estratégia necessita ser inserida em uma etapa de vida, uma forma de detectar suas condições incorporadas no crescimento, com o objetivo de que cada atividade seja atingida conforme as exigências estabelecidas pelo sua condição.

Um dos benefícios de ocupar-se com esse ciclo de vida é que tem um fluxo de acontecimentos simples e do mesmo modo um protótipo eficaz incluso na engenharia de software. Contudo é fundamental que o usuário como o engenheiro encontra-se de acordo que prototipação será o protótipo a ser adotado para a evolução do sistema, e que este auxiliará como mecanismo desenvolvimento do sistema final.

Segundo Pressman (2011), a prototipação inicia com o estudo e coleta das exigências. Neste seguimento é necessário estabelecer os detalhes fundamentais e exigências a serem analisados pelo utilizador, portanto a Prototipação tem em vista elaborar um código em pouco tempo e que satisfaça exclusivamente as necessidades do cliente. Não adianta realizar uma extensa pesquisa e um planejamento prolongado, porque a prototipaçao apenas atrasará o desenvolvimento nesta circunstância.

Posteriormente a pesquisa necessitará ser realizado um estudo de imediato, analisando somente os baixos pontos no qual o usuário possua atividade, como entradas e saídas fundamentais, interface básica e de qualquer aspecto cordial, contudo não se impressionando com aspecto e aparências definitivas. O usuário necessitará ter um contato inicial com os seguimentos básicos e poderá mostrar todos os obstáculos, várias dificuldades, atuais conceitos e tudo que precisar que altere, portanto o modelo conduzirá à medida que o usuário pode solicitar e vai auxilia-lo a compreender como a aplicação irá atuar.

Após o usuário ter seu conhecimento inicial com o modelo do sistema, o software receberá um aperfeiçoamento, que seria o atendimento de determinadas obrigações solicitadas pelo usuário, mudanças e correções. O aperfeiçoamento presta como acerto que apresentará até que nível o programa consegue aperfeiçoar a atividade e satisfazer as necessidades, portanto nem tudo é apurado. Esta fase se renova ate que o modelo esteja atuando de maneira a satisfazer quase todas as exigências do usuário e agir de forma agradável.

Depois de muitos aperfeiçoamentos e reprogramações, o modelo pode ser enfim excluído. Este é um segmento difícil de tal maneira para o desenvolvedor como para o usuário. O usuário não permanecerá contente em escutar que deverá ser criado software atual, e o engenheiro do programa não tem muita calma para continuar incrementando tudo mais que uma vez. Entretanto essa é a finalidade da prototipação, estipular todas as dificuldades do usuário e buscar as resposta de maneira prática ao apresentar soluções e auxiliar na compreensão para o desenvolvedor e engenheiro.

A conveniência de excluir o modelo é que o mesmo recebeu muitas alterações no código que certamente está com algumas funcionalidades irregulares, atributos sem formatação, projeto confuso e improviso por todas as lógicas. A realidade é que não será excluído o sistema, mas somente será formado um atual segundo o modelo, mas uma vez que já conhecendo as soluções aguardadas, quais as implementações serão realizadas em cada ciclo e terá um modelo em seu código, de maneira simples para realizar correções após um período longo.

A necessidade de entendimento entre o usuário e o engenheiro se faz absolutamente essencial, visto que depois de muito tempo de utilização do software funcionando, este deverá alterar e por último conseguir aspectos adequados. A prototipação não afasta a importância de documentar o código, todavia dá a chance de se estabelecer as documentações indispensáveis como o andamento ao mesmo período, obtendo soluções em um reduzido período e a desempenho de uma excelente atividade entre o campo de análise de sistema, engenharia de software e programação.

2.6 Programação Orientada a Objetos (POO).

Revoluções tecnológicas no campo da Informática têm gerado um dever de prática e manuseio de dados que anteriormente não eram aplicados. Os modelos de informações complexas, como os objetos, passaram a ser trabalhados por intermédio dos códigos de programação, que herdaram o conceito de Linguagem de Programação Orientada a Objetos.

Apoiada na ideia dos tipos de Álgebra, a orientação a objetos apontou na década de 60, no entanto apenas na década de 90 iniciou a ser largamente aplicada em computadores. Ela tem como causa essencial demonstrar o universo atual e a maneira de comunicar-se com os objetos, adequando à formação e o desempenho em uma entidade exclusiva.

De acordo com Camara (2006), orientação a objetos tem o propósito de entalhar o mundo real, e assegurar que os encargos de suporte serão superiores perante esta situação. Isso é concebível, uma vez que empregado uma linguagem de programação orientada a objetos pode-se atingir um progresso mais ágil, como este progresso acontece em módulos, em segmentos de códigos semelhantes aos objetos e suas ligações. Por meio da orientação a objetos consegue-se adquirir um nível melhor e eficiência no processo, uma vez que o elemento reusabilidade possibilita que se reempreguem os demais objetos que foram primeiramente concebidos e conseguem ser facilmente adicionados na aplicação. O reemprego, além disso, assegura um maior manuseio da aplicação, uma vez que os ensaios relacionados aos elementos, em seguida foram antecipadamente efetuados, assegurando deste modo a aplicação consistente dos objetos.

Em alguns códigos de programação como o Java, as variáveis como o int e float não são chamadas como objetos. Na linguagem de programação OO clara, a partir de características mais simples, tudo é um objeto, como inteiros e lógicos, até as instâncias de classe mais complicadas, nem todas as linguagens orientadas a objetos atingem a essa fase.

Com finalidade de realizar uma linguagem mais nítida e reaproveitável, a POO usa diversos conteúdos em seu paradigma de elaboração como: classe, objetos, atributos, agregação, encapsulamento, herança, polimorfismo.

Deitel (2010) afirma que classe esclarece os atributos e ações frequentes comuns por meio de um modelo de objeto. Os objetos de um determinado padrão ou categoria distribuem as idênticas ações e atributos.

Objeto é uma classe sendo instanciada. Um objeto é apto a guardar condições por meio de seus atributos e responder a mensagens emitidas a ele, desta forma concatenar e mandar instruções a vários objetos.

Atributos são fatores de uma classe que podem ser notórios superficialmente.

Segundo Goetten Junior e Winck (2006), encapsulamento é reunir em certo local, códigos que por diversas vezes surgiam em um grupo no meio de um software, dando um título a esse conjunto. Dessa forma despontou a definição de sub-rotina, no lugar em que acabasse sendo essencial, coseguiria solicitar a realização do conjunto aplicando a denominação adotada.

Por meio do encapsulamento, é permitido para o programador usar entidades de programa como se fossem caixas-pretas. Em outros termos, o usuário, aguardando ter certa solução e tendo somente um olhar aparente da ação, remete avisos a uma caixa preta sem se incomodar com as particularidades da elaboração.

Herança, segundo Camara (2006) é uma estrutura que aceita que aspectos simples a diferentes classes tornem-se auxiliadoras em uma classe inicial, ou superclasse. A começar de uma classe inicial, diversas classes conseguem ser estabelecidas. Toda classe procedente ou subclasse exibe qualidades (estrutura e métodos) da classe inicial e adiciona a elas o que for estabelecido de características para a própria.

De forma geral, polimorfismo representa "várias formas". Segundo Serson (2008), em Orientação a Objetos polimorfismo é o início do qual duas ou mais classes derivadas de uma mesma superclasse conseguem chamar métodos que possuem igual identidade (assinatura), mas ações diferentes, particularizadas para cada classe derivada, utilizando para isso uma relação a um objeto da forma da superclasse.

Há quatro formas de polimorfismo as quais podem ser vistas a seguir.

Os polimorfismos do padrão ad-hoc (aparente), não possuem movimento, denominados de estáticos. Não há uma maneira exclusiva e organizada de definir a forma de solução de uma atividade em condições da forma da proposta de inicio. Eles se classificam em polimorfismo de sobrecarga e polimorfismo de coerção.

a) ***Sobrecarga*:** Autoriza usar em uma classe idêntica um único detector para apontar diversos métodos variados. A desigualdade está na quantidade e na característica de seus critérios.

b) ***Coerção:*** Possibilita usar critérios de características divergentes das que foram estipuladas na autenticação do método. Desta forma é realizada uma modificação de categorias, tais como:

a) Ad-hoc Universal;
b) Sobrecarga Coerção Inclusão Paramétrico de inteiro para real, para em seguida executar o método.

Os polimorfismos do gênero global são ágeis e possibilitam que um número adote um valor ilimitado de condições:

a) ***Inclusão*:** Propicia elaborar métodos aptos a herdar critérios de várias classes diferenciadas, caso façam parte a uma herança de classes. Em outros termos, conseguimos conceituar um método exclusivo habilitado a realizar ativamente as mesmas atividades para objetos de classes diversas em uma categoria de classes.

Sobreposição**:** É uma condição única de polimorfismo de inclusão. Possibilita que métodos fixados em classes provenientes e com autenticação igual do método identificado na superclasse consigam ser restabelecidos executando desta forma uma pratica atual.

b) ***Paramétrico:*** Admite gerar métodos e formas comuns. Desta maneira, consegue-se remover o perfil dos critérios de acesso e a forma de resposta dos métodos. Esses conceitos são omitidos durante o instante da aplicação.

Em virtude a suas propriedades, os polimorfismos de sobrecarga e de inclusão são os mais empregados.

2.7 Java

Uma linguagem fundamentada em C e C++, designada como Oak (carvalho) pelo seu fundador em consideração a uma árvore que se situava frente à janela do seu escritório na Sun. Constatou-se depois que já existia uma de computador chamada de Oak. Na ocasião em que um grupo da Sun conheceu um café na região com o nome de Java (cidade de origem de um tipo de café importado) este nome foi indicado e ficou.

Segundo Deitel (2001), o Java foi difundido em 1995 como uma forma de integrar assunto ágil a paginas da World Wide Web. Ao invés de páginas somente com conteúdo e figuras estáticas, as páginas pessoais da Web obtiveram animações com sons, clipes, movimentações, interatividade, brevemente, cenas tridimensionais. Os meios de Java são justamente o que as instituições e as associações necessitam para satisfazer as exigências de procedimento de dados atuais. Neste caso em seguida avistamos em Java a capacidade para ser uma das fundamentais linguagens de programação de emprego comum do mundo.

O Java é empregado para elaborar softwares institucionais de imensa importância, possibilitando realizar diversos padrões de softwares, com divergentes finalidades, viabilizam programas para aparelhos de uso geral e para diversas finalidades, quem a adota não corre a ameaça da ausência da mesma durante um intervalo de tempo.

Para Costa (2009, p.171), “Java é uma linguagem que oferece portabilidade. Um programa portável é aquele que necessita de muito pouca ou nenhuma alteração no seu código para poder ser executado em plataformas diferentes”.

A Java Virtual Machine (JVM) é responsável pela interpretação de um agrupamento individualizado de informações, ou seja, um bytecode, que é o resultado da resposta de um compilador Java.

> JVM é a máquina virtual Java e existe uma para cada plataforma como Windows, Solaris, Sparc, Mac OS e Linux. Executa as instruções que o Copilador gera e contém os mecanismos necessários para garantirem a integridade de classes. As Versões JRE seguem exatamente as mesmas versões do JDK, e, portanto ao desenvolver uma aplicação com uma versão do JDK você deverá utilizar preferencialmente a mesma versão do JRE para executá-la. (BONAN, 2008, p.XXXIII).

É interessante conhecer que o JRE é o Java Runtime Enviroment que é utilizado unicamente para executar os códigos em Java e o JDK é Development Kit, isto é, o conjunto para desenvolver Java.

Para codificar o software, no Java, o programador lança o comando javac. O conversor Java interpreta o software Java para bytecodes – o código compreendido pelo tradutor Java. Se o software converter de modo correto, será criado um arquivo designado.class, esse arquivo comporta os bytecodes os quais irão ser representados ao longo da etapa de implementação.

Para Deitel (2010), programas Java baseiam-se em frações denominadas classes. As classes compreendem elementos denominados métodos que desempenham funções e devolvem dados na ocasião em que as funções são completadas. Desenvolvedor Java extraem benefícios de amplas coletâneas de classes encontradas em bibliotecas de classes Java. As bibliotecas de classes são igualmente denominadas como Java APIS (Applications Programming Interfaces – interfaces de programas aplicativos).

O framework que os programadores mais conhecedores escolheram para incrementar é o eclipse, isto é, através de plug-ins é capaz de programar em diversas linguagens, mesmo assim a favorita é o Java determinadas concessões da linguagem operacional Linux e agora busca como referência o eclipse, um excelente instrumento para o ambiente Java.

2.7.1 Interface Gráfica com o Usuário (AWT/GUI)

Em Java, as classes estão reunidas dois pacotes amplos: java.awt (pacote do núcleo) e javax.swing (pacote de extensão), em que nos apoiamos para elaborar os componentes, assim como proporcionar-lhes aplicabilidade. Os dois pacotes determinam elementos com características diferentes. É fácil criar utilizações gráficas extraordinariamente impressionantes e bonitas por intermédio dos pacotes AWT/Swing, e produzirem aberturas no modelo Windows, os dois são causadores de efeitos visuais, formas gráficas e do processo em questão.

O software Java proporciona realizar execuções na forma de texto e na forma de gráfico, utilizando AWT / Swing.

A AWT é um campo de ligação normalizada para a entrada às variadas bibliotecas de elementos das variadas GUIs adotada pelo Java. Esse conjunto de elementos para elaboração de Interfaces Gráficas está ultrapassado e foi alterada pelo projeto Swing.

Os elementos Swing, uma bagagem proveniente da AWT, são executados sem qualquer código nativo, isto é, além de serem evidentemente mais demorado que os elementos nativos, eles proporcionam maior autonomia aos desenvolvedores.

O pacote javax.swing foi concebido em 1997 e compreende os elementos GUI que se tornaram referência em Java desde a versão 1.2 da Java 2. A maior parte dos elementos Swings (assim são denominados), são redigidos, trabalhados e expostos totalmente em Java, sendo denominados como elementos Java puro. Passam a ter um “J”, no nome de determinados elementos, como, amostra: JLabel, JButton, JFrame, JPanel, etc.

De acordo com Deitel (2010), um campo de comunicação gráfica com o cliente (GUI) exibe uma interface inovadora, adequando-se a um software um “visual” e “desempenho” diferenciados, permitindo a diferenciados programas uma coleção de elementos evidentes de interface com o cliente. As GUIS permitem ao usuário gastar menos tempo tentando lembrar o que faz certas sequências de pressionamento de tecla, despendendo mais tempo utilizando o programa de uma maneira produtiva.

A GUI é um componente com que o utilizador atua mutuamente por intermédio do mouse ou teclado.

O Swing também propicia agilidade para customizar o visual e a forma dos elementos conforme a maneira própria de cada plataforma, ou ainda modifica-los ao mesmo tempo em que o software está senso rodado.

As alternativas são a caracterização com o estilo do Apple Macintosh, do Microsoft Windows, ou do Motif (UNIX).

2.7.2 Hibernate

Hibernate é uma ferramenta open source e pertence à JBoss Inc. Refere-se a um código e download livres, recursos podem ser elaboradas e adequadas ou incorporadas ao Hibernate, deixando esse framework compacto.

Segundo Antonio e Ferro (2009 p.7), "Hibernate é um framework que tem alto desempenho para persistência de objetos usando o modelo relacional e também serviços de consulta (query)".

O Hibernate admite atuar com persistência em cima do banco de dados, livre da obrigação de ter códigos SQL na classe da linguagem Java, como a maior parte dos desenvolvedores de sistemas adota o emprego de banco de dados conectados. As anotações do banco de dados necessitam ser alterados em objetos e os dados compreendidos nos objetos devem ser persistidas em configuração de linha e colunas.

Denominamos isso de "Mapeamento Objeto - Relacional".

O Hibernate segundo Bauer e King (2004), permite o desligamento entre a classe de negócio e a classe de persistência ao mesmo tempo em que cria a relação objeto-relacional. Ele possibilita que as informações persistidas mudem para a classe de negócios como objetos, utilizando o modo de programação orientado a objetos.

Hibernate permanece disposto como uma camada no meio do programa e do banco de dados, cuidando de sustentar e salvar os objetos. A camada de Persistência é de absoluto destaque, uma vez que ela tem o dever de encapsular todo o método útil com o objetivo que um objeto ou um conjunto de objetos sejam preservados ou restaurados de um ambiente de arquivamento. A aplicação da classe de Persistência faz com que o banco de dados seja mais seguro, desta maneira separa as atividades que têm permissão livre ao banco de dados.

O emprego do Hibernate na arquitetura do sistema foi em razão de alguns benefícios na sua aplicação:

a) Base de código livre e de utilização gratuita;
b) Reutilização de código;
c) Diminuir o período de elaboração e o gasto imediato;

Para a elaboração da classe de persistência aplicando o Hibernate, foram geradas classes Java conforme as tabelas atuais no banco de dados. Estas classes futuramente serão

aplicadas para as operações de inclusão, alteração e listagem de informações direto da origem. E todas as atividades subsequentes executadas serão compromissos do Hibernate.

2.7.3 JPA

O conceito da JPA é que possa ser possível trabalhar sem interrupções com as classes e não ter que empregar as informações provenientes de cada banco de dados, essa tarefa será realizada pela JPA e não mais pelo desenvolvedor.

A JPA usufrui de excelentes conceitos de conhecimentos como a persistência TopLink, JDO e Hibernate. É aplicável com o ambiente Java SE, bem como Java EE e possibilita dispor de proveitos da API de persistência modelo.

Associa denominações de classes e atributos com nomes de tabelas e de colunas, além de estabelecer uma referência de escolha de chaves estrangeiras e tabelas associativas. Deste modo, estas informações devem ser notificadas somente no momento em que permaneçam distante do modelo. (DEMICHIEL; KEITH, 2006).

É constituído na ideia POJO (Plain Old JavaObject) e agrega conceitos de consagrados frameworks de persistências para normalizar a relação Obeto/Relacional no Java. Na JPA, os arranjos persistentes são chamados entidades (Entities). Uma entidade é um arranjo simples (POJO) que retrata um grupo de elementos persistidos no banco de dados.

> No contexto da JPA, chamamos de entidades as classes que estão vinculadas com uma unidade de persistência. Como a JPA trata de mapeamento objeto-relacional, a unidade de persistência é um banco de dados, e de modo geral, uma entidade estará associada a uma tabela nesse banco. (CORDEIRO, 2013, p.28).

A JPA aceita dois sistemas de consultas (querys) para comunicar-se com o banco de dados, sistema SQL e a Java Persistence Query Language (JPQL).

O sistema primário do JPA é a JPQL que oferece como fundamental benefício à autonomia do banco de dados.

2.7.4 MVC

O desenvolvimento do software seguiu a arquitetura MVC (Model-view-Controller).

Esse conceito visa dividir o desenvolvimento de sistemas em camadas separadas que trabalham em conjunto formando um projeto completo. O uso desse conceito auxilia o processo de desenvolvimento, por propor uma melhor organização de código-fonte. (SAMPAIO, 2007, p. 80).

Segundo Jobstraibizer (2009), o MVC, desagrega em três classes uma função, são elas Modelo, Visão e Controle. Definição de engenharia de software para desenvolvimento e design.

A Camada de Modelo baseia-se nas camadas do código e na obtenção dos dados.

Perceba que cada classe dispõe de uma função específica adentro do software. É o ambiente no qual deve ser adicionada a camada de persistência (DAO - *Data Access Object*). Essencialmente essa classe aborda os conceitos e entradas referentes a banco de dados.

A camada de **Visão** é associada à aparência da aplicação, isto é, as imagens que serão apresentadas para o cliente. Nessa camada somente os requisitos de imagem devem ser elaborados, como janelas, botões e mensagens, com a finalidade de auxiliar o suporte, e deste modo à interface modificada por algum web designer sem intervir no restante da aplicação.

Já a camada de Controle, é na qual toda a fundamentação da aplicação é realizada. Essa camada destina-se a elaboração dos códigos de transação e aprovações, que podem ser operações do usuário ou do sistema, realizando comunicação com o Model, como a confirmação de ações aprovadas e o processamento de informações do cliente.

O MVC é um modelo de Arquitetura de Software e não um modelo de Projeto é um dos mais usados e propagados pelos programadores especialmente pela utilidade e finalidade.

Os benefícios da aplicação do MVC no sistema são na organização, permitindo uma maior compreensão da arquitetura do código, no momento em que será usado por outro programador que inicie as atividades no programa. Agilidade no tratamento e verificação das exceções. É capaz de mostrar em qual classe a falha acontece, ao invés de vasculhar o código atrás de falha. Outro ponto importante do MVC é a segurança da transição de dados.

2.7.5 Banco de Dados

Um banco de dados é uma organização de dados. O Banco de dados utilizado no sistema foi o mysql, que é uma estrutura de armazenamento de informações relacionais de código aberto, idealizado a proporcionar uma assistência a banco de dados cliente/servidor, disponibilizando velocidade e agilidade. Além de proporcionar uma relação com diversas ferramentas de gerenciamento e interfaces de programação, assim como C, C++, Eiffel, Java, Perl, PHP, Python e Tcl. Consegue ser executado em qualquer servidor e comportar diversas aplicações cliente.

Segundo Deitel (2010 p.899), “os sistemas de banco de dados atuais mais populares são os bancos de dados relacionais, nos quais os dados são armazenados sem levar em conta sua estrutura física”.

Para Ferrari (2006), o SQL (Structured Query Language) é empregado como linguagem padrão, pelos bancos de dados relacionais. Onde todas as informações são interpretadas como interações matemáticas. Os dados são organizados em tabelas, separadas em linhas e colunas: as linhas são os registros e as colunas representam os campos.

Para utilizar as propriedades e utilidades do SQL, necessitamos de um lugar no qual possamos empregar as instruções desta linguagem para manipular os dados. Esse lugar passa a ter o nome de Sistema de Gerenciamento de Banco de Dados Relacionais (SGBDR).

O Mysql é um software gratuito, compatível para os comandos SQL, foi idealizado tendo como modelo os sistemas mais empregados no mercado (Microsoft SQL Server e Oracle).

III. ARDUINO E COMPONENTES

O arduino (ilustrado na Figura 4), é um microprocessador que pode ser configurado, é capaz operar como um minúsculo computador, sempre que integrado com alguns acessórios.

Ele consegue corresponder aos impulsos (eletricidade) provenientes de sensores (foto célula) e utilizar estas informações para verificar estados como a luz em um local e, assim, monitorar uma chave elétrica para ligar a luz no momento em que o local permaneça escuro.

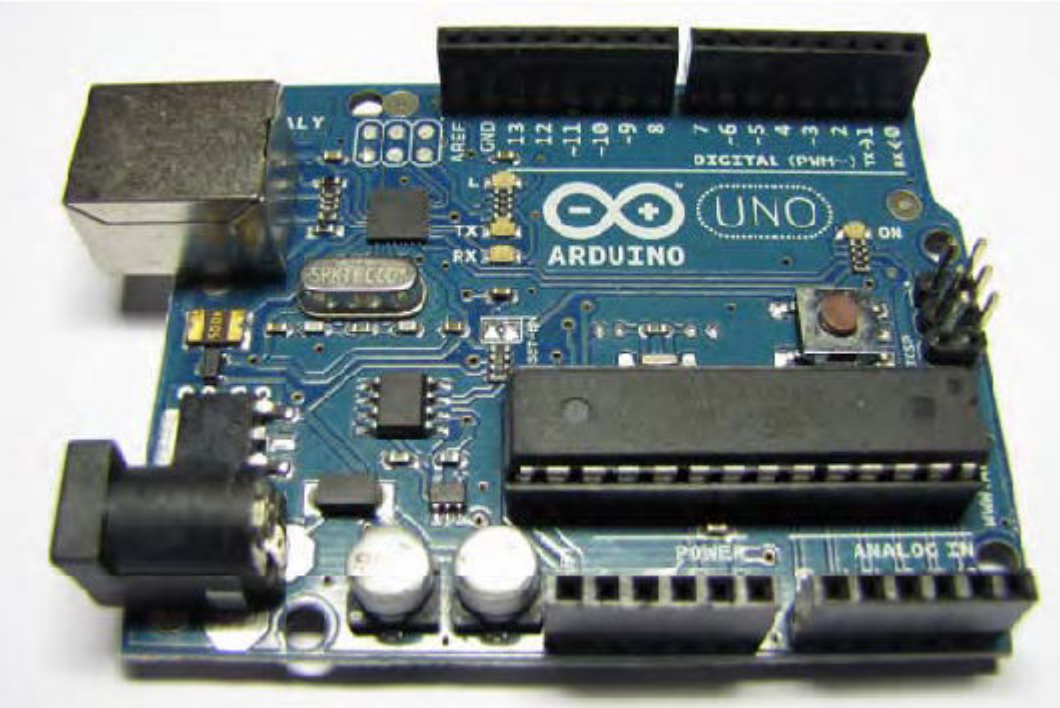

Figura 4 – Arduino Uno. (Fonte: Beginning Arduino, MCROBERTS, 2010 p. 3).

De acordo com Monk (2013), é uma plataforma de microcontrolador que despertou a criatividade dos estudiosos de eletrônica. Para qualquer um que pretenda fazer projetos eletrônicos é uma excelente escolha devido a sua simplicidade de emprego e seu conteúdo aberto.

O arduino adota uma linguagem denominada de Wiring na programação da IDE, fundamentada na linguagem C/ C++, na qual toda a metodologia da programação é elaborada, em seguida por meio da IDE do Arduino transformada em linguagem de máquina e logo após movida para o Atmega328.

Segundo Mcroberts (2010), a funcionalidade do Arduino pode ser aumentada com a utilização de dispositivos, por exemplo, receptores de GPS, monitores LCD, módulos Ethernet, etc) que se encaixam na placa arduino possiblitando novas experiências e auxiliando os projetistas em seus protótipos. Esses dispositivos são chamados de Shields.

3.1 Sensor Magnético Digital

Sensores são mecanismos que sofrem variação de estado de acordo com a relação com o local. Seu hardware pode ser composto por diversos componentes eletrônicos ou por apenas um componente.

Sensores de proximidade magnéticos se baseiam de acordo com Thomazini e Albuquerque (2008 p.46), “no uso de campos magnéticos e convertem esse campo em um sinal elétrico. Esses sensores podem ser eletrônicos e a ampola reed”.

Uma chave magnética tem a mesma semelhança de uma chave digital, porém seu funcionamento é através de um imã, é também chamada de “reed switch”. Refere-se a dois contatos de metal que em condição normal permanecem separados. Entretanto, na existência de um campo magnético, os contatos se encostam, sendo capazes de conduzir a corrente elétrica. As junções encontram-se no interior de um invólucro de vidro, sendo conservadas isoladas da corrosão atmosférica (Figura 5).

Figura 5 – Chave Magnética. (Fonte: **Maxwell Bohr:** Instrumentação Eletrônica, PATSKO, 2006 p.69).

Existem diversas utilidades disponíveis para o sensor magnético. Por exemplo, pode ser empregado para controlar produtos. Nessa situação, o imã fica regado a uma peça de modelo, enquanto o sensor fica fixado perto a ele para que possam comunicar-se.

O modelo mais convencional de utilização de chaves magnéticas é em mecanismos de alarme.

Esses objetos são instalados em batentes de portas e janelas. Um imã é instalado na porta de forma que no momento em que ela está fechada, a chave, também esteja. Entretanto, se a porta for aberta, o imã em consequência se distanciará da chave, que se abrirá. Se o alarme encontrar-se ligado, logo ele será disparado.

A chave magnética é um elemento respectivamente sensível, assim sendo são fundamentais determinadas precauções ao longo de sua aplicação, como ao contorcer suas extremidades, a fim de fixa-la numa placa de circuito impresso, deve ser conservado um determinado afastamento da sua cápsula, impedindo que a estrague.

A chave magnética pode ser utilizada com o mesmo circuito destinado às chaves digitais.

3.2 Sensor de Pressão

Constituído de um condutor ou semicondutor, que altera conforme a pressão empregada sobre ele a sua resistência. Portanto, de acordo com a pressão investida, a tensão elétrica da saída analógica sofre uma oscilação.

O sensor de peso, também é chamado de sensor de pressão ou de força, é outro sensor resistivo de simples utilização. Por intermédio desse elemento conseguimos mensurar o peso de determinada peça ou uma pressão concentrada sobre ela.

No meio industrial, o controle utilizando esse elemento na entrada de matéria-prima e saída do produto acabado, como em indústrias alimentícias, por exemplo, é muito importante. Permite conter gastos e efetuar certas técnicas de modo mais competente.

Ficou estabelecida a aplicação da Célula de Carga como transdutor do projeto. A célula de carga é um transdutor, que conforme a força que é acumulada contra seu corpo altera sua tensão elétrica. Na estrutura física da célula encontram-se os strains gauges, que se resumem em sensores de deformação bastante precisos, que alteram sua resistência conforme a pressão que acumulada sobre eles (Figura 6).

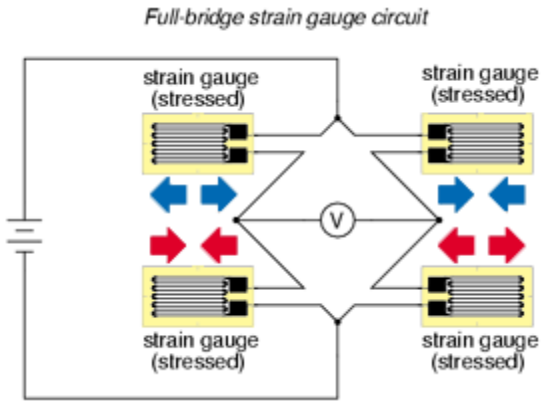

Figura 6 – Célula de Carga. (Fonte: **Célula de Carga:** Aplicação e Princípio de Funcionamento, VIEGAS, 2014, p.15).

Para Thomazini e Albuquerque (2008), estes instrumentos normalmente permanecem em ligação de ponte de wheatstone, assim, no momento que é investida uma tensão contra esta ponte, há uma alteração de pressão contra a estrutura da célula, alterando a resistência produzindo uma tensão de saída.

A célula de carga é bastante utilizada em balanças, e com isso, fomos à busca de uma para desmontar e retirar a célula de carga (Figura 7).

Figura 7 – Célula de Carga. (Fonte: **Célula de Carga:** Aplicação e Princípio de Funcionamento, VIEGAS, 2014, p.19).

A célula de carga possui quatro fios, dois para alimentação e dois para a sua saída. A oscilação de tensão correspondente ao peso que é aplicado sobre a célula é muito pequeno, na ordem de 0,7mV para uma força proporcional ao um peso de 10 kg.

Esse componente pode ser instalado num robô ou em determinado sistema eletrônico com uma finalidade muito importante, que é a de sensor de toque.

IV. PROJETO

Para o desenvolvimento do gerenciamento de controle de estoque usando o arduino, foi necessário um estudo prévio para definir a abrangência do sistema, suas funcionalidades, requisitos e o *hardware* necessário para que o software promova os resultados esperados.

Ao projetar o SIGMETAL a principal meta foi automatizar os diversos processos operacionais de uma empresa metalúrgica que eram feitos de forma manual, como por exemplo, o cadastro de produtos, clientes, fornecedores, e por fim o processo de estocagem.

Como linguagem de programação utilizou-se o Java, que é orientada a objetos, *open source,* multiplataforma, e consolidada no mercado por sua utilização em vários sistemas.

A funcionalidade do sistema consiste em agilizar os processos internos de uma empresa metalúrgica, dentre eles, cadastro e venda de produtos e principalmente o controle da estocagem, com isso utilizou-se o arduino acoplado ao sistema como experimento no controle

da pesagem da matéria prima na entrada, e na contagem do produto acabado na saída, por meio de um sensor ligado ao arduino, visando diminuir os gastos da automação do processo.

4.1 Diagrama de Classes

O diagrama de classes mostra um conjunto de classes e seus relacionamentos, o SIGMETAL é composto por 23 classes.

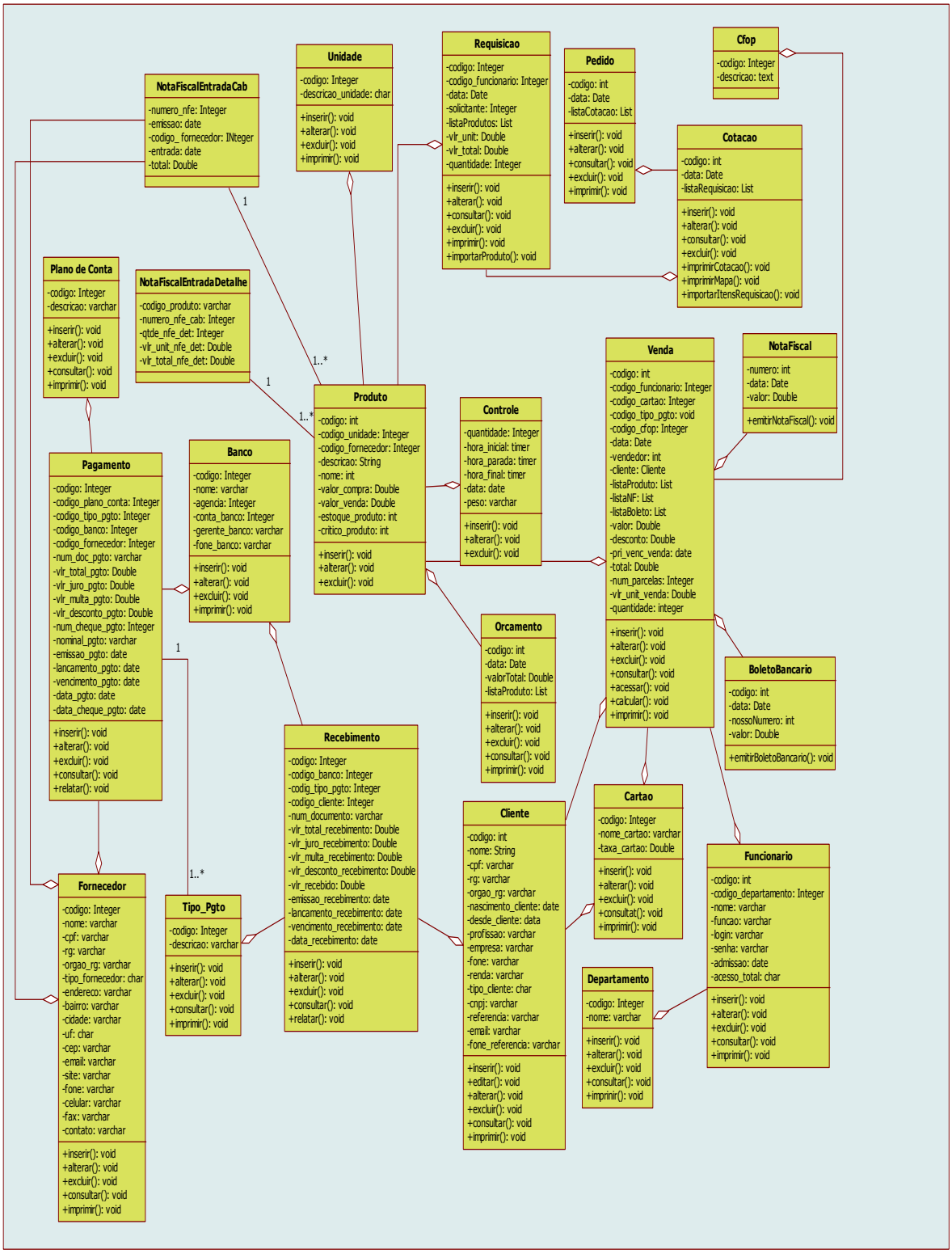

Figura 8 – **Diagrama de Classes** (Fonte: Próprio autor).

4.2 Caso de Uso

O Diagrama de Casos de Uso descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário, auxiliando a comunicação entre os analistas e o cliente.

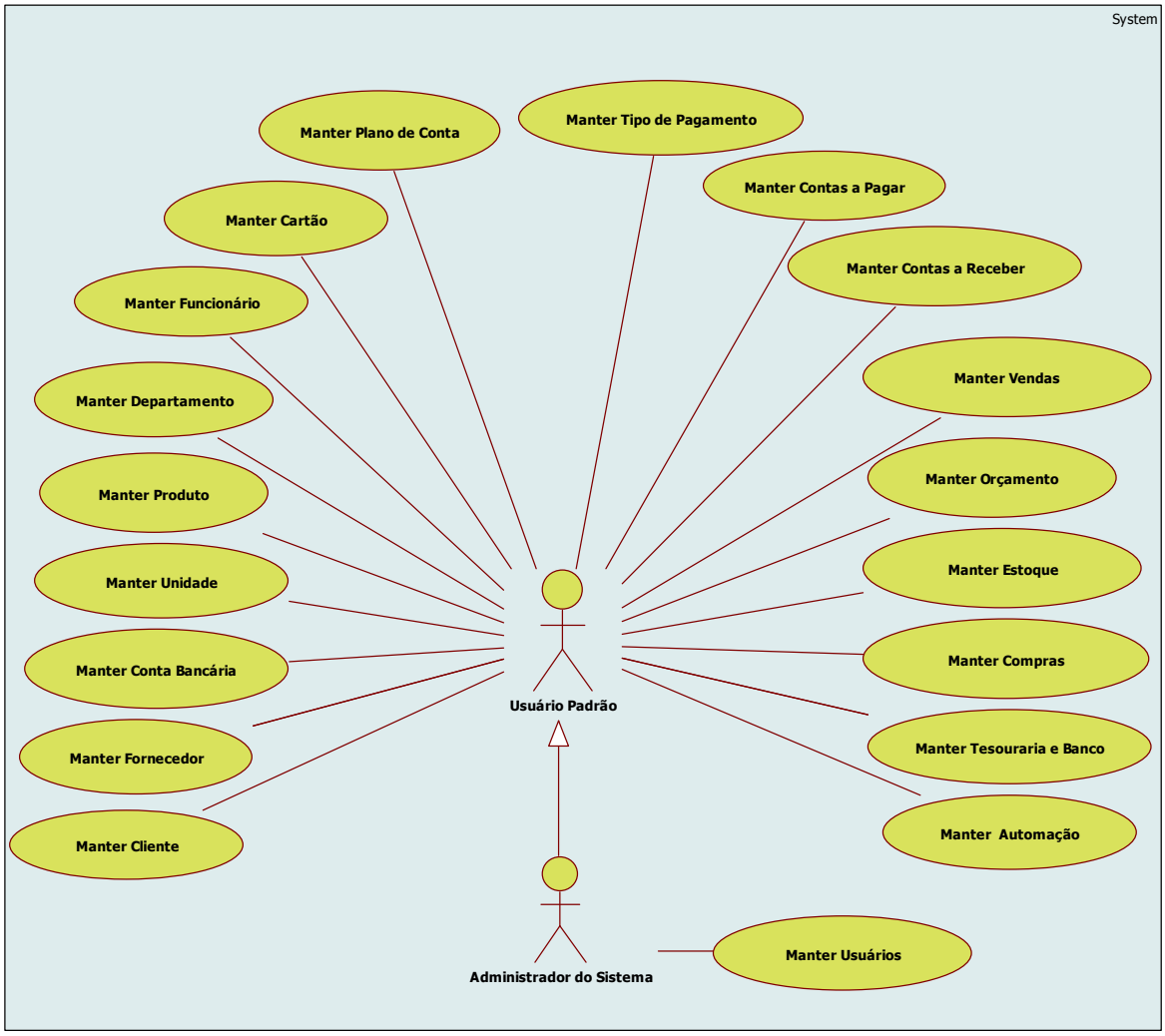

Figura 9 – **Diagrama de Caso de Uso** (Fonte: Próprio autor).

4.3 Diagrama de Sequência

Consiste em um diagrama que tem o objetivo de mostrar como as mensagens entre os objetos são trocadas no decorrer do *tempo* para a realização de uma operação.

A figura abaixo destaca o diagrama de entrada de estoque.

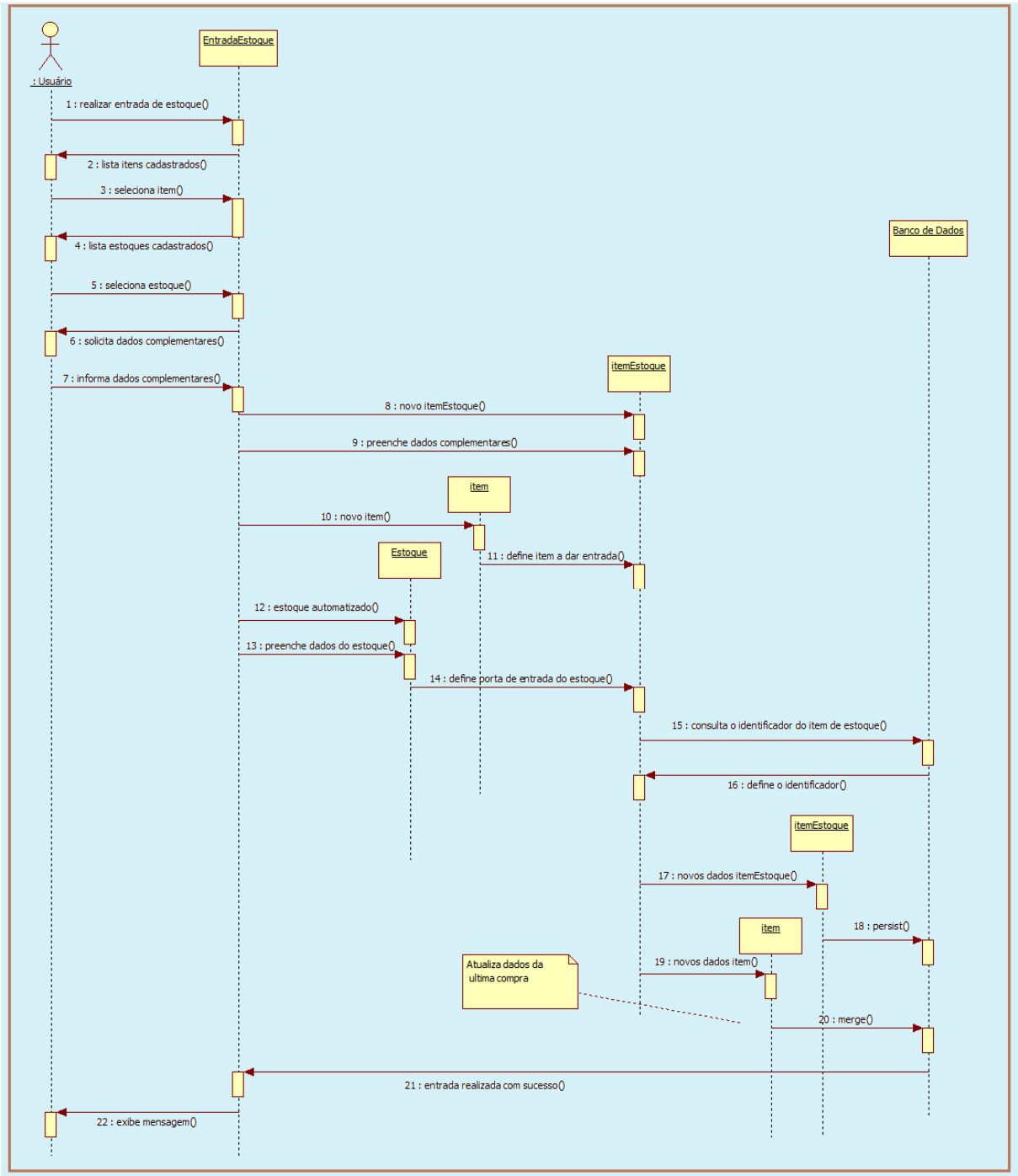

Figura 10 – **Diagrama de Sequência Entrada de Estoque** (Fonte: Próprio autor).

4.4 Diagrama de Atividades

É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra e serão empregados para fazer a modelagem de aspectos dinâmicos do sistema. A figura 11, demonstra o diagrama de atividades emitir pedido de compra.

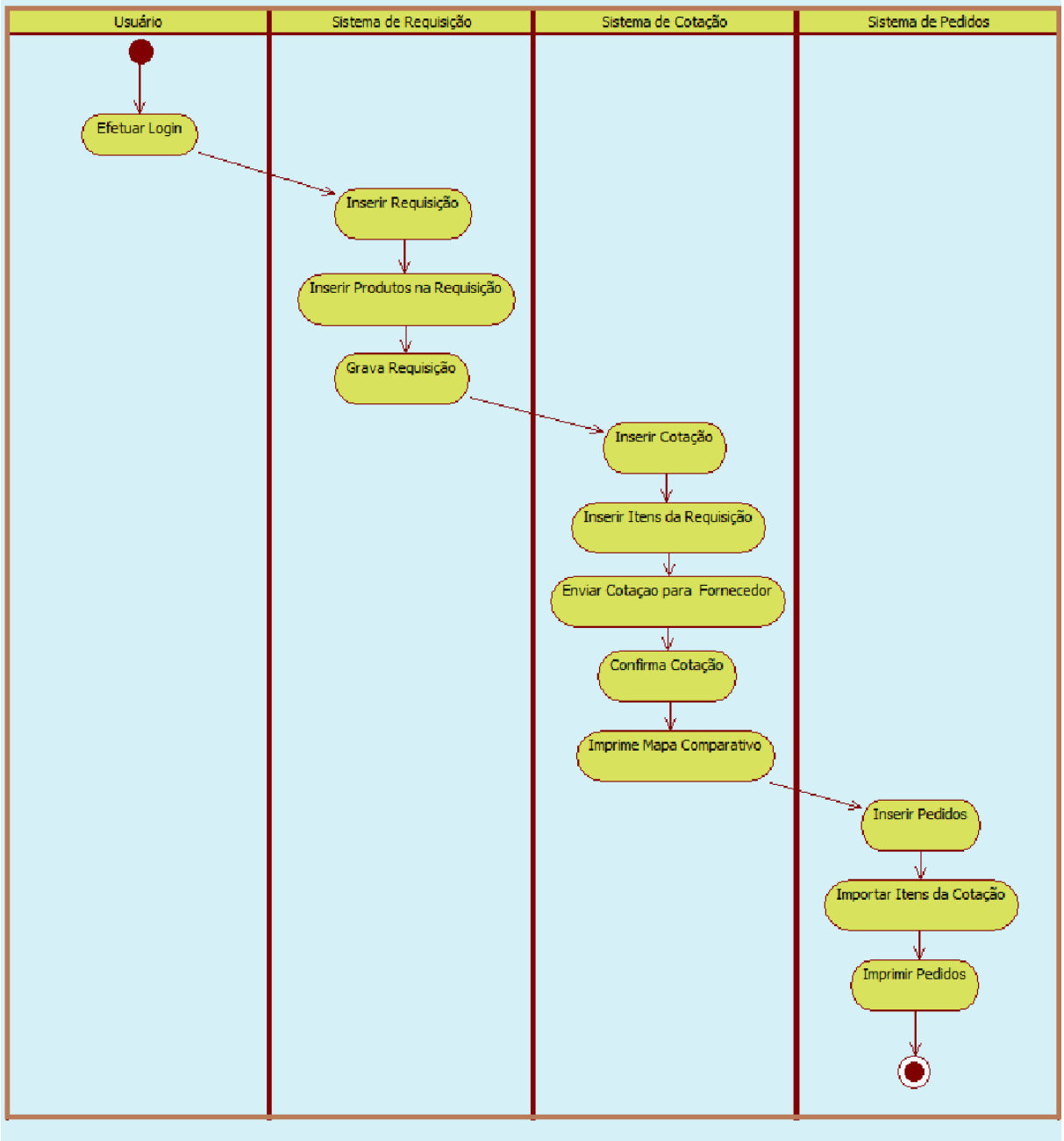

Figura 11 – **Diagrama de atividades Emitir Pedido de Compra** (Fonte: Próprio autor).

4.5 Interfaces do SIGMETAL

Logo abaixo serão apresentadas algumas das interfaces gráficas do sistema.

Figura 12 – **Tela Inicial do Sigmetal** (Fonte: Próprio autor).

A figura 12 apresenta a tela de inicial do sistema (menu), através dela o usuário tem a opção de escolher uma funcionalidade, como por exemplo, cadastro de clientes, produtos, vendas, fornecedores, dentre outros.

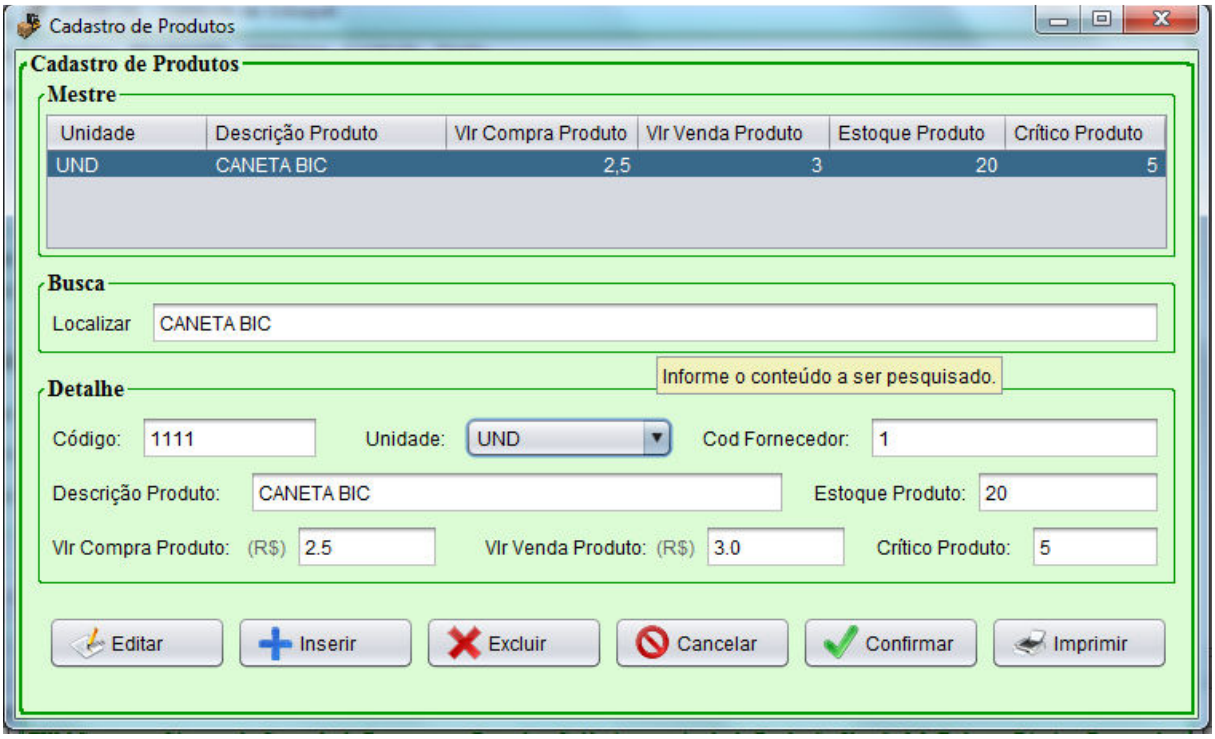

Figura 13 – **Tela Cadastro de Podutos** (Fonte: Próprio autor).

A figura 13 ilustra a tela de cadastro de produtos, com as seguintes opções: uma tabela mestre, que mostra os detalhes dos produtos cadastrados no banco de dados do sistema. Logo abaixo da mesma um campo de busca, onde irá fazer uma pesquisa, digitando o nome ou valor de qualquer um dos itens dispostos na referida tabela. Após selecionar o item no campo busca ele será destacado na cor azul e automaticamente apresentado seus dados nos campos de texto da opção de quadro detalhe, em seguida basta marcar o botão confirma e os dados serão gravados no banco de dados do Sigmetal.

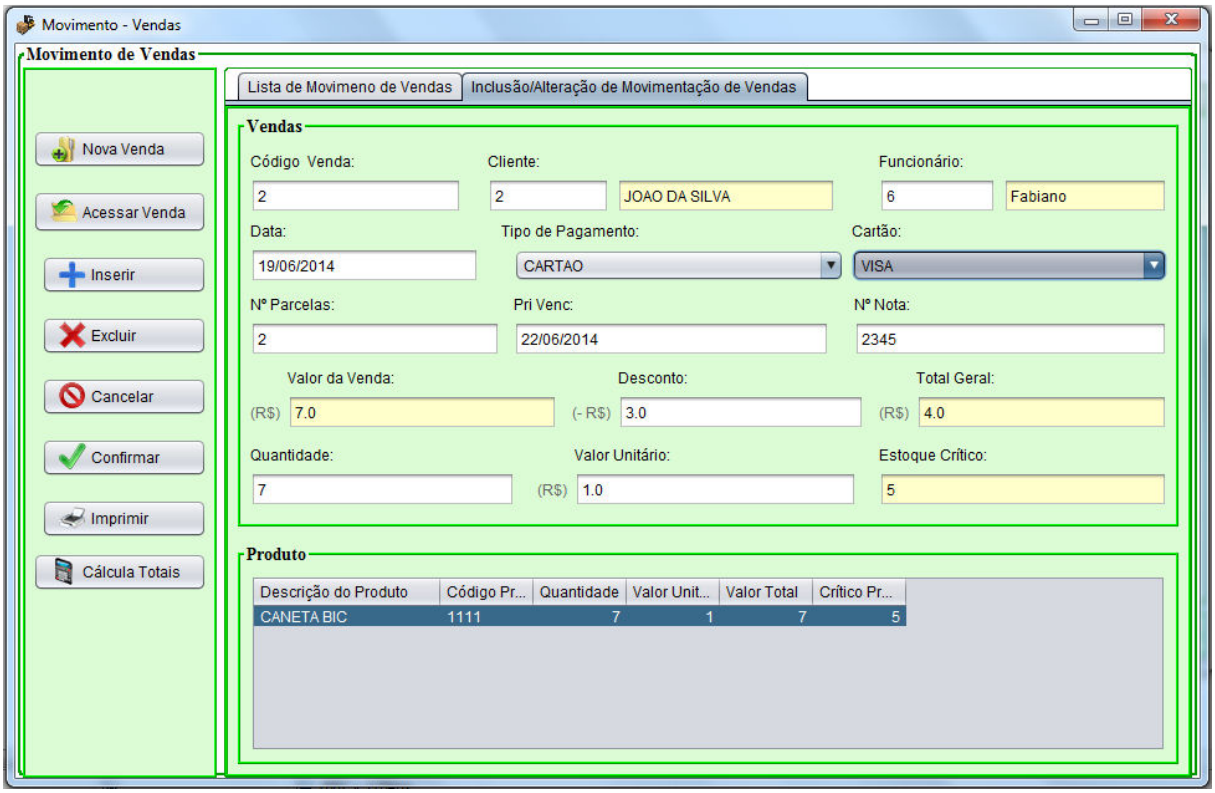

Figura 14 – **Tela Movimento de Vendas** (Fonte: Próprio autor).

A figura 14 mostra a tela movimento de vendas, onde é possível inserir os dados de uma nova venda, bem como acessar vendas anteriores. Além de várias outras funcionalidades, em destaque como a informação ao usuário da quantidade mínima em estoque e a não concretização da venda por falta de produto em estoque.

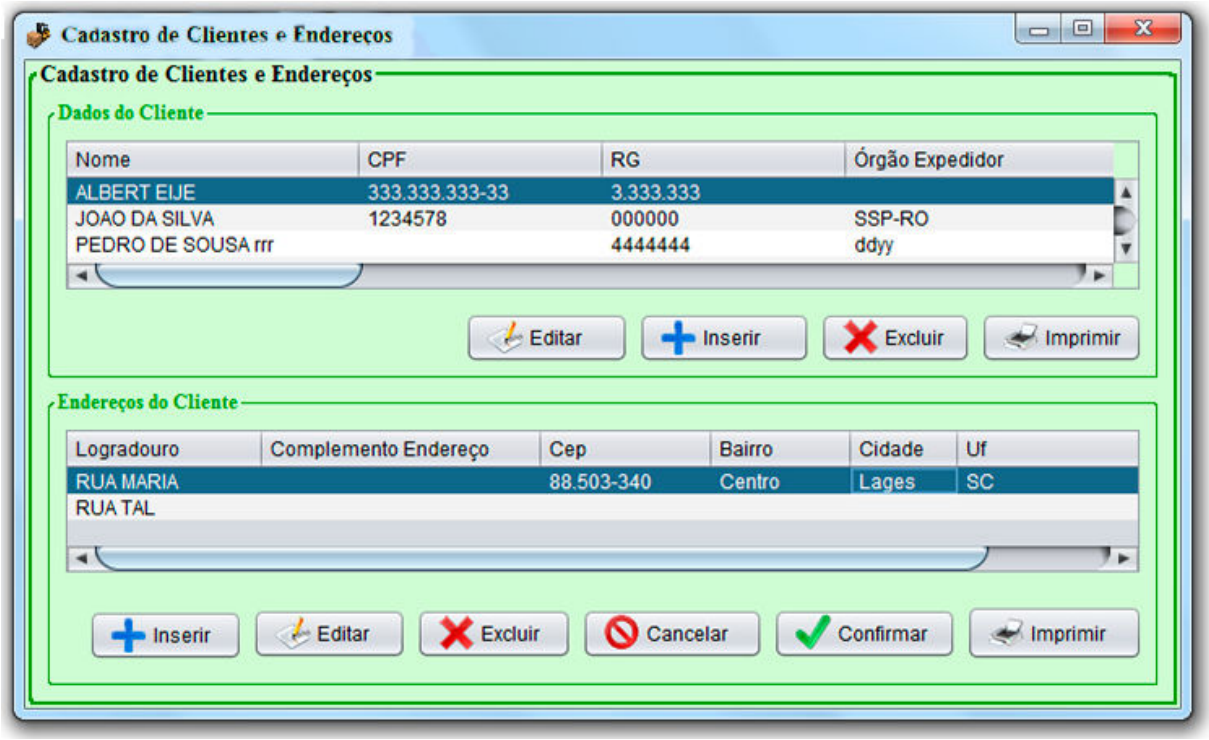

Figura 15– **Tela Cadastro de Clientes** (Fonte: Próprio autor).

A figura 15 destaca a tela de cadastro de clientes, nela é possível acessar e selecionar os quadros: dados e endereços do cliente, na mesma ao selecionar o botão inserir abre uma nova tela para cadastro de novos clientes ao banco de dados, conforme mostra a figura abaixo (figura 16).

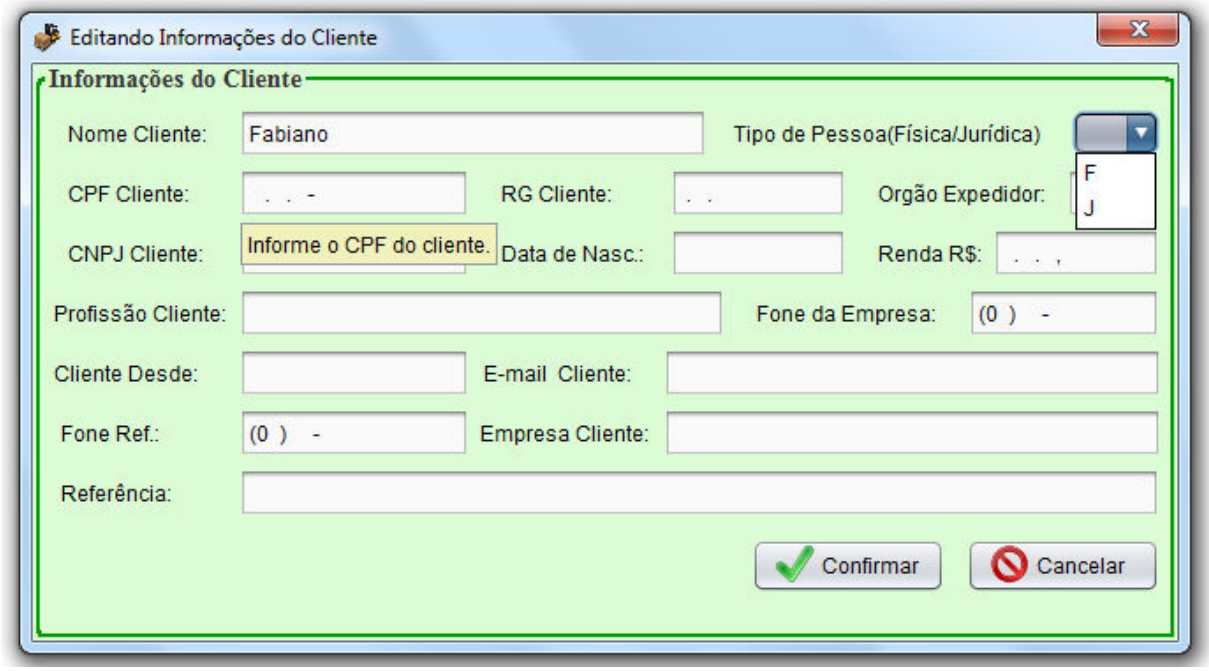

Figura 16– **Tela Editando Informações do Cliente** (Fonte: Próprio autor).

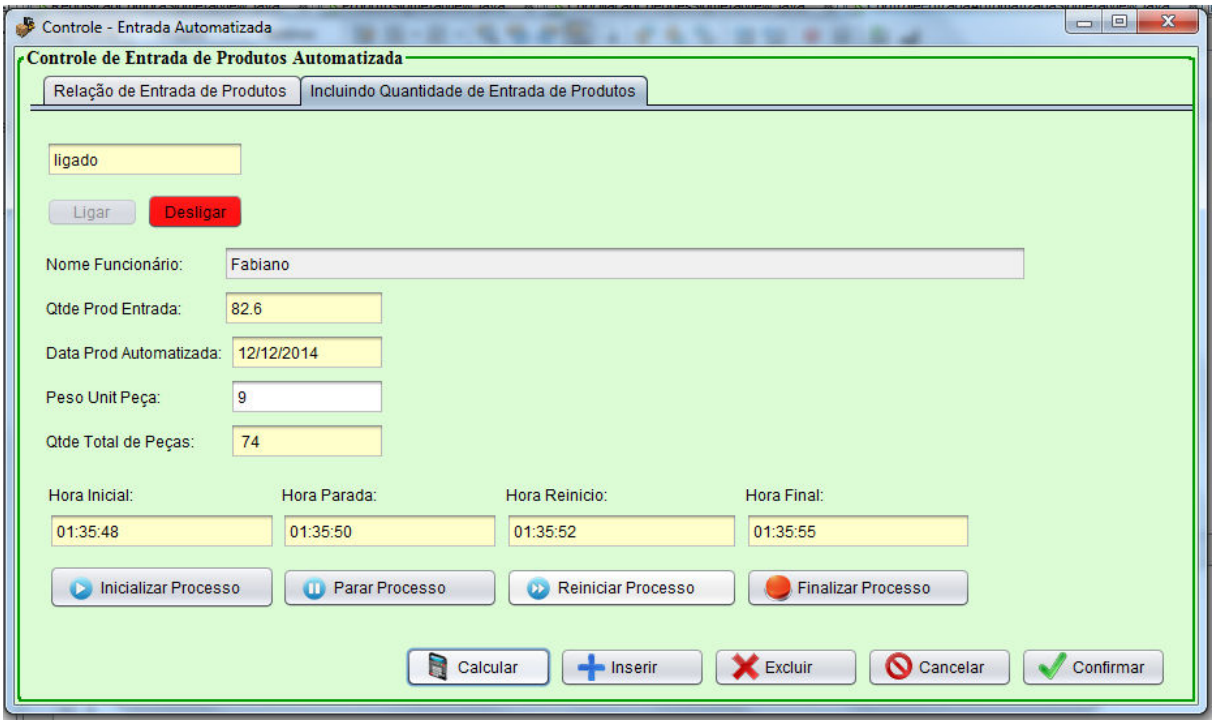

Figura 17– **Tela de Controle de Entrada de Produtos Automatizada** (Fonte: Próprio autor).

A figura 17 destaca a tela de controle de Entrada de Produtos Automatizada, nela é possível inserir, acessar e selecionar os cadastros referentes à entrada da matéria prima. O controle da entrada é feito através da pesagem da matéria prima de entrada, utilizando um sensor de pressão conectado ao arduino.

Logo abaixo a figura 18 mostra a tela de listagem de dados que também faz parte do módulo de controle de Entrada de Produtos Automatizada.

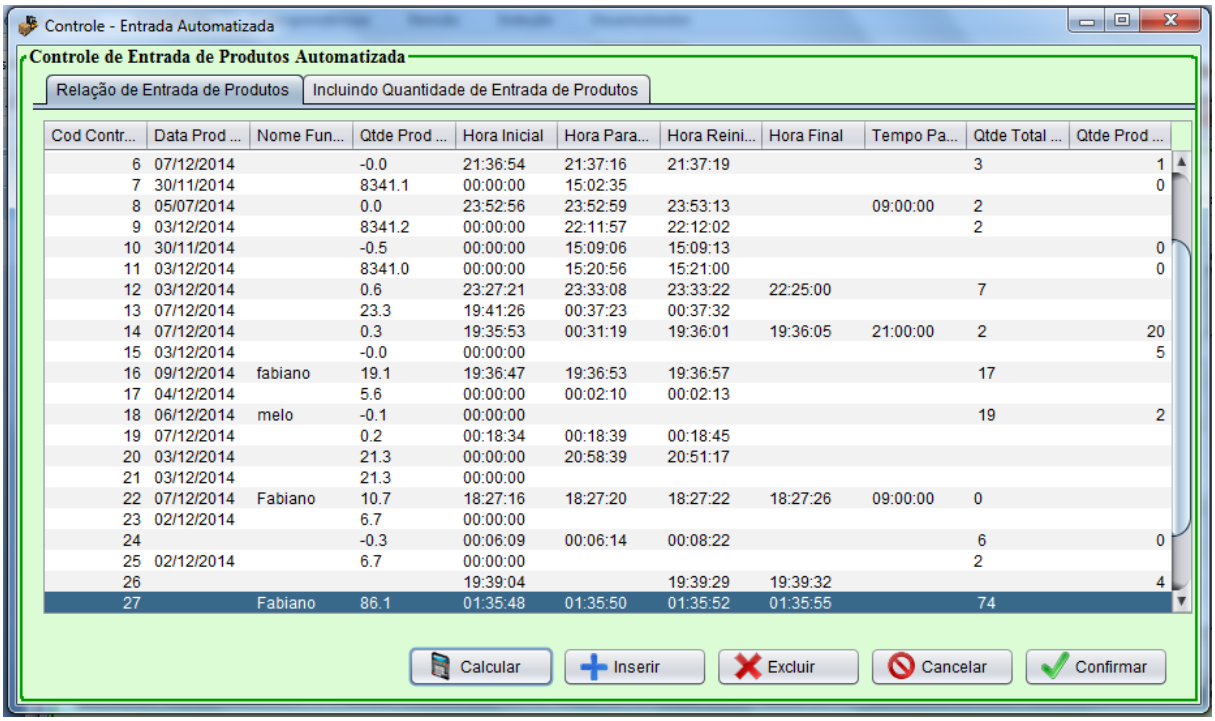

Figura 18– **Tela de Controle de Entrada de Produtos Automatizada – Relação de Entrada de Produtos (Fonte: Próprio autor).**

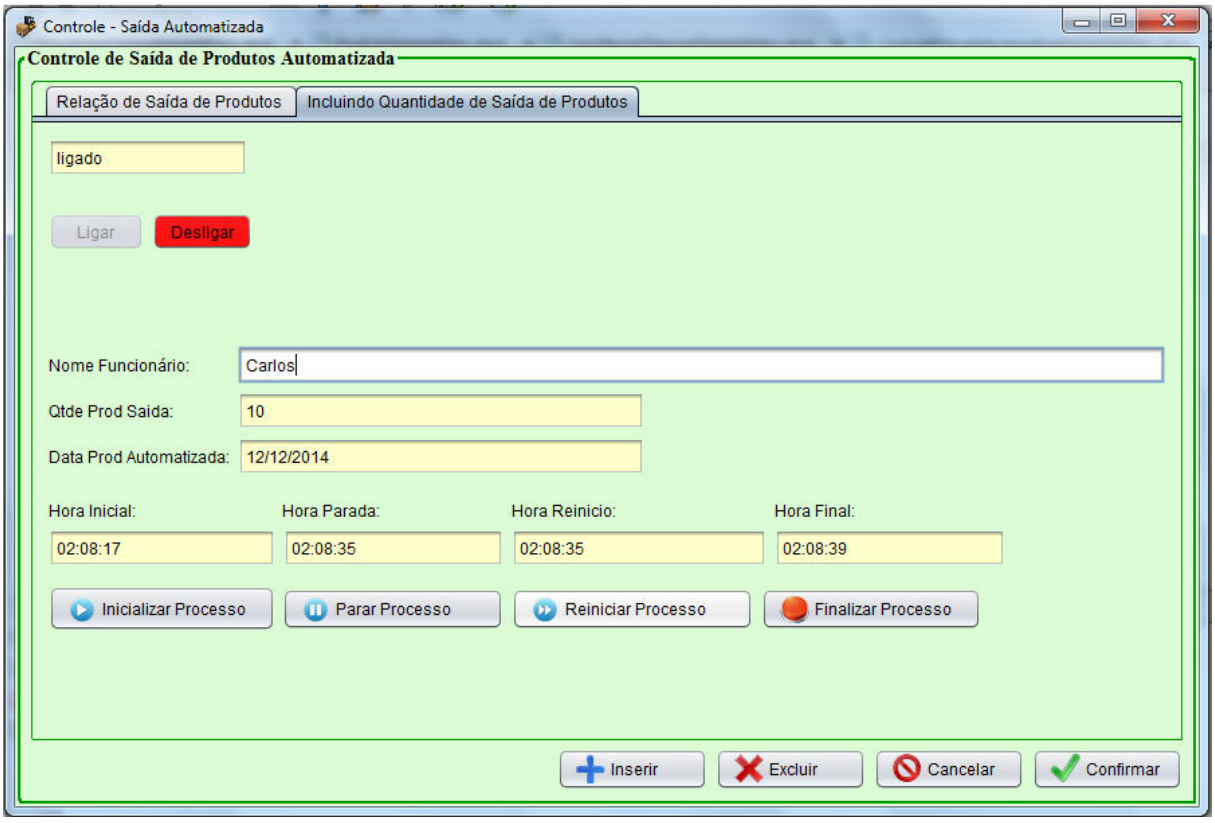

Figura 19– **Tela de Controle de Saída de Produtos Automatizada** (Fonte: Próprio autor).

A figura 19 destaca a tela de controle de Saída de Produtos Automatizada, nela é possível inserir, acessar e selecionar os dados referentes à saída de produtos já elaborados. O controle da saída é feito através da contagem do produto final, utilizando um sensor magnético conectado ao arduino, com a finalidade de detectar a passagem do produto e enviar ao sistema.

O sensor magnético usado neste sistema vai fixado a uma esteira rolante.

Logo abaixo a figura 20 mostra a tela de listagem de dados que também faz parte do módulo de controle de Saída de Produtos Automatizada.

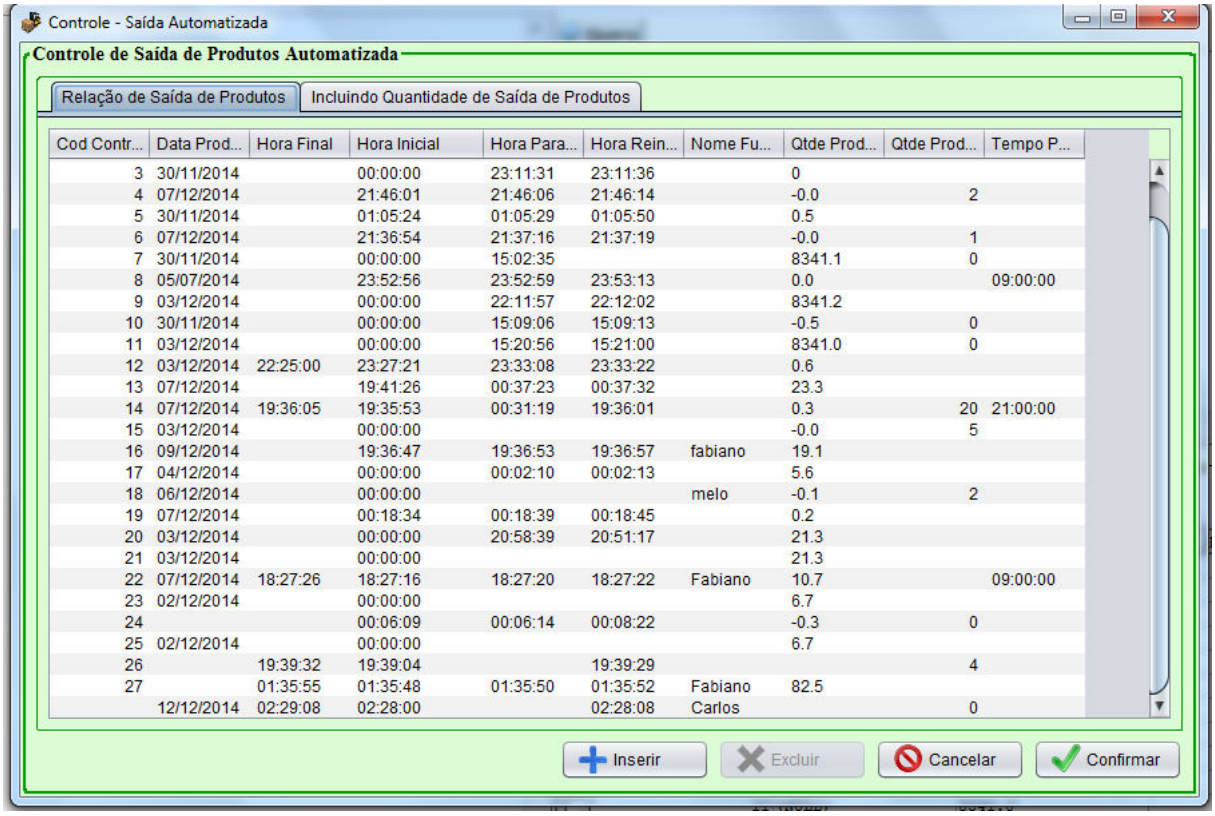

Figura 20– **Tela de Controle de Saída de Produtos Automatizada – Relação de Saída de Produtos (Fonte: Próprio autor).**

V. CONSIDERAÇÕES FINAIS

Conforme os objetivos estabelecidos, conclui-se que o trabalho possibilitou o desenvolvimento de um sistema de gerenciamento de estoque de matéria prima para metalurgias usando arduino, no controle de entrada de matérias primas e saída de produtos acabados. A fim de agilizar os processos de acesso ao estoque da empresa e a venda de produtos, os quais antes eram realizados manualmente, através de anotações, utilizando formulários pré – impressos, ou no computador através de planilhas eletrônicas e arquivos de texto.

O sistema automatizado desenvolvido ficou mais seguro, com acesso facilitado às informações, além de agilizar as atividades de tomada de decisão realizadas pelo administrador.

Através do desenvolvimento de rotinas de cadastros de clientes, fornecedores, produtos, emissão de ordem de compra de matérias primas e venda de produtos, dentre outras funcionalidades do sistema, obtêm-se um controle sobre estas operações, permitindo assim gerenciá-las. Pode-se também realizar consultas e emitir relatórios básicos, permitindo acesso às informações desejadas.

Com o desenvolvimento do trabalho ampliou-se o conhecimento sobre novas ferramentas e aplicativos, dentre eles o arduino que tem a função de automatizar o processo de entrada da matéria-prima e saída do produto final, que auxiliam no desenvolvimento do sistema. Em particular, a falta de experiência com programação e a falta de domínio em uma linguagem específica, acarretou em dificuldades que foram superadas através de pesquisas, leituras, conhecimentos adquiridos no curso e auxílio de recursos eletrônicos, permitindo assim a conclusão deste.

VI. REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIO, Erik Aceiro ; FERRO, Milene. Análise Comparativa Entre os Principais Frameworks de Desenvolvimento JAVA. Disponível em: <http: // wright. ava. ufsc. br/~ alice /conahpa/anais/2009/cd_conahpa2009/papers/final139.pdf>. Acesso em: 21. maio. 2014.

BALLOU, Ronald H. **Logística empresarial**: transporte, administração de materiais e distribuição física. São Paulo: Atlas, 1993. 352p.

BALLOU, Ronald H. Gerenciamento da Cadeia de Suprimentos/Logística Empresarial. 5. ed. Porto Alegre: Bookman, 2006.410 p.

BATISTA, Emerson de Oliveira. **Sistema de Informação:** o uso consciente da tecnologia para o gerenciamento. São Paulo: Saraiva, 2004. 282 p.

BAUER, Christian; KING, Gavin. **Hibernate Inaction.** São Paulo: Manning Publications, 2004. 408 p.

BERTAGLIA, Paulo R.. **Logística e gerenciamento da cadeia de abastecimento**. São Paulo: Saraiva, 2003. 579 p.

BONAN, Adilson Rodrigues. **Java:** dicas e truques. Rio de Janeiro: Alta Books, 2008. 565 p.

CAMARA, Fábio. **Orientação a Objeto Com.net.** 2.ed.São Paulo:Visual Books, 2006.124 p.

CORDEIRO, Gilliard. **Aplicação Java para web com JSF e JPA.** São Paulo: Casa do Código, 2013. 270 p.

CORRÊA, Henrique L., GIANESI, Irineu G. N. CAON, Mauro. **Planejamento, programação e controle da produção.** 2. ed. São Paulo: Editora Atlas S.A., 1998. 411 p.

COSTA, Daniel G.. **Java:** dicas e truques. Rio de Janeiro: Brasport, 2009. 340 p.

CURY, Antonio. **Organização e métodos:** uma visão holística. São Paulo: Atlas S.a., 2000. 589 p.

D.CALLISTER JUNIOR, William; RETHWISCH, David G.. **Ciência e Engenharia de Materiais uma Introdução.** 8. ed. São Paulo: Ltc (grupo Gen), 2012. 844 p.

DEITEL, H. M.; DEITEL, P. J. **Java**: Como Programar. 3. ed. Porto Alegre: Bookman, 2001. 1201 p.

DEITEL, Harvey M.; DEITEL, Paul J. **Java:** Como Programar. 8. ed. São Paulo: Pearson, 2010. 1114 p.

DEMICHIEL, L.; KEITH, M. (Lideres). **JSR 220: Enterprise JavaBeans, version 3.0** – Java Persistence API. Final Release. California, Estados Unidos da America: [s.n.], 2006. Disponivel em: <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>. Acesso em: 24 mai. 2014.

DIAS, Marco Aurélio P. Administração de materiais: princípios, conceitos e gestão. 5. ed. São Paulo: Atlas, 2005.182p.

DIETER, George E. **Metalúrgica Mecânica.** 2. ed. Rio de Janeiro: Guanabara Dois, 1981.653p.

FERRARI, Fabrício Augusto. **Criando Banco de Dados com MySQL.** São Paulo: Digerati, 2006. 128 p.

FLEURY, Paulo F.; WANKE, Peter; FIGUEIREDO, Kebler F.. **Logística empresarial: a perspectiva brasileira.** São Paulo: Atlas, 2000. 288 p.

FRANCISCHINI, Paulino. **Administração de materiais e do patrimônio**. São Paulo: Pioneira, 2002. 310 p.

GARCIA, E. S. et al.; **Gestão de Estoques: Otimizando a Logística e a Cadeia de Suprimentos**. Rio de Janeiro: E-Papers Serviços Editoriais, 2006. 144p.

GUEDES, Gilleanes T.A. **Uml 2 - Uma Abordagem Prática.** 2. ed. São Paulo: Novatec, 2011. 488 p.

GORDON, S. R.; GORDON, J. R. **Sistemas de informação:** uma abordagem gerencial. Rio de Janeiro: LTC, 2006. 377 p.

GOETTEN Junior, Vicente; WINCK, Diogo Vinícius. **AspectJ:** Programação Orientada a Aspectos com Java. São Paulo: Novatec, 2006. 228 p.

JOBSTRAIBIZER, Flávia. **Guia profissional PHP.** São Paulo: Digerati Books, 2009. 112 p.

LAUDON, C.K.; LAUDON, P.J. **Sistemas de informação gerenciais:** administrando a empresa digital**.** 5.ed. São Paulo: Prentice Hall, 2004. 562p.

MARTINS, Petrônio Garcia. **Administração de materiais e recursos patrimoniais**. São Paulo, Saraiva, 2001. 353 p.

MCROBERTS, Michael. **Beginning Arduino.** New Work: Apress, 2010. 472 p.

MELO, I. S. **Administração de sistemas de informação**. 1. ed. São Paulo: Pioneira, 2006. 178p.

MONK, Simon. **Programação com Arduino.** São Paulo: Bookman, 2013. 160 p.

MORAES, Cícero Couto de; CASTRUCCI, Plínio de Lauro. **Engenharia de automação industrial**. 2. ed. Rio de Janeiro, LTC, 2007.347 p.

O'BRIEN, James A.; MARAKAS, George M.. **Administração de Sistemas de Informação.** 15. ed. São Paulo: Tecbooks, 2013. 620 p.

OLIVEIRA, Djalma de Pinho Rebouças de. **Sistemas de informações gerenciais: estratégicas táticas operacionais**. 6. ed. São Paulo: Atlas, 2004. 285p.

POZO, Hamilton. **Administração de recursos materiais e patrimoniais:** uma abordagem logística. 3. ed. São Paulo: Atlas, 2004. 204p.

PRESSMAN, Roger S.. **Engenharia de Software:** Uma Abordagem Profissional. 7. ed. São Paulo: Bookman, 2011. 780 p.

REZENDE, Denis Alcides. **Planejamento de Sistemas de Informação e Informática:** guia prático para planejar a tecnologia da informação integrada ao planejamento estratégico das organizações. 4. ed. São Paulo: Atlas, 2011. 168 p.

ROSÁRIO, João Maurício. **Automação Industrial.** São Paulo: Baraúna, 2009. 515 p.

SAMPAIO, Cleuton. **Guia do Java: Enterprise Edition5: desenvolvendo aplicações corporativas**. Rio de Janeiro: Brasport, 2007. 200 p.

SERSON, Roberto Rubinstein. **Programação Orientada a Objetos com Java 6:** Curso universitário. São Paulo: Brasport, 2008. 492 p.

SILVEIRA, Paulo Rogério da; SANTOS, Winderson E. dos. **Automação e Controle Discreto.** 7. ed. São Paulo: Érica, 2009. 256 p.

SLACK, Nigel *et al*. **Administração da Produção**. 3. ed. São Paulo: Atlas,1999. 747 p.

SOMMEVILLE, Ian. **Engenharia de *software***. 8.ed. São Paulo: Addison Wesley,2007.568 p.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga de. **Sensores Industriais:** Fundamentos e Aplicações. 6. ed. São Paulo: Érica, 2008. 224 p.

VASCONCELOS, Antônio Galvão. **Estoques elevados podem fechar portas.** Artigo publicado em 2003, no Diário Catarinense**.** Disponível em: <http: //www.varejista. com.br/novo_site/desc_materia.asp?id=20490>. Acesso em: 02. maio. 2014.

VIANA, João José. **Administração de Materiais:** Um Enfoque Prático. 1. ed. São Paulo: Atlas, 2002. 443 p.

VIEGAS, Hygor. **Célula de Carga:** Aplicação e Princípio de Funcionamento -. 2014. Disponível em: <http://pt.scribd.com/doc/218503634/Celula-de-Carga-Aplicacao-e-Principio-de-Funcionamento-www-ctai-com-br>. Acesso em: 15 abr. 2014.

VII. ANEXOS

A figura 21 ilustra o diagrama de caso de uso compras numa visão mais detalhada do diagrama geral.

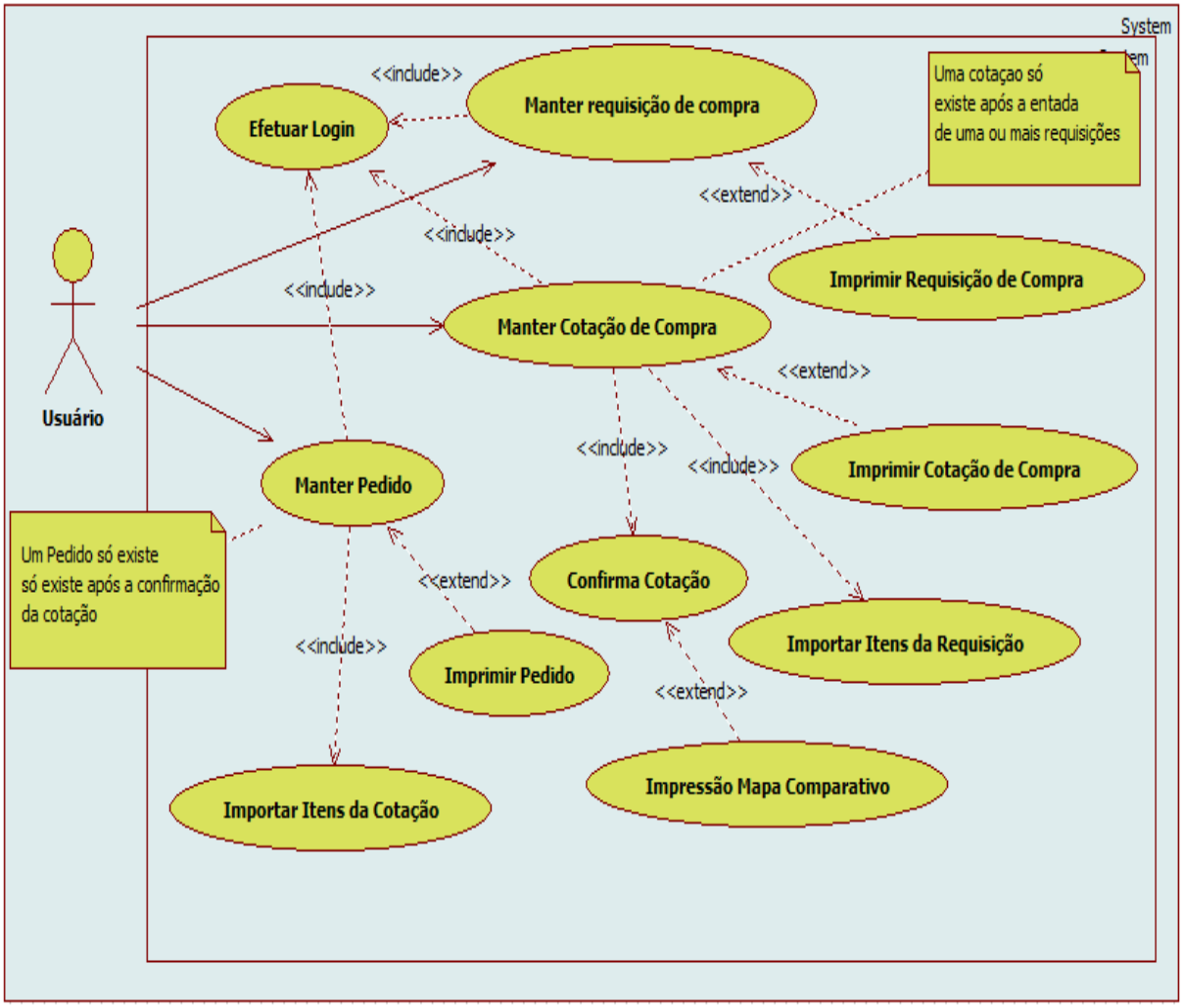

Figura 21 – **Diagrama de Caso de Uso Compras** (Fonte: Próprio autor).

A figura 22 mostra o diagrama de sequência manter usuário, numa visão parcial do diagrama completo.

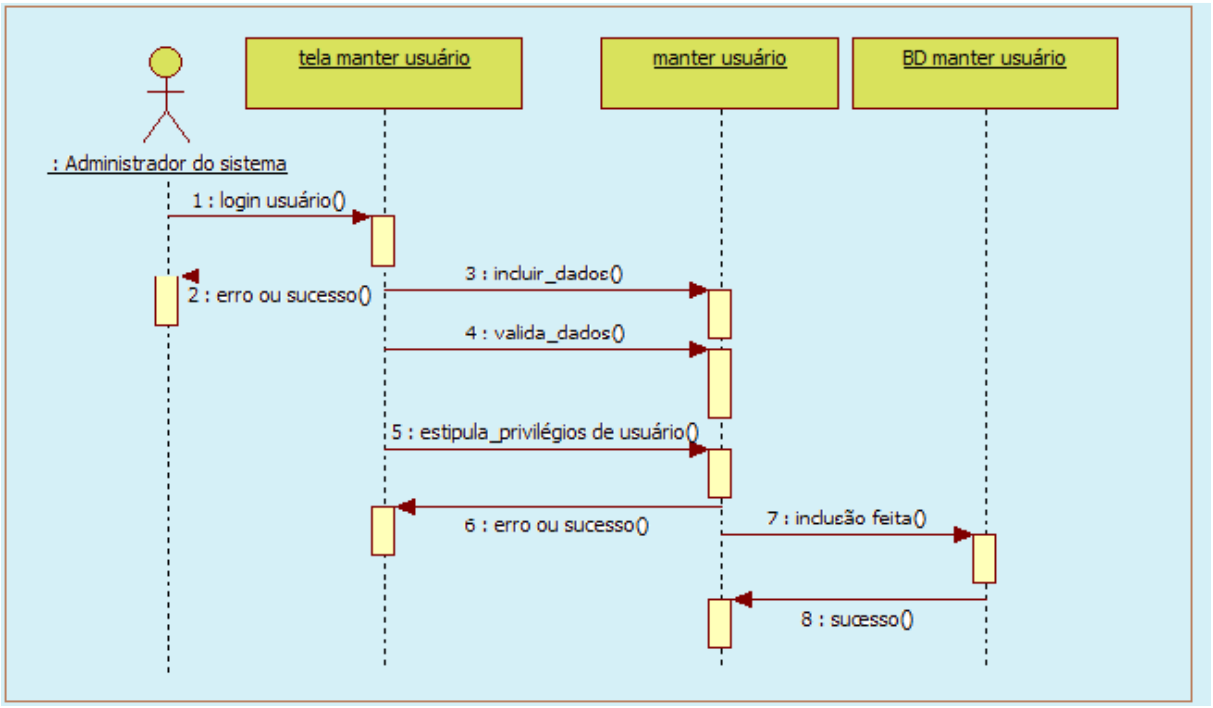

Figura 22 – **Diagrama de sequência manter usuário** (Fonte: Próprio autor).

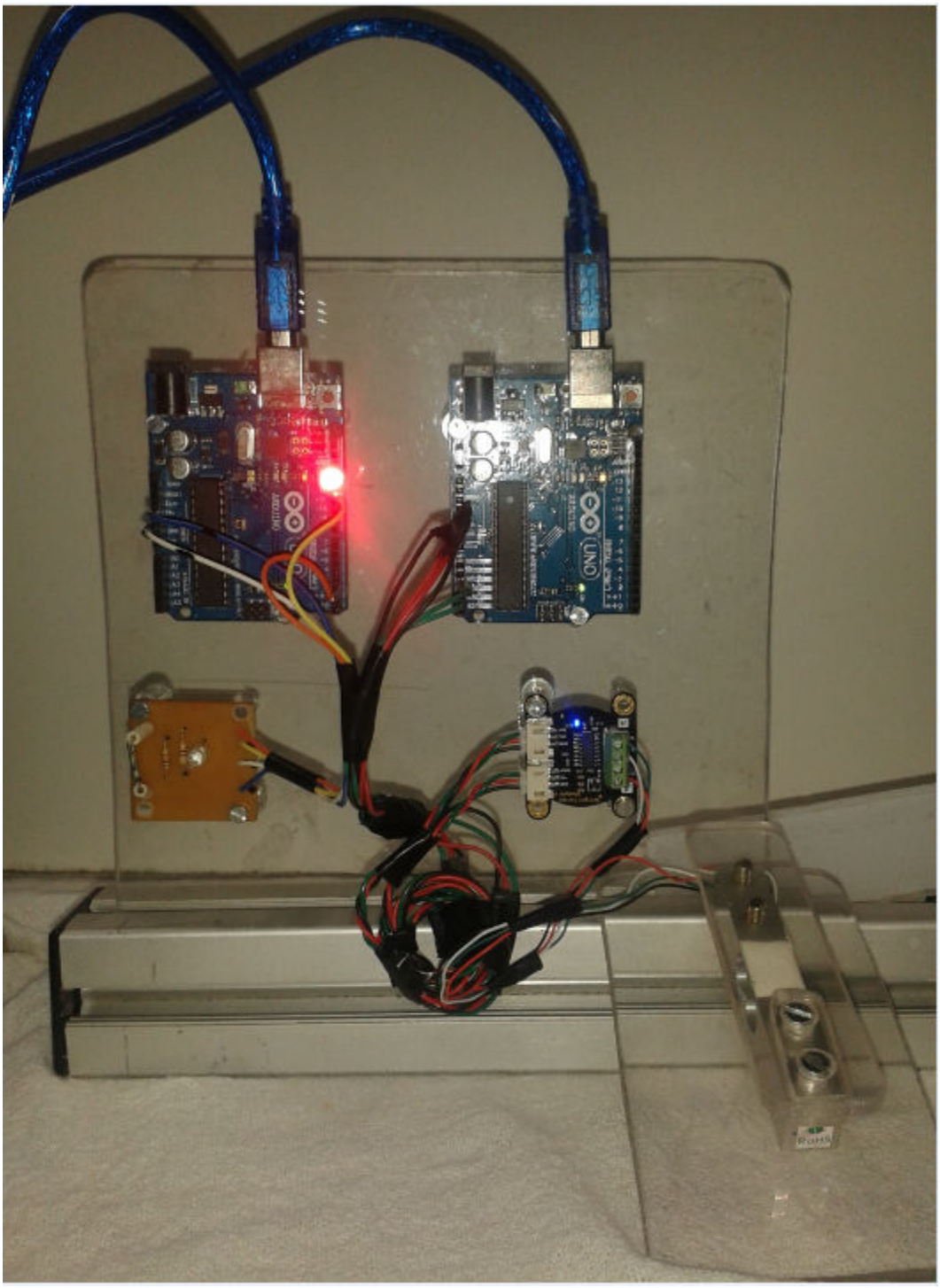

Figura 23 – **Foto do Hardware contendo Arduinos e Sensores** (Fonte: Próprio autor).

A figura 23 mostra a montagem da parte do hardware, que é interligada ao software. Sendo a mesma composta por dois arduinos uno, e dois sensores, um sensor de pressão e o outro sensor magnético.

Código fonte do sensor magnético utilizado no arduino

Sensor_Magnetico

```
int sm = 2; //variável do tipo inteiro; sm = 2 pino recebe entrada de corrente da chave magnética.
int ledPin = 13; // Led que indica que o circuito esta ligado.
int ledPin1 = 12; // int variável do tipo inteiro; ledPinl = 13 pino do Led que indica a contagem com a aproximação.
int numero = 0; // variável de inicialização das condições de estado do sistema(ligado, desligado, parado, reiniciado).
int smCounter = 0; // variável para armazenar o valor de leitura, para cada vez que o botão é apertado.


//Função setup, executado uma vez ao ligar o Arduino.
void setup(){

 pinMode(ledPin,OUTPUT); //declaração da porta digital de saída.
 pinMode(ledPin1,OUTPUT); // declaração da porta digital do pino de saída do sensor magnético.
 pinMode(sm, OUTPUT); // declaração da porta digital de saída do sensor magnético.
 pinMode(smCounter,OUTPUT); //declaração da porta digital de saída do contador do sensor magnético .
 Serial.begin (9600); //Ativando o serial monitor que exibir os valores lidos no sensor.

}


//Função loop, executado enquanto o Arduino estiver ligado.
void loop( ){

 if (Serial.available ( ) > 0){ //condição que compara e retorna a quantidades de bytes disponíveis para leitura no buffer de leitura.
                  // A leitura dos dados só e realizada quando há dados disponíveis.
    char numero = Serial.read ( ); //Lê o byte mais recente apontado no buffer de entrada da serial.

     if (numero > 0){
      if (numero == '1'){ // condição que indica led ligado e inicio do contador.

        digitalWrite (ledPin , HIGH); // ascende a luz (envia corrente)
         Serial.println ( "Ligado" ); // mensagem que será impressa na porta serial quando digitado o número 1.
          Serial.println (smCounter = 0); // mostra na serial contador igual a zero.

      }
```

```
    else if (numero == '0'){ // condição que indica led desligado e finaliza o contador.

                digitalWrite (ledPin, LOW); //  apaga a luz (e não envia corrente)
                Serial.println (smCounter = 0); // Mostra na Serial
                digitalWrite (sm, 0);

              Serial.println ( "Desligado" ); // mensagem que será impressa na porta serial
                quando digitado o número 0.
                   delay (20000); // intervalo de tempo de espera em segundos

                }
                 if (numero == '5'){ // condição que indica circuito parado.

                digitalWrite (sm, 0); // define o contador.

                    Serial.println ( "Parado" ); // mensagem que será impressa na porta
                        serial quando digitado o número 5.

                    delay (20000); // intervalo de tempo de espera em segundos
                }
    else if (numero == '8'){ // condição que indica circuito reinicializado.


                    Serial.println("Reiniciado"); // mensagem que será impressa na porta serial
quando digitado o número 8.
                    Serial.println(smCounter = smCounter); // Mostra na Serial o valor de
contagem.

                   Delay (500); // intervalo de tempo de espera em segundos
                }
     }

  }

   if (digitalRead (sm)){ //ler o pino do sensor magnético.


     smCounter++; // Incrementa o contador

    digitalWrite (ledPin1, HIGH); // envia corrente para a chave magnética.
          delay (500); // intervalo de tempo de espera em segundos

          digitalWrite (ledPin1, LOW); // não envia corrente para a chave magnética.
          delay (500); // intervalo de tempo de espera em segundos
          Serial.println (smCounter, DEC); // Mostra na Serial

   }

}
```

Código fonte do sensor de pressão utilizado no arduino

```
#include <Hx711.h> // biblioteca Hx711
Hx711 scale(A2, A3); // escala da classe Hx711  referente aos pinos A2, A3do aduino.

void setup() { //Função setup, executado uma vez ao ligar o Arduino.
  Serial.begin(9600); //Ativando o serial monitor que exibir os valores lidos no sensor.
}
//Função loop, executado enquanto o Arduino estiver ligado.
void loop() {
  Serial.print(scale.getGram(),  1);  //mostra  na  serial  o  valor  pego  através  do  método  get
                                      pegando os dados da classe Hx711.

  Serial.println ( " g " ); // mostra na serial a mensagem g.
  delay(200); // intervalo de tempo de espera em segundos
}
```

Classe Hx711 (biblioteca)

```
#ifndef HX711_H_ // chamada inicial da biblioteca HX711_H.
#define HX711_H_ // chamada final da biblioteca HX711_H
#include "Arduino.h" // chamada da biblioteca "Arduino.h"

class Hx711
{
public: // modificador public deixará visível a classe ou membro para todas as outras classes,
        subclasses e pacotes do projeto Java.
        Hx711(uint8_t pin_din, uint8_t pin_slk); // variáveis  Hx711
        virtual ~Hx711( ); //verifica se Hx711está pronto
        long getValue( ); // pega o valor e armazena no formato long
        long averageValue(byte times = 25); // Retorna uma leitura média; tempo  =  quantidade
                                              de tempo para leitura
        void setOffset(long offset); // usado para a tara
```

```
        void setScale(float scale = 1992.f); // calibra o sensor de pressão
        float getGram( ); // pega o valor float da grama atual


private:
        const uint8_t _pin_dout; //saída de dados  serial Pin
        const uint8_t _pin_slk; // entrada de dados serial Pin
        long _offset; // usado para a tara
        float _scale; // Usado para retornar o peso em gramas, kg
};
#endif /* HX711_H_ */
```

Código fonte da biblioteca Hx711.h para sensor de pressão utilizado no arduino

```
#include "Hx711.h" // biblioteca Hx711.h
 // variáveis  Hx711
Hx711::Hx711(uint8_t pin_dout, uint8_t pin_slk) :
                _pin_dout(pin_dout), _pin_slk(pin_slk)
{
        pinMode(_pin_slk, OUTPUT);  // define o pino digital como saída
        pinMode(_pin_dout, INPUT);  // define o pino digital como entrada

        digitalWrite(_pin_slk, HIGH); //define pino slk ligado
        delayMicroseconds(100); // faz pausa de 100 microsegundos
        digitalWrite(_pin_slk, LOW); //define pino slk desligado

        averageValue( ); // media do valor
        this->setOffset(averageValue());
        this->setScale(); // mostra escala
}

Hx711::~Hx711()
{

}
// media de tempo
long Hx711::averageValue(byte times)
{
        long sum = 0;
        for (byte i = 0; i < times; i++)
        {
                sum += getValue();
        }
```

```
	return sum / times;
}

long Hx711::getValue() //Pega o valor e armazena
{
	byte data[3];

	while (digitalRead(_pin_dout))
		;

	for (byte j = 0; j < 3; j++)
	{
		for (byte i = 0; i < 8; i++)
		{
			digitalWrite(_pin_slk, HIGH);
			bitWrite(data[2 - j], 7 - i, digitalRead(_pin_dout));
			digitalWrite(_pin_slk, LOW);
		}
	}

	digitalWrite(_pin_slk, HIGH);
	digitalWrite(_pin_slk, LOW);

	return ((long) data[2] << 16) | ((long) data[1] << 8) | (long) data[0];
}
// mostra o valor de tara
void Hx711::setOffset(long offset)
{
	_offset = offset;
}
// mostra o valor de Hx711calibrado
void Hx711::setScale(float scale)
{
	_scale = scale;
}
 //pega o valor de Hx711 da grama atual
float Hx711::getGram()
{
	long val = (averageValue() - _offset);
	return (float) val / _scale;
}
```

Comunicação Java + Arduino

```
import gnu.io.CommPortIdentifier;
import gnu.io.NoSuchPortException;
import gnu.io.SerialPort;
import java.io.IOException;
import java.io.OutputStream;
import javax.swing.JOptionPane;

public class ControlePorta {
  private OutputStream serialOut;
  private int taxa;
  private String portaCOM;

  /**
   * Construtor da classe ControlePorta
   * @param portaCOM - Porta COM que será utilizada para enviar os dados para o arduino
   * @param taxa - Taxa de transferência da porta serial geralmente é 9600
   */
  public ControlePorta(String portaCOM, int taxa) {
    this.portaCOM = portaCOM;
    this.taxa = taxa;
    this.initialize();
  }
  /**
   * Médoto que verifica se a comunicação com a porta serial está ok
   */
  private void initialize() {
    try {
      //Define uma variável portId do tipo CommPortIdentifier para realizar a comunicação
serial
      CommPortIdentifier portId = null;
      try {
        //Tenta verificar se a porta COM informada existe
        portId = CommPortIdentifier.getPortIdentifier(this.portaCOM);
      }catch (NoSuchPortException npe) {
        //Caso a porta COM não exista será exibido um erro
        JOptionPane.showMessageDialog(null, "Porta COM não encontrada.",
            "Porta COM", JOptionPane.PLAIN_MESSAGE);
      }
      //Abre a porta COM
      SerialPort port = (SerialPort) portId.open("Comunicação serial", this.taxa);
      serialOut = port.getOutputStream();
      port.setSerialPortParams(this.taxa, //taxa de transferência da porta serial
                  SerialPort.DATABITS_8, //taxa de 10 bits 8 (envio)
                  SerialPort.STOPBITS_1, //taxa de 10 bits 1 (recebimento)
                  SerialPort.PARITY_NONE); //receber e enviar dados
    }catch (Exception e) {
```

```
        e.printStackTrace();
    }
}

 /**
  * Método que fecha a comunicação com a porta serial
  */
 public void close() {
   try {
      serialOut.close();
   }catch (IOException e) {
     JOptionPane.showMessageDialog(null, "Não foi possível fechar porta COM.",
             "Fechar porta COM", JOptionPane.PLAIN_MESSAGE);
   }
 }

 /**
  * @param opcao - Valor a ser enviado pela porta serial
  */
 public void enviaDados(int opcao){
   try {
     serialOut.write(opcao); //escreve o valor na porta serial para ser enviado
   } catch (IOException ex) {
      JOptionPane.showMessageDialog(null, "Não foi possível enviar o dado. ",
             "Enviar dados", JOptionPane.PLAIN_MESSAGE);
   }
 }
}
```

Código fonte da tela principal (menu).

C: \ Users \ Lg \ Desktop \ Sigmetal \ src \ br \ com \ Estoque de \ cadastro \ Sigmetal \ gui \ MenuSigmetal.java

```
/*
    Diretório onde esta armazenada pasta contendo os arquivos do sistema Sigmetal

*/

pacote br.com.estoque.cadastro.sigmetal.gui; // Local onde são armazenados as classes no sistema

//Pacotes contendo importações dos dados das classes em questão para a classe Menu Sigmetal

 import br.com.estoque.compras.sigmetal.gui.ConfirmaCotacaoCompraSigmetalView;
import br.com.estoque.compras.sigmetal.gui.CotacaoCompraSigmetalView;
 import br.com.estoque.compras.sigmetal.gui.PedidoCompraSigmetalView;
 import br.com.estoque.controle.sigmetal.gui.PortasSerialSigmetalView;
 import br.com.estoque.controle_estoque.sigmetal.gui.AjustePrecosControle_EstoqueSigmetalView;
 import br.com.estoque.controle_estoque.sigmetal.gui.EntradaProdutoControle_EstoqueSigmetalView;
```

```
import br.com.estoque.financeiro.sigmetal.gui.ConciliacaoChequesSigmetalView;
import br.com.estoque.financeiro.sigmetal.gui.MovimentoBancarioSigmetalView;
import br.com.estoque.financeiro.sigmetal.gui.PagamentoSigmetalView;
import br.com.estoque.financeiro.sigmetal.gui.RecebimentoSigmetalView;
import br.com.estoque.movimento.sigmetal.gui.MovimentoOrcamentoVendaSigmetalView;
import br.com.estoque.movimento.sigmetal.gui.MovimentoVendaSigmetalView;
import java.awt.Toolkit;
import javax.swing.JFrame;

/ **
*
* @ author  Fabiano  Hoefling  Melo
 * /

público  classe  MenuSigmetal  estende javax.swing.JFrame { // Classe de livre acesso usando componentes
.swing

                // JFrame  classe do Swing do Java que é responsável por desenhar uma tela e tratar eventos na
mesma

   private String [] args ;

   / **
    * Cria nova  forma  MenuSigmetal
    * /
   público  MenuSigmetal () { // Detalhes do construtor
      initComponents (); // Faz a montagem do framework SwingBean, criando o painel através de um arquivo
XML,

      inicializar (); // Método de inicialização do menu
   }

    // Método de inicialização da imagem de ícone da barra superior do sistema
   privado  vazio  initialize () { //Método
       este . setIconImage (Toolkit. getDefaultToolkit () getImage (getClass () getResource (.. " cargo1.png " )));
      setIconImage (Toolkit. getDefaultToolkit () getImage (.
            " C: \ \ Users \ \ Lg \ \ desktop \ \ SIGMetal Java Desktop \ \ Imagens \ \ PNG \ \ cargo1.png " ));
// Local de armazenamento da imagem

   }

   /**
    * Este método é chamado de dentro do construtor para inicializar o formulário.
     * ATENÇÃO: Não modifique este código. O conteúdo deste método é sempre
     * Regenerado pelo editor de formulários.
    */

   @SuppressWarnings( " unchecked " )
    // <editor-fold defaultstate="collapsed" desc="Generated code">

   privadas  vazio  initComponents ( )  {  //Método que preenche um slot vazio com uma nova janela,
utilizado no rearranjo das janelas.
      BindingGroup  =  new  org . JDesktop . beansbinding . BindingGroup ( ) ;

      jMenuItem31  =  new  javax . swing . JMenuItem ( ) ; //
      jPanel2  =  new  javax . swing . JPanel ( ) ; // Inicializa o container visual contendo jPanel1, jPanel2 e
jPanel3
      jPanel1  =  new  javax . swing . JPanel ( ) ; // Inicializa o container visual contendo todos os botões do
Menu Sigmetal
```

```
        jButton1 = new javax . swing . JButton ( ) ; // Inicializa o  Botão  Cliente
        jButton2 = new javax . swing . JButton ( ) ; // Inicializa o  Botão  Produtos
        jButton3 = new javax . swing . JButton ( ) ; // Inicializa o  Botão  Vendas
        jButton4 = new javax . swing . JButton ( ) ; // Inicializa o  Botão  Estoque
        jButton5 = new javax . swing . JButton ( ) ; // Inicializa o  Botão  Sair
        jPanel3 = new javax . swing . JPanel ( ) ; //Inicializa o Painel 3
        jLabel2 = new javax . swing . JLabel ( ) ; //Inicializa o Painel 3
        jPanel4 = new javax . swing . JPanel ( ) ;
        jLabel1 = new javax . swing . JLabel ( ) ;
        jMenuBar1 = new javax . swing . JMenuBar ( ) ;
        jMenu1 = new javax . swing . JMenu ( ) ;
        jMenuItem1 = new javax . swing . JMenuItem ( ) ;
        jMenuItem2 = new javax . swing . JMenuItem ( ) ;
        jMenuItem3 = new javax . swing . JMenuItem ( ) ;
        jSeparator1 = new javax . swing . JPopupMenu . Separator ( ) ;
        jMenuItem4 = new javax . swing . JMenuItem ( ) ;
        jMenuItem5 = new javax . swing . JMenuItem ( ) ;
        jSeparator2 = new javax . swing . JPopupMenu . Separator ( ) ;
        jMenuItem6 = new javax . swing . JMenuItem ( ) ;
        jMenuItem7 = new javax . swing . JMenuItem ( ) ;
        jSeparator3 = new javax . swing . JPopupMenu . Separator ( ) ;
        jMenuItem8 = new javax . swing . JMenuItem ( ) ;
        jMenuItem32 = new javax . swing . JMenuItem ( ) ;
        jMenuItem10 = new javax . swing . JMenuItem ( ) ; // Menu referente  ao cadastro do plano de contas
        jMenuItem11 = new javax . swing . JMenuItem ( ) ; // Menu referente  ao Cadastro de Tipos de
Pagamento / Recebimento";
        jSeparator5 = new javax . swing . JPopupMenu . Separator ( ) ; // Separa um menu de outro
        jMenu2 = new javax . swing . JMenu ( ) ;
        jMenuItem9 = new javax . swing . JMenuItem ( ) ; // Menu referente  ao acesso a contas a pagar
        jMenuItem13 = new javax . swing . JMenuItem ( ) ;
        jMenuItem14 = new javax . swing . JMenuItem ( ) ;
        jMenuItem30 = new javax . swing . JMenuItem ( ) ;
        jMenu5 = new javax . swing . JMenu ( ) ;
        jMenuItem15 = new javax . swing . JMenuItem ( ) ;
        jMenuItem23 = new javax . swing . JMenuItem ( ) ;
        jMenu6 = new javax . swing . JMenu ( ) ;
        jMenuItem19 = new javax . swing . JMenuItem ( ) ;
        jMenuItem20 = new javax . swing . JMenuItem ( ) ;
        jMenuItem21 = new javax . swing . JMenuItem ( ) ;
        jMenuItem22 = new javax . swing . JMenuItem ( ) ;
        jMenu7 = new javax . swing . JMenu ( ) ;
        jMenuItem16 = new javax . swing . JMenuItem ( ) ;
        jMenuItem17 = new javax . swing . JMenuItem ( ) ;
        jMenuItem18 = new javax . swing . JMenuItem ( ) ;
        jMenu3 = new javax . swing . JMenu ( ) ;
        jMenuItem24 = new javax . swing . JMenuItem ( ) ;
        jMenuItem25 = new javax . swing . JMenuItem ( ) ;
        jSeparator4 = new javax . swing . JPopupMenu . Separator ( ) ;
        jMenu8 = new javax . swing . JMenu ( ) ;
        jMenuItem27 = new javax . swing . JMenuItem ( ) ;
        jMenuItem28 = new javax . swing . JMenuItem ( ) ;
        jMenu9 = new javax . swing . JMenu ( ) ;
        jMenuItem29 = new javax . swing . JMenuItem ( ) ;
        jMenu4 = new javax . swing . JMenu ( ) ;
        jMenuItem26 = new javax . swing . JMenuItem ( ) ;
        jMenuItem12 = new javax . swing . JMenuItem ( ) ;

        jMenuItem31 . setText ( " jMenuItem31 " ) ;
```

```
        setDefaultCloseOperation ( javax . balanço . WindowConstants . EXIT_ON_CLOSE ) ; // Maximizar e
minimizar a janela so sisrema
        setTitle ( " SIGMetal - Controle de Estoque " ) ; // Mostra o Título
        setBounds ( new java . awt . Rectangle ( 749 , 451 , 2 , 1 ) ) ; // Indica o tamanho da janela
        setIconImage ( getIconImage ( ) ) ; // Pega a imagem e depois executa
        setPreferredSize ( new java . awt . Dimension ( 640 , 450 ) ) ; // Dimensões da imagem na tela

        org . jdesktop . beansbinding . Binding binding = org . jdesktop . beansbinding . Bindings .
createAutoBinding ( org . jdesktop . beansbinding . AutoBinding . UpdateStrategy . READ_WRITE , this , org
. jdesktop . beansbinding . ELProperty . create ( " ${iconImage} " ) , this , org . jdesktop . beansbinding .
BeanProperty . create ( " iconImage " ) ) ;
        bindingGroup . addBinding ( binding ) ;

        jPanel2 . setBackground ( new java . awt . Color ( 242 , 255 , 255 ) ) ; // Indica a cor do Painel1
        jPanel2 . setPreferredSize ( new java . awt . Dimension ( 600 , 400 ) ) ; // Tamanho do Painel
        jPanel2 . setRequestFocusEnabled ( false ) ;
        jPanel2 . setVerifyInputWhenFocusTarget ( false ) ;

        jPanel1 . setBackground ( new java . awt . Color ( 219 , 252 , 212 ) ) ; Indica a cor do Painel1
        jPanel1 . setBorder ( javax . swing . BorderFactory . createEtchedBorder ( new java . awt . Color ( 0 ,
153 , 0 ) , new java . awt . Color ( 102 , 255 , 153 ) ) ) ;
        jPanel1 . setAlignmentY ( 0.9F ) ;
        jPanel1 . setAutoscrolls ( true ) ;
        jPanel1 . setDebugGraphicsOptions ( javax . swing . DebugGraphics . NONE_OPTION ) ;
        jPanel1 . setPreferredSize ( new java . awt . Dimension ( 398 , 100 ) ) ;

        jButton1 . setFont ( novo java . awt . Font ( " Times New Roman " , 1 , 14 ) ) ; / / NOI18N
        jButton1 . setIcon ( new javax . swing . ImageIcon ( " C: \\ Users \\ Lg \\ Desktop \\ Aulatobias \\
Sigmetal \\ src \\ imagens \\ pessoas.png " ) ) ; // NOI18N
        jButton1 . setText ( " Clientes " ) ;
        jButton1 . setHorizontalAlignment ( javax . swing . SwingConstants . LEFT ) ;
        jButton1 . addActionListener ( new java . awt . event . ActionListener ( ) {
            public void actionPerformed ( java . awt . event . ActionEvent evt ) {
                jButton1ActionPerformed ( evt ) ;
            }
        } ) ;

        jButton2 . setFont ( novo java . awt . Font ( " Times New Roman " , 1 , 14 ) ) ; / / NOI18N
        jButton2 . setIcon ( new javax . swing . ImageIcon ( " C: \\ Users \\ Lg \\ Desktop \\ Aulatobias \\
Sigmetal \\ src \\ imagens \\ Produtos.png " ) ) ; // NOI18N
        jButton2 . setText ( " Produtos " ) ;
        jButton2 . setHorizontalAlignment ( javax . swing . SwingConstants . LEFT ) ;
        jButton2 . setHorizontalTextPosition ( javax . swing . SwingConstants . RIGHT ) ;
        jButton2 . addActionListener ( new java . awt . event . ActionListener ( ) {
            public void actionPerformed ( java . awt . event . ActionEvent evt ) {
                jButton2ActionPerformed ( evt ) ;
            }
        } ) ;

        \\
        jButton3 . setFont ( novo java . awt . Font ( " Times New Roman " , 1 , 14 ) ) ; / / NOI18N
        jButton3 . setIcon ( new javax . swing . ImageIcon ( " C: \\ Users \\ Lg \\ Desktop \\ Aulatobias \\
Sigmetal \\ src \\ imagens \\ Vendas.png " ) ) ; // NOI18N
        jButton3 . setText ( " Vendas " ) ;
        jButton3 . setHorizontalAlignment ( javax . swing . SwingConstants . LEFT ) ;
        jButton3 . setHorizontalTextPosition ( javax . swing . SwingConstants . RIGHT ) ;

        jButton4 . setFont ( novo java . awt . Font ( " Times New Roman " , 1 , 14 ) ) ; / /
          NOI18N
        jButton4 . setText ( " Estoque de " ) ;
```

```
        jButton5 . setFont ( novo  java . awt . Font ( " Times New Roman " , 1 , 14 ) ) ; / / NOI18N
        jButton5 . setIcon ( new  javax . swing . ImageIcon ( " C: \\ Users \\ Lg \\ Desktop \\ Aulatobias \\
Sigmetal \\ src \\ imagens \\ Sair.png " ) ) ; // NOI18N
        jButton5 . setText ( " Sair " ) ;
        jButton5 . setHorizontalAlignment ( javax . swing . SwingConstants . LEADING ) ;
        jButton5 . setHorizontalTextPosition ( javax . swing . SwingConstants . RIGHT ) ;

        javax . swing . GroupLayout  jPanel1Layout  =  new  javax . swing . GroupLayout ( jPanel1 ) ;
        jPanel1 . setLayout ( jPanel1Layout ) ;
        jPanel1Layout . setHorizontalGroup (
            jPanel1Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addGroup ( jPanel1Layout . createSequentialGroup ( )
                . addContainerGap ( )
                . addComponent ( jButton1 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                . addPreferredGap ( javax . swing . LayoutStyle . ComponentPlacement . RELATED )
                . addComponent ( jButton2 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                . addPreferredGap ( javax . swing . LayoutStyle . ComponentPlacement . RELATED )
                . addComponent ( jButton3 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                . addPreferredGap ( javax . swing . LayoutStyle . ComponentPlacement . RELATED )
                . addComponent ( jButton4 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                . addPreferredGap ( javax . swing . LayoutStyle . ComponentPlacement . RELATED )
                . addComponent ( jButton5 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                . addContainerGap ( ) )
        ) ;
        jPanel1Layout . setVerticalGroup (
            jPanel1Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addGroup ( jPanel1Layout . createSequentialGroup ( )
                . addContainerGap ( )
                . addGroup ( jPanel1Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment .
BASELINE , false )
                    . addComponent ( jButton1 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                    . addComponent ( jButton2 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                    . addComponent ( jButton3 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                    . addComponent ( jButton4 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
                    . addComponent ( jButton5 , javax . swing . GroupLayout . DEFAULT_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE ) )
                . addContainerGap ( javax . swing . GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE ) )
        ) ;

        jPanel3 . setBackground ( new  java . awt . Color ( 255 , 255 , 255 ) ) ;
        jPanel3 . setBorder ( javax . swing . BorderFactory . createLineBorder ( new  java . awt . Color ( 0 , 153 ,
0 ) , 2 ) ) ;
        jPanel3 . setAutoscrolls ( true ) ;
        jPanel3 . setFocusTraversalPolicyProvider ( true ) ;
        jPanel3 . setPreferredSize ( new  java . awt . Dimension ( 640 , 284 ) ) ;

        jLabel2 . setHorizontalAlignment ( javax . swing . SwingConstants . CENTER ) ;
        jLabel2 . setIcon ( new  javax . swing . ImageIcon ( getClass ( ) . getResource ( "
/imagens/telaPrincipal.png " ) ) ) ; // NOI18N
        jLabel2 . setDebugGraphicsOptions ( javax . swing . DebugGraphics . BUFFERED_OPTION ) ;
```

```
        jLabel2 . setHorizontalTextPosition ( javax . swing . SwingConstants . CENTER ) ;
        jLabel2 . setMaximumSize ( new java . awt . Dimension ( 2000 , 1000 ) ) ;
        jLabel2 . setNextFocusableComponent ( jLabel2 ) ;
        jLabel2 . setPreferredSize ( new java . awt . Dimension ( 1000 , 500 ) ) ;

        javax . swing . GroupLayout jPanel3Layout = new javax . swing . GroupLayout ( jPanel3 ) ;
        jPanel3 . setLayout ( jPanel3Layout ) ;
        jPanel3Layout . setHorizontalGroup (
            jPanel3Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addComponent ( jLabel2 , javax . swing . GroupLayout . PREFERRED_SIZE , 0 , Short .
MAX_VALUE )
        ) ;
        jPanel3Layout . setVerticalGroup (
            jPanel3Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addComponent ( jLabel2 , javax . swing . GroupLayout . DEFAULT_SIZE , 296 , Short .
MAX_VALUE )
        ) ;

        jPanel4 . setBackground ( new java . awt . Color ( 219 , 252 , 212 ) ) ;
        jPanel4 . setBorder ( javax . swing . BorderFactory . createEtchedBorder ( new java . awt . Color ( 0 ,
153 , 0 ) , new java . awt . Color ( 102 , 255 , 153 ) ) ) ;

        jLabel1 . setBackground ( novo java . awt . Cor ( 219 , 252 , 212 ) ) ;
        jLabel1 . setFont ( novo java . awt . Font ( " Times New Roman " , 1 , 12 ) ) ; //


        Systems - Sistema de Controle de Estoque de Entrada e Saida com Automatizada da Produção. Versão
1.0. TODOS OS DIREITOS RESERVADOS. " ) ;
        jLabel1 . setAlignmentX ( 0.5F ) ;
        jLabel1 . setAutoscrolls ( verdadeiros ) ;

        javax . swing . GroupLayout jPanel4Layout = new javax . swing . GroupLayout ( jPanel4 ) ;
        jPanel4 . setLayout ( jPanel4Layout ) ;
        jPanel4Layout . setHorizontalGroup (
            jPanel4Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addComponent ( jLabel1 , javax . swing . GroupLayout . DEFAULT_SIZE , 729 , Short .
MAX_VALUE )
        ) ;
        jPanel4Layout . setVerticalGroup (
            jPanel4Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addComponent ( jLabel1 , javax . swing . GroupLayout . DEFAULT_SIZE , 32 , Short .
MAX_VALUE )
        ) ;

        jLabel1 . getAccessibleContext ( ) . setAccessibleName ( " .. Sistemas FHM - Sistema de Controle de
Estoque de Entrada e Saida com Automatizada da Produção Versão 1.0 TODOS OS DIREITOS
RESERVADOS " ) ; // Mostra o rodapé com texto

        javax . swing . GroupLayout jPanel2Layout = new javax . swing . GroupLayout ( jPanel2 ) ;
        jPanel2 . setLayout ( jPanel2Layout ) ;
        jPanel2Layout . setHorizontalGroup (
            jPanel2Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addComponent ( jPanel1 , javax . swing . GroupLayout . DEFAULT_SIZE , 733 , Short .
MAX_VALUE )
            . addComponent ( jPanel4 , javax . swing . GroupLayout . Alignment . TRAILING , javax . swing .
GroupLayout . DEFAULT_SIZE , javax . swing . GroupLayout . DEFAULT_SIZE , Short . MAX_VALUE )
            . addComponent ( jPanel3 , javax . swing . GroupLayout . Alignment . TRAILING , javax . swing .
GroupLayout . DEFAULT_SIZE , 733 , Short . MAX_VALUE )
        ) ;
        jPanel2Layout . setVerticalGroup (
```

```
        jPanel2Layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
        . addGroup ( jPanel2Layout . createSequentialGroup ( )
            . addComponent ( jPanel1 , javax . swing . GroupLayout . PREFERRED_SIZE , 83 , javax . swing .
GroupLayout . PREFERRED_SIZE )
            . addPreferredGap ( javax . swing . LayoutStyle . ComponentPlacement . RELATED )
            . addComponent ( jPanel3 , javax . swing . GroupLayout . DEFAULT_SIZE , 300 , Short .
MAX_VALUE )
            . addPreferredGap ( javax . swing . LayoutStyle . ComponentPlacement . RELATED )
            . addComponent ( jPanel4 , javax . swing . GroupLayout . PREFERRED_SIZE , javax . swing .
GroupLayout . DEFAULT_SIZE , javax . swing . GroupLayout . PREFERRED_SIZE ) )
    ) ;

    jMenuBar1 . setBackground ( new java . awt . Color ( 255 , 255 , 255 ) ) ;
    jMenuBar1 . setBorder ( javax . swing . BorderFactory . createLineBorder ( new java . awt . Color ( 0 ,
153 , 0 ) , 2 ) ) ;

    jMenu1 . setMnemonic ( ' c ' ) ;
    jMenu1 . setText ( " Cadastro " ) ;
    jMenu1 . setFont ( new java . awt . Font ( " SansSerif " , 1 , 12 ) ) ; // NOI18N

    jMenuItem1 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_C , java . awt . event . InputEvent . CTRL_MASK ) ) ;
    jMenuItem1 . setFont ( new java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
    jMenuItem1 . setMnemonic ( ' c ' ) ;
    jMenuItem1 . setText ( " Clientes " ) ;
    jMenuItem1 . addActionListener ( new java . awt . event . ActionListener ( ) {
        public void actionPerformed ( java . awt . event . ActionEvent evt ) {
            jMenuItem1ActionPerformed ( evt ) ;
        }
    } ) ;
    jMenu1 . add ( jMenuItem1 ) ;

    jMenuItem2 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_F , java . awt . event . InputEvent . CTRL_MASK ) ) ;
    jMenuItem2 . setFont ( new java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
    jMenuItem2 . setMnemonic ( ' f ' ) ;
    jMenuItem2 . setText ( " Fornecedores " ) ;
    jMenuItem2 . addActionListener ( new java . awt . event . ActionListener ( ) {
        public void actionPerformed ( java . awt . event . ActionEvent evt ) {
            jMenuItem2ActionPerformed ( evt ) ;
        }
    } ) ;
    jMenu1 . add ( jMenuItem2 ) ;

    jMenuItem3 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_B , java . awt . event . InputEvent . CTRL_MASK ) ) ;
    jMenuItem3 . setFont ( new java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
    jMenuItem3 . setMnemonic ( ' b ' ) ;
    jMenuItem3 . setText ( " Contas Bancárias " ) ;
    jMenuItem3 . addActionListener ( new java . awt . event . ActionListener ( ) {
        public void actionPerformed ( java . awt . event . ActionEvent evt ) {
            jMenuItem3ActionPerformed ( evt ) ;
        }
    } ) ;
    jMenu1 . add ( jMenuItem3 ) ;
    jMenu1 . add ( jSeparator1 ) ;

    jMenuItem4 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_U , java . awt . event . InputEvent . CTRL_MASK ) ) ;
    jMenuItem4 . setFont ( new java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
```

```
        jMenuItem4 . setMnemonic ( ' u ' ) ;
        jMenuItem4 . setText ( " Unidades " ) ;
        jMenuItem4 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem4ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem4 ) ;

        jMenuItem5 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_P ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem5 . setFont ( new  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ;  // NOI18N
        jMenuItem5 . setMnemonic ( ' p ' ) ;
        jMenuItem5 . setText ( " Produtos " ) ;
        jMenuItem5 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem5ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem5 ) ;
        jMenu1 . add ( jSeparator2 ) ;

        jMenuItem6 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_D ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem6 . setFont ( new  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ;  // NOI18N
        jMenuItem6 . setMnemonic ( ' D ' ) ;
        jMenuItem6 . setText ( " Departamentos " ) ;
        jMenuItem6 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem6ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem6 ) ;

        jMenuItem7 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_N ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem7 . setFont ( new  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ;  // NOI18N
        jMenuItem7 . setMnemonic ( ' n ' ) ;
        jMenuItem7 . setText ( " Funcionários " ) ;
        jMenuItem7 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem7ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem7 ) ;
        jMenu1 . add ( jSeparator3 ) ;

        jMenuItem8 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_A ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem8 . setFont ( new  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ;  // NOI18N
        jMenuItem8 . setMnemonic ( ' a ' ) ;
        jMenuItem8 . setText ( " Cartões " ) ;
        jMenuItem8 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem8ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem8 ) ;
```

```
        jMenuItem32 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_E ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem32 . setText ( " Cfop " ) ;
        jMenuItem32 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem32ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem32 ) ;

        jMenuItem10 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_L ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem10 . setFont ( new  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ; // NOI18N
        jMenuItem10 . setMnemonic ( ' l ' ) ;
        jMenuItem10 . setText ( " Plano de Contas " ) ;
        jMenuItem10 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem10ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem10 ) ;

        jMenuItem11 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_T ,  java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem11 . setFont ( new  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ; // NOI18N
        jMenuItem11 . setMnemonic ( ' T ' ) ;
        jMenuItem11 . setText ( " Tipo de Pgto/Recbto " ) ;
        jMenuItem11 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem11ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu1 . add ( jMenuItem11 ) ;
        jMenu1 . add ( jSeparator5 ) ;

        jMenuBar1 . adicionar ( jMenu1 ) ;

        jMenu2 . setMnemonic ( ' m ' ) ;
        jMenu2 . setText ( " Movimento " ) ;
        jMenu2 . setFont ( new  java . awt . Font ( " SansSerif " ,  1 ,  12 ) ) ; // NOI18N

        jMenuItem9 . setFont ( novo  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ; / / NOI18N
        jMenuItem9 . setText ( " Contas a Pagar " ) ;
        jMenuItem9 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem9ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu2 . add ( jMenuItem9 ) ;

        jMenuItem13 . setFont ( novo  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ; / / NOI18N
        jMenuItem13 . setText ( " Contas a Receber " ) ;
        jMenuItem13 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem13ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu2 . add ( jMenuItem13 ) ;

        jMenuItem14 . setFont ( novo  java . awt . Font ( " SansSerif " ,  0 ,  12 ) ) ; / / NOI18N
```

```
jMenuItem14 . setText ( " Vendas " ) ;
jMenuItem14 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem14ActionPerformed ( evt ) ;
    }
} ) ;
jMenu2 . add ( jMenuItem14 ) ;

jMenuItem30 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem30 . setText ( " Orçamentos " ) ;
jMenuItem30 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem30ActionPerformed ( evt ) ;
    }
} ) ;
jMenu2 . add ( jMenuItem30 ) ;

jMenu5 . setText ( " Controle de Estoque de " ) ;
jMenu5 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N

jMenuItem15 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem15 . setText ( " Entrada de NF " ) ;
jMenuItem15 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem15ActionPerformed ( evt ) ;
    }
} ) ;
jMenu5 . add ( jMenuItem15 ) ;

jMenuItem23 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem23 . setText ( " Atualização de Preços " ) ;
jMenuItem23 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem23ActionPerformed ( evt ) ;
    }
} ) ;
jMenu5 . add ( jMenuItem23 ) ;

jMenu2 . adicionar ( jMenu5 ) ;

jMenu6 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenu6 . setLabel ( " Compras " ) ;

jMenuItem19 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem19 . setText ( " requisição " ) ;
jMenu6 . adicionar ( jMenuItem19 ) ;

jMenuItem20 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem20 . setText ( " Cotação " ) ;
jMenuItem20 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem20ActionPerformed ( evt ) ;
    }
} ) ;
jMenu6 . add ( jMenuItem20 ) ;

jMenuItem21 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem21 . setText ( " Confirma Cotação " ) ;
jMenuItem21 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
```

```
        jMenuItem21ActionPerformed ( evt ) ;
    }
} ) ;
jMenu6 . add ( jMenuItem21 ) ;

jMenuItem22 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem22 . setText ( " Pedido " ) ;
jMenuItem22 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem22ActionPerformed ( evt ) ;
    }
} ) ;
jMenu6 . add ( jMenuItem22 ) ;

jMenu2 . adicionar ( jMenu6 ) ;

jMenu7 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenu7 . setLabel ( " Tesouraria e Banco " ) ;

jMenuItem16 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem16 . setText ( " Emissão de Cheques " ) ;
jMenu7 . adicionar ( jMenuItem16 ) ;

jMenuItem17 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem17 . setText ( " Conciliação de Cheques " ) ;
jMenuItem17 . addActionListener ( new java . awt . event . ActionListener ( ) {
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem17ActionPerformed ( evt ) ;
    }
} ) ;
jMenu7 . add ( jMenuItem17 ) ;

jMenuItem18 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem18 . setText ( " Movimento " ) ;
jMenuItem18 . addActionListener ( new java . awt . event . ActionListener ( ) { // Ação do menu
    public void actionPerformed ( java . awt . event . ActionEvent evt ) {
        jMenuItem18ActionPerformed ( evt ) ;
    }
} ) ;
jMenu7 . add ( jMenuItem18 ) ; //Menu(movimento) e submenu

jMenu2 . adicionar ( jMenu7 ) ;

jMenuBar1 . adicionar ( jMenu2 ) ;
jMenu2 . getAccessibleContext ( ) . setAccessibleName ( "" ) ;

jMenu3 . setMnemonic ( ' u ' ) ;
jMenu3 . setText ( " Utilitários " ) ;
jMenu3 . setFont ( new java . awt . Font ( " SansSerif " , 1 , 12 ) ) ; // NOI18N

jMenuItem24 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem24 . setText ( " Calculadora " ) ;
jMenu3 . adicionar ( jMenuItem24 ) ;

jMenuItem25 . setFont ( novo java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
jMenuItem25 . setText ( " Calendário " ) ;
jMenu3 . adicionar ( jMenuItem25 ) ;
jMenu3 . adicionar ( jSeparator4 ) ;

jMenu8 . setText ( " Sistema de Segurança " ) ;
```

```
        jMenu8 . setFont ( novo  java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N

        jMenuItem27 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent
.VK_F10 , 0 ) ) ;
        jMenuItem27 . setFont ( new  java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // Edita a fonte de texto
        jMenuItem27 . setText ( " Trocar Senha Corrente " ) ;
        jMenu8 . adicionar ( jMenuItem27 ) ;

        jMenuItem28 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_F12 , 0 ) ) ; //Atalho para item menu
        jMenuItem28 . setFont ( new  java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
        jMenuItem28 . setText ( " Definir do Nível de Acesso " ) ;
        jMenu8 . adicionar ( jMenuItem28 ) ;

        jMenu3 . adicionar ( jMenu8 ) ;

        jMenuBar1 . adicionar ( jMenu3 ) ;

        jMenu9 . setMnemonic ( ' o ' ) ;
        jMenu9 . setText ( " Controle " ) ;
        jMenu9 . setFont ( new  java . awt . Font ( " SansSerif " , 1 , 12 ) ) ; // NOI18N// Fonte do texto do menu

        jMenuItem29 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_O , java . awt . event . InputEvent . CTRL_MASK ) ) ;
        jMenuItem29 . setFont ( new  java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
        jMenuItem29 . setText ( " Lista Porta Serial " ) ;
        jMenuItem29 . setActionCommand ( "" ) ;
        jMenuItem29 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem29ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu9 . add ( jMenuItem29 ) ;

        jMenuBar1 . adicionar ( jMenu9 ) ;

        jMenu4 . setMnemonic ( ' a ' ) ;
        jMenu4 . setText ( " Ajuda " ) ;
        jMenu4 . setFont ( new  java . awt . Font ( " SansSerif " , 1 , 12 ) ) ; // NOI18N

        jMenuItem26 . setFont ( novo  java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; / / NOI18N
        jMenuItem26 . setText ( " Sobre o Sitema " ) ;
        jMenu4 . adicionar ( jMenuItem26 ) ;

        jMenuItem12 . setAccelerator ( javax . swing . KeyStroke . getKeyStroke ( java . awt . event . KeyEvent .
VK_S , java . awt . event . InputEvent . CTRL_MASK ) ) ; //Acelerador  para chamar operação (atalho)
        jMenuItem12 . setFont ( new  java . awt . Font ( " SansSerif " , 0 , 12 ) ) ; // NOI18N
        jMenuItem12 . setMnemonic ( ' s ' ) ;
        jMenuItem12 . setText ( " Sair " ) ;
        jMenuItem12 . addActionListener ( new  java . awt . event . ActionListener ( )  {
            public  void  actionPerformed ( java . awt . event . ActionEvent  evt )  {
                jMenuItem12ActionPerformed ( evt ) ;
            }
        } ) ;
        jMenu4 . add ( jMenuItem12 ) ;

        jMenuBar1 . adicionar ( jMenu4 ) ;

        setJMenuBar ( jMenuBar1 ) ;
```

```
        javax . swing . GroupLayout  layout  =  new  javax . swing . GroupLayout ( getContentPane ( ) ) ;
        getContentPane ( ) . setLayout ( layout ) ;
        layout . setHorizontalGroup (
            layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING )
            . addComponent ( jPanel2 ,  javax . swing . GroupLayout . Alignment . TRAILING ,  javax . swing .
GroupLayout . DEFAULT_SIZE ,  733 ,  Short . MAX_VALUE )
        ) ;
        layout . setVerticalGroup (
            layout . createParallelGroup ( javax . swing . GroupLayout . Alignment . LEADING ) //Posicionamento
e alinhamento da barra de menus no topo da tela
            . addGroup ( javax . swing . GroupLayout . Alignment . TRAILING ,  layout . createSequentialGroup ( )
                . addComponent ( jPanel2 ,  javax . swing . GroupLayout . DEFAULT_SIZE ,  431 ,  Short .
MAX_VALUE )
                . addContainerGap ( ) )
        ) ;

        BindingGroup . ligamento ( ) ;

        java . awt . Dimension  screenSize  =  java . awt . Toolkit . getDefaultToolkit ( ) . getScreenSize ( ) ;
        setBounds ( ( screenSize . width - 749 ) / 2 ,  ( screenSize . height - 503 ) / 2 ,  749 ,  503 ) ;
    } // </ Editor-fold>

    privado  vazio  jMenuItem2ActionPerformed ( java . awt . evento . ActionEvent  evt )  {  // Realiza a ação
do evento do item de menu  " Cadastro de Fornecedores "
        String args [] = new String [1];
        args [0] = " Cadastro de Fornecedores " ;
        . FornecedorSigmetalView principal (args);
    }

    privado  vazio  jMenuItem3ActionPerformed ( java . awt . evento . ActionEvent  evt )  {     // Realiza a
ação do evento do item de menu  " Cadastro das Contas Bancárias "
        String args [] = new String [1];
        args [0] = " Cadastro das Contas Bancárias " ;
        . BancoSigmetalView principal (args);
    }

    privado  vazio  jMenuItem12ActionPerformed ( java . awt . evento . ActionEvent  evt )  {  // Realiza a
ação do evento do item de menu  sair
        . Sistema de saída (0);
    }

    privado  vazio  jMenuItem8ActionPerformed ( java . awt . evento . ActionEvent  evt )  {  // Realiza a ação
do evento do item de menu  " Cadastro de CARTOES "
        String args [] = new String [1];
        args [0] = " Cadastro de CARTOES " ;
        . CartaoSigmetalView principal (args);
    }

    privado  vazio  jMenuItem11ActionPerformed ( java . awt . evento . ActionEvent  evt )  {  // Realiza a
ação do evento do item de menu  Cadastro de TIPOS de Pagamento / Recebimento " ;
        String args [] = new String [1];
        args [0] = " Cadastro de TIPOS de Pagamento / Recebimento " ;
        . TipoPgtoSigmetalView1 principal (args);
    }

    privado  vazio  jButton1ActionPerformed ( java . awt . evento . ActionEvent  evt )  {  //

    }
```

```
   privado vazio jMenuItem29ActionPerformed ( java . awt . evento . ActionEvent evt ) {   // Realiza a
ação do evento do item de menu " Lista Portas Seriais "
        String args [] = new String [1];
        args [0] = " Lista Portas Seriais " ;
        . PortasSerialSigmetalView principais (args);

   }

   privado vazio jMenuItem5ActionPerformed ( java . awt . evento . ActionEvent evt ) {   // Realiza a ação
do evento do item de menu " Cadastro de Produtos "
        String args [] = new String [1];
        args [0] = " Cadastro de Produtos " ;
        . ProdutoSigmetalView principal (args);
    }

   privado vazio jMenuItem4ActionPerformed ( java . awt . evento . ActionEvent evt ) {  // Realiza a ação
do evento do item de menu " Cadastro de Unidades "
        String args [] = new String [1];
        args [0] = " Cadastro de Unidades " ;
        . UnidadeSigmetalView principais (args);

   }

   privado vazio jMenuItem10ActionPerformed ( java . awt . evento . ActionEvent evt ) {  // Realiza a
ação do evento do item de menu " Cadastro fazer Plano de Contas "
        String args [] = new String [1];
        args [0] = " Cadastro fazer Plano de Contas " ;
        . PlanoContaSigmetalView principal (args);
    }

   privado vazio jMenuItem6ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação
do evento do item de menu " Cadastro de Departamentos "
        String args [] = new String [1];
        args [0] = " Cadastro de Departamentos " ;
        . DepartamentoSigmetalView principal (args);
    }

   privado vazio jMenuItem7ActionPerformed ( java . awt . evento . ActionEvent evt ) {  // Realiza a ação
do evento do item de menu " Cadastro de Funcionários "
        String args [] = new String [1];
        args [0] = " Cadastro de Funcionários " ;
        . FuncionarioSigmetalView principal (args);
    }

   privado vazio jMenuItem14ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação
do evento do item de menu " Movimento – Vendas”
        String args [] = new String [1];
        args [0] = " Movimento - Vendas " ;
        . MovimentoVendaSigmetalView principais (args);

   }

   privado vazio jMenuItem1ActionPerformed ( java . awt . evento . ActionEvent evt ) {   // Realiza a ação
do evento do item de menu " Cadastro de Clientes "
        String args [] = new String [1];
        args [0] = " Cadastro de Clientes " ;
        ClienteSigmetalView. principais (args);
    }
```

```
    privado vazio jMenuItem30ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação do evento do item de menu Movimento - Orçamento de Vendas "
        String args [] = new String [1];
        args [0] = " Movimento - Orçamento de Vendas " ;
        . MovimentoOrcamentoVendaSigmetalView principal (args);
    }

    privado vazio jMenuItem20ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação do evento do item de menu " Movimento - Cotação de Compras "
        String args [] = new String [1];
        args [0] = " Movimento - Cotação de Compras " ;
        . CotacaoCompraSigmetalView principal (args);
    }

    privado vazio jMenuItem21ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação do evento do item de menu Movimento - Confirmação Cotação de Compras "
        String args [] = new String [1];
        args [0] = " Movimento - Confirmação Cotação de Compras " ;
        . ConfirmaCotacaoCompraSigmetalView principal (args);
    }

    privado vazio jMenuItem22ActionPerformed ( java . awt . evento . ActionEvent evt ) {// Realiza a ação do evento do item de menu " Movimento - Pedido de Compras "
        String args [] = new String [1];
        args [0] = " Movimento - Pedido de Compras " ;
        . PedidoCompraSigmetalView principal (args);
    }

    privado vazio jMenuItem15ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação do evento do item de menu " Movimento - Estoque - Entrada de Produtos
        String args [] = new String [1];
        args [0] = " Movimento - Estoque - Entrada de Produtos " ;
        . EntradaProdutoControle_EstoqueSigmetalView principal (args);
    }

    privado vazio jMenuItem23ActionPerformed ( java . awt . evento . ActionEvent evt ) {  // Realiza a ação do evento do item de menu " Movimento - Estoque - AJUSTE de Preços "
        String args [] = new String [1];
        args [0] = " Movimento - Estoque - AJUSTE de Preços " ;
        . AjustePrecosControle_EstoqueSigmetalView principal (args);
    }

    privado vazio jMenuItem9ActionPerformed ( java . awt . evento . ActionEvent evt ) {// Realiza a ação do evento do item de menu " Movimento - Contas a Pagar "
        String args [] = new String [1];
        args [0] = " Movimento - Contas a Pagar " ;
        . PagamentoSigmetalView principal (args);
    }

    privado vazio jMenuItem13ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação do evento do item de menu " Movimento - Contas a Receber "
         String args [] = new String [1];
        args [0] = " Movimento - Contas a Receber " ;
        . RecebimentoSigmetalView principal (args);
    }

    privado vazio jMenuItem17ActionPerformed ( java . awt . evento . ActionEvent evt ) {  // Realiza a ação do evento do item de menu Movimento - Conciliação de Cheques "
         String args [] = new String [1];
        args [0] = " Movimento - Conciliação de Cheques " ;
```

```
    . ConciliacaoChequesSigmetalView principal (args);
  }

  privado vazio jMenuItem18ActionPerformed ( java . awt . evento . ActionEvent evt ) {  // Realiza a
ação do evento do item de menu " Movimento - Movimento de Banco "
    String args [] = new String [1];
     args [0] = " Movimento - Movimento de Banco " ;
    . MovimentoBancarioSigmetalView principal (args);
  }

  privado vazio jMenuItem32ActionPerformed ( java . awt . evento . ActionEvent evt ) { // Realiza a ação
do evento do item de menu " Cadastro de CFOP "
    String args [] = new String [1];
    args [0] = " Cadastro de CFOP " ;
    . CfopSigmetalView principal (args);
  }

  privado vazio jButton2ActionPerformed ( java . awt . evento . ActionEvent evt ) { //
    / / TODO adicione seu código de manipulação aqui:
  }

  / / Declaração de variáveis - não


  private javax.swing.JButton jButton1; // Variável do botão com o texto Clientes na tela do menu
  private javax.swing.JButton jButton2; // Variável do botão com o texto Produtos na tela do menu
  private javax.swing.JButton jButton3; // Variável do botão com o texto Vendas na tela do menu
  private javax.swing.JButton jButton4; // Variável do botão com o texto Estoque na tela do menu
  private javax.swing.JButton jButton5; // Variável do botão com o texto Sair na tela do menu
  private javax.swing.JLabel jLabel1; //Variável do campo de legenda, texto
  private javax.swing.JLabel jLabel2;
  private javax.swing.JMenu jMenu1;
  private javax.swing.JMenu jMenu2;
  private javax.swing.JMenu jMenu3;
  private javax.swing.JMenu jMenu4;
  private javax.swing.JMenu jMenu5;
  private javax.swing.JMenu jMenu6;
  private javax.swing.JMenu jMenu7;
  private javax.swing.JMenu jMenu8;
  private javax.swing.JMenu jMenu9;
  private javax.swing.JMenuBar jMenuBar1;
  private javax.swing.JMenuItem jMenuItem1;
  private javax.swing.JMenuItem jMenuItem10;
  private javax.swing.JMenuItem jMenuItem11;
  private javax.swing.JMenuItem jMenuItem12;
  private javax.swing.JMenuItem jMenuItem13;
  private javax.swing.JMenuItem jMenuItem14;
  private javax.swing.JMenuItem jMenuItem15;
  private javax.swing.JMenuItem jMenuItem16;
  private javax.swing.JMenuItem jMenuItem17;
  private javax.swing.JMenuItem jMenuItem18;
  private javax.swing.JMenuItem jMenuItem19;
  private javax.swing.JMenuItem jMenuItem2;
  private javax.swing.JMenuItem jMenuItem20;
  private javax.swing.JMenuItem jMenuItem21;
  private javax.swing.JMenuItem jMenuItem22;
  private javax.swing.JMenuItem jMenuItem23;
  private javax.swing.JMenuItem jMenuItem24;
  private javax.swing.JMenuItem jMenuItem25;
  private javax.swing.JMenuItem jMenuItem26;
```

```
    private javax.swing.JMenuItem jMenuItem27;
    private javax.swing.JMenuItem jMenuItem28;
    private javax.swing.JMenuItem jMenuItem29;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem30;
    private javax.swing.JMenuItem jMenuItem31;
    private javax.swing.JMenuItem jMenuItem32;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JMenuItem jMenuItem5;
    private javax.swing.JMenuItem jMenuItem6;
    private javax.swing.JMenuItem jMenuItem7;
    private javax.swing.JMenuItem jMenuItem8;
    private javax.swing.JMenuItem jMenuItem9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPopupMenu.Separator jSeparator1; //Variável do separador de menu item
    private javax.swing.JPopupMenu.Separator jSeparator2;
    private javax.swing.JPopupMenu.Separator jSeparator3;
    private javax.swing.JPopupMenu.Separator jSeparator4;
    private javax.swing.JPopupMenu.Separator jSeparator5;
    private org.jdesktop.beansbinding.BindingGroup bindingGroup;


  // Fim da declaração de variáveis

    públicos  estáticos  vazio  principais ( finais String args []) { // Método construtor estático (vazio)
        java.awt.EventQueue. invokeLater ( new Runnable () {
            @ Override
            público  vazio  run () {// Método para  rodar o sftware
                JFrame frame = new JFrame (); //Inicializa a janela
                frame.setExtendedState (JFrame. MAXIMIZED_BOTH );
                frame.pack ();
                novo MenuSigmetal () setVisible (. verdadeiro ); //Inicializa a tela onde vão ser colocados os botões
e as imagens na janela



            }
```