

UNIP - UNIVERSIDADE PAULISTA
PROJETO INTEGRADO MULTIDISCIPLINAR – PIM I
CURSO SUPERIOR DE TECNOLOGIA DE ANALISE E DESENVOLVIMENTO
DE SISTEMAS

CLAYTON BELARMINO DA SILVA
ERIK HIDEYUKI YOSHIMOTO SEKI
GABRIEL FERNANDES LEMOS
GABRIEL FRANCO GARCIA RODRIGUES DE PAULA
MATHEUS NUNES NEPOMUCENO

DESENVOLVIMENTO DE SISTEMA PARA PIZZARIA

SÃO PAULO

2019

CLAYTON BELARMINO DA SILVA

ERIK HIDEYUKI YOSHIMOTO SEKI

GABRIEL FERNANDES LEMOS

GABRIEL FRANCO GARCIA RODRIGUES DE PAULA

MATHEUS NUNES NEPOMUCENO

DESENVOLVIMENTO DE SISTEMA PARA PIZZARIA

Projeto de integração
multidisciplinar PIM apresentada como
exigência para a conclusão de semestre,
junto à Universidade Paulista UNIP, sob a
orientação do professor Ivan Geza Borbely

SÃO PAULO

2019

A Deus que nos criou e que foi criativo nesta tarefa. Seu fôlego de vida nos foi sustento e deu coragem para questionar realidades e propor sempre um novo mundo de possibilidades.

À UNIP (Universidade Paulista) pelo ambiente criativo e amigável que proporciona, pela oportunidade de fazer o curso, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbramos um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Agradecemos a todos os professores por nos proporcionarem o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional. Pelo tanto que se dedicaram a nós, não somente por terem nos ensinado, mas por terem nos feitos aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os nossos eternos. Ao nosso orientador, pelo empenho dedicado à elaboração deste projeto.

SUMÁRIO

Conteúdo

1 INTRODUÇÃO	12
1.1 APRESENTAÇÃO GERAL	13
1.2 OBJETIVOS GERAIS E ESPECÍFICOS.....	13
1.3 JUSTIFICATIVA.....	13
1.4 METODOLOGIA	14
2 PLANEJAMENTO DO SISTEMA	15
2.1.1 MENU.....	15
2.1.2 RELATÓRIOS	15
2.1.3 PERDA.....	15
2.2 DESCRIÇÃO DO CONTEXTO DA APLICAÇÃO	16
2.3 PROPOSTA SISTÊMICA.....	16
3 DESENVOLVIMENTO	16
3.1 LINGUAGEM C E SUAS FERRAMENTAS.....	17
3.1.1 FERRAMENTAS	17
3.1.1.1 DECLARAÇÃO DE VARIÁVEIS	17
3.1.1.2 ESTRUTURA DE DECISÃO.....	18
3.1.1.3 ESTRUTURA DE REPETIÇÃO	18
3.2 FLUXOGRAMA MODULAR.....	18
3.3 APLICAÇÃO DA ENGENHARIA DE SOFTWARE	26
3.3.1 ESCOPO.....	27
3.3.2 ENGENHARIA DE REQUISITOS.....	27
3.4 JUSTIFICATIVA DO MODELO DE PROCESSO.....	28
3.4.1 IDENTIFICAÇÃO E LEVANTAMENTO DE REQUISITOS	29
3.4.2 CONSTRUÇÃO DO PROTÓTIPO	29
3.5 DESCRIÇÃO DO CACULO DE APOIO AOS RELATÓRIOS	33
3.6 DIAGRAMA DA REPRESENTAÇÃO DA REDE DE COMUNICAÇÃO	33
3.6.1 ESPECIFICAÇÃO DIAGRAMA DE REDES.....	34
3.7 CRONOGRAMA DE DESENVOLVIMENTO E IMPLANTAÇÃO.....	36
4 IMPLANTAÇÃO DO SISTEMA.....	37
4.1 DESCRIÇÃO DO PROCESSO DE IMPLANTAÇÃO	37
4.1.1 DETALHES DO LOGIN.....	37

4.1.1 TELA DE MENU.....	39
4.1.2 CADASTROS.....	40
4.1.3 VENDAS	44
4.1.4 RELATÓRIOS	48
4.1.5 OPÇÕES.....	50
4.2 MANUAL DE INSTALAÇÃO DO SOFTWARE.....	50
4.3 MANUAL DE CONFIGURAÇÃO DA REDE	51
4.4 MANUAL DE TREINAMENTO DOS USUARIOS	56
4.5 GLOSSÁRIO DO SISTEMA.....	58
5 CONCLUSÃO.....	63

LISTA DE FIGURAS

Figura 1: Dados de fluxograma	19
Figura 2: Fluxograma de Login e senha	19
Figura 3: Dados de Cadastro do Produto	20
Figura 4: Dados de Cadastro de Cliente	21
Figura 5: Dados de Cadastro de Funcionários	21
Figura 6: Dados de Cadastro do Fornecedor	21
Figura 7: Dados de Cadastro dos Gastos	22
Figura 8: Opção de Voltar e Sair	22
Figura 9: Fluxograma de Venda	22
Figura 10: Fluxograma de Relatórios	23
Figura 11: Opções e Sair	23
Figura 12: Dados de Cadastro do Produto (user padrão)	24
Figura 13: Dados de Cadastro do Cliente (user padrão)	25
Figura 14: Dados de Cadastro do Fornecedor (user padrão)	25
Figura 15: Opção de voltar e sair (user padrão)	25
Figura 16: Fluxograma de Venda (user padrão)	26
Figura 17: Opção e Sair (user padrão)	26
Figura 18: Requisitos de software	28
Figura 19: Estrutura de Menu	29
Figura 20: Protótipo tela de menu	30
Figura 21: Protótipo tela de cadastro	31
Figura 22: Protótipo tela de venda	31
Figura 23: Protótipo tela de relatórios	32
Figura 24: Protótipo tela de opções	32
Figura 25: Calculo por unidade	33
Figura 26: Calculo da rede	33
Figura 27: Primeira unidade	34
Figura 28: Segunda unidade	35
Figura 29: Terceira unidade	35
Figura 30: Cronograma	36
Figura 31: Tela de Login (ADMIN)	38
Figura 32: Tela de login (usuário padrão)	38

Figura 33: Tela de menu	39
Figura 34: Tela de menu do usuário padrão.....	39
Figura 35: Tela de Cadastro.....	40
Figura 36: Tela de cadastro do produto	41
Figura 37: Tela de Cadastro dos Clientes	41
Figura 38: Tela de Cadastro dos Funcionários.....	42
Figura 39: Tela de Cadastro dos Fornecedores	42
Figura 40: Tela de Cadastro dos Gastos.....	43
Figura 41: Consultar os Gastos.....	43
Figura 42: Resultado do gastos da unidade 1	44
Figura 43: Tela de Vendas	45
Figura 44: Tela de gerar nova venda	45
Figura 45: Tela de gerar nova venda (2)	46
Figura 46: Tela de Resultado de venda	46
Figura 47: Tela de listar venda	47
Figura 48: Tela de resultado de listar venda	47
Figura 49: Inserir,Alterar, Apagar e Listar	48
Figura 50: Tela de Relatórios	48
Figura 52: Tela de faturamento Mensal.....	49
Figura 53: Tela de Faturamento total	49
Figura 54: Tela de Opções.....	50
Figura 55: Baixar sistema.....	50
Figura 56: main.exe.....	50
Figura 57: Sistema	51
Figura 58: Tela inicial	51
Figura 59: Painel de controle.....	52
Figura 60: Rede e internet.....	52
Figura 61: Centro de rede e compartilhamento	53
Figura 62: Configurar uma nova rede.....	53
Figura 63: Conecte-se à internet.....	54
Figura 64: PPPoE.....	54
Figura 65: Login	55
Figura 66: Finalizado.....	56

Figura 67: Tela principal	56
Figura 68: Tela de cadastros.....	57
Figura 69: Tela de vendas.....	57
Figura 70: Tela de relatórios.....	58

RESUMO

É de grande importância o esclarecimento do nosso projeto com base na situação que foi proporcionado a nós, dito isso, a construção foi feita a partir da necessidade na criação de uma rede para três unidades de uma determinada pizzaria, trazendo controle e auxílio para o manuseio do cliente, foi desenvolvido com ferramentas em linguagem C, armazenando em arquivos texto, produzindo de acordo com as necessidades do problema solicitado. Portanto, adicionamos algumas características significativas para a elaboração do sistema, que é: gerenciamento do projeto, qualidade do produto em desenvolvimento, domínio de cada integrante do grupo e divisões de tarefas.

Para uma melhor eficiência do nosso produto sistemático, é necessário estabelecermos etapas e requisitos que o tornarão fundamental em seu uso, dessa maneira, produzimos um esqueleto com algumas alternativas, como por exemplo, opções de cadastro, vendas, relatórios, opções de ajuda e etc. Contudo, o desenvolvimento da rede proporcionará diversas atividades que irá auxiliar o usuário em inúmeras ocasiões. Apresentando as opções de manuseio do cliente anteriormente, não podemos esquecer os relatórios de vendas organizados sequencialmente por: sabores mais pedidos por unidade, unidade que obtém maior lucro e totalização diária e mensal do faturamento por unidade e o total da rede.

Em vista disso, desenvolvemos de acordo com as necessidades apresentadas, com objetivo de construirmos uma rede viável com ótimo desempenho e configurações sustentáveis em suas funcionalidades.

PALAVRAS CHAVE: pizzaria, linguagem C, controle, requisitos, relatórios

ABSTRACT

It is very important to clarify our project based on the situation that was provided to us, the construction was made from the need to create a network for three units of a pizzeria, bringing control and aid for handling. Was developed with tools in C language, storing in text files, producing according to the needs of the requested problem. Therefore, we have added some significant features to the design of the system, which are: project management, quality of product under development, domain of each group member and division of tasks.

For better efficiency of our systematic product, we need to establish steps and requirements that will make it fundamental in its use, so will produce a skeleton with some alternatives, such as registration options, sales, reports, help options etc. However, the development from the network will provide various activities that will assist the user on numerous occasions. Introducing the customer's handling options earlier, let us not forget the sales reports organized sequentially by: flavors plus orders per unit, highest profitable unit and daily and monthly total billing per unit and total network.

In view of this, we develop according to the needs presented, aiming to build a viable network with great performance and sustainable configurations in its functionality.

KEYWORDS: pizzeria, C language, control, requirements, reports

1 INTRODUÇÃO

A primeira versão de C foi criada por Dennis Ritchie em 1972 nos laboratórios Bell para ser incluído como um dos softwares a serem distribuídos juntamente com o sistema operacional Unix do computador PDP-11, na equipe certificada por Ken Thompson.

Ao ponto de vista técnico, o surgimento do C iniciou com a linguagem ALGOL 60, definida em 1960. ALGOL era uma linguagem de alto nível, que permitia ao programador trabalhar "longe da máquina", sem se preocupar com os aspectos de como cada comando ou dado era armazenado ou processado. Foi criado para substituir o FORTRAN. ALGOL não teve sucesso, talvez por tentar ser de muito alto nível em uma época em que a maioria dos sistemas operacionais exigiam do usuário um grande conhecimento de *hardware*.

Em 1967 surgiu CPL (*Combined Programming Language*) nas universidades de Londres e Cambridge com o objetivo, segundo a equipe do projeto, de "trazer ALGOL à terra", ou "manter contato com a realidade de um computador real". Da mesma forma de ALGOL, CPL não foi bem aceita, em especial pelos projetistas de sistemas operacionais que a consideravam difícil de implementar. Ainda em 1967, em Cambridge, Martin Richards criou o BCPL (Basic CPL), uma simplificação do CPL, tentando manter apenas as "boas coisas do CPL".

Em 1970, Ken Thompson, chefe da equipe que projetou o UNIX para o PDP11 do Bell Labs, implementou um compilador para uma versão mais reduzida do CPL. Batizou a linguagem de B. Tanto BCPL quanto B mostravam-se muito limitadas, prestando-se apenas para certas classes de problemas. Isto se fez sentir especialmente na primeira versão do PDP11, lançado no mercado em 1971. Um dos fatores que levou à isto foi a intenção do grupo responsável pelo UNIX de reescrevê-lo todo em uma linguagem de alto nível, e para isto B era considerado lento. Estes problemas levaram a que o projetista Dennis Ritchie, do Bell Labs, fosse encarregado de projetar uma nova linguagem, sucessora do B, que viria então, a ser chamada de C.

A linguagem C buscou manter o "contato com o computador real" e ainda sim dar ao programador novas condições para o desenvolvimento de programas em áreas diversas, como comercial, científica e de engenharia. Por muitos anos (aproximadamente 10) a sintaxe (formato) tida como padrão da linguagem C foi aquela fornecida com o UNIX versão 5.0 do Bell Labs. A principal documentação deste padrão encontra-se na publicação "**The C Programming Language**", de Brian Kernighan e Dennis Ritchie (K&R), tida como a "bíblia da linguagem C".

O mais interessante desta versão de C era que os programas-fonte criados para rodar em um tipo de computador podiam ser transportados e recompilados em outros sem grandes problemas. A esta característica dá-se o nome de **portabilidade**. Com ela, uma empresa que desenvolve um programa pode fazê-lo rodar em diferentes computadores sem ter um elevado custo a cada vez que isto for feito. Em 1985, ANSI (**American National Standards Institute**) estabeleceu um padrão oficial de C o chamado "**C ANSI**". (Sarrogia, 2019)

1.1 APRESENTAÇÃO GERAL

Este documento tem como objetivo explicar as soluções de um determinado problema de rede entre três unidades de pizzaria, oferecendo controle e facilitação no gerenciamento de seu negócio. Portanto, nosso grupo obtém finalidade de conduzir o desenvolvimento com base no plano de ação para um sistema solicitado.

É de grande importância colocarmos em pauta os tópicos mais importantes de nosso sistema e seu escopo adjacente. Cadastro de produtos, funcionários, clientes, fornecedor e gastos (todos com a funcionalidade de salvar, alterar e excluir), pedidos solicitados pelo cliente, relatórios e opções.

Desta forma, nosso software foi dedicado para apresentar serventia e aplicações onde abrangerá todo o sistema proposto para as unidades com base em sua situação problema.

Fonte: Elaboração própria

1.2 OBJETIVOS GERAIS E ESPECÍFICOS

O objetivo geral do nosso aplicativo é a criação de uma ferramenta de uso simultâneo entre uma rede de três pizzarias trazendo facilidade de utilização e ampliando um novo recurso para a empresa que hoje não possui esse software. Já os objetivos específicos é a construção de um aplicativo que traga o máximo de recursos para a rede de pizzaria, sendo eles:

Atendimento online, pois hoje só temos atendimento físico ou por telefone; Abrangência no nosso cardápio para atender ao maior número de públicos e centralizando o meio de comunicação entre as redes e os clientes; Relatórios com faturamento mensal, semana e diário para acompanhamento de evolução das pizzarias.

Fonte: Elaboração própria

1.3 JUSTIFICATIVA

O projeto se consiste em funcionalidades que toda pizzaria precisa, porém, este sistema contém funções automatizadas e seguras para que o funcionário trabalhe de forma eficiente e prática, sempre visando a produtividade, consequentemente gerando mais rentabilidade, maior desempenho e agilidade nas tarefas diárias da pizzaria.

Fonte: Elaboração própria

1.4 METODOLOGIA

A Metodologia é o estudo dos métodos. Ou então as etapas a seguir num determinado processo. Tem como finalidade captar e analisar as características dos vários métodos disponíveis, avaliar suas capacidades, potencialidades, limitações ou distorções e criticar os pressupostos ou as implicações de sua utilização. Além de ser uma disciplina que estuda os métodos, a metodologia é também considerada uma forma de conduzir a pesquisa ou um conjunto de regras para ensino de ciência e arte.

A Metodologia é a explicação minuciosa, detalhada, rigorosa e exata de toda ação desenvolvida no método (caminho) do trabalho de pesquisa. É a explicação do tipo de pesquisa, do instrumental utilizado (questionário, entrevista etc.), do tempo previsto, da equipe de pesquisadores e da divisão do trabalho, das formas de tabulação e tratamento dos dados, enfim, de tudo aquilo que se utilizou no trabalho de pesquisa.

Em Gestão de Projetos, existe a metodologia geral e a metodologia detalhada. A metodologia pode ser dividida em vários métodos até chegar num determinado objetivo.

Deve-se notar que a palavra metodologia é muitas vezes usada onde seria mais adequado usar método. O termo metodologia inclui os seguintes conceitos, em relação a uma disciplina particular ou campo de estudo:

Coleção de teorias, conceitos e ideias; estudo comparativo de diferentes enfoques; Crítica de um método individual. (TANIA, 2009).

2 PLANEJAMENTO DO SISTEMA

O projeto integrado multidisciplinar apresentado tem como o objetivo principal a construção de um sistema para uma pizzaria. Para isso, foram estudados os conceitos de planejamento de uma pizzaria, buscando avaliar a situação atual da empresa e determinar quais seria os principais pontos que deveriam ser inseridas dentro de um novo projeto. Ao seu termino, o software deve apresentar resultados que contribuam para a rotina diária da pizzaria, tais como, relatórios de vendas, ganhos, perdas, etc. Com isso, o sistema irá ser de grande ajuda para uma tomada de decisão.

2.1 CENÁRIO DETALHADO DA SITUAÇÃO PROBLEMA

Inicialmente uma rede de pizzarias possui apenas três unidades próximas entre si, localizadas no centro de uma cidade do Brasil. A rede de pizzarias necessita desenvolver um sistema para o gerenciamento e controle do negócio focado nos atendimentos ao cliente. Como a empresa não possui capital para grandes investimentos em infraestrutura, bom como software caros, optou-se por desenvolver o sistema na linguagem C em modo console, armazenando os dados em arquivos de texto com os diários para contabilização.

Cada unidade da pizzaria possui um computador onde é executado o programa para atendimento aos clientes a ser desenvolvimento. As três unidades da rede estão interligadas pela internet formando uma pequena intranet. Em uma das unidades há uma quarta máquina para onde os arquivos de fechamento diário das operações é copiado ao final do expediente.

2.1.1 MENU

Tela de login, tela para cadastros de bebidas, sabores, pizzaiolos, pedidos, clientes, reclamações e elogios dos clientes, promoções, controle do estoque da matéria prima para elaboração das pizzas.

2.1.2 RELATÓRIOS

Sabores mais pedidos por unidade da rede, unidade da rede que mais vende, totalização diária e mensal do faturamento por unidade de rede e total da rede.

2.1.3 PERDA

Cancelamentos de pedidos e tudo aquilo que influencia o faturamento da pizzaria.

Fonte: Elaboração própria

2.2 DESCRIÇÃO DO CONTEXTO DA APLICAÇÃO

O sistema possui dois tipos de login. Master e usuário padrão, onde o Master teria acesso a tudo que o software tem a oferecer, cadastros de produtos, clientes, funcionários, fornecedores, vendas e cancelamento de vendas, relatórios de faturamento, sejam eles diários, mensais e totais, entre outras coisas. Já o usuário padrão, somente a vendas, isso por motivos de segurança e hierarquia.

O programa contém 5 telas essenciais para uma pizzaria. Tela de cadastro, vendas, gastos, relatórios e opções, onde a primeira seria para cadastrar novos clientes, fornecedores, produtos, etc. A segunda seria para uma nova venda ou cancela-la. A visão de relatórios é uma visão gerencial, e somente o Master tem acesso. Nessa visão ele consegue gerar relatórios de sabores mais pedidos, quantidade de vendas por unidade, faturamento diário, mensal e total.

As opções é uma aba onde os dois logins possui acesso, nele o funcionário consegue alterar sua senha, pedir ajuda e saber mais sobre o sistema.

Fonte: Elaboração própria

2.3 PROPOSTA SISTÊMICA

O programa feito em linguagem C em modo console é totalmente capaz de manter as três unidades funcionando em perfeito estado, todas com as mesmas opções de menu, relatório e perdas.

A opção de menu é o pilar do sistema, pois nele são efetuados os cadastros, elogios, reclamações e controle do estoque. Relatórios são para uma melhor tomada de decisão, ele apresenta sabores mais pedidos por unidade, rede que mais vende, totalização diária e mensal do faturamento por unidade de rede e total da rede. A ultima visão é a de perda, também utilizado para tomada de decisão que contém visões como cancelamentos de pedidos e tudo aquilo que influencia o faturamento da pizzaria.

Fonte: Elaboração própria

3 DESENVOLVIMENTO

Neste capítulo, as etapas do desenvolvimento deste trabalho são detalhadas. Inicialmente, é mostrada uma visão geral sobre o planejamento do sistema, mostrando as principais características e detalhando as técnicas e recursos utilizados. Depois, são apresentados os atores, que são os usuários do sistema; inicialmente, o sistema engloba o controle de uma pizzaria, porém, a possibilidade de expansão do sistema para outros setores é grande. Em seguida, são apresentadas, detalhadamente, a implantação e todas as funcionalidades e

restrições do sistema. Na última sessão, explica-se sobre o manual de instalação, configuração da rede e treinamento para o usuário onde ele poderá usar seus conhecimentos adquiridos para enfim, fazer uso do software

Fonte: Elaboração própria

3.1 LINGUAGEM C E SUAS FERRAMENTAS

A linguagem C é uma linguagem de alto nível, genérica. Foi desenvolvida por programadores para programadores tendo como meta características de flexibilidade e portabilidade. O C é uma linguagem que nasceu juntamente com o advento da teoria de linguagem estruturada e do computador pessoal. Assim tornou-se rapidamente uma linguagem “popular” entre os programadores. O C foi usado para desenvolver o sistema operacional UNIX, e hoje está sendo usada para desenvolver novas linguagens, entre elas a linguagem C++ e Java.

O C é uma linguagem de alto nível com uma sintaxe bastante estruturada e flexível tornando sua programação bastante simplificada e programas em C são compilados, gerando programas executáveis.

Essa linguagem compartilha recursos tanto de alto quanto de baixo nível, pois permite acesso e programação direta do microprocessador. Com isto, rotinas cuja dependência do tempo é crítica, podem ser facilmente implementadas usando instruções em Assembly. Por esta razão o C é a linguagem preferida dos programadores de aplicativos e de grande portabilidade. O compilador C gera códigos mais enxutos e velozes do que muitas outras linguagens, embora estruturalmente simples (poucas funções intrínsecas) o C não perde funcionalidade pois permite a inclusão de uma farta quantidade de rotinas do usuário. Os fabricantes de compiladores fornecem uma ampla variedade de rotinas pré-compiladas em bibliotecas. (Brasil Escola, 2017)

3.1.1 FERRAMENTAS

Existem muitos códigos em linguagem C, porém, iremos destacar os mais importantes, com isso, podemos escrever um algoritmo para resolução de um problema por intermédio de qualquer linguagem. A seguir mostramos alguns exemplos de trechos de códigos escritos na linguagem C.

3.1.1.1 DECLARAÇÃO DE VARIÁVEIS

Em C, como na maioria das linguagens, as variáveis devem ser declaradas no início do programa. Estas variáveis podem ser de vários tipos: int (inteiro), float (real de simples precisão) e outras que serão vistas no capítulo 2. No exemplo acima num, raiz, inf e sup são declaradas como variáveis reais, enquanto i é declarada como uma variável inteira.

3.1.1.2 ESTRUTURA DE DECISÃO

Permite direcionar o fluxo lógico para dois blocos distintos de instruções conforme uma condição de controle.

```
If (condição) {  
    Bloco 1;  
} else {  
    Bloco2;  
};
```

3.1.1.3 ESTRUTURA DE REPETIÇÃO

Permite executar repetidamente um bloco de instruções até que uma condição de controle seja satisfeita.

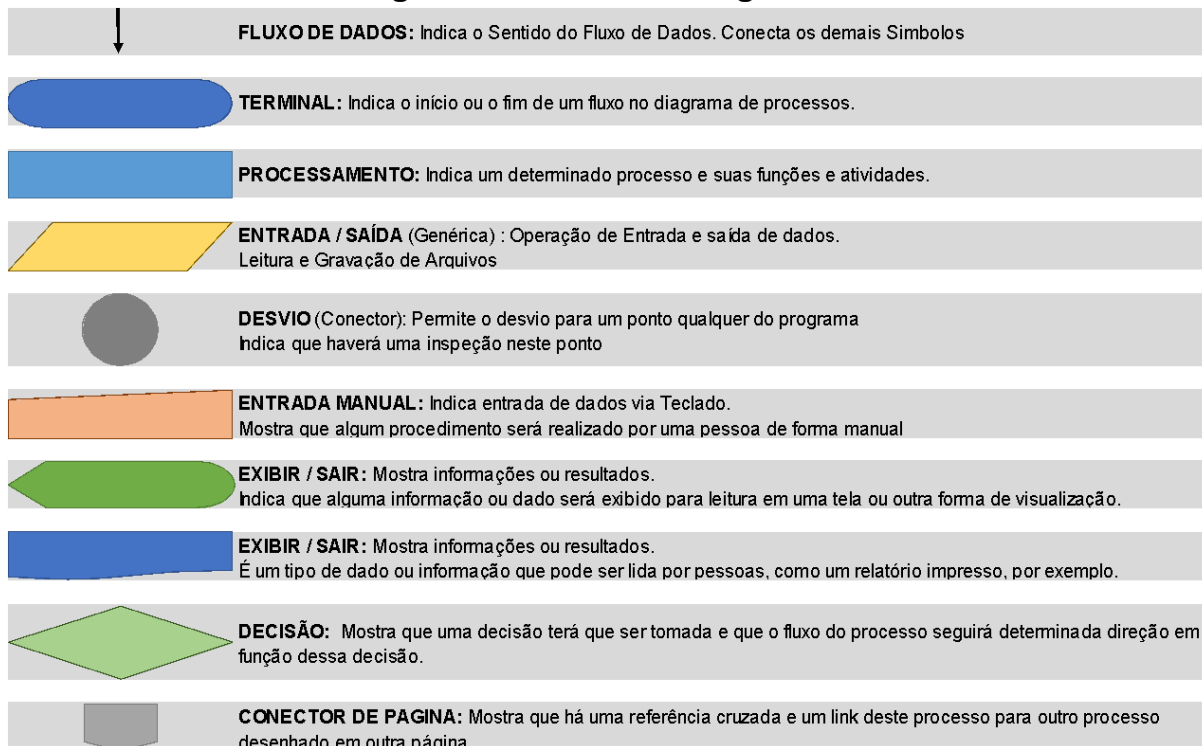
```
do {  
    bloco;  
} while (condição);
```

3.2 FLUXOGRAMA MODULAR

Passo a Passo Fluxograma – Pizzaria

Aqui iremos lhes mostrar passo a passo do fluxograma do nosso projeto e alguns símbolos “Expressões” que utilizamos. Lembrando que podemos ter símbolos não utilizados, mas deixamos para que sejam de fácil identificação.

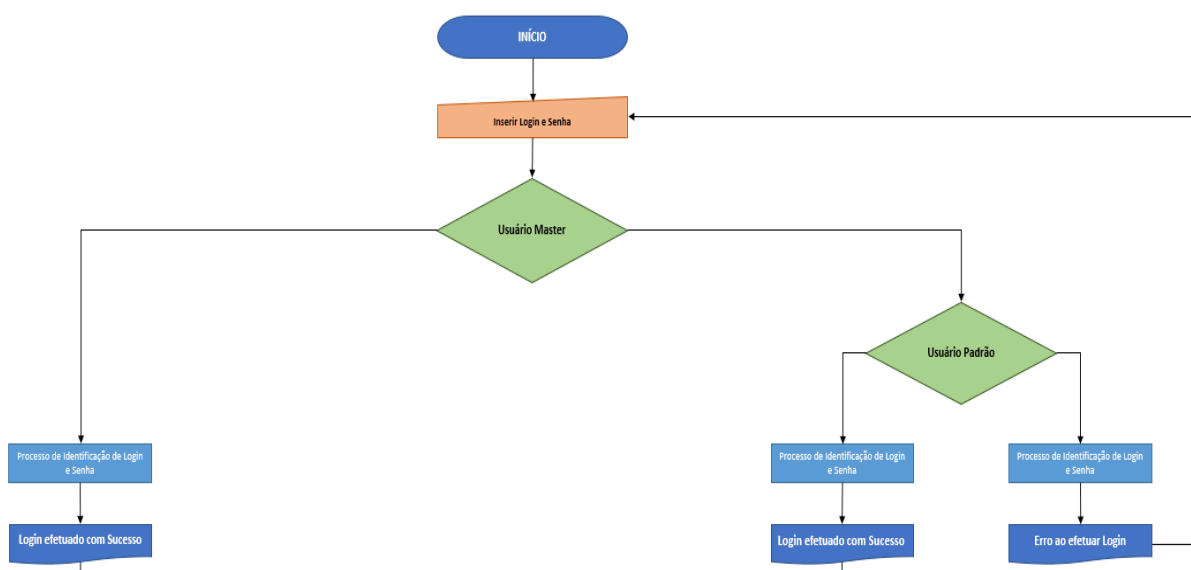
Figura 1: Dados de fluxograma



Iniciando o processo iremos inserir **Login e Senha** -> O Software irá identificar se o **“LOGIN e SENHA”** informado trata-se de um perfil de Administrador (Master) ou Usuário Padrão (User), caso ele informe o login e senha inválidos ele será redirecionado p/ a tela de acesso novamente até que insira as informações corretas.

Segue Início do fluxograma:

Figura 2: Fluxograma de Login e senha



O Login feito com sucesso iremos iniciar pelo acesso do **“Administrador” – MASTER.**

Nele teremos as telas iniciais de:

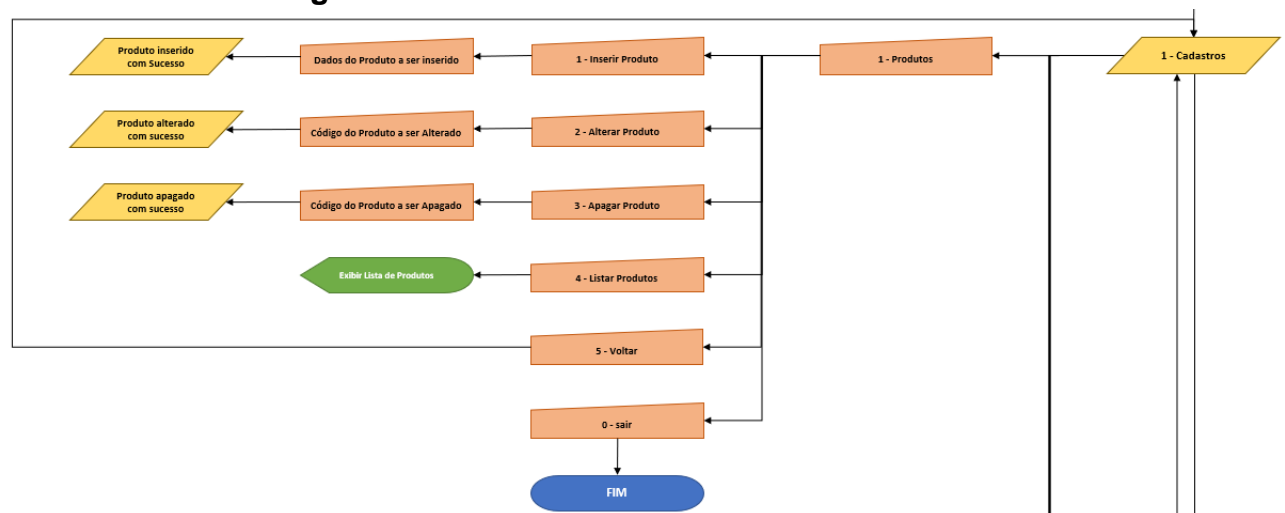
Cadastros
Vendas
Relatórios
Opções
Sair

Iniciamos por **Cadastros** teremos as seguintes opções:

Produtos
Clientes
Funcionários
Fornecedor
Gastos
Voltar
Sair

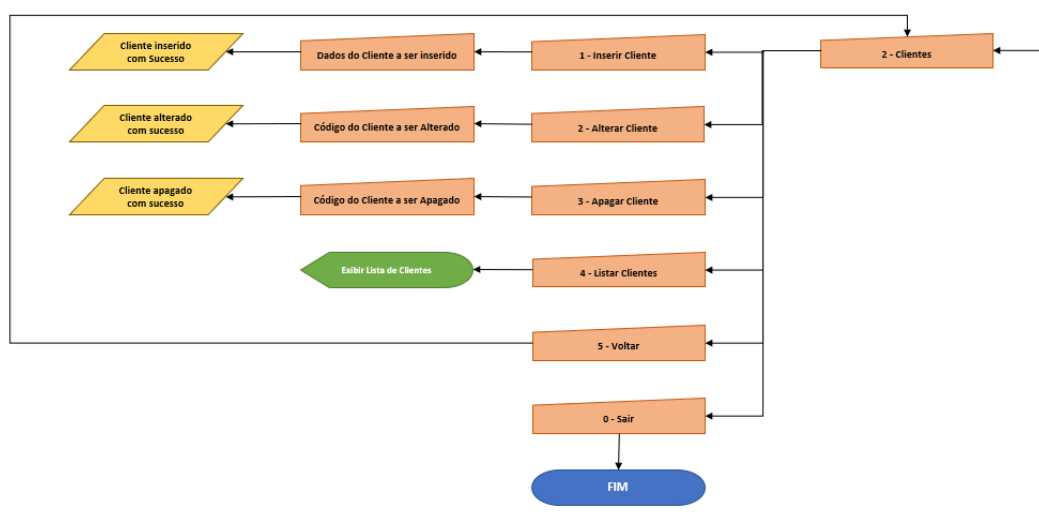
E começaremos por **Produtos** e teremos as opções conforme imagem abaixo:

Figura 3: Dados de Cadastro do Produto



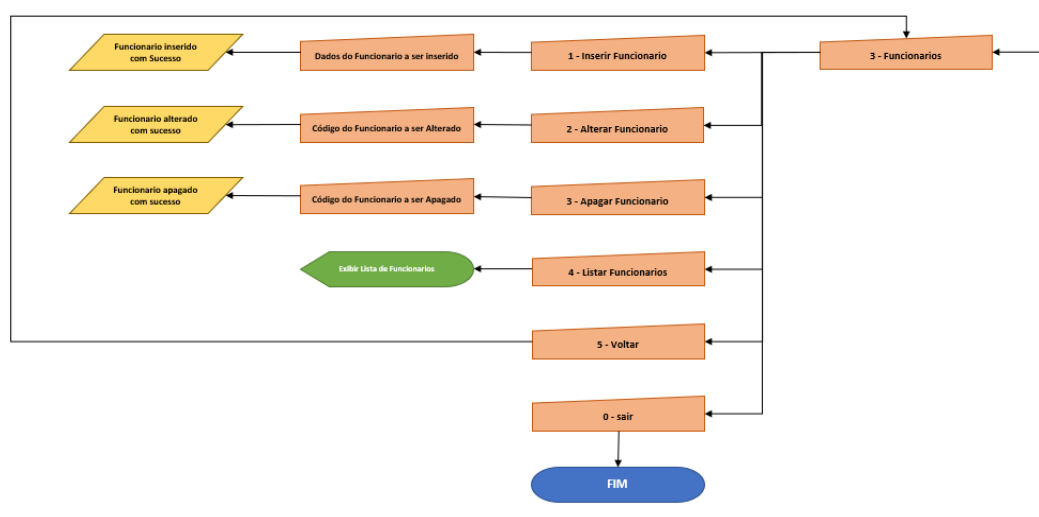
Na segunda etapa teremos **Clientes** e teremos as opções:

Figura 4: Dados de Cadastro de Cliente



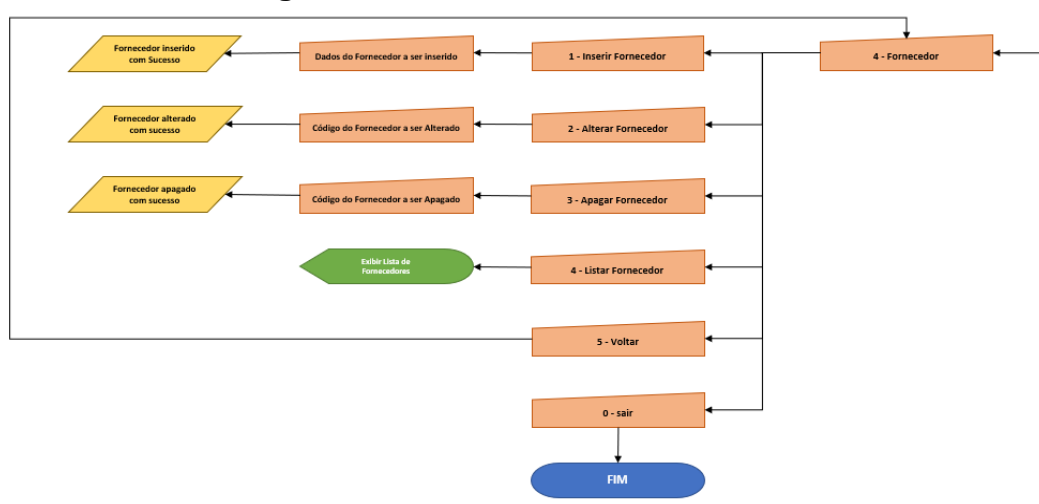
Na terceira etapa teremos **Funcionários**:

Figura 5: Dados de Cadastro de Funcionários



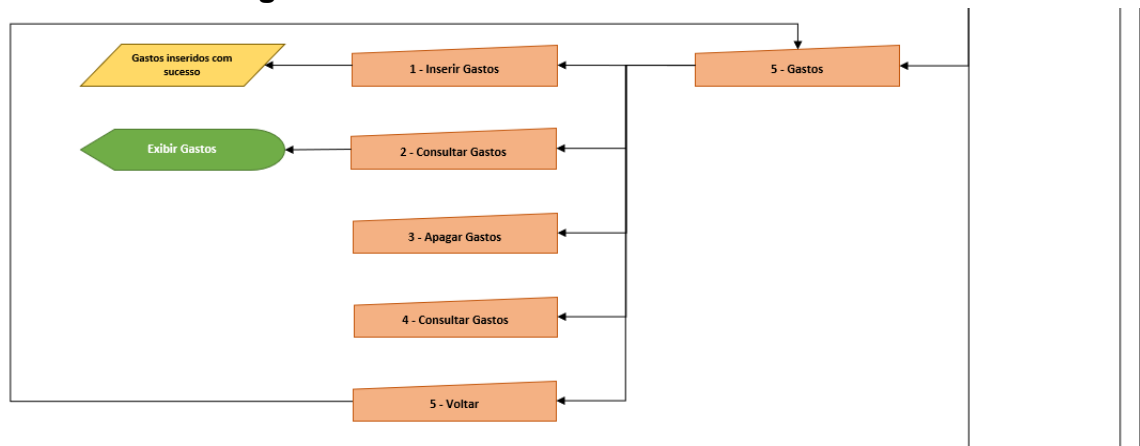
Na quarta etapa teremos **Fornecedor**:

Figura 6: Dados de Cadastro do Fornecedor



Na quinta etapa teremos **Gastos**:

Figura 7: Dados de Cadastro dos Gastos

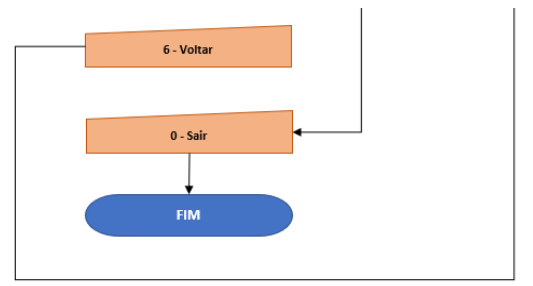


Em seguida teremos **Voltar e Sair**:

Voltar: Iremos p/ a tela de **Cadastros, Vendas, Relatórios, Opções e Sair**.

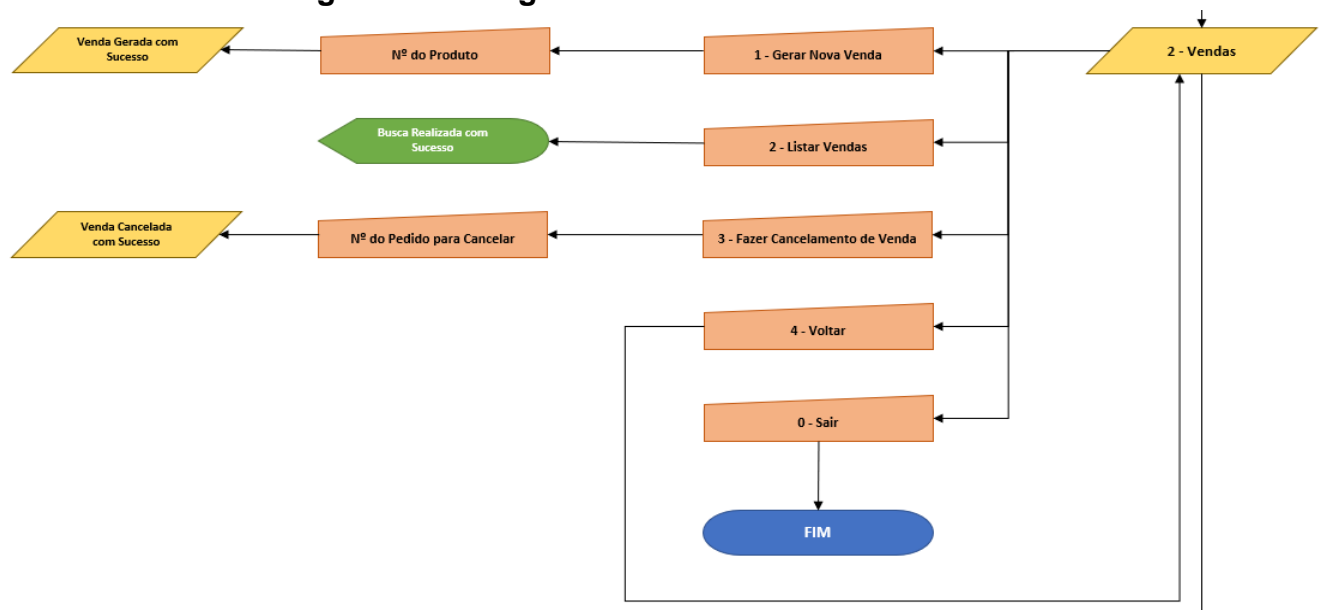
Sair: Ele irá encerrar o programa.

Figura 8: Opção de Voltar e Sair



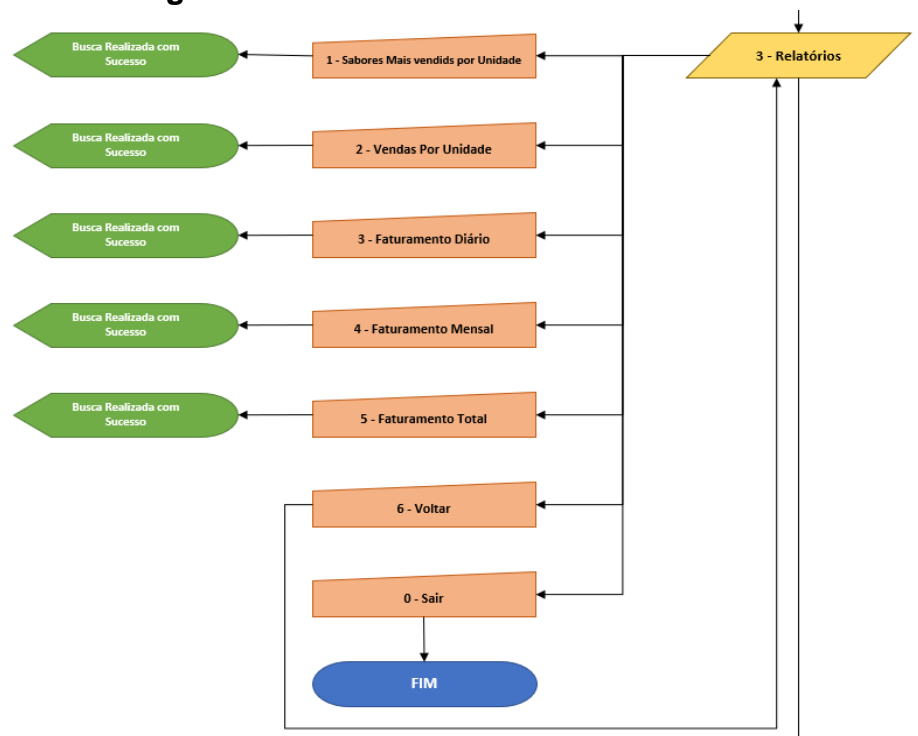
Agora passaremos para a etapa de **Vendas** e teremos as opções conforme imagem abaixo:

Figura 9: Fluxograma de Venda



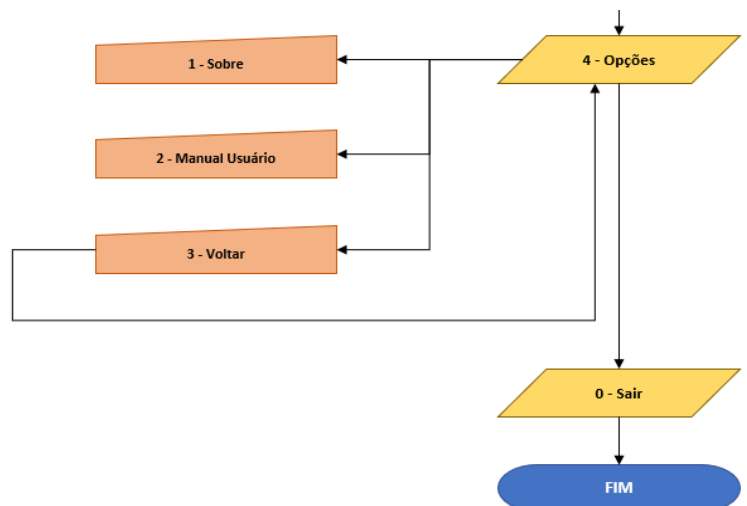
Agora passaremos para a etapa **Relatórios** e teremos as opções conforme imagem abaixo:

Figura 10: Fluxograma de Relatórios



Agora passaremos para a etapa **Opções e Sair** e teremos as opções conforme imagem abaixo:

Figura 11: Opções e Sair



E nessa parte encerramos as telas do usuário **“Administrador – MASTER”**.

Agora vamos para as telas do usuário **“Usuário Padrão – USER”**.

Continuando pela mesma lógica da primeira imagem, o usuário realiza o login, caso

ele insira a informação correta ele já é liberado as telas, caso contrário o programa solicita que o mesmo informe novamente o **LOGIN e SENHA**. Feito o login.

Nele teremos as telas iniciais de:

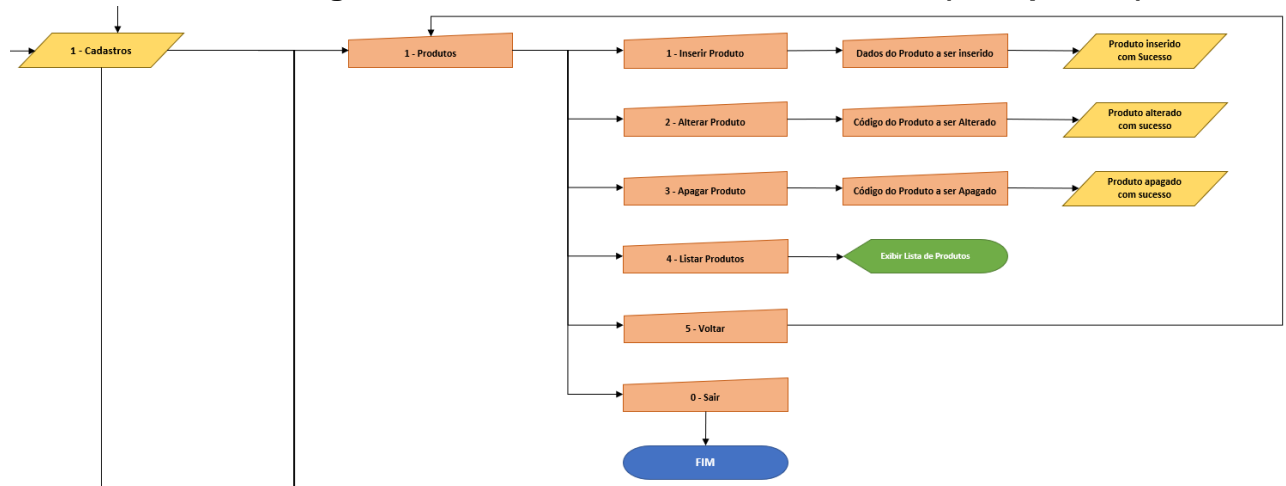
Cadastros
Vendas
Opções
Sair

Iniciamos por **Cadastros** teremos as seguintes opções:

Produtos
Clientes
Funcionários
Fornecedor
Voltar
Sair

E começaremos por **Produtos** e teremos as opções conforme imagem abaixo:

Figura 12: Dados de Cadastro do Produto (user padrão)



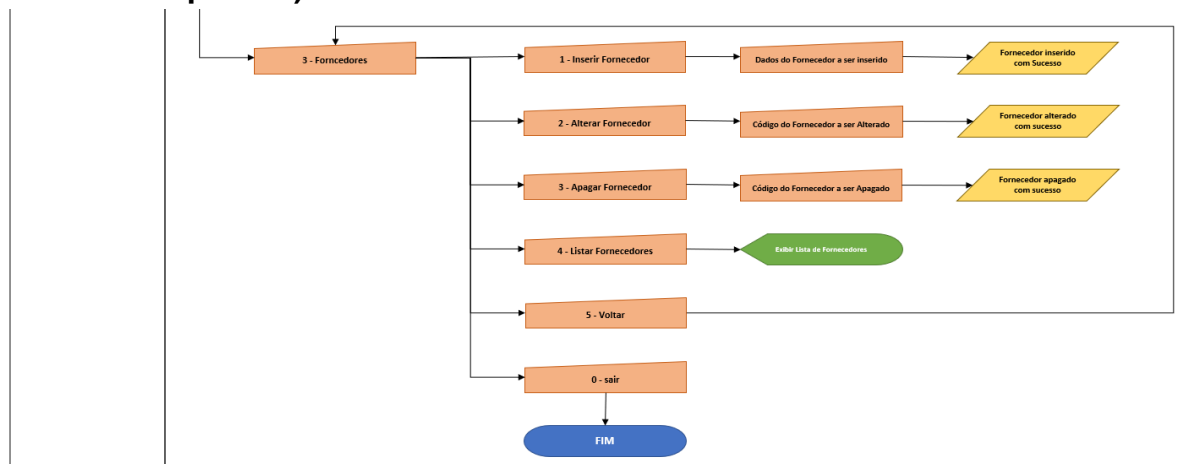
Na segunda etapa teremos **Clientes** e teremos as opções:

Figura 13: Dados de Cadastro do Cliente (user padrão)



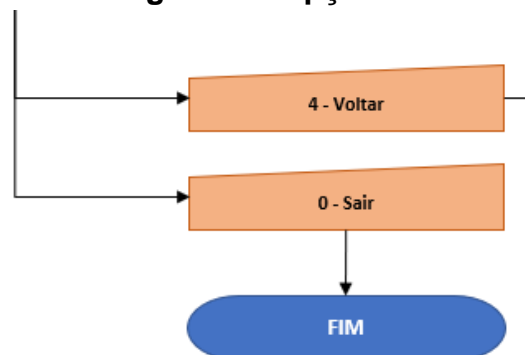
Na terceira etapa teremos **Fornecedor**:

Figura 14: Dados de Cadastro do Fornecedor (user padrão)



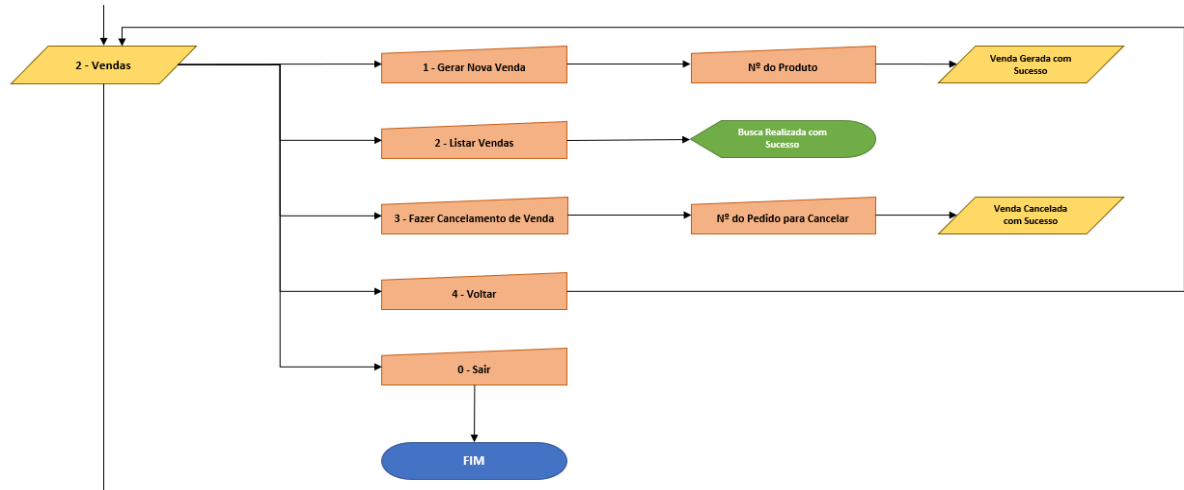
Agora passaremos para a etapa **Voltar e Sair** e teremos as opções conforme imagem abaixo:

Figura 15: Opção de voltar e sair (user padrão)



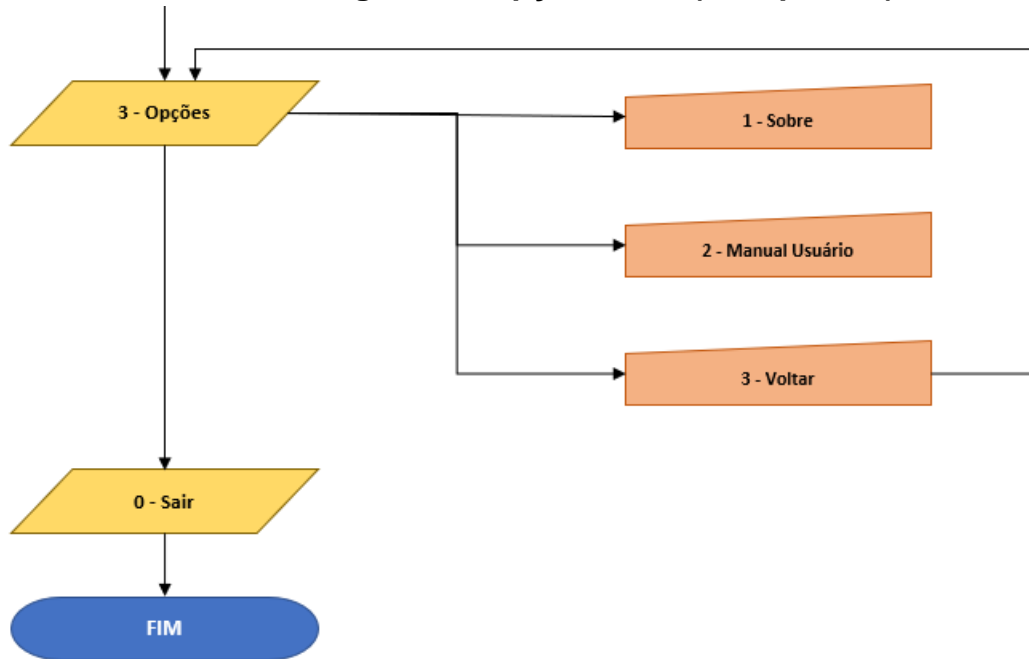
Agora passaremos para a etapa de **Vendas** e teremos as opções conforme imagem abaixo:

Figura 16: Fluxograma de Venda (user padrão)



Agora passaremos para a etapa de **Opções e Sair** e teremos as opções conforme imagem abaixo:

Figura 17: Opção e Sair (user padrão)



E assim encerramos nosso fluxograma.

3.3 APLICAÇÃO DA ENGENHARIA DE SOFTWARE

A engenharia de software baseada em conhecimento técnico, é estipulada para o desenvolvimento de um sistema perfeito, ou seja, aquele que obtém todos os meios de controle para diversas atividades que ele enfrentará em determinados ambientes de implantação. Contudo a Engenharia de Software é uma área da computação voltada à especificação, desenvolvimento e manutenção de sistemas

de software, com aplicação de tecnologias e práticas de gerenciamento de projetos e outras disciplinas, visando organização, produtividade e qualidade.

3.3.1 ESCOPO

Desenvolvemos um software para atender as necessidades de uma rede de pizzaria, caracterizando a conformidade de suas aplicações e viabilizando os melhores atendimentos conforme o seu desempenho e sua situação problema. Portanto, com a equipe de projeto, estabelecemos

3.3.2 ENGENHARIA DE REQUISITOS

Declarado anteriormente, conseguimos adquirir informações consistentes através de discussões e pontos relativos ao desenvolvimento do sistema, levantando as seguintes abordagens de requisitos.

Figura 18: Requisitos de software

REQUISITOS DE SOFTWARE

Uma melhor viabilidade do sistema



Fonte: Elaboração própria

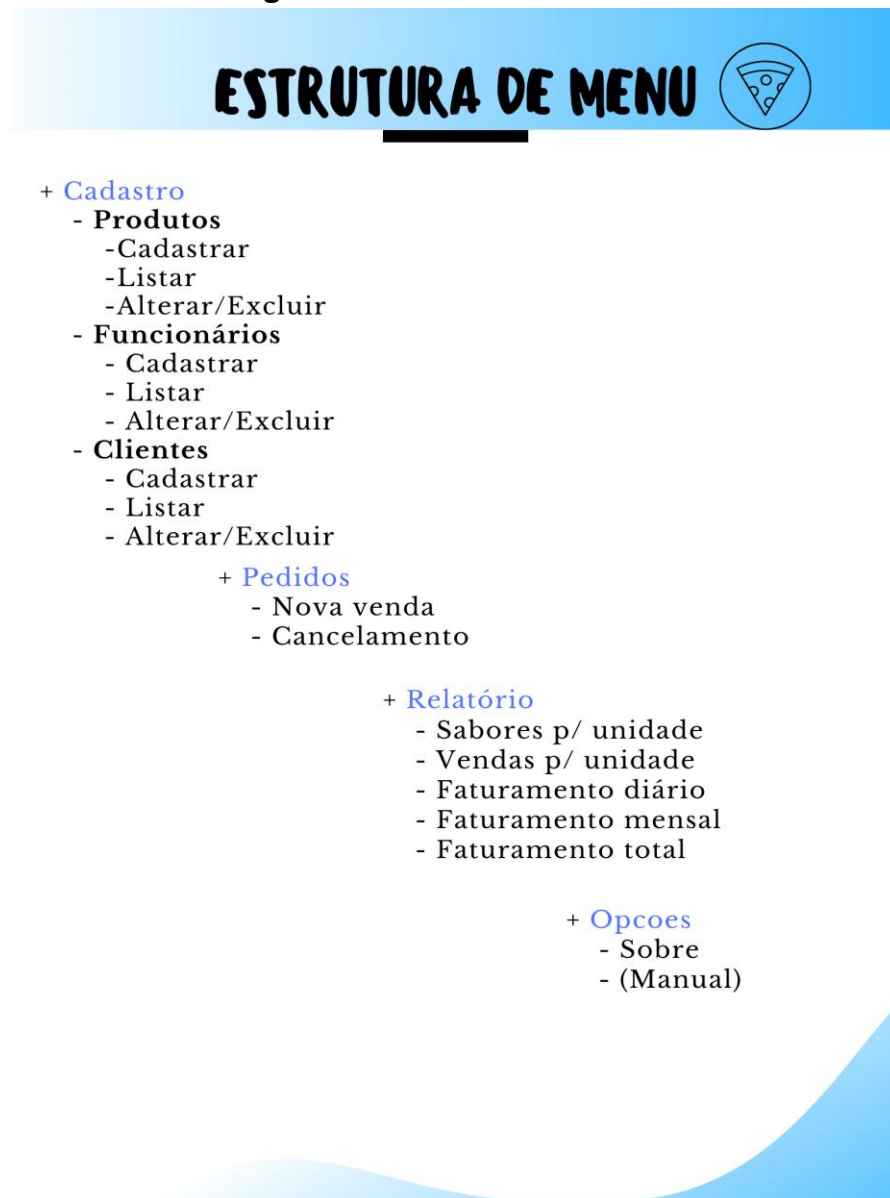
3.4 JUSTIFICATIVA DO MODELO DE PROCESSO

Metodologia estabelecida: Prototipação

Estabelecemos o modelo de prototipação com intuito de atingirmos os objetivos com base em seu ciclo de criação e seu desenvolvimento, portanto, identificamos as etapas de processo que são fundamentais na prototipação e incluímos em nosso projeto, deixando claro e viável para a concepção dos demais colaboradores e envolvidos no planejamento.

3.4.1 IDENTIFICAÇÃO E LEVANTAMENTO DE REQUISITOS

Figura 19: Estrutura de Menu



Fonte: Elaboração própria

3.4.2 CONSTRUÇÃO DO PROTÓTIPO

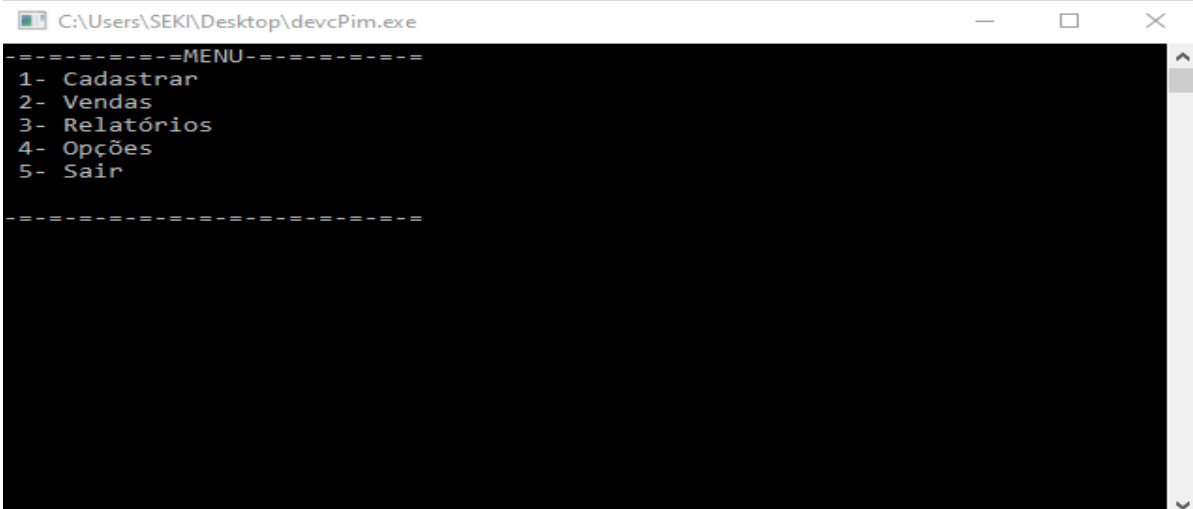
Protótipo é o termo usado para se referir ao que foi criado pela primeira vez, servindo de modelo ou molde para futuras produções, dito isso, ele poderá restituir para um esboço mais específico e detalhado.

Para enfatizar a construção do nosso projeto, esquematizamos um protótipo tornando uma melhor visualização na elaboração de nosso sistema. Definimos um molde para o envolvimento da rapidez e da economia de um experimento de um projeto. Por conseguinte, observamos cada peça do nosso protótipo e apropriamos o que poderia ser aperfeiçoado ou retirado, através do desenvolvimento e da confecção elaboramos as etapas técnicas para a evolução do projeto.

É relevante esclarecer que o nosso molde tem como objetivo ser um protótipo evolutivo, ou seja, a cada instante que construímos ou adicionamos uma especificação/opção, o produto vai progredindo.

Pode-se observar abaixo um exemplo do nosso protótipo baseado na construção do nosso sistema de rede para uma pizzaria. Portanto, adicionamos as exigências que um sistema deve obter.

Figura 20: Protótipo tela de menu



```
C:\Users\SEKI\Desktop\devcPim.exe
-----MENU-----
1- Cadastrar
2- Vendas
3- Relatórios
4- Opções
5- Sair
-----
```

Figura 21: Protótipo tela de cadastro

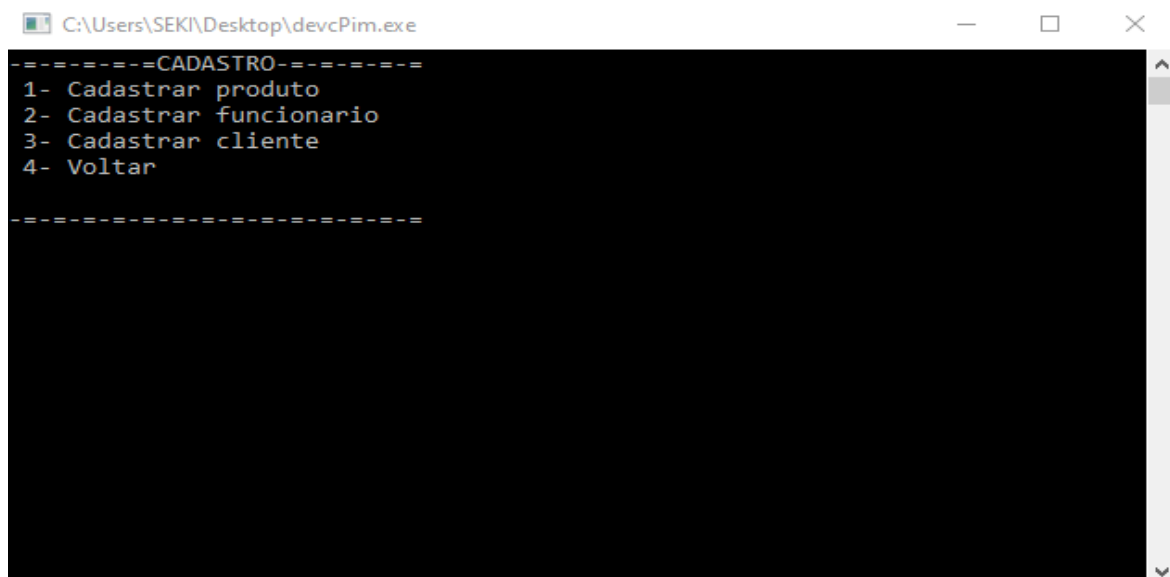


Figura 22: Protótipo tela de venda

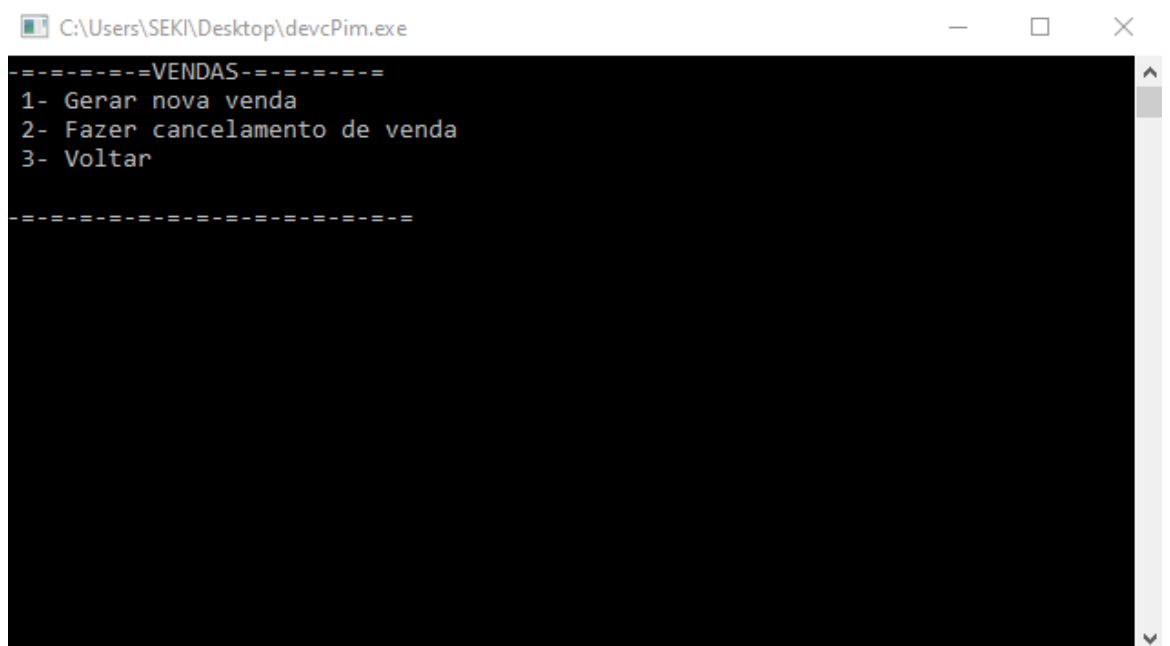


Figura 23: Protótipo tela de relatórios

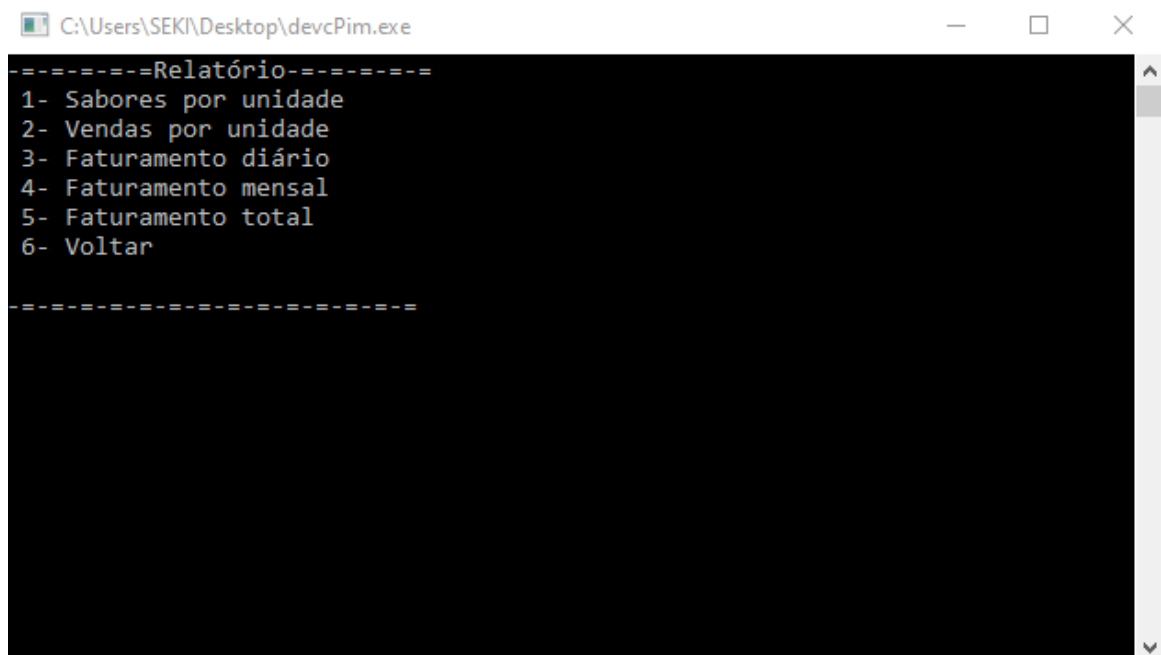
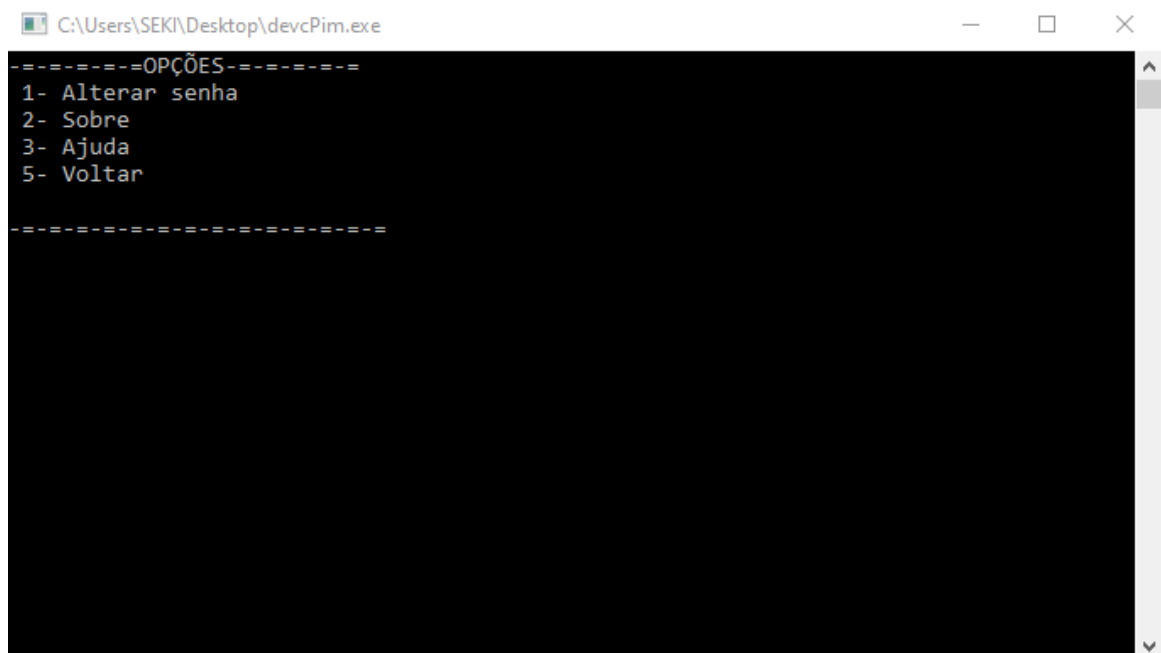


Figura 24: Protótipo tela de opções



Fonte: Elaboração própria

3.5 DESCRIÇÃO DO CACULO DE APOIO AOS RELATÓRIOS

Os cálculos utilizados para faturamento por unidade são os seguintes:

Figura 25: Calculo por unidade

```
case '2':
    system("cls");
    while((opcao=FuncoesMenuRelatorios(MenuGastosUnidade))!= SAIR)
    switch(opcao){
        case '1':
            AbrirArquivoGastos1();
            TotalGastos1=CalcularGasto();
            fclose(fp);
            AbrirArquivoVenda1();
            TotalVenda1=CalcularVenda();
            fclose(fp);
            TotalUnidade1=TotalVenda1-(TotalGastos1);
            system("cls");
            printf("\n0 faturamento da unidade1 é de: %.2f\n\n",TotalUnidade1);
            system("pause");
            system("cls");
            break;
```

O faturamento da unidade é a venda subtraído pelos gastos das unidades. Já o faturamento da rede são a soma de todas as unidades.

Figura 26: Calculo da rede

```
case '3':
    TotalRede=TotalUnidade1+TotalUnidade2+TotalUnidade3;
    printf("\n0 faturamento da rede é de: %.2f\n\n",TotalRede);
    system("pause");
    system("cls");
    break;
```

Fonte: Elaboração própria

3.6 DIAGRAMA DA REPRESENTAÇÃO DA REDE DE COMUNICAÇÃO

O diagrama de redes representa o sequenciamento lógico-temporal dos elementos terminais de um projeto a serem concluídos, fazendo um estudo entre os elementos terminais e suas dependências – nada mais do que colocar todas as atividades a serem executadas uma obra e relacioná-las em uma folha de papel. Enquanto uma EAP nos oferece uma visualização do “todo” decomposto em “partes”, o Diagrama de Redes nos entrega uma visualização do “antes” e do “depois”, facilitando a identificação de limitações de trabalho, de modo a não possuir relações circulares ou redundantes entre cada atividade. Este método, por sua grande disseminação, possui diferentes formas de representação, com isso, um diagrama de rede é uma representação visual de uma rede de computadores ou telecomunicações. Ele mostra os componentes que constituem uma rede e como eles interagem, incluindo roteadores, dispositivos, hubs, firewalls etc.

Os diagramas a seguir mostram a representação na prática de três unidades de uma pizzaria, isso para um melhor entendimento de como funciona a rede e uma visão mais ampla sobre a pizzaria. (Victor, 2016)

3.6.1 ESPECIFICAÇÃO DIAGRAMA DE REDES

O projeto tem o download disponibilizado na internet, dito isso, ele passa pelo Firewall até chegar ao servidor dedicado que está hospedado a pizzaria, onde ele irá escolher que tipo de sistema operacional será instalado e qual tipo de hardware, além disso, o servidor dedicado contém algumas funcionalidades de grande ajuda ao cliente. Atualizações do sistema operacional; Atualizações para os aplicativos instalados; Monitoramento de servidor e aplicativos; Firewall de manutenção; Detecção de intrusão; Cópias de segurança de dados; Recuperação de desastres.

Contudo, os dados vão para o *Switch* e através de um cabo de par trançado passam a ser disponibilizados para o *Desktop* e ao telefone, onde o funcionário irá trabalhar normalmente. (Qostecnologia, 2015)

Figura 27: Primeira unidade

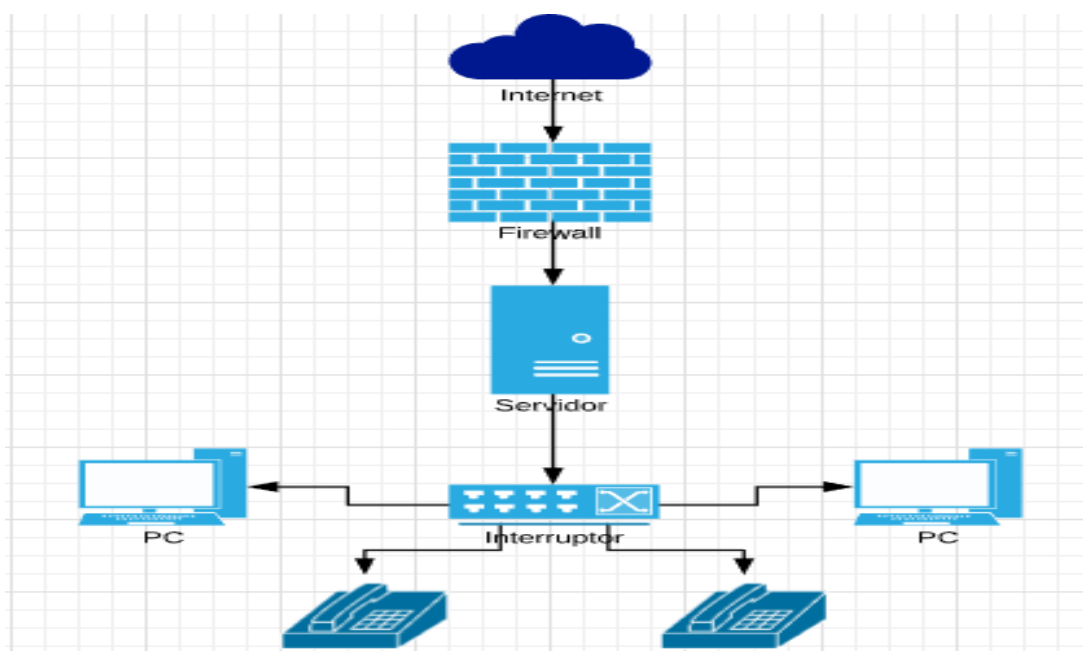


Figura 28: Segunda unidade

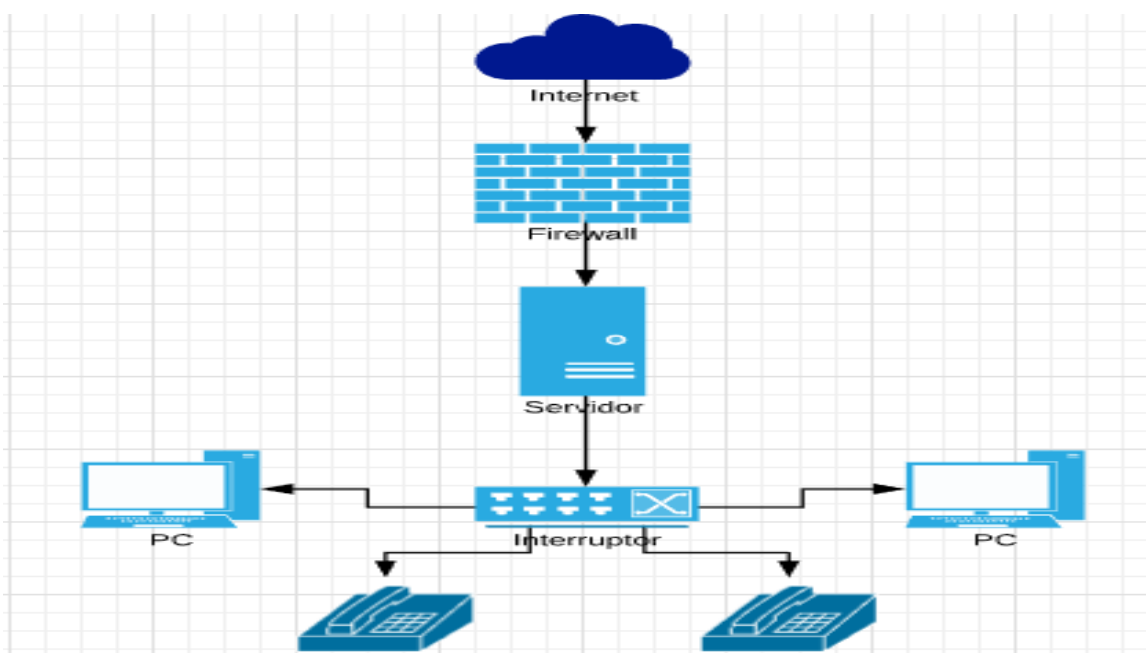
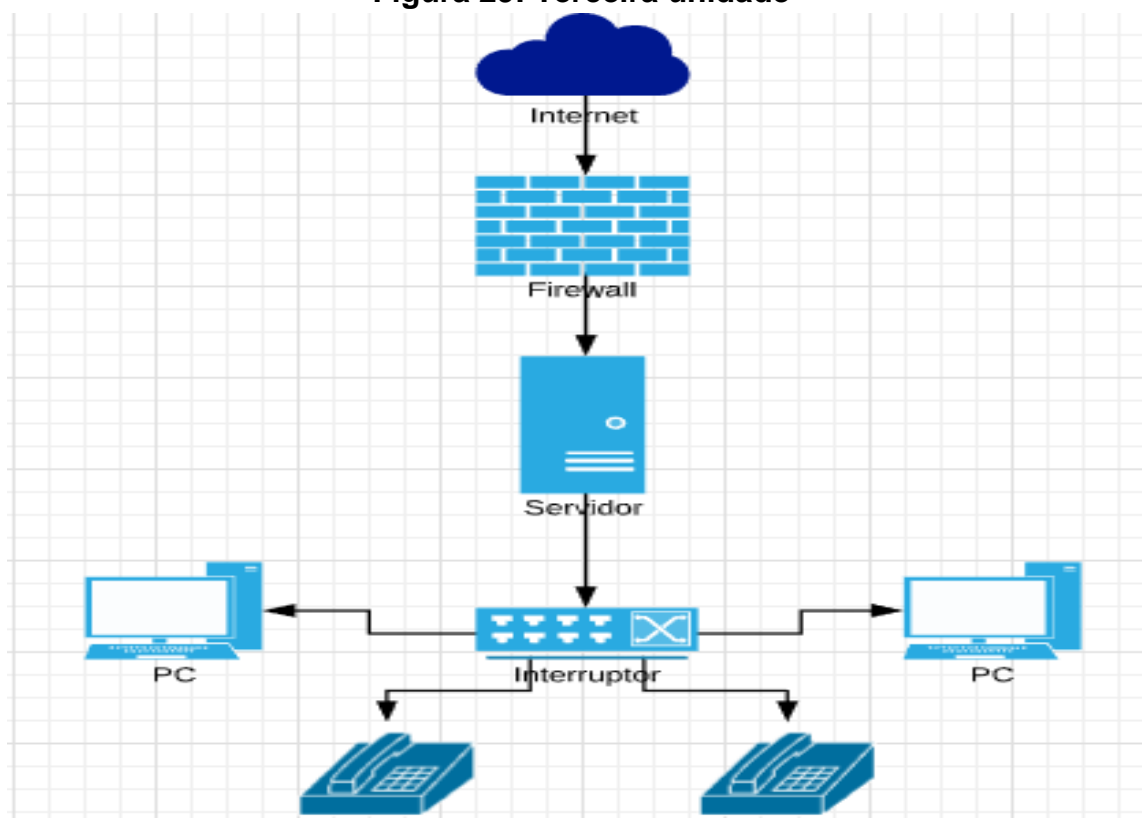


Figura 29: Terceira unidade



Fonte: Elaboração própria

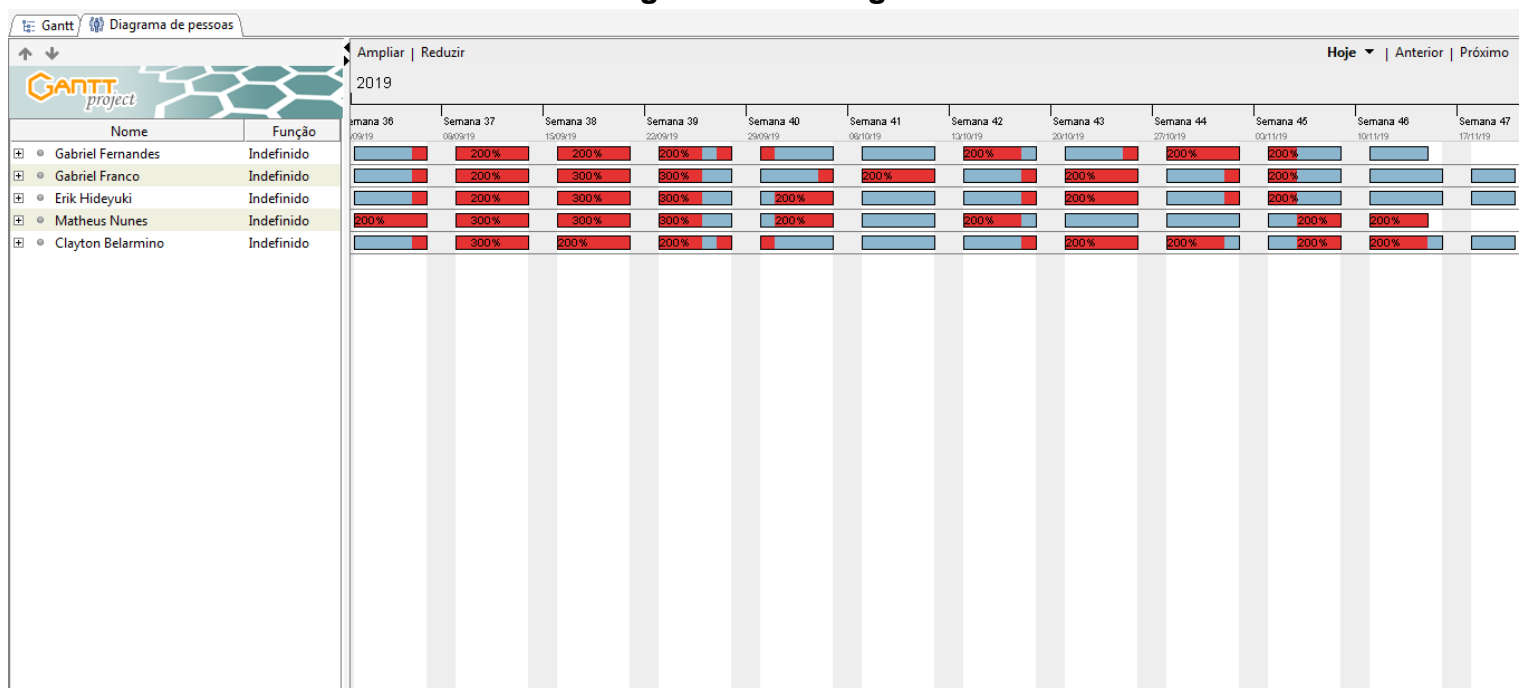
3.7 CRONOGRAMA DE DESENVOLVIMENTO E IMPLANTAÇÃO

O Cronograma de Projetos é uma ferramenta bem mais conhecida — inclusive pelos profissionais com pouco conhecimento das técnicas de gerenciamento de projetos. Pode ser entendido como uma matriz que revela graficamente para cada item da EAP, em uma escala de tempo, o período que deve ser realizado.

A duração é o intervalo entre o início e o término de uma tarefa específica, sem levar em conta o número de pessoas necessárias para que isso aconteça.

O projeto define o Cronograma de Projetos como uma ferramenta de comunicação que demonstra todo o trabalho que precisa ser feito, quais os recursos da organização que serão empregados e quais os prazos que precisam ser cumpridos para que este trabalho seja realizado. Ele deve prever e refletir todos os esforços para a entrega do projeto finalizado. (Santos, 2014)

Figura 30: Cronograma



Fonte: Elaboração própria

4 IMPLANTAÇÃO DO SISTEMA

Evidenciando a implantação do sistema de como aplicamos e desenvolvemos com base em seu ciclo de vida, o projeto técnico obteve pontos significativos em seu processo, mapeamento, entre outros fatores. Portanto, trabalhamos e elaboramos de forma eficiente para o desempenho do produto em equipe, deixando claro e objetivo todas as etapas de ação e também todo conhecimento que buscamos e aplicamos. Assim sendo, toda as ideias, metas, abordagens, resultados e atividades designadas a este projeto serão organizados e apresentados neste documento, trazendo entendimento detalhado.

4.1 DESCRIÇÃO DO PROCESSO DE IMPLANTAÇÃO

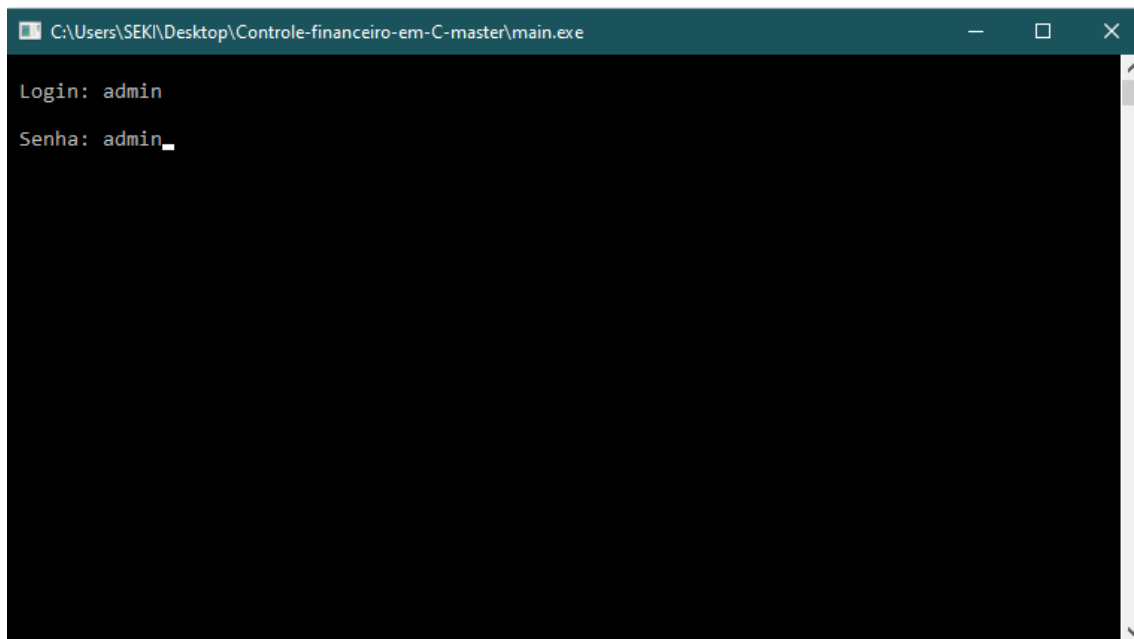
Para uma melhor identificação em cada tarefa, separamos os pontos importantes, trazendo uma melhor viabilidade do que o programa é constituído.

4.1.1 DETALHES DO LOGIN

Criamos o software com objetivo de controle e gerenciamento, contudo, há alguns critérios que necessitarão para o manuseio deste, opções de auxílio e gestão, dito isso, em nosso projeto, possuirá dois logins (processo para acessar um sistema informático), um login para usuário comum e outro para administrador.

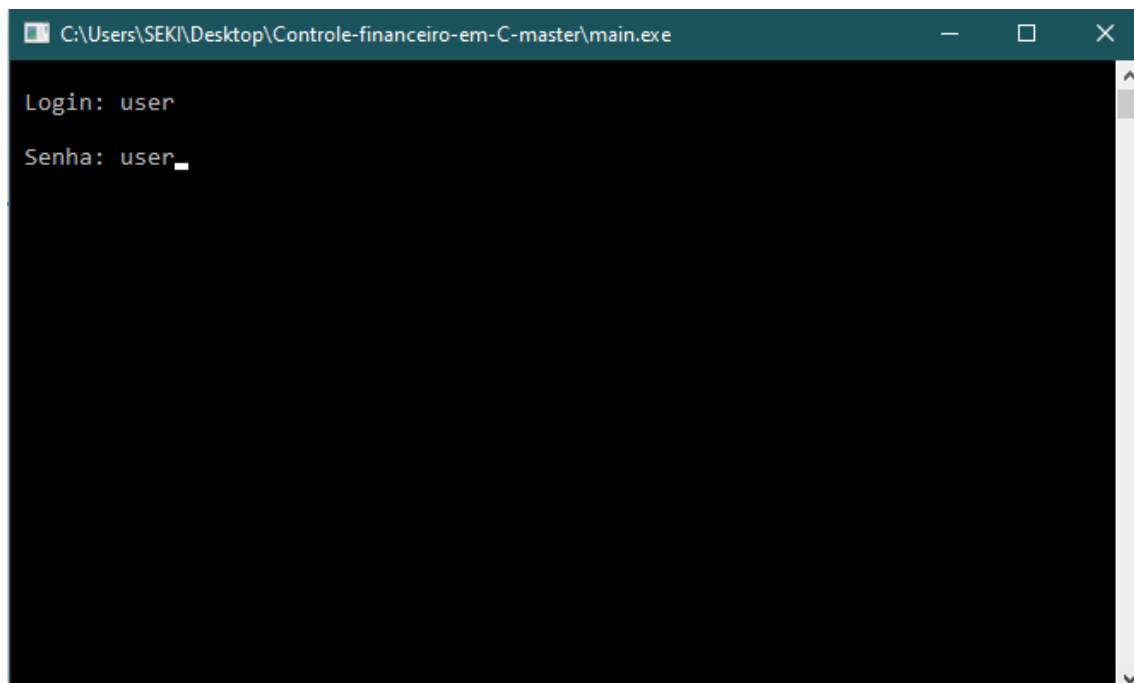
Já o login de admin terá acesso total nas opções do sistema que um usuário comum não conseguiria manusear, tal como relatórios e o cadastro de gastos que serão salvos no arquivo texto.

Figura 31: Tela de Login (ADMIN)



O login de usuário padrão estará utilizando algumas funções de Cadastro de produto, fornecedor, cliente e opções. Desse modo, haverá restrições para o indivíduo não obter a possibilidade de manipular registros importantes no sistema.

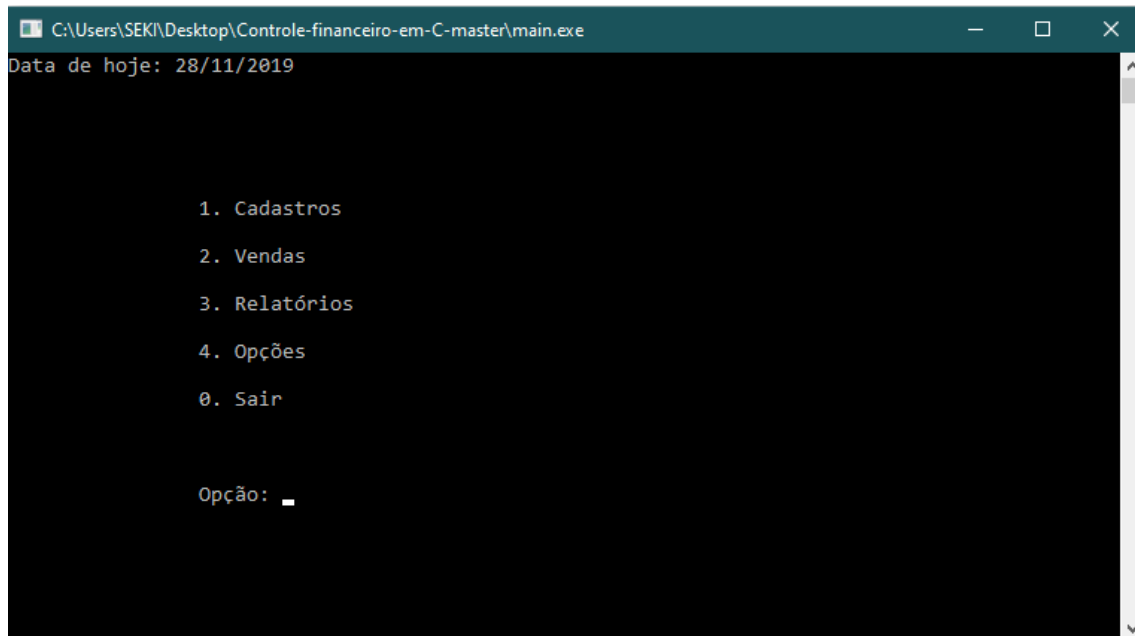
Figura 32: Tela de login (usuário padrão)



4.1.1 TELA DE MENU

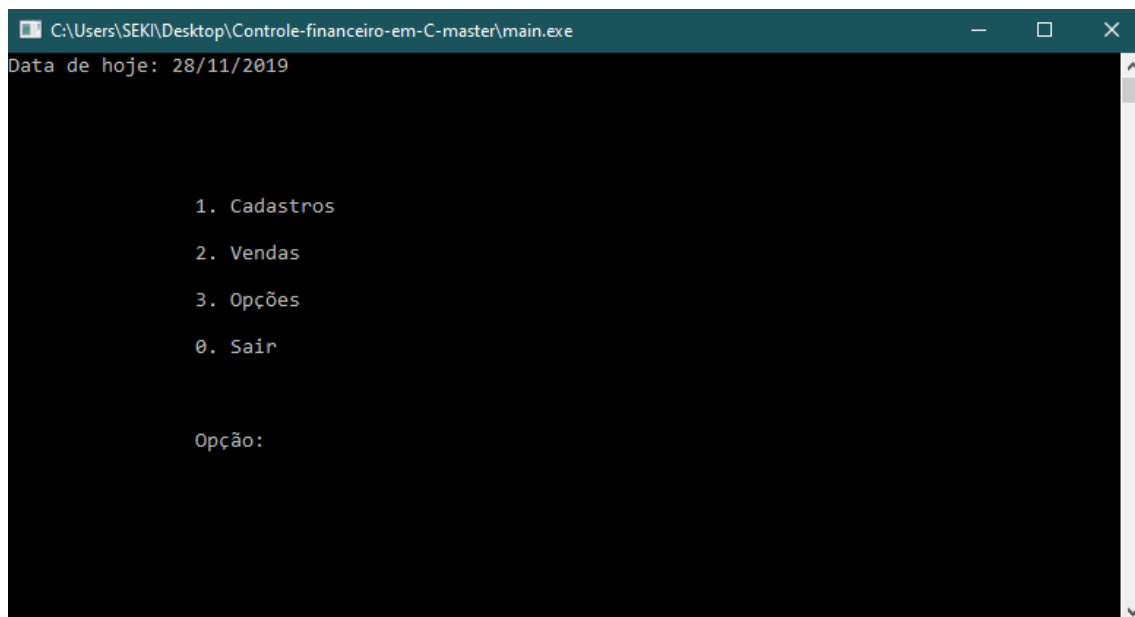
Logo após inserir as informações no login, sendo usuário padrão ou admin, o sistema encaminhará para a tela de Menu. Deste modo, a tela vai demonstrar algumas opções como:

Figura 33: Tela de menu



Percebe-se que na tela abaixo, o login de usuário padrão não tem a opção de relatórios.

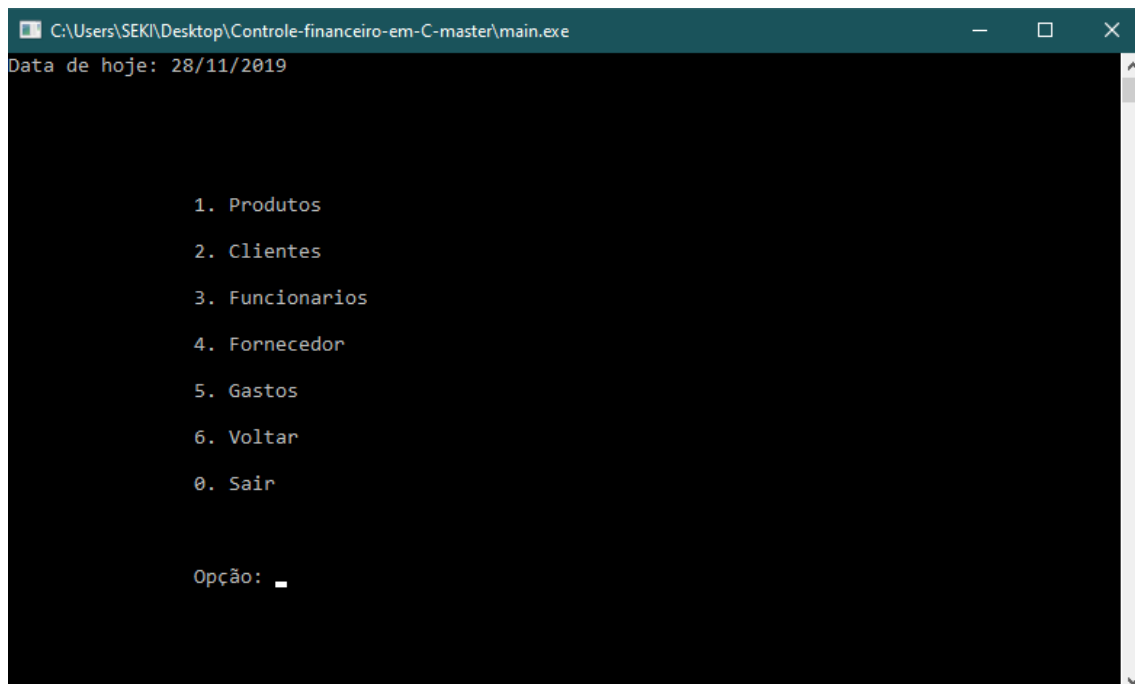
Figura 34: Tela de menu do usuário padrão



4.1.2 CADASTROS

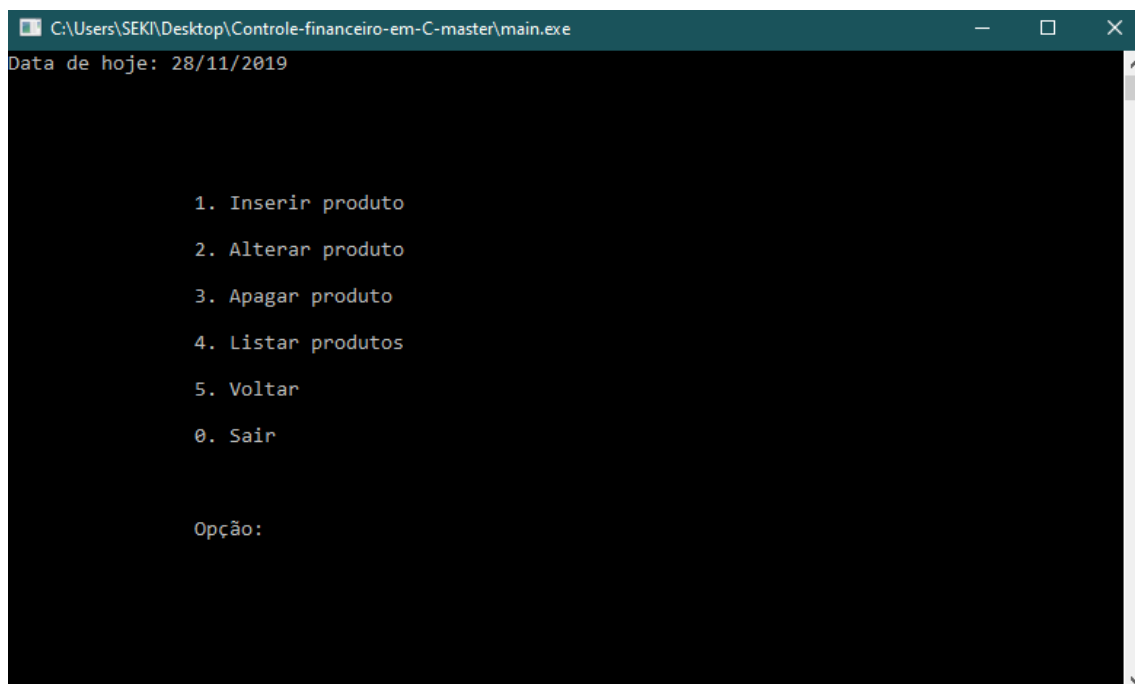
Optando pelo cadastro, o usuário estará selecionando toda forma de arquivamento de um determinado dado, ou seja, se houver a necessidade de registrar, o sistema oferecerá diversas atividades que o ajudarão para concluir seu objetivo. Portanto, as inúmeras escolhas serão de :

Figura 35: Tela de Cadastro



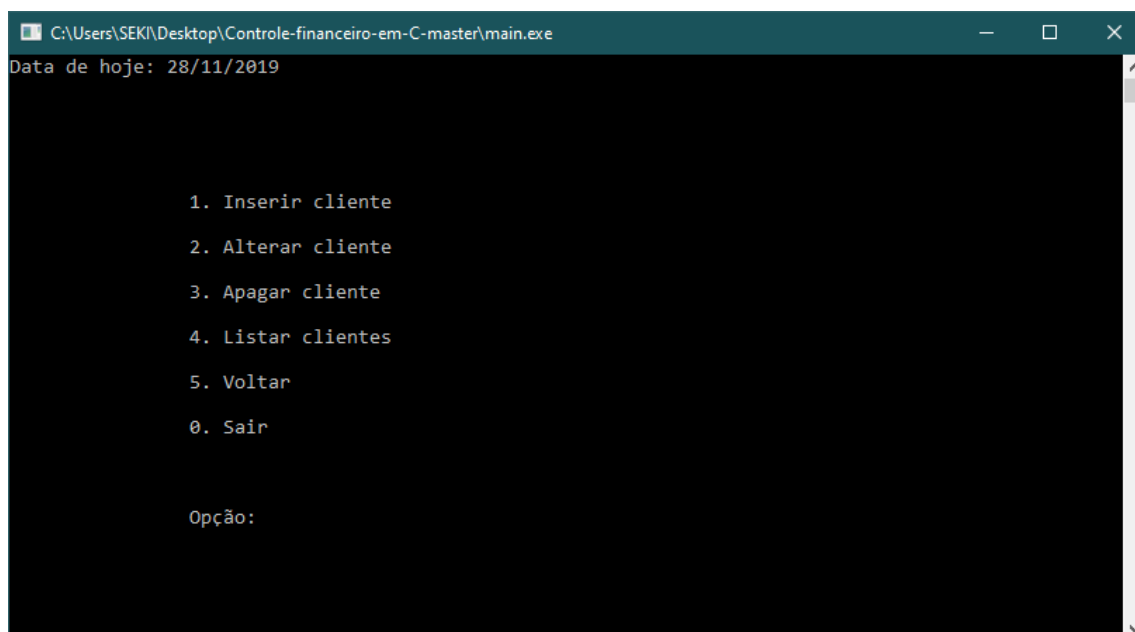
Cadastro de produto: onde realizará as etapas para se registrar um determinado produto, validando seu ID, quantidade, valor de custo, valor de venda, valor de promoção (caso não haja, poderá registrar o valor de venda igualmente).

Figura 36: Tela de cadastro do produto



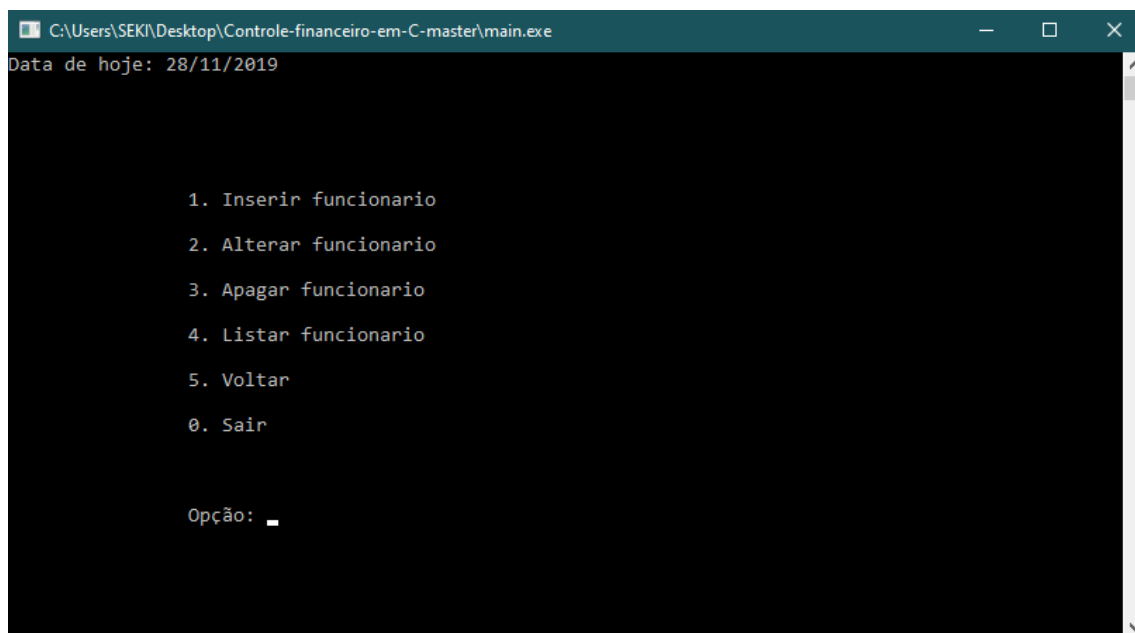
Cadastro de clientes: nome do cliente, CPF, RG, número de celular, e-mail, endereço e observações relativos ao atendimento da pizzeria.

Figura 37: Tela de Cadastro dos Clientes



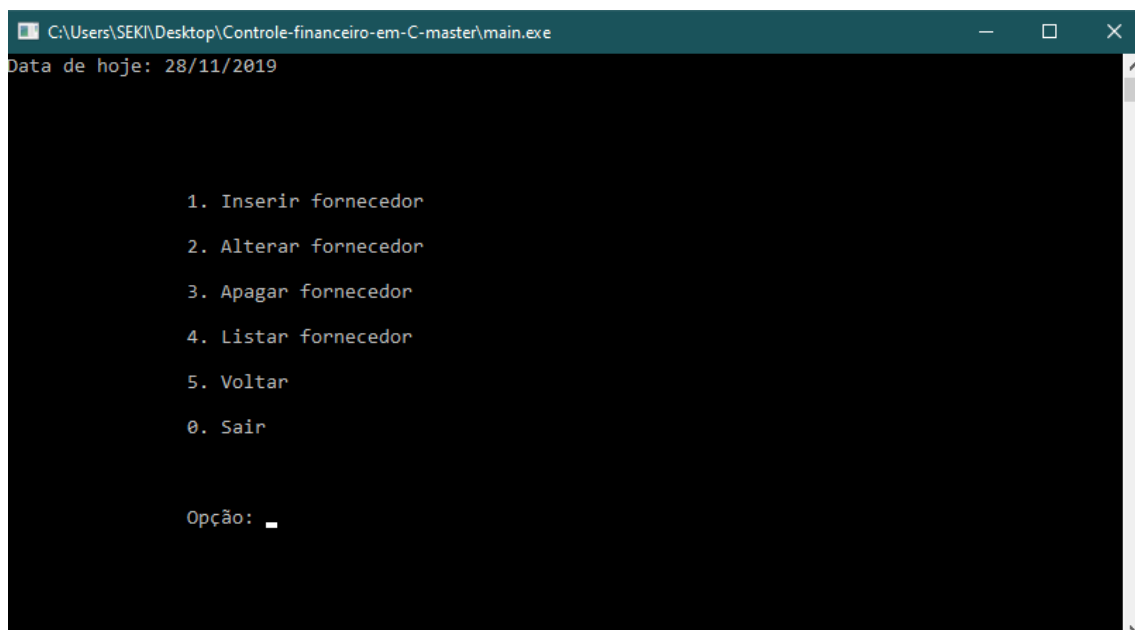
Cadastro de funcionários: nome do funcionário, CPF, RG, número de celular, e-mail, endereço e o seu salário adjacente.

Figura 38: Tela de Cadastro dos Funcionários



Cadastro de fornecedores: nome do fornecedor, CPF ou CNPJ, RG, número de celular, e-mail, endereço e observações.

Figura 39: Tela de Cadastro dos Fornecedores



Cadastro dos gastos: terá opções de adicionar gastos com base na unidade desejada, alterar, apagar ou consultar.

Figura 40: Tela de Cadastro dos Gastos



Após selecionar a opção de consultar os gastos, o usuário poderá optar por escolher qual gastos por unidade ele deseja ver. No exemplo abaixo foi escolhida a unidade 1 e seu resultado de gastos em diversas categorias.

Figura 41: Consultar os Gastos

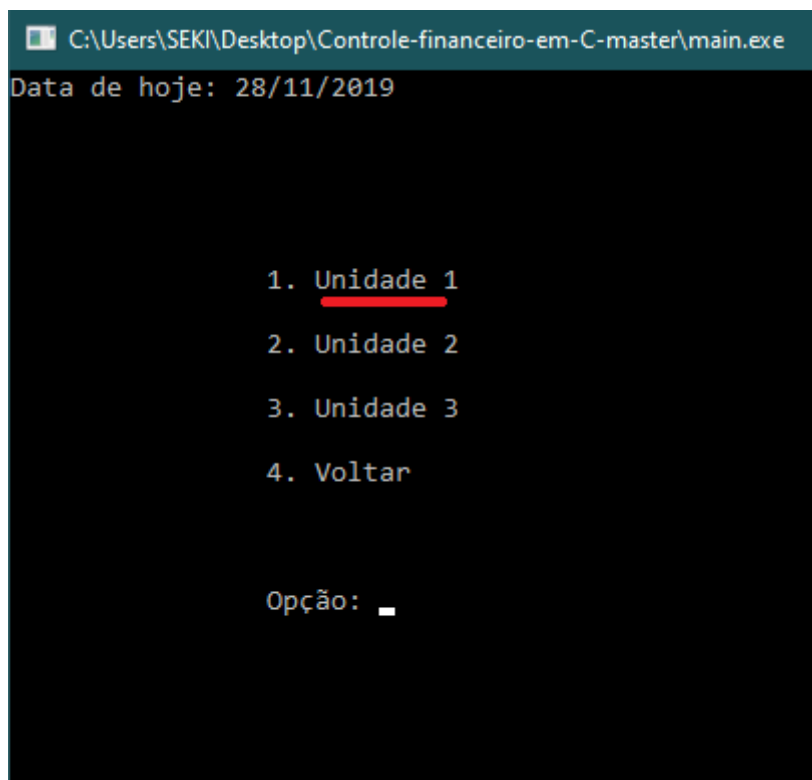
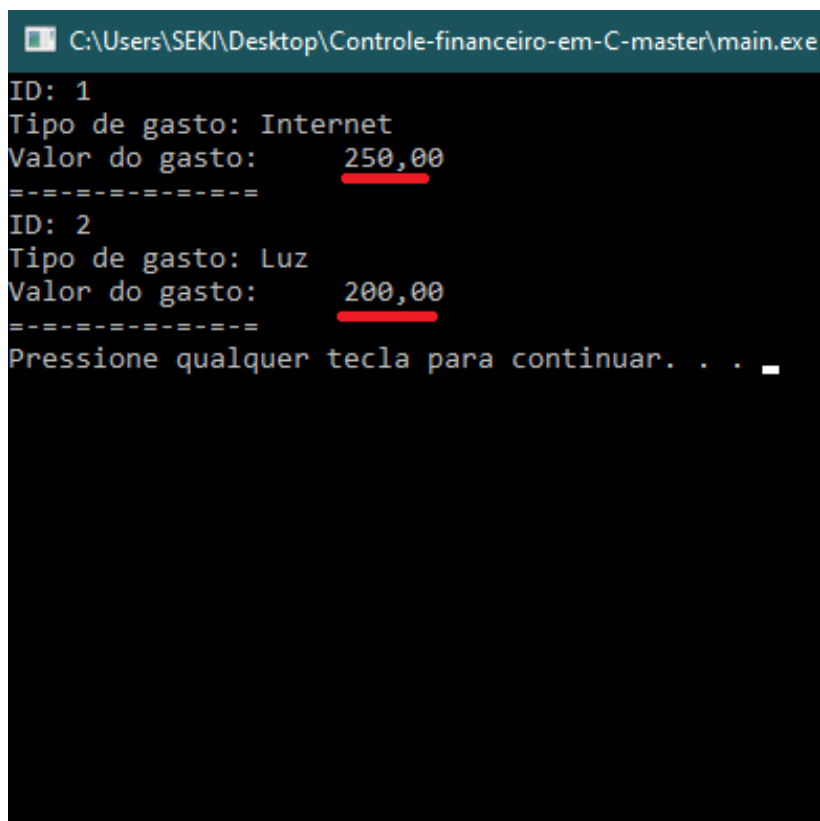


Figura 42: Resultado do gastos da unidade 1



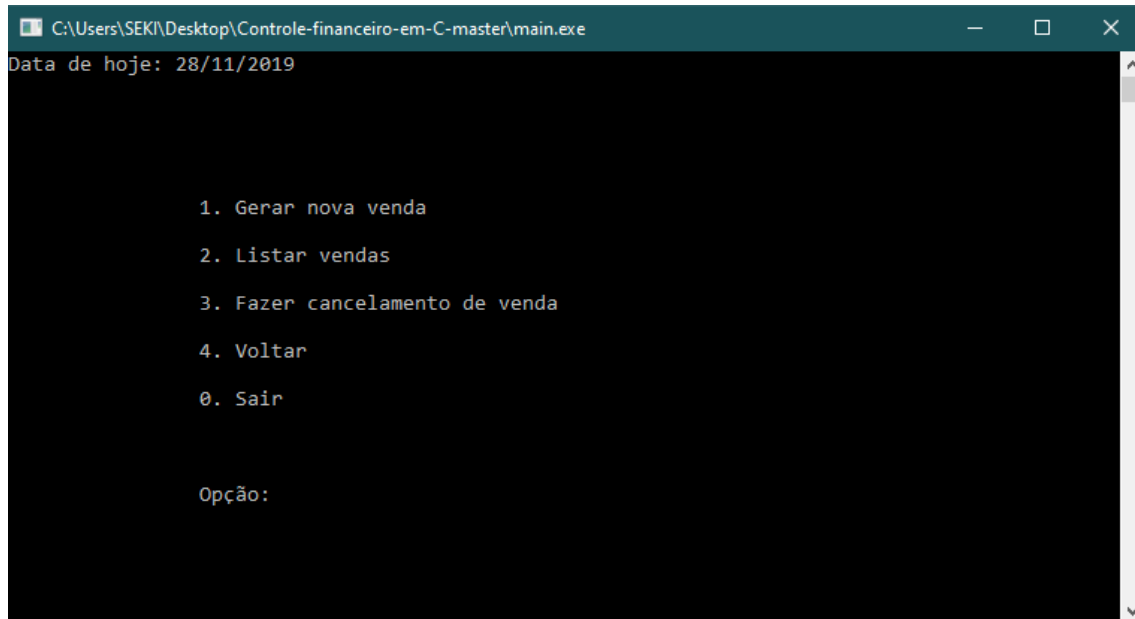
```
C:\Users\SEKI\Desktop\Controle-financeiro-em-C-master\main.exe
ID: 1
Tipo de gasto: Internet
Valor do gasto: 250,00
=====
ID: 2
Tipo de gasto: Luz
Valor do gasto: 200,00
=====
Pressione qualquer tecla para continuar. . . _
```

Gastos de 450,00.

4.1.3 VENDAS

Nesta etapa, estará apresentando o componente de Vendas, ou seja, receber algumas alternativas de gerar uma nova venda, listar vendas por unidade ou fazer o cancelamento de tal.

Figura 43: Tela de Vendas



Após optar por gerar uma nova venda, o sistema pedirá para você solicitar o número do produto, cliente e a quantidade, dito isso, dependendo das suas preferências, vai surgir as informações daquilo que o usuário escolheu. No final, ele apresentará o resultado e o valor gerado pela suas escolhas. As figuras de exemplos estão logo abaixo.

Figura 44: Tela de gerar nova venda

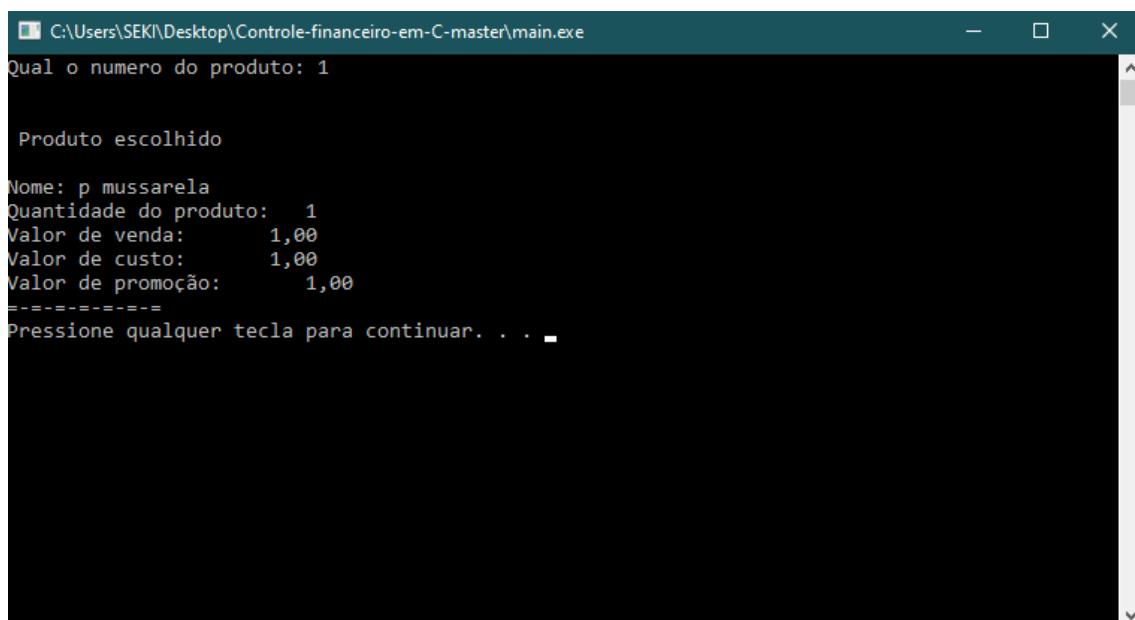


Figura 45: Tela de gerar nova venda (2)

```
C:\Users\SEKI\Desktop\Controle-financeiro-em-C-master\main.exe
Qual o numero do cliente: 1

Cliente escolhido
Nome: Gabriel
CPF: 123123123
RG: 12321323
Celular: 123213
Email: 123231123@311kjal.om
Endereço: odsnhadnsoa
Observação: oiasdpasd
=====
Pressione qualquer tecla para continuar. . . _
```

Figura 46: Tela de Resultado de venda

```
C:\Users\SEKI\Desktop\Controle-financeiro-em-C-master\main.exe
=====RESUMO=====
Cliente: Gabriel
=====
Produto: p mussarela
Valor unitário: 1
Quantidade: 1
=====
Total: 1
=====
Pressione qualquer tecla para continuar. . . _
```

← Resultado

No final, após inserir toda as escolhas e informações, ele vai perguntar se realmente o usuário quer salvar ou não.

Portanto, escolhendo a opção de Listar Venda, abrirá uma nova tela dizendo as informações de Venda de um determinado produto e de uma certa Unidade.

Figura 47: Tela de listar venda

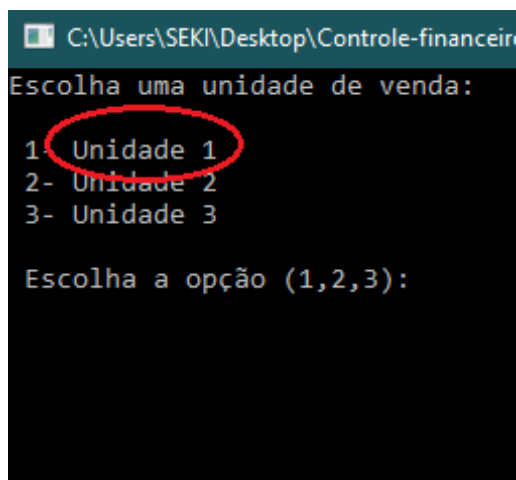
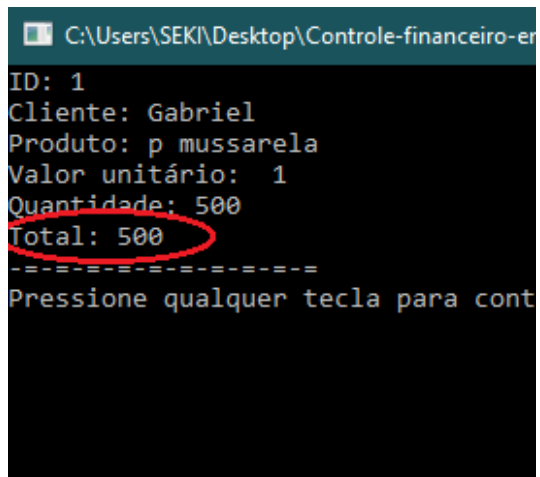
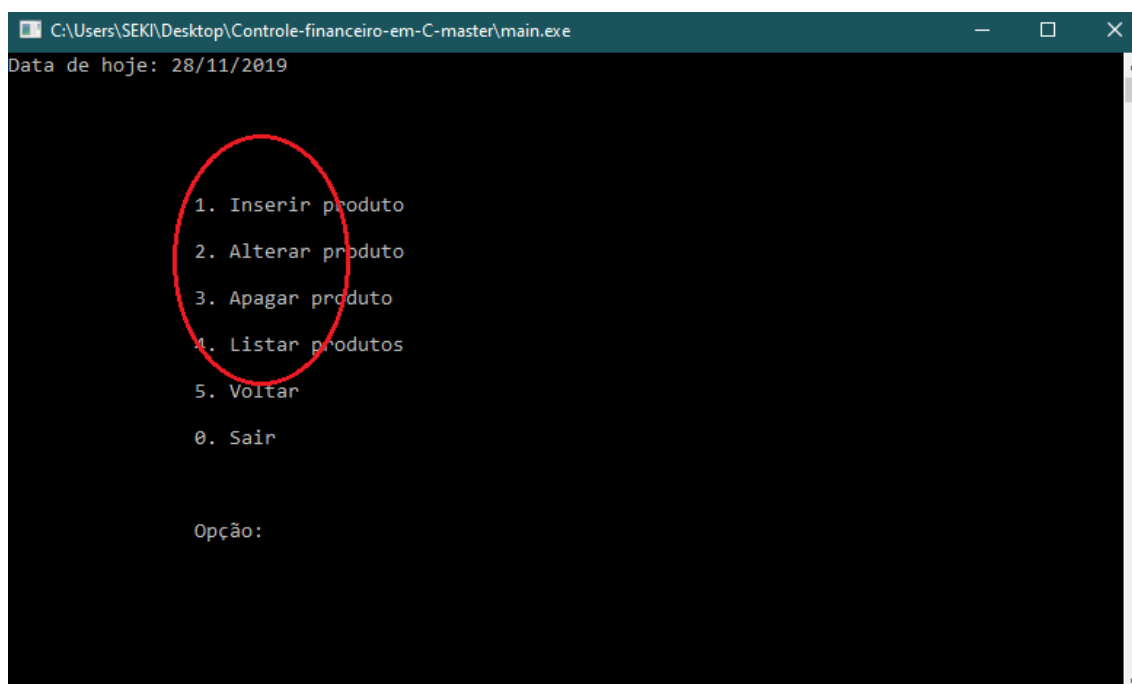


Figura 48: Tela de resultado de listar venda



É importante citar que as opções de cadastro e vendas permanecerão organizadas para inserir, alterar, apagar ou listar, caso ocorra algum erro ou preferência no momento de inserção.

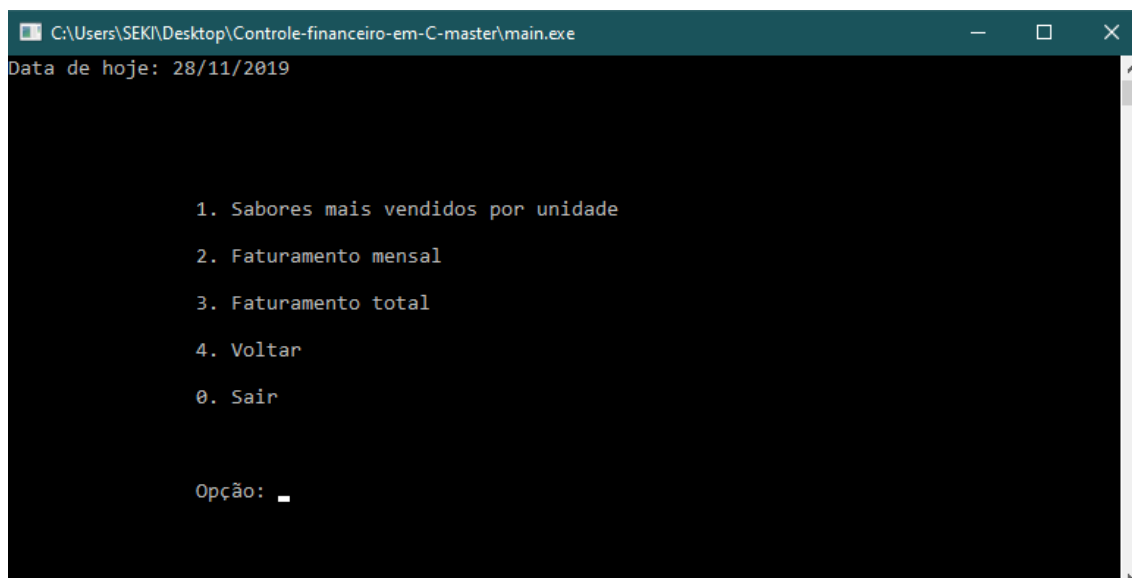
Figura 49: Inserir, Alterar, Apagar e Listar



4.1.4 RELATÓRIOS

Na tela de Relatórios, consequentemente, disponibilizará os dados e as informações propostas pelos registros colocados em cadastros de produtos e vendas.

Figura 50: Tela de Relatórios



Sabores mais vendidos por unidade

Faturamento mensal

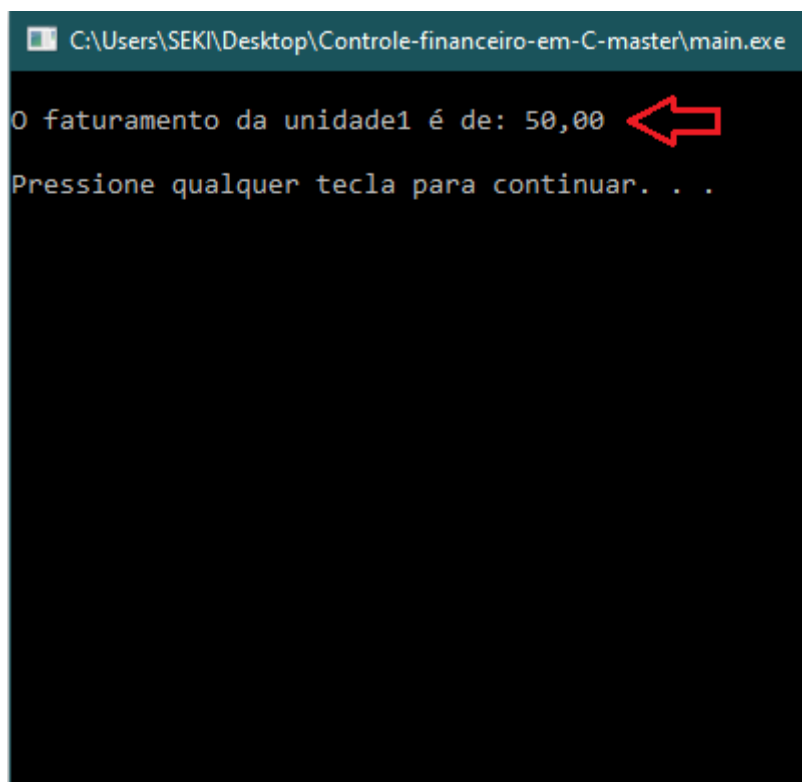
O faturamento mensal é levantado pelas Vendas e pelos Gastos, dito isso, na tela aparecerá o valor gerado, sendo positivo ou negativo. Pegamos um exemplo para melhor entendimento, solicitamos o relatório de faturamento mensal da unidade 1 com base nos gastos e vendas, ou seja :

Os gastos da unidade 1 foi de R\$ 450,00

E a venda da unidade 1 foi de R\$ 500,00

Consequentemente, o faturamento da unidade 1 será de...

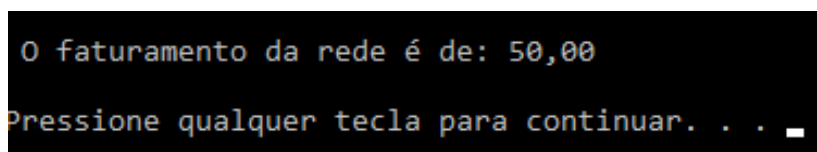
Figura 52: Tela de faturamento Mensal



Faturamento total

É de grande importância a verificação dos lucros, portanto, para registrar o faturamento total entre "3" unidades, pegamos seus valores mensais e apenas calculamos com objetivo de analisar futuras pretensões.

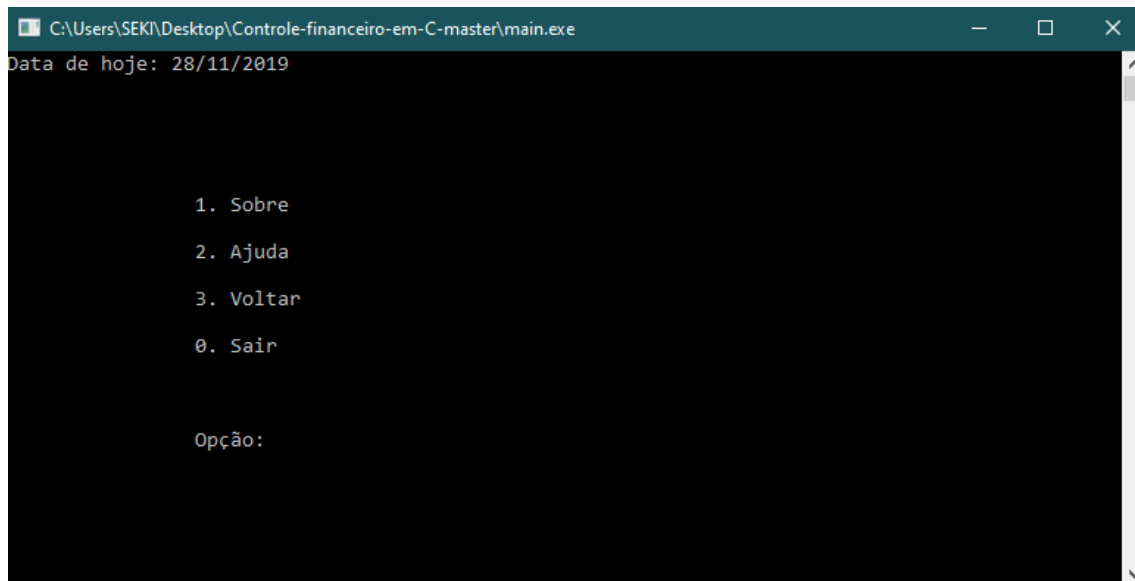
Figura 53: Tela de Faturamento total



4.1.5 OPÇÕES

Na tela de opções terão sobre (com os nomes dos colaboradores, ideia do projeto e informações relevantes do sistema), ajuda para auxílio e a seleção de voltar.

Figura 54: Tela de Opções



4.2 MANUAL DE INSTALAÇÃO DO SOFTWARE

O projeto será enviado por e-mail para o cliente. O sistema irá vir em um arquivo zip com o nome “Controle-financeiro-em-C-“onde dentro irá conter o sistema. Clica no arquivo baixado.

Figura 55: Baixar sistema



Em seguida, abra o executável main.exe

Figura 56: main.exe

Name	Size	Packed	Type	Modified	CRC32
Pasta de arquivos					
main.exe	63.289	19.555	Aplicativo	28/11/2019 14:20	5C2E4470

Feito isso, faça seu login no sistema e comece a usufruir das suas funcionalidades.

Figura 57: Sistema



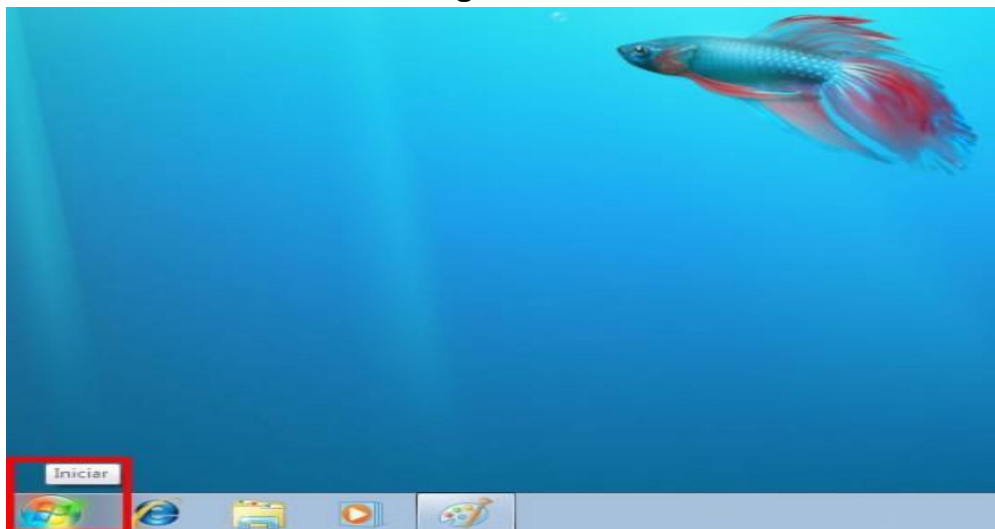
Fonte: Elaboração própria

4.3 MANUAL DE CONFIGURAÇÃO DA REDE

Este manual foi feito pelo sistema operacional Windows 7.

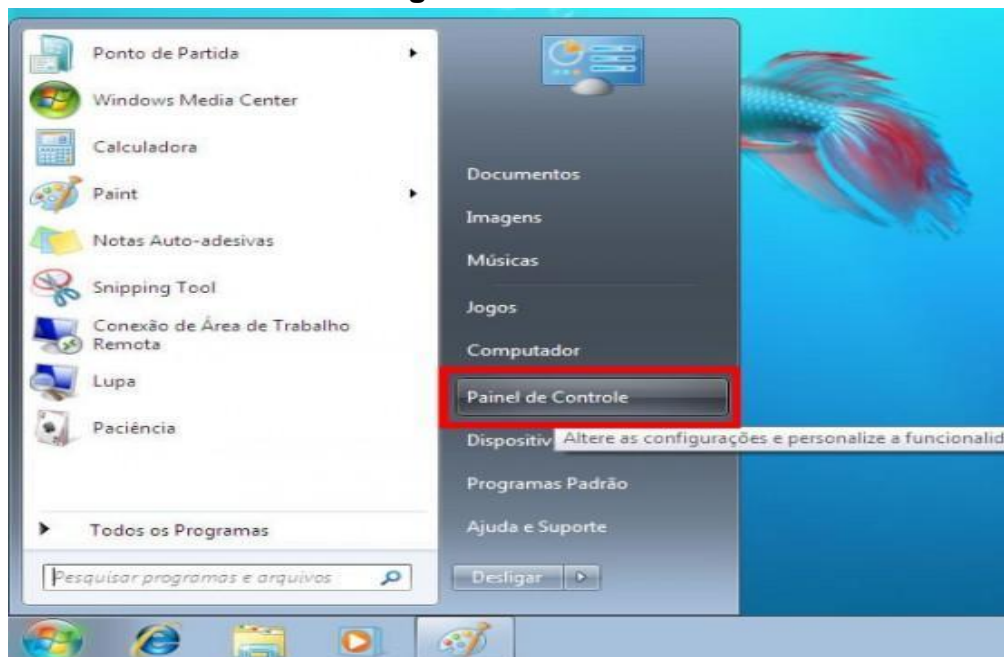
Para iniciar a configuração Banda Larga, Clique em "Iniciar".

Figura 58: Tela inicial



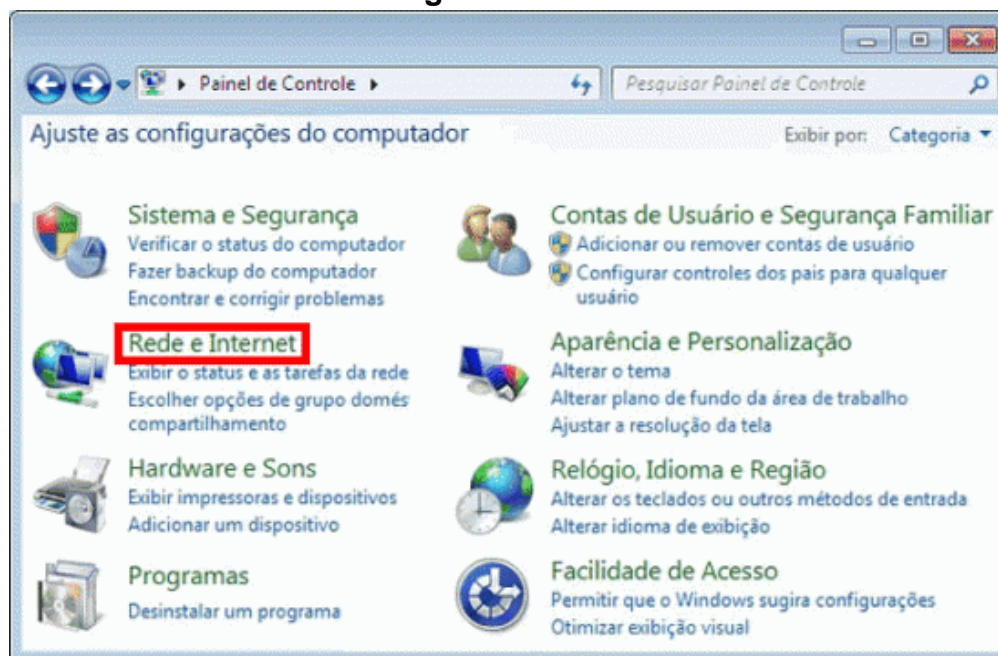
Em seguida clique em "Painel de Controle".

Figura 59: Painel de controle



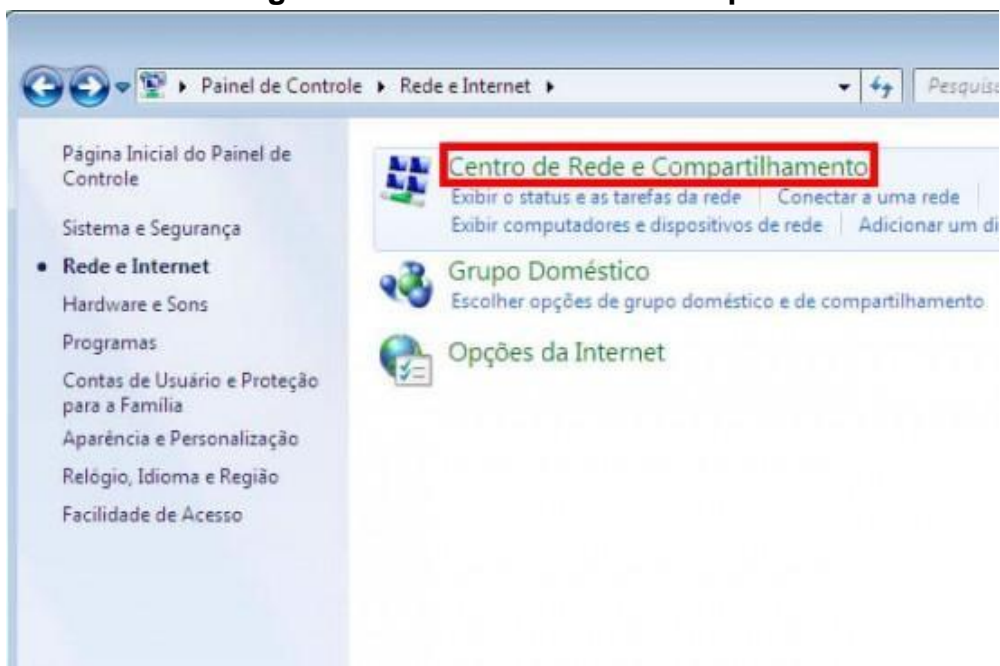
Clique na opção "Rede e Internet"

Figura 60: Rede e internet



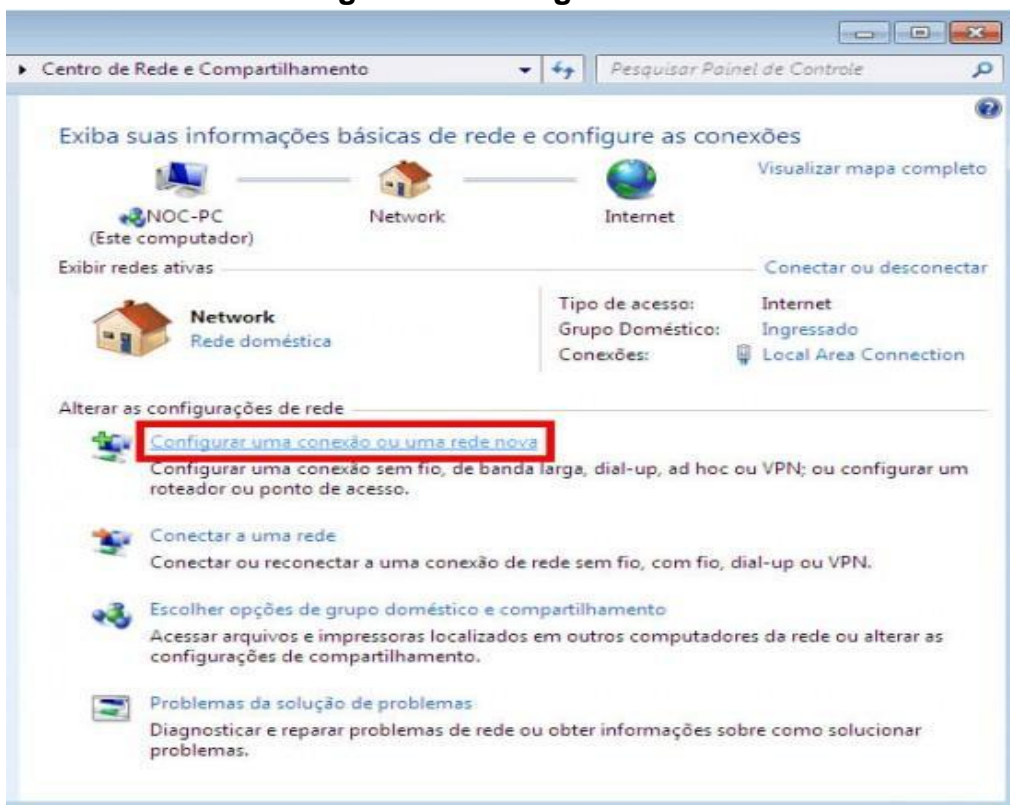
Em seguida escolha a opção "Centro de Rede e Compartilhamento"

Figura 61: Centro de rede e compartilhamento



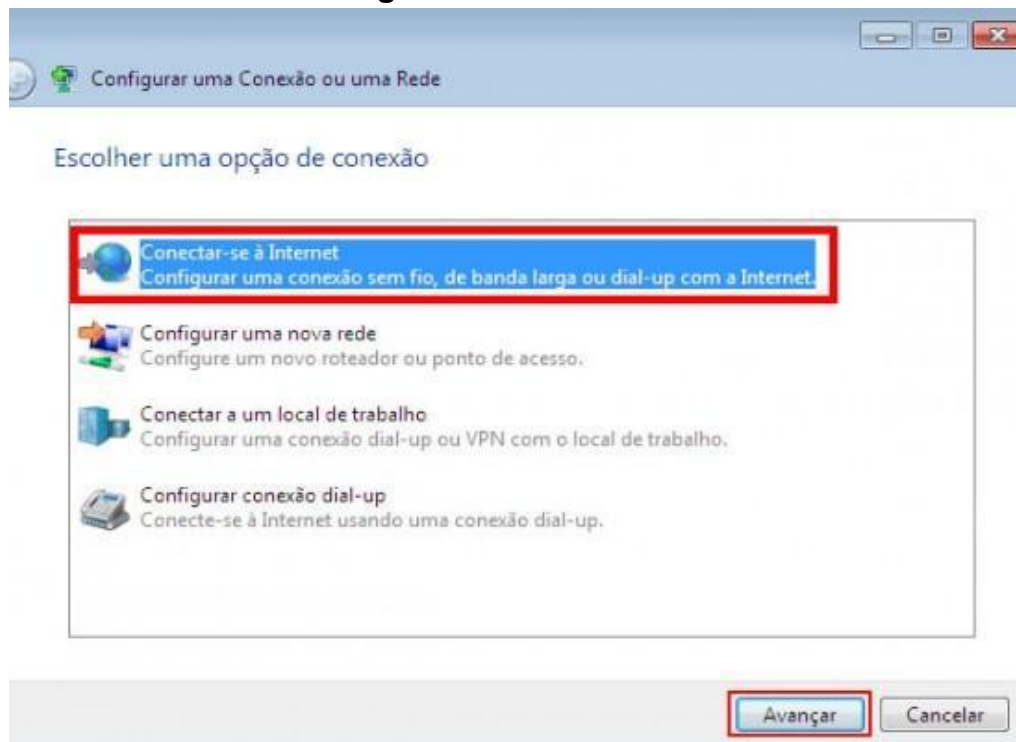
No menu à direita, clique na opção "Configurar uma conexão ou uma rede nova".

Figura 62: Configurar uma nova rede



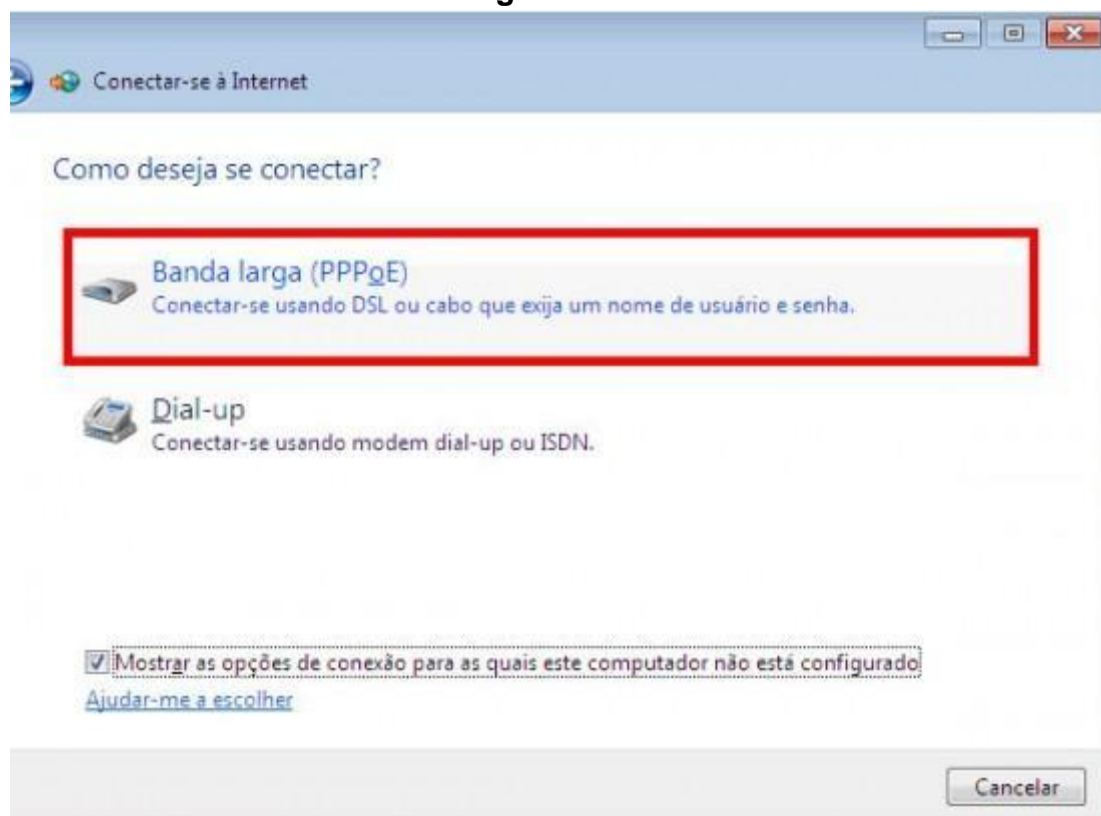
Clique na opção de conexão "Conectar-se à Internet" e depois em "Avançar".

Figura 63: Conecte-se à internet



Clique em "Banda larga (PPPoE)".

Figura 64: PPPoE



Preencha o campo Nome de usuário com o endereço de e-mail de conexão completo (Ex.: username@terra.com.br), abaixo, insira a senha e em seguida dê um nome para a conexão. Importante ressaltar, marque a opção "Lembrar esta senha" para não precisar digitá-la sempre que for estabelecer a conexão. Marque a opção "Permitir que outras pessoas usem esta conexão" caso o computador seja acessado por outras pessoas através de suas contas de usuário. Clique em "Conectar".

Figura 65: Login

Conectar-se à Internet

Digite as informações do provedor de serviços de Internet

Nome de usuário: [Nome de usuário Terra]

Senha: [Senha atribuída a você pelo provedor]

☐ Mostrar caracteres

☐ Lembrar esta senha

Nome da conexão: TERRA

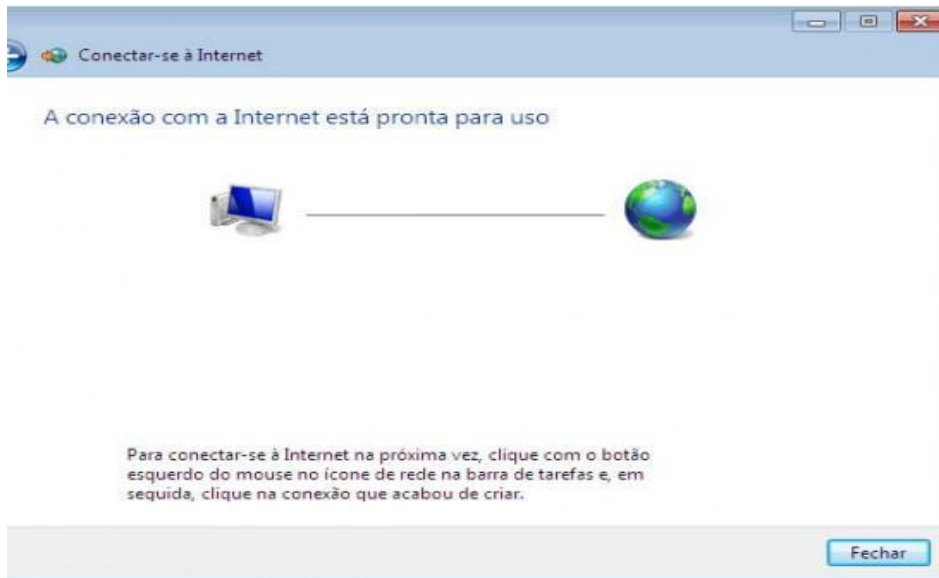
☐ Permitir que outras pessoas usem esta conexão
Esta opção permite que qualquer pessoa com acesso a este computador use a conexão.

[Não tenho um provedor](#)

Conectar Cancelar

Aguarde enquanto o sistema estabelece a conexão e feche a tela após a conexão ser estabelecida.

Figura 66: Finalizado



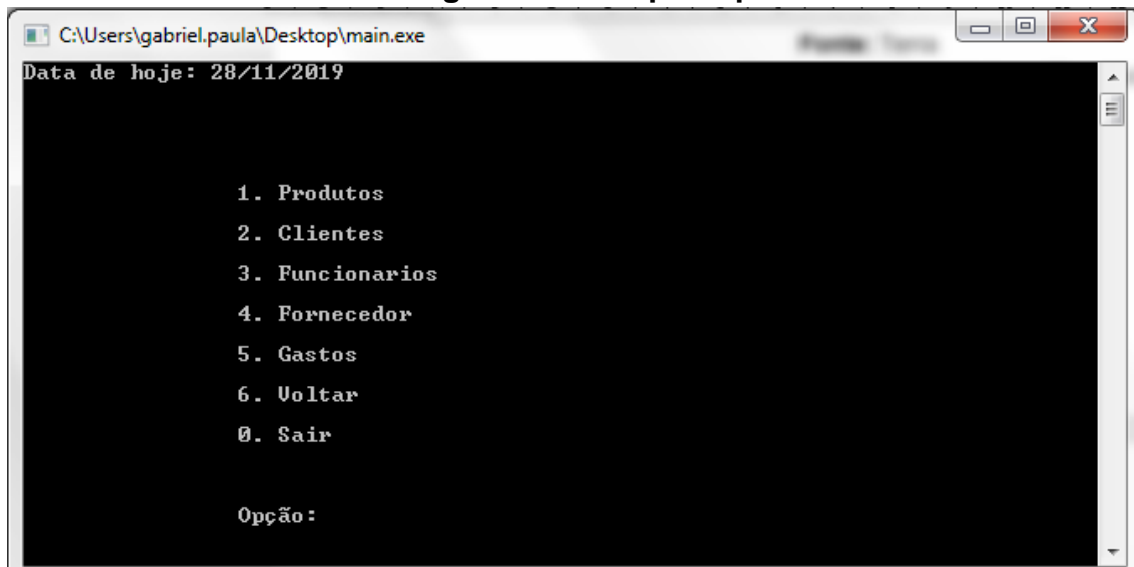
Fonte: Terra

Sempre que desejar conectar, basta acessar a conexão criada e clicar em “Conectar”.

4.4 MANUAL DE TREINAMENTO DOS USUARIOS

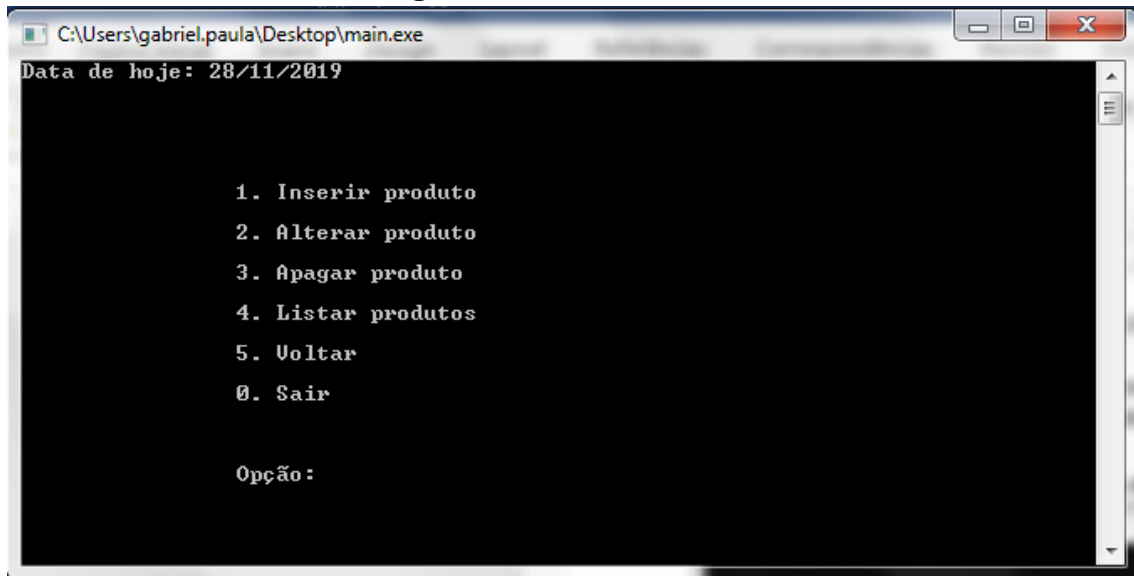
O sistema é bem intuitivo e de fácil aprendizado. A tela mais importante é a de cadastros, na qual poderá cadastrar Produtos, clientes, funcionários e gastos

Figura 67: Tela principal



Para cadastrar um produto, aperte a tecla 1. Em seguida o programa irá lhe solicitar o que deseja fazer, seja inserir, alterar, apagar ou ligar um produto.

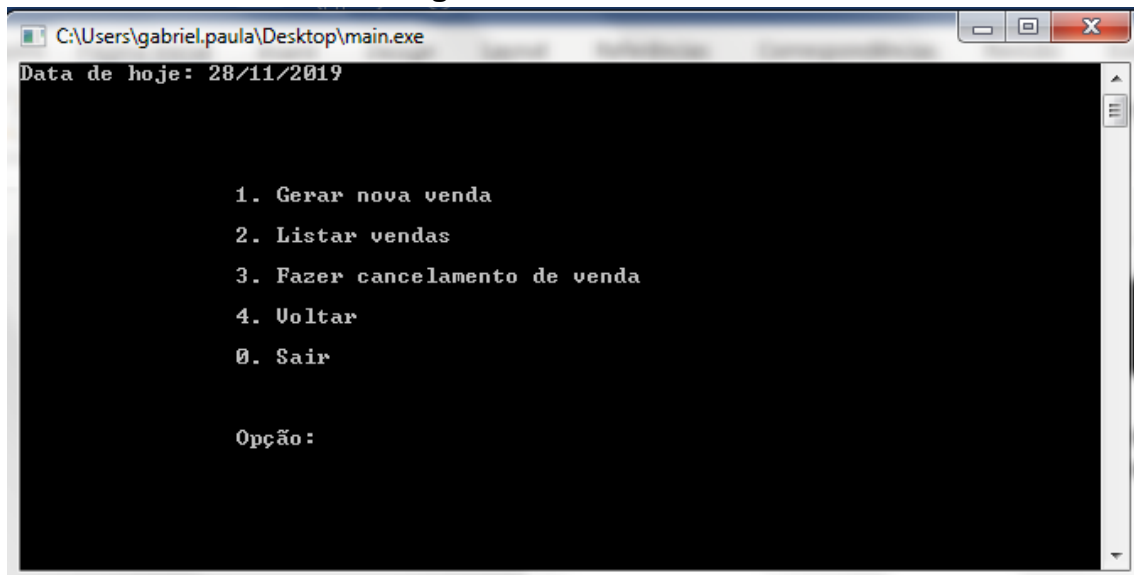
Figura 68: Tela de cadastros



Aperte o número em seu teclado referente ao que deseja fazer e faça sua ação. O mesmo funciona para cadastros de clientes, funcionários e gastos.

O menu de vendas é utilizado para realizar vendas, ele contém as opções de gerar nova venda, listar vendas e fazer o cancelamento de uma venda

Figura 69: Tela de vendas

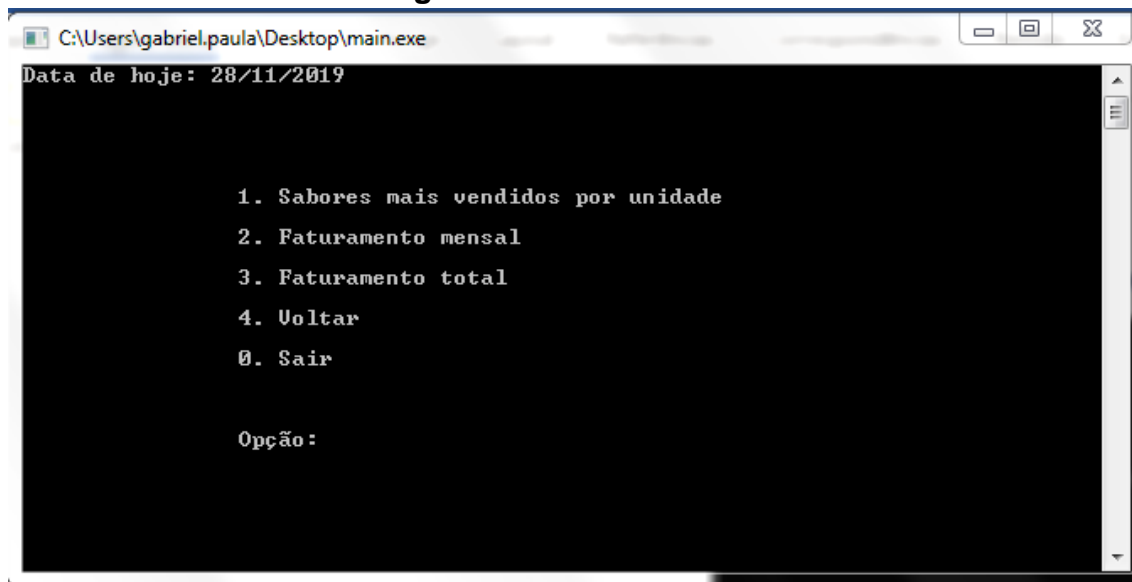


Aperte o número em seu teclado referente ao que deseja fazer e faça sua ação.

A tela de relatórios somente o administrador do sistema tem acesso, pois nela tem contém informação de faturamento onde um funcionário comum não tem acesso.

Essa opção tem disponível sabores mais vendidos por unidade, faturamento mensal e faturamento total. Para visualizar essas opções, basta aperta em seu teclado o número referente a ação que irá fazer.

Figura 70: Tela de relatórios



Fonte: Elaboração própria

4.5 GLOSSÁRIO DO SISTEMA

Glossário de termos

Letra A

Arquivo texto (txt) - arquivo texto é uma espécie de ficheiro informático que é estruturado como uma sequência de linhas. Um arquivo de texto existe dentro de um computador do sistema de arquivos.

ALGOL - O ALGOL é uma família de linguagens de programação de alto nível voltadas principalmente para aplicações científicas.

Aplicação - Software que faz uso de serviços de rede tais como transferência de arquivos, login remoto e correio eletrônico.

Assembly - Assembly ou linguagem de montagem é uma notação legível por humanos para o código de máquina que uma arquitetura de computador específica usa, utilizada para programar códigos entendidos por dispositivos computacionais, como microprocessadores e microcontroladores.

Aplicativo de Software - Software aplicativo, aplicativo ou aplicação, é o software projetado para executar um grupo de funções, tarefas ou atividades coordenadas para o benefício do usuário.

Letra B

Banda Larga - O termo banda larga pode apresentar diferentes significados em diferentes contextos. A recomendação I.113 do setor de Padronização da UIT define banda larga como a capacidade de transmissão.

Letra C

Cabo par trançado - O Cabo par trançado é um tipo de cabo que possui pares de fios entrelaçados um ao redor do outro para cancelar as interferências eletromagnéticas.

Conexão - Ligação entre computadores e dispositivos para que dados sejam transferidos: impressora sem conexão; conexão à internet.

Cliente - Em informática, é um processo ou programa que requisita serviços a um servidor.

Letra D

Desktop - Traduzido do inglês, é um computador pessoal projetado para uso regular em um único local em ou próximo a uma mesa ou mesa, devido ao tamanho e aos requisitos de energia.

Download - Termo utilizado para descrever a gravação de um arquivo no computador do usuário a partir de um site na internet.

Letra E

Engenharia de Software - Engenharia de software é uma área da computação voltada à especificação, desenvolvimento, manutenção e criação de software, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade.

Engenharia de Requisitos - A engenharia de requisitos é um processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo.

EAP - Em Gerência de projetos, uma Estrutura Analítica de Projetos, do Inglês, Work breakdown structure é um processo de subdivisão das entregas e do trabalho do projeto em componentes menores e mais facilmente gerenciáveis.

Letra F

Fluxograma - Fluxograma é um tipo de diagrama, e pode ser entendido como uma representação esquemática de um processo ou algoritmo.

Firewall - Um firewall é um dispositivo de uma rede de computadores que tem por objetivo aplicar uma política de segurança a um determinado ponto da rede.

Letra G

Gestão de Projetos - Gestão de projetos ou gerência de projetos, gerenciamento de projetos ou ainda administração de projetos é a área da administração que aplica os conhecimentos, as habilidades e as técnicas.

Letra H

Hardware - O hardware é um termo técnico que foi traduzido para a língua portuguesa como equipamento, pode ser definido como um termo geral da língua inglesa que se refere a equipamentos físicos.

HUB - Hub ou concentrador é o processo pelo qual se transmite ou difunde determinada informação, tendo, como principal característica, que a mesma informação está sendo enviada para muitos receptores ao mesmo tempo.

Letra I

Internet – Rede mundial de computadores, também conhecida por web.

Letra J

Letra K

Letra L

Linguagem C - **C** é uma **linguagem** de programação compilada de propósito geral, estruturada, imperativa, procedural, padronizada por Organização Internacional para Padronização (ISO)

Login - Em termos informáticos, login ou logon ou sign-in, é o processo para acessar um sistema informático restrito feita através da autenticação ou identificação do utilizador.

Letra M

Mainframe - Um mainframe é um computador de grande porte dedicado normalmente ao processamento de um volume enorme de informações.

Letra N

Letra O

Online – Possui como significado literal “em linha”. Define o status do objeto que está conectado à rede.

Letra P

Prototipação - A prototipagem de software é a atividade de criação de protótipos de aplicativos de software, ou seja, versões incompletas do programa de software que está sendo desenvolvido.

Password (Senha) – Código de acesso a um determinado serviço ou rede.
PDF (Portable Document Format) – É um formato de arquivo criado pela empresa Adobe Systems para que qualquer documento seja visualizado, independente de qual tenha sido o programa que o originou.

Letra Q

Letra R

Rede - genericamente um conjunto de computadores ligados que se comunicam entre si.

Recuperação de Desastres - Recuperação de desastre, ou Recuperação de Sistemas, do inglês disaster recovery, envolve um conjunto de políticas e procedimentos para permitir a recuperação ou continuação da infraestrutura de tecnologia e sistemas vitais na sequência de um desastre natural ou provocado pelo homem.

Roteador - Um roteador é um dispositivo que encaminha pacotes de dados entre redes de computadores, criando um conjunto de redes de sobreposição.

Letra S

Software - é uma sequência de instruções escritas para serem interpretadas por um computador com o objetivo de executar tarefas específicas.

Sistema operacional - é um programa ou um conjunto de programas cuja função é gerenciar os recursos do **sistema** (definir qual programa recebe atenção do processador, gerenciar memória, criar um **sistema** de arquivos, etc).

Servidor - Em informática, um servidor é um software ou computador, com sistema de computação centralizada que fornece serviços a uma rede de computadores, chamada de cliente.

Switch - Termo com diversos significados (switch equipamento para rede ou aplicação em programação).

Sintaxe - Sintaxe é o estudo das regras que regem a construção de frases nas línguas naturais. A sintaxe é a parte da gramática que estuda a disposição das palavras na frase e das frases no discurso, incluindo a sua relação lógica, entre as múltiplas combinações possíveis para transmitir um significado completo e compreensível.

Letra T

Template – É um modelo a ser seguido, com uma estrutura predefinida que facilita o desenvolvimento e criação do conteúdo a partir de algo construído.

Letra U

Usabilidade - **Usabilidade** é a medida de qualidade e eficiência da experiência do usuário com um determinado produto.

Letra V

Variáveis - Na programação, uma variável é um objeto capaz de reter e representar um valor ou expressão. Enquanto as variáveis só "existem" em tempo de execução, elas são associadas a "nomes", chamados identificadores, durante o tempo de desenvolvimento.

Letra W

Windows - Microsoft Windows é uma família de sistemas operacionais desenvolvidos, comercializados e vendidos pela Microsoft.

Letra X

Letra Y

Letra Z

5 CONCLUSÃO

O desenvolvimento do projeto possibilitou a criação de um *software* para uma pizzaria, podendo aprimorar sua agilidade com a demanda que entra todos os dias, além disso, diminuiu o tempo de espera para realização de um pedido, motivando as duas partes envolvidas. De modo geral, os clientes são em sua grande maioria, pessoas de todas as idades. O sistema possibilitou um grande aprendizado, pois com ele, conseguimos desenvolver habilidades que antes não tínhamos, como: melhor interpretação de diagrama de redes, construção de modelos a partir da engenharia de software, raciocínio lógico para programação, etc.

Contudo, com este cenário que temos mediante a sistemas de pizzaria, estamos otimistas em relação ao nosso projeto, pois ele irá beneficiar não só a empresa, mas também o funcionário que irá desenvolver a agilidade necessária para manusear o *software*.

E por fim, este estudo foi de suma importância para conciliar os conhecimentos construídos em sala de aula com o estudo proposto, podendo identificar a viabilidade para dessa forma não construir um sistema de qualquer forma, mas sim, fazê-lo de forma organizada, prática e com responsabilidade, e certamente enriqueceu não só na forma teórica, mas também na forma prática que o mesmo proporcionou.

6 LINHA DE CÓDIGO

```
#include
<stdio.h>
>

#include <stdlib.h>
#include <locale.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <time.h> /*Biblioteca para trabalhar com data */

#define ARQUIVO_PRODUTO "produto.txt"
#define ARQUIVO_CLIENTE "cliente.txt"
#define ARQUIVO_FUNCIONARIO "funcionario.txt"
#define ARQUIVO_FORNECEDOR "fornecedor.txt"
#define ARQUIVO_GASTOS_1 "gastoUnidade1.txt"
#define ARQUIVO_GASTOS_2 "gastoUnidade2.txt"
#define ARQUIVO_GASTOS_3 "gastoUnidade3.txt"
#define ARQUIVO_VENDA_1 "Venda1.txt"
#define ARQUIVO_VENDA_2 "Venda2.txt"
#define ARQUIVO_VENDA_3 "Venda3.txt"
#define SAIR '0'

/**
=====
Sistema de gestão e controle financeiro para uma pizzeria.
=====
Autores: Gabriel Fernandes Lemos,
Erik Hideyuki Yoshimoto Seki,
Gabriel Franco Garcia Rodrigues de Paula,
Matheus Nunes Nepomuceno,
Clayton Belarmino da Silva,
=====
Data de inicio: 01/10/2019
=====
**/

char *MenuPrincipal[]={
    "1. Cadastros",
    "2. Vendas",
    "3. Relatórios",
    "4. Opções",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
```



```

};

char *MenuPrincipalUsuario[]={
    "1. Cadastros",
    "2. Vendas",
    "3. Opções",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuCadastro[]={
    "1. Produtos",
    "2. Clientes",
    "3. Funcionarios",
    "4. Fornecedor",
    "5. Gastos",
    "6. Voltar",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuCadastroUsuario[]={
    "1. Produtos",
    "2. Clientes",
    "3. Fornecedor",
    "4. Voltar",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuProduto[]={
    "1. Inserir produto",
    "2. Alterar produto",
    "3. Apagar produto",
    "4. Listar produtos",
    "5. Voltar",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuCliente[]={
    "1. Inserir cliente",
    "2. Alterar cliente",
    "3. Apagar cliente",
    "4. Listar clientes",
    "5. Voltar",

```

```

        "0. Sair",
        NULL /* ACABARAM AS OPÇÕES*/
    };

char *MenuFuncionario[]={
    "1. Inserir funcionario",
    "2. Alterar funcionario",
    "3. Apagar funcionario",
    "4. Listar funcionario",
    "5. Voltar",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuFornecedor[]={
    "1. Inserir fornecedor",
    "2. Alterar fornecedor",
    "3. Apagar fornecedor",
    "4. Listar fornecedor",
    "5. Voltar",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuVendas[]={
    "1. Gerar nova venda",
    "2. Listar vendas",
    "3. Fazer cancelamento de venda",
    "4. Voltar",
    "0. Sair",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuOpcoes[]={
    "1. Sobre",
    "2. Manual",
    "3. Voltar",
    NULL /* ACABARAM AS OPÇÕES*/
};

char *MenuGastos[]={
    "1. Inserir Gastos",
    "2. Alterar Gastos",
    "3. Apagar Gastos",
    "4. Consultar Gastos",
    "5. Voltar",

```

```

        NULL
    };

    char *MenuGastosUnidade[]={
        "1. Unidade 1",
        "2. Unidade 2",
        "3. Unidade 3",
        "4. Voltar",
        NULL
    };

    char *MenuRelatorios[]={
        "1. Sabores mais vendidos por unidade",
        "2. Faturamento mensal",
        "3. Faturamento total",
        "4. Voltar",
        "0. Sair",
        NULL /* ACABARAM AS OPÇÕES*/
    };

    typedef struct Produto{
        char nomeProduto[30];
        int quantidadeProduto;
        float valorPromocaoProduto, valorVendaProduto, valorCustoProduto;
        char status; /* '*' indica que o produto está apagado*/
    }Produto;

    typedef struct Cliente{
        char nomeCliente[50],cpfCliente[14], rgCliente[12],
        celularCliente[14], emailCliente[50], enderecoCliente[100],
        observacaoCliente[1000];
        char status; /* '*' indica que o produto está apagado*/
    }Cliente;

    typedef struct Funcionario{
        float salarioFuncionario;
        char cpfFuncionario[14], rgFuncionario[12], celularFuncionario[13],
        emailFuncionario[100], enderecoFuncionario[100], nomeFuncionario[100];
        char status; /* '*' indica que o produto está apagado*/
    }Funcionario;

    typedef struct Fornecedor{
        char nomeFornecedor[50],cnpjFornecedor[14], rgFornecedor[12],
        celularFornecedor[14], emailFornecedor[50], enderecoFornecedor[100],
        observacaoFornecedor[1000];
    }Fornecedor;

```

```

        char status; /* '*' indica que o produto está apagado*/
    }Fornecedor;

typedef struct Gastos{
    char tipo[50];
    int unidadeGasto;
    float valorGasto;
    char status; /* '*' indica que o produto está apagado*/
}Gastos;

typedef struct Venda{
    char nomeCliente[30], nomeProduto[30];
    int quantidadeProduto;
    float valorUnitarioProduto, total;
    char status; /* '*' indica que o produto está apagado*/
}Venda;

FILE *fp; /*variável global pois é útil ao longo do programa*/

/*Variáveis para trabalhar com vendas*/
Produto ProdutoVendido;
Cliente ClienteEscolhido;
int quantidade;
long int idProdutoVenda;
float total;

/* lê os valores inseridos pelo usuário*/
void Ler_Gastos(Gastos *g){
    int opcaoDigitada=0;

    do{
        system("cls");
        printf("Escolha uma unidade para inserir o gasto: \n\n 1-
Unidade 1 \n 2- Unidade 2 \n 3- Unidade 3 \n");
        printf ("\n Escolha a opção (1,2,3): ");
        scanf("%d",&g->unidadeGasto);
    }while((g->unidadeGasto< 1) || (g->unidadeGasto > 3));
    system("cls");
    printf("Digite o valor dos gastos: ");
    scanf("%f",&g->valorGasto);
    system("cls");
    fflush(stdin);
    printf("Digite o tipo de gasto: ");
    gets(g->tipo);
    system("cls");
    fflush(stdin);

```

```

}

void Ler_Produto(Produto *p){
    int opcaoDigitada = 0;

    printf("Digite o nome do produto:  ");
    gets(p->nomeProduto);
    system("cls");

    printf("Digite a quantidade:  ");
    scanf("%d",&p->quantidadeProduto);
    system("cls");

    printf("Digite o valor de custo:  ");
    scanf("%f",&p->valorCustoProduto);
    system("cls");

    printf("Digite o valor de venda:  ");
    scanf("%f",&p->valorVendaProduto);
    system("cls");

    printf("Digite o valor de promoção (Se não houver promoção repita o
valor de venda):  ");
    scanf("%f",&p->valorPromocaoProduto);
    system("cls");

    p->status = ' ';
    fflush(stdin);
}

void Ler_Cliente(Cliente *c){
    printf("Digite o nome do cliente:  ");
    gets(c->nomeCliente);
    system("cls");

    printf("Digite o CPF do cliente:  ");
    gets(c->cpfCliente);
    system("cls");

    printf("Digite o RG do cliente:  ");
    gets(c->rgCliente);
    system("cls");

    printf("Digite o celular do cliente:  ");
    gets(c->celularCliente);
    system("cls");
}

```

```

printf("Digite o email do cliente:  ");
gets(c->emailCliente);
system("cls");

printf("Digite o endereço do cliente:  ");
gets(c->enderecoCliente);
system("cls");

printf("Digite o observação do cliente:  ");
gets(c->observacaoCliente);
system("cls");

c->status = ' ';
fflush(stdin);
}

void Ler_Funcionario(Funcionario *f){
printf("Digite o nome do funcionario:  ");
gets(f->nomeFuncionario);
system("cls");

printf("Digite o CPF do funcionario:  ");
gets(f->cpfFuncionario);
system("cls");

printf("Digite o RG do funcionario:  ");
gets(f->rgFuncionario);
system("cls");

printf("Digite o celular do funcionario:  ");
gets(f->celularFuncionario);
system("cls");

printf("Digite o email do funcionario:  ");
gets(f->emailFuncionario);
system("cls");

printf("Digite o endereço do funcionario:  ");
gets(f->enderecoFuncionario);
system("cls");

printf("Digite o salario do funcionario:  ");
scanf("%f",&f->salarioFuncionario);
system("cls");

```

```

        f->status = ' ';
        fflush(stdin);
    }

    void Ler_Fornecedor(Fornecedor *f){

        printf("Digite o nome do fornecedor:  ");
        gets(f->nomeFornecedor);
        system("cls");

        printf("Digite o CPF/CNPJ do fornecedor:  ");
        gets(f->cnnpjFornecedor);
        system("cls");

        printf("Digite o RG do fornecedor:  ");
        gets(f->rgFornecedor);
        system("cls");

        printf("Digite o celular do fornecedor:  ");
        gets(f->celularFornecedor);
        system("cls");

        printf("Digite o email do fornecedor:  ");
        gets(f->emailFornecedor);
        system("cls");

        printf("Digite o endereço do fornecedor:  ");
        gets(f->enderecoFornecedor);
        system("cls");

        printf("Digite o observação do fornecedor:  ");
        gets(f->observacaoFornecedor);
        system("cls");

        f ->status = ' ';
        fflush(stdin);
    }

    void Atribuir_Valores_Venda(Venda *v){

        strcpy(v->nomeCliente,ClienteEscolhido.nomeCliente);

        strcpy(v->nomeProduto,ProdutoVendido.nomeProduto);

        v->quantidadeProduto = quantidade;
    }

```

```

        v->valorUnitarioProduto = ProdutoVendido.valorPromocaoProduto;

        v->total = total;

        v->status = ' ';
        fflush(stdin);
    }

    /*Mostra na tela os produtos existentes nos arquivos*/
    void Mostrar_Gastos(Gastos g){
        printf("Tipo de gasto: %-30s \n",g.tipo);
        printf("Valor do gasto: %10.2f\n",g.valorGasto);
        printf("-----\n");
    }

    void Mostrar_Venda(Venda v){
        printf("Cliente: %s \n",v.nomeCliente);
        printf("Produto: %s \n",v.nomeProduto);
        printf("Valor unitário: %2.f \n",v.valorUnitarioProduto);
        printf("Quantidade: %d \n", v.quantidadeProduto);
        printf("Total: %2.f\n", v.total);
        printf("-----\n");
    }

    void Mostrar_Produto(Produto p){
        printf("Nome: %-30s \n",p.nomeProduto);
        printf("Quantidade do produto: %3d \n",p.quantidadeProduto);
        printf("Valor de venda: %10.2f\n",p.valorVendaProduto);
        printf("Valor de custo: %10.2f\n",p.valorCustoProduto);
        printf("Valor de promoção: %10.2f\n",p.valorPromocaoProduto);
        printf("-----\n");
    }

    void Mostrar_Cliente(Cliente c){
        printf("Nome: %-30s \n",c.nomeCliente);
        printf("CPF: %-30s \n",c.cpfCliente);
        printf("RG: %-30s \n",c.rgCliente);
        printf("Celular: %-30s \n",c.celularCliente);
        printf("Email: %-30s \n",c.emailCliente);
        printf("Endereço: %-30s \n",c.enderecoCliente);
        printf("Observação: %-30s \n",c.observacaoCliente);
        printf("-----\n");
    }

    void Mostrar_Funcionario(Funcionario f){
        printf("Nome: %-30s \n",f.nomeFuncionario);

```



```

        printf("CPF: %-30s \n",f.cpfFuncionario);
        printf("RG: %-30s \n",f.rgFuncionario);
        printf("Celular: %-30s \n",f.celularFuncionario);
        printf("Email: %-30s \n",f.emailFuncionario);
        printf("Endereço: %-30s \n",f.enderecoFuncionario);
        printf("Salário: %10.2f\n",f.salarioFuncionario);
        printf("=-=-=-=-=-\n");
    }

void Mostrar_Fornecedor(Fornecedor f){
    printf("Nome: %-30s \n",f.nomeFornecedor);
    printf("CPF/CNPJ: %-30s \n",f.cnpjFornecedor);
    printf("RG: %-30s \n",f.rgFornecedor);
    printf("Celular: %-30s \n",f.celularFornecedor);
    printf("Email: %-30s \n",f.emailFornecedor);
    printf("Endereço: %-30s \n",f.enderecoFornecedor);
    printf("Observação: %-30s \n",f.observacaoFornecedor);
    printf("=-=-=-=-=-\n");
}

/*Adiciona valores ao arquivo*/
void Adiciona_Gasto(Gastos g){
    fseek(fp, 0L, SEEK_END);
    if(fwrite(&g, sizeof(g), 1, fp)!=1)
        printf("Adicionar Gasto: Falhou a escrita de gastos");
}

void Adiciona_Venda(Venda v){
    fseek(fp, 0L, SEEK_END);
    if(fwrite(&v, sizeof(v), 1, fp)!=1)
        printf("Adicionar Venda: Falhou a escrita de venda");
}

void Adiciona_Produto(Produto p){
    fseek(fp, 0L, SEEK_END);
    if(fwrite(&p, sizeof(p), 1, fp)!=1)
        printf("Adicionar Produto: Falhou a escrita do produto");
}

void Adiciona_Cliente(Cliente p){
    fseek(fp, 0L, SEEK_END);
    if(fwrite(&p, sizeof(p), 1, fp)!=1)
        printf("Adicionar Cliente: Falhou a escrita do produto");
}

void Adiciona_Funcionario(Funcionario f){

```

```

        fseek(fp, 0L, SEEK_END);
        if(fwrite(&f, sizeof(f), 1, fp)!=1)
            printf("Adicionar funcionario: Falhou a escrita do funcionario");
    }

```

```

void Adiciona_Fornecedor(Fornecedor f){
    fseek(fp, 0L, SEEK_END);
    if(fwrite(&f, sizeof(f), 1, fp)!=1)
        printf("Adicionar Fornecedor: Falhou a escrita do fornecedor");
}

```

/*Verificar se o arquivo já existe. Se não existir, ele é criado se já existir, abre-o em modo de leitura e escrita (r+b)*/

```

void AbrirArquivoGastos1(){
    fp=fopen(ARQUIVO_GASTOS_1,"r+b");
    if(fp == NULL){
        fp=fopen(ARQUIVO_GASTOS_1,"w+b");
        if(fp==NULL){
            printf("Erro: Impossível criar arquivo de gastos\n");
            exit(1);
        }
    }
}

```

```

void AbrirArquivoGastos2(){
    fp=fopen(ARQUIVO_GASTOS_2,"r+b");
    if(fp == NULL){
        fp=fopen(ARQUIVO_GASTOS_2,"w+b");
        if(fp==NULL){
            printf("Erro: Impossível criar arquivo de gastos\n");
            exit(1);
        }
    }
}

```

```

void AbrirArquivoGastos3(){
    fp=fopen(ARQUIVO_GASTOS_3,"r+b");
    if(fp == NULL){
        fp=fopen(ARQUIVO_GASTOS_3,"w+b");
        if(fp==NULL){
            printf("Erro: Impossível criar arquivo de gastos\n");
            exit(1);
        }
    }
}

```

```
}
```

```
void AbrirArquivoProduto(){
```

```
    fp= fopen(ARQUIVO_PRODUTO, "r+b"); //tentar abrir
```

```
    if(fp==NULL){
```

```
        fp = fopen(ARQUIVO_PRODUTO, "w+b"); // criar o arquivo
```

```
        if(fp==NULL){
```

```
            printf("Erro fatal: impossível criar arquivo de produtos\n");
```

```
            exit(1);
```

```
        }
```

```
    }
```

```
}
```

```
void AbrirArquivoCliente(){
```

```
    fp= fopen(ARQUIVO_CLIENTE, "r+b"); //tentar abrir
```

```
    if(fp==NULL){
```

```
        fp = fopen(ARQUIVO_CLIENTE, "w+b"); // criar o arquivo
```

```
        if(fp==NULL){
```

```
            printf("Erro fatal: impossível criar arquivo de clientes\n");
```

```
            exit(1);
```

```
        }
```

```
    }
```

```
}
```

```
void AbrirArquivoFuncionario(){
```

```
    fp= fopen(ARQUIVO_FUNCIONARIO, "r+b"); //tentar abrir
```

```
    if(fp==NULL){
```

```
        fp = fopen(ARQUIVO_FUNCIONARIO, "w+b"); // criar o arquivo
```

```
        if(fp==NULL){
```

```
            printf("Erro fatal: impossível criar arquivo de  
funcionario\n");
```

```
            exit(1);
```

```
        }
```

```
    }
```

```
}
```

```
void AbrirArquivoFornecedor(){
```

```
    fp= fopen(ARQUIVO_FORNECEDOR, "r+b"); //tentar abrir
```

```
    if(fp==NULL){
```

```
        fp = fopen(ARQUIVO_FORNECEDOR, "w+b"); // criar o arquivo
```

```
        if(fp==NULL){
```

```
            printf("Erro fatal: impossível criar arquivo de  
fornecedor\n");
```

```
            exit(1);
```

```
        }
```

```

    }
}

void AbrirArquivoVenda1(){
    fp= fopen(ARQUIVO_VENDA_1, "r+b"); //tentar abrir
    if(fp==NULL){
        fp = fopen(ARQUIVO_VENDA_1, "w+b"); // criar o arquivo
        if(fp==NULL){
            printf("Erro fatal: impossível criar arquivo de venda\n");
            exit(1);
        }
    }
}

void AbrirArquivoVenda2(){
    fp= fopen(ARQUIVO_VENDA_2, "r+b"); //tentar abrir
    if(fp==NULL){
        fp = fopen(ARQUIVO_VENDA_2, "w+b"); // criar o arquivo
        if(fp==NULL){
            printf("Erro fatal: impossível criar arquivo de venda\n");
            exit(1);
        }
    }
}

void AbrirArquivoVenda3(){
    fp= fopen(ARQUIVO_VENDA_3, "r+b"); //tentar abrir
    if(fp==NULL){
        fp = fopen(ARQUIVO_VENDA_3, "w+b"); // criar o arquivo
        if(fp==NULL){
            printf("Erro fatal: impossível criar arquivo de venda\n");
            exit(1);
        }
    }
}

/*Exibe a data do dia */
void FuncaoExibirData(){
    time_t mytime;
    mytime = time(NULL);
    struct tm tm = *localtime(&mytime);
    printf("Data de hoje: %d/%d/%d\n", tm.tm_mday, tm.tm_mon + 1,
tm.tm_year + 1900);
}

/* Faz um menu simples com as opções do vetor de strings.

```

seleciona a opção, usando o primeiro caracter de cada string.
devolve o primeiro caracter da opção.

```
*/  
char FuncoesMenuPrincipal(char *opcoes[]){  
    int i;  
    char ch;  
    while(1){  
  
        FuncaoExibirData();  
  
        printf("\n\n\n\n\n");  
        for(i=0; opcoes[i]!=NULL; i++)  
            printf("\t\t%s\n\n",opcoes[i]);  
  
        printf("\n\n\t\tOpção: ");  
        ch = getchar(); fflush(stdin);  
        for(i=0; opcoes[i]!= NULL; i++)  
            if(opcoes[i][0]==ch)  
                return ch;  
    }  
}  
  
char FuncoesMenuProduto(char *opcoes[]){  
    int i;  
    char ch;  
    while(1){  
  
        FuncaoExibirData();  
  
        printf("\n\n\n\n\n");  
        for(i=0; opcoes[i]!=NULL; i++)  
            printf("\t\t%s\n\n",opcoes[i]);  
  
        printf("\n\n\t\tOpção: ");  
        ch = getchar(); fflush(stdin);  
        for(i=0; opcoes[i]!= NULL; i++)  
            if(opcoes[i][0]==ch)  
                return ch;  
    }  
}  
  
char FuncoesMenuCliente(char *opcoes[]){  
    int i;  
    char ch;  
    while(1){
```

```

        FuncaoExibirData();

        printf("\n\n\n\n\n");
        for(i=0; opcoes[i]!=NULL; i++)
            printf("\t\t%s\n\n",opcoes[i]);

        printf("\n\n\t\tOpção: ");
        ch = getchar(); fflush(stdin);
        for(i=0; opcoes[i]!= NULL; i++)
            if(opcoes[i][0]==ch)
                return ch;
    }
}

char FuncoesMenuFornecedor(char *opcoes[]){
    int i;
    char ch;
    while(1){

        FuncaoExibirData();

        printf("\n\n\n\n\n");
        for(i=0; opcoes[i]!=NULL; i++)
            printf("\t\t%s\n\n",opcoes[i]);

        printf("\n\n\t\tOpção: ");
        ch = getchar(); fflush(stdin);
        for(i=0; opcoes[i]!= NULL; i++)
            if(opcoes[i][0]==ch)
                return ch;
    }
}

char FuncoesMenuFuncionario(char *opcoes[]){
    int i;
    char ch;
    while(1){

        FuncaoExibirData();

        printf("\n\n\n\n\n");
        for(i=0; opcoes[i]!=NULL; i++)
            printf("\t\t%s\n\n",opcoes[i]);

        printf("\n\n\t\tOpção: ");
        ch = getchar(); fflush(stdin);
    }
}

```

```

        for(i=0; opcoes[i] != NULL; i++)
            if(opcoes[i][0]==ch)
                return ch;
    }
}

char FuncoesMenuCadastro(char *opcoes[]){
    int i;
    char ch;
    while(1){

        FuncaoExibirData();

        printf("\n\n\n\n\n");
        for(i=0; opcoes[i] != NULL; i++)
            printf("\t\t%s\n", opcoes[i]);

        printf("\n\n\t\tOpção: ");
        ch = getchar(); fflush(stdin);
        for(i=0; opcoes[i] != NULL; i++)
            if(opcoes[i][0]==ch)
                return ch;
    }
}

char FuncoesMenuVendas(char *opcoes[]){
    int i;
    char ch;
    while(1){

        FuncaoExibirData();

        printf("\n\n\n\n\n");
        for(i=0; opcoes[i] != NULL; i++)
            printf("\t\t%s\n", opcoes[i]);

        printf("\n\n\t\tOpção: ");
        ch = getchar(); fflush(stdin);
        for(i=0; opcoes[i] != NULL; i++)
            if(opcoes[i][0]==ch)
                return ch;
    }
}

char FuncoesMenuOpcoes(char *opcoes[]){
    int i;

```

```

char ch;
while(1){

    FuncaoExibirData();

    printf("\n\n\n\n\n");
    for(i=0; opcoes[i]!=NULL; i++)
        printf("\t\t%s\n\n", opcoes[i]);

    printf("\n\n\t\tOpção: ");
    ch = getchar(); fflush(stdin);
    for(i=0; opcoes[i]!= NULL; i++)
        if(opcoes[i][0]==ch)
            return ch;
    }
}

char FuncoesMenuGastos(char *opcoes[]){
int i;
char ch;
while(1){

    FuncaoExibirData();

    printf("\n\n\n\n\n");
    for(i=0; opcoes[i]!=NULL; i++)
        printf("\t\t%s\n\n", opcoes[i]);

    printf("\n\n\t\tOpção: ");
    ch = getchar(); fflush(stdin);
    for(i=0; opcoes[i]!= NULL; i++)
        if(opcoes[i][0]==ch)
            return ch;
    }
}

char FuncoesMenuRelatorios(char *opcoes[]){
int i;
char ch;
while(1){

    FuncaoExibirData();

    printf("\n\n\n\n\n");
    for(i=0; opcoes[i]!=NULL; i++)
        printf("\t\t%s\n\n", opcoes[i]);

```



```

        printf("\n\n\t\tOpção: ");
        ch = getchar(); fflush(stdin);
        for(i=0; opcoes[i] != NULL; i++)
            if(opcoes[i][0]==ch)
                return ch;
    }
}

//Faz o CRUD de produto
void InserirProduto(){
    system("cls");
    Produto x;
    Ler_Produto(&x);
    Adiciona_Produto(x);
}

void AlterarProduto(){
    system("cls");
    Produto x;
    long int id;
    printf("Qual o numero do produto: ");
    scanf("%ld", & id); fflush(stdin);
    if(fseek(fp, (id-1)*sizeof(Produto), SEEK_SET)!=0){
        printf("produto inexistente!!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Produto), 1, fp) != 1){
        printf("Problemas na leitura do produto!!!");
        system("pause");
        return;
    }

    if(x.status=='*'){
        printf("Um produto apagado não pode ser alterado!!! \n\n");
        system("pause");
        return;
    }

    printf("\n\n produtos Atuais \n\n");
    Mostrar_Produto(x);
    printf("\n\n Novos produtos \n\n");
    Ler_Produto(&x);
    // recuar um produto no arquivo
    fseek(fp, -(long) sizeof(Produto), SEEK_CUR);

```

```

        // reescrever o produto;
        fwrite(&x, sizeof(Produto), 1, fp);
        fflush(fp); /*despejar os arquivos no disco rígido*/
        printf("\nProduto alterado com sucesso!\n");
        system("pause");
    }

void ApagarProduto(){
    system("cls");
    Produto x;
    long int id;
    char resp;

    printf("Qual o numero do produto: ");
    scanf("%ld", &id); fflush(stdin);
    if(fseek(fp, (id - 1)*sizeof(Produto), SEEK_SET)!= 0){
        printf("produto inexistente ou problemas no produto!!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Produto), 1, fp)!= 1){
        printf("Problema na leitura do produto!!!");
        system("pause");
        return;
    }
    if(x.status=='*'){
        printf("produto já está apagado!!!\n\n");
        return;
        system("pause");
    }
    printf("\n\n produtos atuais \n\n");
    Mostrar_Produto(x);
    printf("\n\n Apagar o produto (s/n)???: ");
    resp = getchar();
    fflush(stdin);
    if(toupper(resp)!= 'S')
        return;

    x.status= '*';
    // recuar um produto no arquivo
    fseek(fp, -(long) sizeof(Produto), SEEK_CUR);
    // reescrever o produto;
    fwrite(&x, sizeof(Produto), 1, fp);
    fflush(fp); /*Despejar os arquivos no disco rígido*/
    system("cls");
    printf("\nProduto excluido com sucesso!\n");
}

```

```

}

void ListarProduto(){
    system("cls");
    Produto reg;
    rewind(fp);
    int cont=0;
    while(1){
        if(fread(&reg, sizeof(reg), 1, fp) != 1){
            break; /*Sair do laço*/
        }else{
            cont=cont+1;
        }
        if(reg.status=='*'){
            continue; /*Passa ao próximo*/
        }
        printf("ID: %d\n", cont);
        Mostrar_Produto(reg);
    }
}

//Faz inserção e leitura de venda
void InserirVenda(){
    system("cls");
    Venda v;
    Atribuir_Valores_Venda(&v);
    Adiciona_Venda(v);
}

void ListarVenda(){
    system("cls");
    Venda reg;
    rewind(fp);
    int cont=0;
    while(1){
        if(fread(&reg, sizeof(reg), 1, fp) != 1){
            break; /*Sair do laço*/
        }else{
            cont=cont+1;
        }
        if(reg.status=='*'){
            continue; /*Passa ao próximo*/
        }
        printf("ID: %d\n", cont);
        Mostrar_Venda(reg);
    }
}

```

```

}

void ApagarVenda(){
    system("cls");
    Venda x;
    long int id;
    char resp;

    printf("Qual o numero da venda: ");
    scanf("%ld", & id); fflush(stdin);
    if(fseek(fp, (id - 1)*sizeof(Venda), SEEK_SET)!= 0){
        printf("Venda inexistente ou problemas na venda!!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Venda), 1, fp)!= 1){
        printf("Problema na leitura da Venda!!!");
        system("pause");
        return;
    }
    if(x.status=='*'){
        printf("Venda já está apagada!!!\n\n");
        system("pause");
        return;
    }
    printf("\n\n Vendas atuais \n\n");
    Mostrar_Venda(x);
    printf("\n\n Apagar a venda (s/n)???: "); resp = getchar();
    fflush(stdin);
    if(toupper(resp)!= 'S')return;

    x.status= '*';
    // recuar um Venda no arquivo
    fseek(fp, -(long) sizeof(Venda), SEEK_CUR);
    // reescrever a Venda;
    fwrite(&x, sizeof(Venda), 1, fp);
    fflush(fp); /*Despejar os arquivos no disco rígido*/
    system("cls");
    printf("\nVenda excluida com sucesso!\n");
}

//Faz inserção e leitura de gasto
void InserirGasto(){
    system("cls");
    Gastos g;
    Ler_Gastos(&g);
}

```

```

        Adiciona_Gasto(g);
    }

void AlterarGasto(){
    system("cls");
    Gastos x;
    long int id;
    printf("Qual o numero de id do gasto: ");
    scanf("%ld", & id); fflush(stdin);
    if(fseek(fp, (id-1)*sizeof(Gastos), SEEK_SET)!=0){
        printf("Gasto inexistente!!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Gastos), 1, fp)!= 1){
        printf("Problemas na leitura de Gastos!!!");
        system("pause");
        return;
    }

    if(x.status=='*'){
        printf("Um gasto apagado não pode ser alterado!!! \n\n");
        system("pause");
        return;
    }

    printf("\n\n Gastos Atuais \n\n");
    Mostrar_Gastos(x);
    printf("\n\n Novos Gastos \n\n");
    Ler_Gastos(&x);
    // recuar um produto no arquivo
    fseek(fp, -(long) sizeof(Gastos), SEEK_CUR);
    // reescrever o produto;
    fwrite(&x, sizeof(Gastos), 1, fp);
    fflush(fp); /*despejar os arquivos no disco rígido*/
    printf("\nGasto alterado com sucesso!\n");
    system("pause");
}

void ApagarGasto(){
    system("cls");
    Gastos x;
    long int id;
    char resp;

    printf("Qual o numero do id do gasto: ");

```

```

scanf("%ld", & id); fflush(stdin);
if(fseek(fp, (id - 1)*sizeof(Gastos), SEEK_SET)!= 0){
    printf("Gasto inexistente ou problemas nos gastos!!!");
    system("pause");
    return;
}
if(fread(&x, sizeof(Gastos), 1, fp)!= 1){
    printf("Problema na leitura do gasto!!!");
    system("pause");
    return;
}
if(x.status=='*'){
    printf("Gasto já está apagada!!!\n\n");
    system("pause");
    return;
}
printf("\n\n Gastos atuais \n\n");
Mostrar_Gastos(x);
printf("\n\n Apagar o gasto (s/n)???: "); resp = getchar();
fflush(stdin);
if(toupper(resp)!= 'S')return;

x.status= '*';
// recuar um Venda no arquivo
fseek(fp, -(long) sizeof(Gastos), SEEK_CUR);
// reescrever a Venda;
fwrite(&x, sizeof(Gastos), 1, fp);
fflush(fp); /*Despejar os arquivos no disco rígido*/
system("cls");
printf("\nGasto excluido com sucesso!\n");
}

void ListarGasto(){
    system("cls");
    Gastos reg;
    rewind(fp);
    int cont=0;
    while(1){
        if(fread(&reg, sizeof(reg), 1, fp)!= 1){
            break; /*Sair do laço*/
        }else{
            cont=cont+1;
        }
        if(reg.status=='*'){
            continue; /*Passa ao próximo*/
        }
    }
}

```

```

        printf("ID: %d\n", cont);
        Mostrar_Gastos(reg);
    }
}

//Soma todos os gastos de acordo com a unidade escolhida
int CalcularGasto(){
    float valor=0, valorTotalGastos;
    int i,j;
    Gastos x[700];
    rewind (fp);
    do{

        fread(&x, sizeof(x), 1, fp);
        for(i=0; i<50; i++){
            if(x[i].status == '*') {
                continue;
            }

            if(valor == 0){
                valor = x[i].valorGasto;
            }
            valor= valor + x[i].valorGasto;
        }

    }while(i!=50);
    return (valor);
}

//Faz o CRUD de cliente
void InserirCliente(){
    system("cls");
    Cliente x;
    Ler_Cliente(&x);
    Adiciona_Cliente(x);
}

void AlterarCliente(){
    system("cls");
    Cliente x;
    long int id;
    printf("Qual o numero do cliente: ");
    scanf("%ld", & id); fflush(stdin);
}

```

```

        if(fseek(fp, (id-1)*sizeof(Cliente), SEEK_SET)!=0){
            printf("Cliente inexistente!!!");
            system("pause");
            return;
        }
        if(fread(&x, sizeof(Cliente), 1, fp)!= 1){
            printf("Problemas na leitura do cliente!!!");
            system("pause");
            return;
        }

        if(x.status=='*'){
            printf("Um cliente apagado não pode ser alterado!!! \n\n");
            system("pause");
            return;
        }

        printf("\n\n Clientes Atuais \n\n");
        Mostrar_Cliente(x);
        printf("\n\n Novos clientes \n\n");
        Ler_Cliente(&x);
        // recuar um cliente no arquivo
        fseek(fp, -(long) sizeof(Cliente), SEEK_CUR);
        // reescrever o cliente;
        fwrite(&x, sizeof(Cliente), 1, fp);
        fflush(fp); /*despejar os arquivos no disco rígido*/
        printf("\nCliente alterado com sucesso!\n");
        system("pause");
    }

    void ApagarCliente(){
        system("cls");
        Cliente x;
        long int id;
        char resp;

        printf("Qual o numero do cliente: ");
        scanf("%ld", & id); fflush(stdin);
        if(fseek(fp, (id - 1)*sizeof(Cliente), SEEK_SET)!= 0){
            printf("Cliente inexistente ou problemas no cliente!!!");
            system("pause");
            return;
        }
        if(fread(&x, sizeof(Cliente), 1, fp)!= 1){
            printf("Problema na leitura do cliente!!!");
            system("pause");
        }
    }

```



```

        return;
    }
    if(x.status=='*'){
        printf("Cliente já está apagado!!!\n\n");
        return;
        system("pause");
    }
    printf("\n\n Clientes atuais \n\n");
    Mostrar_Cliente(x);
    printf("\n\n Apagar o cliente (s/n)???: "); resp = getchar();
    fflush(stdin);
    if(toupper(resp) != 'S')return;

    x.status= '*';
    // recuar um cliente no arquivo
    fseek(fp, -(long) sizeof(Cliente), SEEK_CUR);
    // reescrever o cliente;
    fwrite(&x, sizeof(Cliente), 1, fp);
    fflush(fp); /*Despejar os arquivos no disco rígido*/
    system("cls");
    printf("\nCliente excluido com sucesso!\n");
}

void ListarCliente(){
    system("cls");
    Cliente reg;
    rewind(fp);
    int cont=0;
    while(1){
        if(fread(&reg, sizeof(reg), 1, fp) != 1){
            break; /*Sair do laço*/
        }else{
            cont=cont+1;
        }
        if(reg.status=='*'){
            continue; /*Passa ao próximo*/
        }
        printf("ID: %d\n", cont);
        Mostrar_Cliente(reg);
    }
}

//Faz CRUD de funcionario
void InserirFuncionario(){
    system("cls");
    Funcionario x;

```

```

        Ler_Funcionario(&x);
        Adiciona_Funcionario(x);
    }

void AlterarFuncionario(){
    system("cls");
    Funcionario x;
    long int id;
    printf("Qual o numero do funcionario: ");
    scanf("%ld", & id); fflush(stdin);
    if(fseek(fp, (id-1)*sizeof(Funcionario), SEEK_SET)!=0){
        printf("Funcionario inexistente!!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Funcionario), 1, fp)!= 1){
        printf("Problemas na leitura do funcionario!!!");
        system("pause");
        return;
    }

    if(x.status=='*'){
        printf("Um funcionario apagado não pode ser alterado!!! \n\n");
        system("pause");
        return;
    }

    printf("\n\n Funcionarios Atuais \n\n");
    Mostrar_Funcionario(x);
    printf("\n\n Novos funcionario \n\n");
    Ler_Funcionario(&x);
    // recuar um funcionario no arquivo
    fseek(fp, -(long) sizeof(Funcionario), SEEK_CUR);
    // reescrever o funcionario;
    fwrite(&x, sizeof(Funcionario), 1, fp);
    fflush(fp); /*despejar os arquivos no disco rígido*/
    printf("\nFuncionario alterado com sucesso!\n");
    system("pause");
}

void ApagarFuncionario(){
    system("cls");
    Funcionario x;
    long int id;
    char resp;

```

```

printf("Qual o numero do funcionario: ");
scanf("%ld", & id); fflush(stdin);
if(fseek(fp, (id - 1)*sizeof(Funcionario), SEEK_SET)!= 0){
    printf("Funcionario inexistente ou problemas no funcionário!!!");
    system("pause");
    return;
}
if(fread(&x, sizeof(Funcionario), 1, fp)!= 1){
    printf("Problema na leitura do funcionario!!!");
    system("pause");
    return;
}
if(x.status=='*'){
    printf("Funcionario já está apagado!!!\n\n");
    return;
    system("pause");
}
printf("\n\n Funcionarios atuais \n\n");
Mostrar_Funcionario(x);
printf("\n\n Apagar o funcionario (s/n)???: "); resp = getchar();
fflush(stdin);
if(toupper(resp)!= 'S')return;

x.status= '*';
// recuar um funcionario no arquivo
fseek(fp, -(long) sizeof(Funcionario), SEEK_CUR);
// reescrever o funcionario;
fwrite(&x, sizeof(Funcionario), 1, fp);
fflush(fp); /*Despejar os arquivos no disco rígido*/
system("cls");
printf("\nFuncionario excluido com sucesso!\n");
}

void ListarFuncionario(){
    system("cls");
    Funcionario reg;
    rewind(fp);
    int cont=0;
    while(1){
        if(fread(&reg, sizeof(reg), 1, fp)!= 1){
            break; /*Sair do laço*/
        }else{
            cont=cont+1;
        }
        if(reg.status=='*'){
            continue; /*Passa ao próximo*/
        }
    }
}

```

```

        }
        printf("ID: %d\n", cont);
        Mostrar_Funcionario(reg);
    }
}

//Faz CRUD de fornecedor
void InserirFornecedor(){
    system("cls");
    Fornecedor x;
    Ler_Fornecedor(&x);
    Adiciona_Fornecedor(x);
}

void AlterarFornecedor(){
    system("cls");
    Fornecedor x;
    long int id;
    printf("Qual o numero do fornecedor: ");
    scanf("%ld", & id); fflush(stdin);
    if(fseek(fp, (id-1)*sizeof(Fornecedor), SEEK_SET)!=0){
        printf("Fornecedor inexistente!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Fornecedor), 1, fp)!= 1){
        printf("Problemas na leitura do fornecedor!!!");
        system("pause");
        return;
    }

    if(x.status=='*'){
        printf("Um fornecedor apagado não pode ser alterado!!! \n\n");
        system("pause");
        return;
    }

    printf("\n\n Fornecedor Atuais \n\n");
    Mostrar_Fornecedor(x);
    printf("\n\n Novos fornecedores \n\n");
    Ler_Fornecedor(&x);
    // recuar um fornecedor no arquivo
    fseek(fp, -(long) sizeof(Fornecedor), SEEK_CUR);
    // reescrever o fornecedor;
    fwrite(&x, sizeof(Fornecedor), 1, fp);
    fflush(fp); /*despejar os arquivos no disco rígido*/
}

```

```

        printf("\nFornecedor alterado com sucesso!\n");
        system("pause");
    }

void ApagarFornecedor(){
    system("cls");
    Fornecedor x;
    long int id;
    char resp;

    printf("Qual o numero do Fornecedor: ");
    scanf("%ld", & id); fflush(stdin);
    if(fseek(fp, (id - 1)*sizeof(Fornecedor), SEEK_SET)!= 0){
        printf("Fornecedor inexistente ou problemas no cliente!!!");
        system("pause");
        return;
    }
    if(fread(&x, sizeof(Fornecedor), 1, fp)!= 1){
        printf("Problema na leitura do fornecedor!!!");
        system("pause");
        return;
    }
    if(x.status=='*'){
        printf("Fornecedor já está apagado!!!\n\n");
        return;
        system("pause");
    }
    printf("\n\n Fornecedor atuais \n\n");
    Mostrar_Fornecedor(x);
    printf("\n\n Apagar o fornecedor (s/n)???: "); resp = getchar();
    fflush(stdin);
    if(toupper(resp)!= 'S')return;

    x.status= '*';
    // recuar um fornecedor no arquivo
    fseek(fp, -(long) sizeof(Fornecedor), SEEK_CUR);
    // reescrever o fornecedor;
    fwrite(&x, sizeof(Fornecedor), 1, fp);
    fflush(fp); /*Despejar os arquivos no disco rígido*/
    system("cls");
    printf("\nFornecedor excluido com sucesso!\n");
}

void ListarFornecedor(){
    system("cls");
    Fornecedor reg;

```

```

rewind(fp);
int cont=0;
while(1){
    if(fread(&reg, sizeof(reg), 1, fp) != 1){
        break; /*Sair do laço*/
    }else{
        cont=cont+1;
    }
    if(reg.status=='*'){
        continue; /*Passa ao próximo*/
    }
    printf("ID: %d\n", cont);
    Mostrar_Fornecedor(reg);
}
}

/*Exibe mensagem de despedida ao sair do programa*/
void msgSair(){
    system("cls");
    printf("Obrigado por utilizar nosso software.\n\nTenha um bom dia!\n\n");
    exit(1);
}

void login(){
    char senharecebida[10], loginrecebido[10], opcao;
    char loginmaster[6]="admin";
    char senhamaster[6]="admin";
    char loginpadrao[10]="user";
    char senhapadrao[10]="user";

    do{
        printf("\n Login: ");
        scanf("%s",&loginrecebido);
        fflush(stdin);
        printf("\n Senha: ");
        scanf("%s",&senharecebida);
        fflush(stdin);

        if(strcmp(loginrecebido,loginmaster)==0&&strcmp(senharecebida,senhamaster)
        ==0){
            return;

```

[illegible]


```

        system("cls");
        goto MenuPrincipalUsuario;
        break;
    }

    msgSair();
    break;

case '2':
    system("cls");
    int unidadeVendaProduto;
    while((opcao =
FuncoesMenuVendas(MenuVendas))!= SAIR)
        switch(opcao){
            case '1':
                MontarVenda();
                break;
            case '2':
                do{
                    system("cls");
                    printf("Escolha uma unidade de
venda: \n\n 1- Unidade 1 \n 2- Unidade 2 \n 3- Unidade 3  \n");
                    printf ("\n Escolha a opção
(1,2,3): ");

                    scanf("%d",&unidadeVendaProduto);
                }while((unidadeVendaProduto < 1 ) ||
(unidadeVendaProduto > 3));

                switch (unidadeVendaProduto){
                    case 1:
                        AbrirArquivoVenda1();
                        ListarVenda();
                        system("pause");

                        system("cls");
                        fflush(stdin);
                        fclose(fp);

                        break;
                    case 2:
                        AbrirArquivoVenda2();
                        ListarVenda();
                        system("pause");

                        system("cls");
                        fflush(stdin);
                        fclose(fp);

                        break;
                }
            }
        }
    }
}

```

```

        case 3:
            AbrirArquivoVenda3();
            ListarVenda();
            system("pause");

        system("cls");
        fflush(stdin);
        fclose(fp);

        break;
    }
    break;
case '3':
    do{
        system("cls");
        printf("Escolha uma unidade de
venda: \n\n 1- Unidade 1 \n 2- Unidade 2 \n 3- Unidade 3 \n");
        printf ("\n Escolha a opção
(1,2,3): ");

        scanf("%d",&unidadeVendaProduto);
    }while((unidadeVendaProduto < 1 ) ||
(unidadeVendaProduto > 3));

    switch (unidadeVendaProduto){
        case 1:
            AbrirArquivoVenda1();
            ApagarVenda();
            system("pause");

            system("cls");
            fflush(stdin);

            AdicionarEstoqueProdutoCancelado(idProdutoVenda);
            fclose(fp);

            break;
        case 2:
            AbrirArquivoVenda2();
            ApagarVenda();
            system("pause");

            system("cls");
            fflush(stdin);

            AdicionarEstoqueProdutoCancelado(idProdutoVenda);
            fclose(fp);

            break;
        case 3:
            AbrirArquivoVenda3();
            ApagarVenda();

```

```

                                system("pause");
                                system("cls");
                                fflush(stdin);

AdicionarEstoqueProdutoCancelado(idProdutoVenda);
                                fclose(fp);
                                break;
                                }
                                case '4':
                                system("cls");
                                goto MenuPrincipalUsuario;
                                break;

                                }

                                case '3':
                                system("cls");
                                //MenuOpcoes
                                while((opcao = FuncoesMenuOpcoes(MenuOpcoes))!= SAIR)
                                switch(opcao){
                                case '1':
                                break;
                                case '2':
                                break;
                                case '3':
                                break;
                                case '4':
                                system("cls");
                                goto MenuPrincipalUsuario;
                                break;
                                }
                                break;

                                }
                                msgSair();
                                }
                                else{
                                printf("login incorreto");
                                }
                                }

                                }while(strcmp(loginrecebido,loginmaster)!=0 ||
                                strcmp(loginrecebido,loginpadrao)!=0);

}

/*Funções do menu de venda */

```

```

void TratamentoErroVenda(Produto p){
    //Se estoque de produto for <= 0
    if (p.quantidadeProduto <= 0){
        system("cls");
        printf("Produto informado sem estoque\n");
        system("pause");
        exit(1);
    }

    //Se valor de venda e promocional for <= 0

}

void PesquisarProduto(Produto *p){
pesquisarProduto:

AbrirArquivoProduto();
system("cls");

    Produto x;
    printf("Qual o numero do produto: ");
    scanf("%ld", & idProdutoVenda); fflush(stdin);

    if(fseek(fp, (idProdutoVenda-1)*sizeof(Produto), SEEK_SET)!=0){
        printf("produto inexistente!!!");
        system("pause");
        goto pesquisarProduto;
    }
    if(fread(&x, sizeof(Produto), 1, fp)!= 1){
        printf("Problemas na leitura do produto!!!");
        system("pause");

        goto pesquisarProduto;
    }

    if(x.status=='*'){
        printf("Um produto apagado não pode ser vendido!!! \n\n");
        system("pause");

        goto pesquisarProduto;
    }

    TratamentoErroVenda(x);

    printf("\n\n Produto escolhido \n\n");
    Mostrar_Produto(x);

```

```

        ProdutoVendido = x;
        system("pause");

    }

    void PesquisarCliente(Cliente *c){
        pesquisarCliente:

        AbrirArquivoCliente();
        system("cls");

        long int id;
        Cliente x;

        printf("Qual o numero do cliente: ");
        scanf("%ld", & id); fflush(stdin);

        if(fseek(fp, (id-1)*sizeof(Cliente), SEEK_SET)!=0){
            printf("Cliente inexistente!!!");
            system("pause");
            goto pesquisarCliente;
        }
        if(fread(&x, sizeof(Cliente), 1, fp)!= 1){
            printf("Problemas na leitura do cliente!!!");
            system("pause");
            goto pesquisarCliente;
        }

        if(x.status=='*'){
            printf("Um cliente apagado não pode ser vendido!!! \n\n");
            system("pause");
            goto pesquisarCliente;
        }

        printf("\n\n Cliente escolhido \n\n");
        Mostrar_Cliente(x);

        ClienteEscolhido = x;
        system("pause");
    }

    void ExibirVenda(){
        system("cls");

        printf("=====RESUMO=====\\n");
    }

```

```

        printf("Cliente: %-30s \n", ClienteEscolhido.nomeCliente);
        printf("-----\n");
        printf("Produto: %-30s \n", ProdutoVendido.nomeProduto);
        printf("Valor unitário: %2.f\n", ProdutoVendido.valorPromocaoProduto);
        printf("Quantidade: %d \n", quantidade);
        printf("-----\n");
        //Calcular total da venda
        total = ProdutoVendido.valorPromocaoProduto * quantidade;

        printf("Total: %2.f\n", total);
        printf("-----\n\n");
        system("pause");
    }

void RetirarEstoqueProdutoVendido(long int id){
    AbrirArquivoProduto();

    Produto x;

    if(fseek(fp, (id-1)*sizeof(Produto), SEEK_SET)!=0){
        printf("produto inexistente!!!");
        system("pause");
        return;
    }

    if(fread(&x, sizeof(Produto), 1, fp)!= 1){
        printf("Problemas na leitura do produto!!!");
        system("pause");
        return;
    }

    if(x.status=='*'){
        printf("Um produto apagado não pode ser alterado!!! \n\n");
        system("pause");
        return;
    }

    x.quantidadeProduto = x.quantidadeProduto - 1;
    // recuar um produto no arquivo
    fseek(fp, -(long) sizeof(Produto), SEEK_CUR);
    // reescrever o produto;
    fwrite(&x, sizeof(Produto), 1, fp);
    fflush(fp); /*despejar os arquivos no disco rígido*/
}

```

```

void AdicionarEstoqueProdutoCancelado(long int id){
    AbrirArquivoProduto();

    Produto x;

    if(fseek(fp, (id-1)*sizeof(Produto), SEEK_SET)!=0){
        printf("produto inexistente!!!");
        system("pause");
        return;
    }

    if(fread(&x, sizeof(Produto), 1, fp) != 1){
        printf("Problemas na leitura do produto!!!");
        system("pause");
        return;
    }

    if(x.status=='*'){
        printf("Um produto apagado não pode ser alterado!!! \n\n");
        system("pause");
        return;
    }

    x.quantidadeProduto = x.quantidadeProduto + 1;
    // recuar um produto no arquivo
    fseek(fp, -(long) sizeof(Produto), SEEK_CUR);
    // reescrever o produto;
    fwrite(&x, sizeof(Produto), 1, fp);
    fflush(fp); /*despejar os arquivos no disco rígido*/
}

void MontarVenda(){
    char opcaoDigitada;
    int unidadeVendaProduto;

    system("cls");

    Produto p;
    Cliente c;

    PesquisarProduto(&p);
    PesquisarCliente(&c);
    system("cls");

    printf("Digite a quantidade: ");
    scanf("%d", &quantidade);
}

```



```

ExibirVenda();

system("cls");
printf("Deseja concretizar a venda ? (s/n)");
scanf("%s", &opcaoDigitada);

if ((opcaoDigitada == 's') || (opcaoDigitada == 'S')){
    do{
        system("cls");
        printf("Escolha uma unidade de venda: \n\n 1- Unidade 1 \n\n 2- Unidade 2 \n 3- Unidade 3  \n");
        printf ("\n Escolha a opção (1,2,3): ");
        scanf("%d",&unidadeVendaProduto);
    }while((unidadeVendaProduto < 1 ) || (unidadeVendaProduto > 3));

    switch (unidadeVendaProduto){
        case 1:
            //Caso unidade 1
            AbrirArquivoVenda1();
            InserirVenda();
            printf("Venda salva com sucesso!\n");
            fclose(fp);
            RetirarEstoqueProdutoVendido(idProdutoVenda);
            system("pause");
            system("cls");
            return;
            break;

        case 2:
            //Caso unidade 2
            AbrirArquivoVenda2();
            InserirVenda();
            printf("Venda salva com sucesso!\n");
            fclose(fp);
            RetirarEstoqueProdutoVendido(idProdutoVenda);
            system("pause");
            system("cls");
            return;
            break;

        case 3:
            //Caso unidade 3
            AbrirArquivoVenda3();
            InserirVenda();
            printf("Venda salva com sucesso!\n");
            fclose(fp);
            RetirarEstoqueProdutoVendido(idProdutoVenda);

```

```

        system("pause");
        system("cls");
        return;
        break;
    }
}
else{
    printf("Venda não salva\n");
    system("pause");
    fflush(stdin);
    system("cls");
}
}

int CalcularVenda(){
    float valorVenda=0, valorTotalGastos;
    int i, j;
    Venda x[700];
    rewind (fp);
    do{

        fread(&x, sizeof(x), 1, fp);
        for(i=0; i<50; i++){
            if(x[i].status == '*') {
                continue;
            }

            if(valorVenda == 0){
                valorVenda = x[i].total;
            }
            valorVenda= valorVenda + x[i].total;
        }

    }while(i!=50);
    return (valorVenda);
}

```

//Função que verifica qual a unidade e abre o txt de acordo com a escolha

```

void VerificaGasto(Gastos g){
    Ler_Gastos(&g);
    system("cls");
    if(g.unidadeGasto== 1){
        AbrirArquivoGastos1();
    }
}

```

```

        Adiciona_Gasto(g);
        fclose(fp);
    }
    if(g.unidadeGasto == 2){
        AbrirArquivoGastos2();
        Adiciona_Gasto(g);
        fclose(fp);
    }
    if(g.unidadeGasto == 3){
        AbrirArquivoGastos3();
        Adiciona_Gasto(g);
        fclose(fp);
    }
}

int main(int argc, char *argv[]) {

    setlocale(LC_ALL, "Portuguese"); /* Torna possível a utilização de
    acentuação e caracteres especiais no programa. */

    char opcao;
    float
    TotalGastos1, TotalGastos2, TotalGastos3, TotalVenda1, TotalVenda2, TotalVenda3
    ;
    float TotalUnidade1, TotalUnidade2, TotalUnidade3, TotalRede;
    Gastos g;

    login();
    system("cls");

    //Menu Principal
    menuPrincipal: //GOTO
    while((opcao = FuncoesMenuPrincipal(MenuPrincipal))!= SAIR)
        switch(opcao){
            case '1':
                system("cls");
                menuCadastro: //GOTO
                //Menu Cadastro
                while((opcao =
    FuncoesMenuCadastro(MenuCadastro))!= SAIR)
                    switch(opcao){
                        case '1':
                            system("cls");
                            //Menu Produto
                            AbrirArquivoProduto();

```

```

                                while((opcao =
FuncoesMenuProduto(MenuProduto))!= SAIR)
                                switch(opcao){

                                case '1':

                                InserirProduto();

                                system("cls");

                                printf("\nProduto inserido com sucesso!\n");
                                system("pause");
                                system("cls");
                                break;

                                case '2':

                                AlterarProduto();

                                system("cls");
                                break;

                                case '3':

                                ApagarProduto();
                                system("pause");
                                system("cls");
                                break;

                                case '4':

                                ListarProduto();
                                system("pause");
                                system("cls");
                                break;

                                case '5':

                                system("cls");
                                goto menuCadastro;
                                break;

                                }
                                msgSair();
                                break;
                                case '2':

                                system("cls");

                                //Menu Cliente
                                AbrirArquivoCliente();
                                while((opcao =
FuncoesMenuCliente(MenuCliente))!= SAIR)
                                switch(opcao){

                                case '1':

```

```

InserirCliente();

system("cls");

printf("\nCliente inserido com sucesso!\n");
system("pause");
system("cls");
break;
case '2':

AlterarCliente();

system("cls");
break;
case '3':
    ApagarCliente();
    system("pause");
    system("cls");
    break;
case '4':
    ListarCliente();
    system("pause");
    system("cls");
    break;
case '5':
    system("cls");
    goto menuCadastro;
    break;
}
msgSair();
break;
case '3':
    system("cls");
//Menu Funcionario

while((opcao =
FuncoesMenuFuncionario(MenuFuncionario))!= SAIR)
switch(opcao){

case '1':

InserirFuncionario();

system("cls");

printf("\nFuncionario inserido com sucesso!\n");
system("pause");
system("cls");

```

```
break;

case '2':

AlterarFuncionario();

system("cls");
break;

case '3':

ApagarFuncionario();

system("pause");
system("cls");
break;

case '4':

ListarFuncionario();

system("pause");
system("cls");
break;

case '5':
system("cls");
goto menuCadastro;
break;

}
msgSair();
break;

case '4':
system("cls");
//Menu Fornecedor
AbrirArquivoFornecedor();
while((opcao =
FuncoesMenuFornecedor(MenuFornecedor))!= SAIR)
switch(opcao){

case '1':

InserirFornecedor();

system("cls");

printf("\nFornecedor inserido com sucesso!\n");

system("pause");
system("cls");
break;

case '2':

AlterarFornecedor();

system("cls");
```

```

                                break;

                                case '3':

ApagarFornecedor();

                                system("pause");
                                system("cls");
                                break;

                                case '4':

ListarFornecedor();

                                system("pause");
                                system("cls");
                                break;

                                case '5':
                                    system("cls");
                                    goto menuCadastro;
                                    break;

                                }
                                msgSair();
                                break;
                                case '5':
                                    system("cls");
                                    //Menu Gastos
MenuGastos://goto
                                    fflush(stdin);
                                    while((opcao= FuncoesMenuGastos(MenuGastos))!=
SAIR)

switch(opcao){
case '1':
    VerificaGasto(g);
    system("cls");
    printf("\nGastos inserido com sucesso!!");
    system("pause");
    system("cls");
    break;

case '2':
    system("cls");
    while((opcao=
FuncoesMenuGastos(MenuGastosUnidade))!= SAIR)
        switch(opcao){
            case '1':
                AbrirArquivoGastos1();
                AlterarGasto();
                system("pause");
                system("cls");

```

```

        fclose(fp);
        break;
    case '2':
        AbrirArquivoGastos2();
        AlterarGasto();
        system("pause");
        system("cls");
        fclose(fp);
        break;
    case '3':
        AbrirArquivoGastos3();
        AlterarGasto();
        system("pause");
        system("cls");
        fclose(fp);
        break;
    case '4':
        goto MenuGastos;
        break;
    }
    case '3':
        system("cls");
        while((opcao=
FuncoesMenuGastos(MenuGastosUnidade))!= SAIR)
        switch(opcao){
            case '1':
                AbrirArquivoGastos1();
                ApagarGasto();
                system("pause");
                system("cls");
                fclose(fp);
                break;
            case '2':
                AbrirArquivoGastos2();
                ApagarGasto();
                system("pause");
                system("cls");
                fclose(fp);
                break;
            case '3':
                AbrirArquivoGastos3();
                ApagarGasto();
                system("pause");
                system("cls");
                fclose(fp);
                break;

```



```

        case '4':
            goto MenuGastos;
            break;
        }
        break;
    case '4':
        system("cls");
        while((opcao=
FuncoesMenuGastos(MenuGastosUnidade))!= SAIR)
            switch(opcao){
                case '1':
                    AbrirArquivoGastos1();
                    ListarGasto();
                    system("pause");
                    system("cls");
                    fclose(fp);
                    break;
                case '2':
                    AbrirArquivoGastos2();
                    ListarGasto();
                    system("pause");
                    system("cls");
                    fclose(fp);
                    break;
                case '3':
                    AbrirArquivoGastos3();
                    ListarGasto();
                    system("pause");
                    system("cls");
                    fclose(fp);
                    break;
                case '4':
                    goto MenuGastos;
                    break;
            }
            break;
    case '5':
        system("cls");
        goto menuCadastro;
        break;
    }

    case '6':
        system("cls");
        goto menuPrincipal;
        break;

```

```

        msgSair();
    }
    case '2':
        menuVendas:
        system("cls");
        //MenuVendas
        int unidadeVendaProduto;
        while((opcao =
FuncoesMenuVendas(MenuVendas))!= SAIR)
        switch(opcao){
            case '1':
                MontarVenda();
                break;
            case '2':
                do{
                    system("cls");
                    printf("Escolha uma unidade de
venda: \n\n 1- Unidade 1 \n 2- Unidade 2 \n 3- Unidade 3  \n");
                    printf ("\n Escolha a opção
(1,2,3): ");

                    scanf("%d",&unidadeVendaProduto);
                }while((unidadeVendaProduto < 1 ) ||
(unidadeVendaProduto > 3));

                switch (unidadeVendaProduto){
                    case 1:
                        AbrirArquivoVenda1();
                        ListarVenda();
                        system("pause");

                        system("cls");
                        fflush(stdin);
                        fclose(fp);

                        break;
                    case 2:
                        AbrirArquivoVenda2();
                        ListarVenda();
                        system("pause");

                        system("cls");
                        fflush(stdin);
                        fclose(fp);

                        break;
                    case 3:
                        AbrirArquivoVenda3();

```

```

        ListarVenda();
        system("pause");

        system("cls");
        fflush(stdin);
        fclose(fp);

        break;
    }
    break;
case '3':
    do{
        system("cls");
        printf("Escolha uma unidade de
venda: \n\n 1- Unidade 1 \n 2- Unidade 2 \n 3- Unidade 3  \n");
        printf ("\n Escolha a opção
(1,2,3): ");

        scanf("%d",&unidadeVendaProduto);
    }while((unidadeVendaProduto < 1 ) ||
(unidadeVendaProduto > 3));

    switch (unidadeVendaProduto){
        case 1:
            AbrirArquivoVenda1();
            ApagarVenda();
            system("pause");

            system("cls");
            fflush(stdin);

            AdicionarEstoqueProdutoCancelado(idProdutoVenda);
            fclose(fp);

            break;
        case 2:
            AbrirArquivoVenda2();
            ApagarVenda();
            system("pause");

            system("cls");
            fflush(stdin);

            AdicionarEstoqueProdutoCancelado(idProdutoVenda);
            fclose(fp);

            break;
        case 3:
            AbrirArquivoVenda3();
            ApagarVenda();
            system("pause");

            system("cls");

```

```

        fflush(stdin);

        AdicionarEstoqueProdutoCancelado(idProdutoVenda);
        fclose(fp);
        break;
    }
    case '4':
        system("cls");
        goto menuPrincipal;
        break;

    }

    case '3':
        system("cls");
        //MenuRelatorios
        while((opcao =
FuncoesMenuRelatorios(MenuRelatorios))!= SAIR)
        switch(opcao){
            case '1':
                break;
            case '2':
                system("cls");

while((opcao=FuncoesMenuRelatorios(MenuGastosUnidade))!= SAIR)
        switch(opcao){
            case '1':
                AbrirArquivoGastos1();
                TotalGastos1=CalcularGasto();
                fclose(fp);
                AbrirArquivoVenda1();
                TotalVenda1=CalcularVenda();
                fclose(fp);
                TotalUnidade1=TotalVenda1-(TotalGastos1);
                system("cls");
                printf("\n0 faturamento da unidade1 é de:
%.2f\n\n",TotalUnidade1);

                system("pause");
                system("cls");
                break;
            case '2':
                AbrirArquivoGastos2();
                TotalGastos2=CalcularGasto();
                fclose(fp);
                AbrirArquivoVenda2();
                TotalVenda2=CalcularVenda();

```

```

        fclose(fp);
        TotalUnidade2=TotalVenda2-(TotalGastos2);
        system("cls");
        printf("\nO faturamento da unidade2 é de:
%.2f\n\n",TotalUnidade2);

        system("pause");
        system("cls");
        break;
    case '3':
        AbrirArquivoGastos3();
        TotalGastos3=CalcularGasto();
        fclose(fp);
        AbrirArquivoVenda3();
        TotalVenda3=CalcularVenda();
        fclose(fp);
        TotalUnidade3=TotalVenda3-(TotalGastos3);
        system("cls");
        printf("\nO faturamento da unidade3 é de:
%.2f\n\n",TotalUnidade3);

        system("pause");
        system("cls");
        break;
    case '4':
        system("cls");
        goto menuPrincipal;
        system("cls");
        break;

    }

        break;
    case '3':

TotalRede=TotalUnidade1+TotalUnidade2+TotalUnidade3;
        printf("\n O faturamento da rede é de:
%.2f\n\n",TotalRede);

        system("pause");
        system("cls");

        break;
        case '4':
        system("cls");
        goto menuPrincipal;
        break;

    }

    break;

```

```

case '4':
    system("cls");
    //MenuOpcoes
    while((opcao = FuncoesMenuOpcoes(MenuOpcoes))!= SAIR)
        switch(opcao){
            case '1':
                //Sobre
                system("cls");
                printf("SOBRE O SOFTWARE:\n\n");
                printf("O projeto se consiste em
funcionalidades que toda pizzaria precisa, porém, este sistema contém
funções automatizadas e seguras para que o funcionário trabalhe de forma
eficiente e prática, sempre visando a produtividade, consequentemente
gerando mais rentabilidade, maior desempenho e agilidade nas tarefas
diárias da pizzaria.\n\n");

                printf("CÓDIGO FONTE EM:
\nhttps://github.com/GabrielFernandesLemos/Controle-financeiro-em-C\n\n");
                printf("CONTRIBUIDORES:\n\n");
                printf("- Gabriel Fernandes Lemos
\n\n- Matheus Nunes Nepomuceno\n\n- Erik Hideyuki Yoshimoto Seki\n\n-
Clayton Belarmino da Silva\n\n- Gabriel Franco Garcia Rodrigues de
Paula\n\n");

                system("pause");
                system("cls");
                break;
            case '2':
                //Manual
                printf("Faça seu login no sistema e
comece a usufruir das suas funcionalidades. \n\nEste manual foi feito pelo
sistema operacional Windows 7. \n\nPara iniciar a configuração Banda
Larga, Clique em Iniciar. Em seguida clique em Painel de Controle Clique
na opção Rede e Internet \n\nEm seguida escolha a opção Centro de Rede e
Compartilhamento No menu à direita, clique na opção Configurar uma conexão
ou uma rede nova. \n\nClique na opção de conexão Conectar-se à Internet e
depois em Avançar. \n\nClique em Banda larga (PPPoE). \n\nPreencha o campo
Nome de usuário com o endereço de e-mail de conexão completo (Ex.:
username@terra.com.br), abaixo, insira a senha e em seguida dê um nome
para a conexão. \n\nImportante ressaltar, marque a opção Lembrar esta
senha para não precisar digitá-la sempre que for estabelecer a conexão.
\n\nMarque a opção ☐Permitir que outras pessoas usem esta conexão☐ caso o
computador seja acessado por outras pessoas através de suas contas de
usuário. \n\nClique em ☐Conectar☐. Aguarde enquanto o sistema estabelece a
conexão e feche a tela após a conexão ser estabelecida. \n\nSempre que
desejar conectar, basta acessar a conexão criada e clicar em ☐Conectar☐.
\n\nO sistema é bem intuitivo e de fácil aprendizado. \n\nA tela mais
importante é a de cadastros, na qual poderá cadastrar Produtos, clientes,

```

funcionários e gastos Para cadastrar um produto, aperte a tecla 1. \n\nEm seguida o programa irá lhe solicitar o que deseja fazer, seja inserir, alterar, apagar ou ligar um produto. \n\nAperte o número em seu teclado referente ao que deseja fazer e faça sua ação. \n\nO mesmo funciona para cadastros de clientes, funcionários e gastos. \n\nO menu de vendas é utilizado para realizar vendas, ele contém as opções de gerar nova venda, listar vendas e fazer o cancelamento de uma venda Aperte o número em seu teclado referente ao que deseja fazer e faça sua ação. \n\nA tela de relatórios somente o administrador do sistema tem acesso, pois nela tem contém informação de faturamento onde um funcionário comum não tem acesso. \n\nEssa opção tem disponível sabores mais vendidos por unidade, faturamento mensal e faturamento total. \n\nPara visualizar essas opções, basta aperta em seu teclado o número referente a ação que irá fazer\n\n");

```
        system("pause");
        system("cls");
        break;
    case '3':
        //Voltar
        system("cls");
        goto menuPrincipal;
        break;
    }
    break;
}
msgSair();
return 0;
}
```

REFERÊNCIAS

Sarroglia. **Introdução.** Disponível em:
<<https://www.inf.pucrs.br/~pinho/Laprol/Historico/Historico.htm>>. Acessado em:
03/10/2019

Tania. **Metodologia utilizada.** Disponível em:
<<https://br.answers.yahoo.com/question/index?qid=20090421092029AADqUAB>>.
Acessado em 07/10/2019

Brasilecola. **Linguagem C e suas ferramentas.**
<<https://monografias.brasilecola.uol.com.br/computacao/fundamentos-linguagem-c.htm>>. Acessado em: 29/10/2019

João Victor. **Diagrama da representação da rede de comunicação.**
<<https://www.guiadaengenharia.com/diagramas-redes-elementos/> João>.
Acessado em: 01/11/2019

Qostecnologia. **Especificação do diagrama de redes.**
<<https://www.qostecnologia.com.br/11-tipos-de-servidores-de-redes/>>. Acessado em: 04/11/2019

Santos. **Cronograma de desenvolvimento e implantação.** Disponível em:
<<http://www.projectbuilder.com.br/blog-home/entry/conhecimentos/entenda-a-diferenca-entre-eap-e-cronograma-de-projetos>>. Acessado em: 06/11/2019

Terra. **Manual de configuração da rede.** Disponível em: <
<https://duvidas.terra.com.br/duvidas/3883/como-configuro-minha-conexao-banda-larga-no-windows-7>>. Acessado em: 08/11/2019

Scribd. **Embasamento para o código.** Disponível em:
<<https://pt.scribd.com/document/79965690/Codigo-Limpo-Completo-PT>>.
Acessado em: 05/10/2019