



**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RN
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**Análise e Implementação de um Software para Controle de Vendas e
Estoque para Pequenos Estabelecimentos**

**Análise e Implementação de um Software para Controle de Vendas e
Estoque para Pequenos Estabelecimentos**

Projeto de Final de Curso apresentado ao
curso de Bacharelado em Sistemas de
Informação da Faculdade Natalense para o
Desenvolvimento do Rio Grande do Norte
como requisito parcial para a obtenção do
título de Bacharel em Sistemas de
Informação.

Manoel Messias Braz Junior

Orientador: Felipe Araújo Aleixo, MSc.

NATAL

2005

MANOEL MESSIAS BRAZ JUNIOR
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RN
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Análise e Implementação de um Software para Controle de Vendas e Estoque
para Pequenos Estabelecimentos

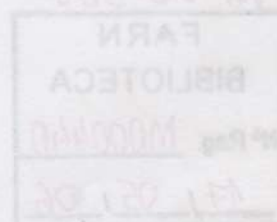
Análise e Implementação de um Software para Controle de Vendas e Estoque para Pequenos Estabelecimentos

Projeto de Final de Curso apresentado ao curso de Bacharelado em Sistemas de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte como requisito parcial para a obtenção do título em Bacharel em Sistemas de Informação.

Orientador: Felliipe Araújo Aleixo, MsC.

NATAL

2005



Catálogo na Publicação – Biblioteca da FARN
Setor de Processos Técnicos

Braz Júnior, Manoel Messias

Análise e implementação de um software para controle e vendas e estoque para pequenos estabelecimentos / Manoel Messias Braz Júnior. – Natal: FARN, 2005.
84 f.

Orientador: Fellipe Araújo Aleixo (M.Sc.)

Monografia (Graduação do Bacharelado em Sistemas de Informação) – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

1. Desenvolvimento de Software – Monografia. 2. Controle de Estoques – Monografia. 3. Software Livre – Monografia. I. Título.

RN/FARN/BC

CDU 004.41

RB-26.560

FARN
BIBLIOTECA
Nº Reg. M000440
17/05/06

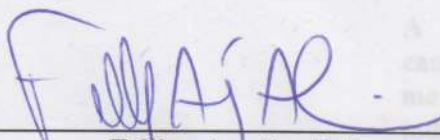
M
004.41
B839a
2005

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RN
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

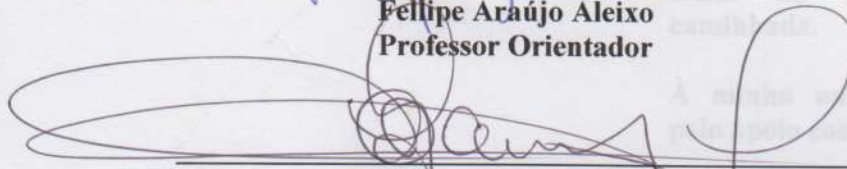
**Análise e Implementação de um Software para Controle de Vendas e Estoque
para Pequenos Estabelecimentos**

Manoel Messias Braz Júnior

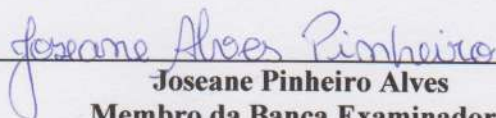
**Monografia apresentada e aprovada em 17 de FEVEREIRO de 2005, pela
banca examinadora composta pelos seguintes membros.**



**Felipe Araújo Aleixo
Professor Orientador**



**Luiz Gonzaga Damasceno
Membro da Banca Examinadora**



**Joseane Pinheiro Alves
Membro da Banca Examinadora**

Natal/RN, 17 de 12 de 05.

Agradecimentos

A Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, pela disponibilidade de recursos para minha formação.

Aos professores de modo geral, pelo auxílio na minha formação acadêmica.

Ao professor Felipe Alcino, pela orientação e paciência na elaboração desta monografia.

Aos amigos de turma que estiveram ao meu lado me encorajando e compartilhando meus conhecimentos.

A Deus que iluminou minha caminhada e minha família que me incentivou e apoiou em todos os momentos dessa caminhada.

À minha namorada, Jacyara, pelo apoio constante.

A todos que me auxiliaram e incentivaram durante meu trabalho.

Agradecimentos

Sumário

1 INTRODUÇÃO.....	11
1.1 Identificativa.....	12
1.2 Metodologia.....	13
2 REFERENCIAL TEÓRICO.....	15
2.1 UML.....	15
2.1.1 Diagrama de Caso De Uso.....	16
A Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, pela disponibilidade de recursos para minha formação.	18
Aos professores de modo geral, pelo auxílio na minha formação acadêmica.	19
Ao professor Fellipe Aleixo, pela orientação e paciência na elaboração desta monografia.	20
Aos amigos de turma que estiveram ao meu lado me ensinando e compartilhando novos conhecimentos.	23
2.2 PHP (Personal Home Page).....	23
2.2.1 Estrutura de um programa PHP.....	26
2.2.2 Estrutura de Controle em PHP.....	26
2.2.2.1 Comandos Condicionais.....	26
2.2.2.2 Comandos de repetição.....	28
2.3 Dreamweaver.....	29
2.4 Apache.....	29
3 ANÁLISE E PROJETO DO SISTEMA DE CONTROLE DE VENDAS E ESTOQUE PARA PEQUENOS ESTABELECIMENTOS.....	31
3.1 Diagrama de Caso de Uso.....	31
3.2 Implementação dos Casos de Uso.....	33
3.2.1 Caso de Uso Fazer Vendas.....	33
3.2.1.1 Diagrama de Atividade Fazer Vendas.....	34
3.2.1.2 Telas e Códigos.....	35
3.2.2 Caso de Uso Visualizar Vendas.....	38
3.2.2.1 Diagrama de Atividade Visualizar Vendas.....	38
3.2.3 Caso de Uso Visualizar Estoque Mínimo.....	39
3.2.3.1 Diagrama de Atividade Visualizar Estoque Mínimo.....	40

2.2.1 Caso de Uso Inserir Produtos.....	40
2.2.2 Diagrama de Atividade Inserir Produtos.....	41

Sumário

1 INTRODUÇÃO.....	11
1.2 Justificativa.....	12
1.3 Metodologia.....	13
2. REFERENCIAL TEÓRICO.....	15
2.1 UML.....	15
2.1.1 Diagrama de Caso De Uso.....	16
2.1.2 Diagrama de Atividade.....	16
2.2 Diagrama de Entidade e Relacionamento.....	18
2.3 Banco de Dados.....	19
2.3.1 PostegreSQL.....	19
2.4 Software Livre.....	20
2.5 HTML.....	22
2.5.1 Comandos HTML – Tag's.....	23
2.6 PHP (Personal Home Page).....	23
2.6.1 Estrutura de um programa PHP.....	26
2.6.2 Estrutura de Controle em PHP.....	26
2.6.2.1 Comandos Condicionais	26
2.6.2.2 Comandos de repetição.....	28
2.7 Dreamweaver.....	29
2.8 Apache.....	29
3. ANÁLISE E PROJETO DO SISTEMA DE CONTROLE DE VENDAS E ESTOQUE PARA PEQUENOS ESTABELECIMENTOS.....	31
3.1 Diagrama de Caso de Uso.....	31
3.2 Implementação dos Casos de Uso.....	33
3.2.1 Caso de Uso Fazer Vendas.....	33
3.2.1.1 Diagrama de Atividade Fazer Vendas.....	34
3.2.1.2 Telas e Códigos.....	35
3.2.2 Caso de Uso Visualizar Vendas	38
3.2.2.1 Diagrama de Atividade Visualizar Vendas.....	38
3.2.3 Caso de Uso Visualizar Estoque Mínimo.....	39
3.2.3.1 Diagrama de Atividade Visualizar Estoque Mínimo.....	40

3.2.4 Caso de Uso Inserir Produtos.....	40
3.2.4.1 Diagrama de Atividade Inserir Produtos.....	41
3.2.4.2 Telas e Códigos.....	42
3.2.5 Caso de Uso Editar Produtos.....	44
3.2.5.1 Diagrama de Atividade Editar Produtos.....	44
3.2.6 Caso de Uso Eliminar Produtos.....	45
3.2.6.1 Diagrama de Atividade Eliminar Produtos.....	46
3.2.7 Caso de Uso Inserir Usuário.....	47
3.2.7.1 Diagrama de Atividade Inserir Usuário.....	47
3.2.8 Caso de Uso Editar Usuário.....	48
3.2.8.1 Diagrama de Atividade Editar Usuário.....	49
3.2.9 Caso de Uso Eliminar Usuário.....	50
3.2.9.1 Diagrama de Atividade Eliminar Usuário.....	50
3.3 Modelo de Dados (Banco de Dados).....	52
4. CONCLUSÃO.....	54
REFERÊNCIAS.....	55
6. ANEXOS	
Figura 15-Menu do gerente.....	36
Figura 16-Tela de fazer vendas.....	36
Figura 17-Consulta no banco de dados.....	37
Figura 18-Recibimento do valor do desconto da quantidade.....	37
Figura 19-Diagrama de atividade visualizar vendas.....	39
Figura 20-Diagrama de atividade visualizar estoque mínimo.....	40
Figura 21-Diagrama de atividade inserir produto.....	41
Figura 22-Tela de controlar produtos.....	42
Figura 23-Tela de cadastro de produtos.....	43
Figura 24-Inserção no banco de dados.....	43
Figura 25-Diagrama de atividade editar produto.....	45
Figura 26-Diagrama de atividade eliminar produto.....	46
Figura 27-Diagrama de atividade inserir usuário.....	48
Figura 28-Diagrama de atividade editar usuário.....	49
Figura 29-Diagrama de Atividade Eliminar Usuário.....	51
Figura 30-Diagrama de entidade e relacionamento.....	52
Figura 31-Modelo lógico do banco de dados.....	53

Figura 29 - Código PHP, Login.....

60

Figura 30 - Código PHP, Menu Gerente.....

61

Lista de Figuras

Figura 1-Diagrama parcial de caso de uso.....	16
Figura 2-Exemplo de um diagrama de atividades.....	17
Figura 3-Elementos que compõem um diagrama de atividades.....	18
Figura 4-Exemplo de um diagrama de atividade e relacionamento.....	19
Figura 5-Exemplo de Tag's.....	23
Figura 6-Diagramas de requisição para o PHP.....	25
Figura 7-Comando IF.....	27
Figura 8-Comando SWITCH.....	27
Figura 9-Comando WHILE.....	28
Figura 10-Comando DO... WHILE.....	28
Figura 11-Comando FOR.....	29
Figura 12-Diagramas de Caso de Uso que representa as ações do Gerente.....	32
Figura 13-Diagramas de Caso de Uso que representa as ações do funcionário.....	32
Figura 14-Diagramas de atividade fazer vendas.....	35
Figura 15-Tela do menu do gerente.....	36
Figura 16-Tela de fazer vendas.....	36
Figura 17-Consulta no banco de dados.....	37
Figura 18-Recebimento do valor do desconto da quantidade.....	37
Figura 19-Diagrama de atividade visualizar vendas.....	39
Figura 20-Diagrama de atividade visualizar estoque mínimo.....	40
Figura 21-Diagrama de atividade inserir produto.....	41
Figura 22-Tela de controlar produtos.....	42
Figura 23-Tela de cadastro de produtos.....	43
Figura 24-Inserção no banco de dados.....	43
Figura 25-Diagrama de atividade editar produto.....	45
Figura 26-Diagrama de atividade eliminar produto.....	46
Figura 27-Diagrama de atividade inserir usuário.....	48
Figura 28-Diagrama de atividade editar usuário.....	49
Figura 29-Diagramas de Atividade Eliminar Usuário.....	51
Figura 30-Diagrama de entidade e relacionamento.....	52
Figura 31-Modelo lógico do banco de dados.....	53

Figura 32: Código PHP, Login.....	60
Figura 33: Código PHP, Menu Gerente.....	61
Figura 34: Código PHP, Menu Funcionário.....	63
Figura 35: Código PHP, Fazer Vendas.....	68
Figura 36: Código PHP, Visualizar Vendas.....	70
Figura 37: Código PHP, Controlar Produtos.....	75
Figura 38: Código PHP, Visualizar Produtos.....	76
Figura 39: Código PHP, Controlar Acesso.....	81
Figura 40: Código PHP, Visualizar Acesso.....	83
Figura 41: Código PHP, Visualizar Estoque Mínimo.....	84

Resumo

Este trabalho apresenta a análise e implementação de uma ferramenta para o controle de estoque e gerência de vendas para uma pequena empresa. Solução esta a ser distribuída na forma de software livre. Este trabalho apresenta através dos casos de uso e dos diagramas de atividade o fluxo seqüencial das atividades do sistema, e as descrições narrativas dos processos. Apresenta também a implementação do sistema em PHP. É apresentada também a modelagem do banco de dados. Conclui que o desenvolvimento e a utilização de sistemas gratuitos podem auxiliar os usuários com inúmeros benefícios, contribuindo para uma considerável melhoria no gerenciamento das vendas.

Palavras chave: 1. Sistema Web, 2. Software Livre, 3. Controle de Estoque

Abstract

This analyzes presents implementation of a tool to control of supply and management to control what a for a small company is selling. This solution is distributed like a free software. It presents through the cases of use and the diagrams of activity the sequential flow of the system's activities and the descriptions, narratives of the processes. It also presents the implementation of the system in PHP. The modeling of the data base is also presented. It concludes that the development and the use of gratuitous systems can assist the users with innumerable benefits, contributing for a considerable improvement in the management of things that they are selling.

Key Words: 1. web system, 2. free software, 3. control of store

Empresas. Essas empresas necessitam de softwares amigáveis para atender as suas necessidades de informatização. Ao mesmo tempo, essas empresas, por serem de pequeno porte, não possuem recursos suficientes para investir nessa área. Para atender a essa demanda, uma solução são os "softwares livres", softwares que podem ser utilizados sem que seja necessário o pagamento de nenhuma licença. Uma parte dessas empresas atua no ramo de vendas diretas aos consumidores. Essas empresas possuem necessidades específicas, como a automatização das vendas, que poderiam ser facilmente atendidas por um software. Software que possa controlar o estoque e monitorar vendas, fazendo uma consulta rápida e fácil de seus produtos.

Com esse pensamento, o estudo realizado teve como objetivo a análise e implementação de um sistema de controle de estoque e de monitoria de venda, baseado na filosofia de software livre.

O sistema é desenvolvido em PHP, este fato, possibilita o desenvolvimento de um site amigável, além disso, ainda é utilizado o PostgreSQL como SGBD (sistema de gerenciamento de banco de dados).

1. INTRODUÇÃO

A presença da informática tem um papel importante na sociedade moderna. Nessa nova sociedade, não só o desempenho de qualquer atividade profissional, como o próprio relacionamento com o meio e com os semelhantes impõem o conhecimento da tecnologia dos computadores. A tecnologia traz vários benefícios, como os sistemas de apoio, possuindo inúmeras aplicações, influenciando diretamente nas atividades profissionais.

Atualmente cresce o número de micro e pequenas empresas. Essas empresas necessitam de softwares amigáveis para atender as suas necessidades de informatização. Ao mesmo tempo, essas empresas, por serem de pequeno porte, não possuem recursos suficientes para investir nessa área. Para atender a essa demanda, uma solução são os “softwares livres”, softwares que podem ser utilizados sem que seja necessário o pagamento de nenhuma licença. Boa parte dessas empresas atua no ramo de vendas diretas aos consumidores. Essas empresas possuem necessidades específicas, como a automatização das vendas, que poderiam ser facilmente atendidas por um software. Software que possa controlar o estoque e monitorar vendas, fazendo uma consulta rápida e fácil de seus produtos.

Com esse pensamento, o estudo realizado teve como objetivo a análise e implementação de um sistema de controle de estoque e de monitoria de venda, baseado na filosofia de software livre.

O sistema é desenvolvido em PHP, este fato, possibilitou o desenvolvimento de um site dinâmico, além disso, ainda é utilizado o PostgreSQL como SGBD (sistema de gerenciamento de banco de dados).

1.2 Justificativa

Com o aumento da utilização de sistemas computacionais como instrumento de apoio às atividades profissionais e com a necessidade de satisfazer os usuários tanto na funcionalidade do sistema como na realização de tarefas no menor espaço de tempo, as empresas buscam suprir as suas necessidades, utilizando softwares que possam garantir um maior conforto e uma maior segurança para quem o maneja.

O sistema proposto será de grande auxílio no processo de monitoria de venda e controle do estoque de uma micro-empresa, que visa ajudar a identificação de produtos por parte de quem o maneja, sendo uma ferramenta muito importante para um maior desenvolvimento da micro-empresa.

Através do sistema proposto o empresário terá uma maior comodidade, pois possuirá em mãos mais informações sobre seu estoque e vendas, gerando assim um maior monitoramento sobre seus produtos, proporcionando um maior controle da empresa.

1.3 Metodologia AL TEÓRICO

A pesquisa proposta neste projeto se classifica como uma pesquisa: exploratória, bibliográfica, qualitativa e laboratorial.

A pesquisa exploratória tem o ambiente natural como sua fonte direta de dados e o pesquisador como seu principal instrumento. Supõe o contato direto e prolongado do pesquisador com o ambiente e a situação que está sendo investigada, via de regra através do trabalho intensivo de campo. Tem como objetivo de encontrar vários conceitos e uma varredura em apostilas, tutoriais, livros, internet, etc. A pesquisa bibliográfica tem por finalidade conhecer as diferentes formas de contribuição científica que se realizaram sobre determinado assunto ou fenômeno, ela é elaborada a partir de materiais já publicados, constituídos principalmente de livros, artigos de periódicos, internet, citações e referências. A pesquisa de laboratório também foi usada pelo fato de se tratar de uma implementação de sistema e porque durante o desenvolvimento e pesquisa, foi utilizado materiais do tipo hardware e software.

1.3.1 Pesquisa (exploratória) bibliográfica

- Software livre;
- Ferramentas abertas para o desenvolvimento.

1.3.2 Observação direta intensiva

- Análise de requisitos do software.

1.3.4 Pesquisa laboratorial

- Implementação do sistema.

A UML é uma linguagem gráfica para a visualização, especificação, construção e documentação de artefatos de sistemas complexos de software. A UML proporciona uma forma-padrão para a preparação de plano de arquitetura de projetos de sistema.

Na linguagem UML, descrevemos o sistema através de diagramas que são uma apresentação gráfica de um conjunto de elementos, geralmente como gráfico de vértices (nós) e arestas (relacionamentos). São desenhados para permitir a visualização de um sistema sob diferentes perspectivas, neste sentido, um diagrama constitui uma projeção de um determinado sistema. Os diagramas utilizados na descrição desse estudo são: Diagrama de Ciclo de Uso e os Diagramas de Atividade.

2. REFERENCIAL TEÓRICO

2.1 UML (*Unified Modeling Language*)

É uma linguagem para a modelagem de sistemas. É usada para a visualização, especificação, construção e documentação de artefatos de um sistema complexo de software.

Os esforços para a criação da UML se iniciaram oficialmente em outubro de 1994, quando Rumbaugh se juntou a Booch na Rational (Rational Software Corporation). O foco inicial do projeto era a unificação dos métodos Booch e OMT (Object Modeling Technique). O esboço da versão 0.8 do Método Unificado (como então era chamado) foi lançado em outubro de 2005. Mais ou menos na mesma época, Jacobson se associou à Rational e o escopo do projeto da UML foi expandido com a finalidade de incorporar o OOSE (Object-Oriented Software Engineering). Os esforços resultaram no lançamento dos documentos da versão 0.9 da UML em junho de 1996. (Booch, Rumbaugh e Jacobson, 2000).

Para Larman (2000), UML é a abreviação de linguagem de modelagem unificada (*Unified Modeling Language*), uma notação (principalmente diagramática) para modelagem de sistemas. Também padroniza artefatos e notações, mas ela não define um processo-padrão de desenvolvimento.

Figura 1 A UML (*Unified Modeling Language*) é a linguagem-padrão para a elaboração de estrutura de projeto de software. A UML poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software. (Booch, Rumbaugh e Jacobson, 2000, p.13).

A UML é uma linguagem gráfica para a visualização, especificação, construção e documentação de artefatos de sistemas complexos de software. A UML proporciona uma forma-padrão para a preparação de plano de arquitetura de projetos de sistema.

Na linguagem UML descrevemos o sistema através de diagramas que são uma apresentação gráfica de um conjunto de elementos, geralmente como gráfico de vértices (itens) e arcos (relacionamentos). São desenhados para permitir a visualização de um sistema sob diferentes perspectivas, neste sentido, um diagrama constitui uma projeção de um determinado sistema. Os diagramas utilizados na descrição desse estudo são: Diagrama de Caso de Uso e os Diagramas de Atividade.

2.1.1 Diagrama de Caso de Uso

Segundo Larman (2000), caso de uso são descrições narrativas dos processos do domínio num formato de prosa estruturada. Como documento narrativo que descreve a sequência de eventos de um autor (um agente externo) que usa um sistema para completar um processo, caso de uso não são exatamente especificação de requisitos ou especificação funcional, mas ilustram e implicam requisitos na história que eles contam.

Um diagrama de caso de uso ilustra um conjunto de caso de uso para um sistema, os atores e a relação entre os atores e os casos de uso, Conforme a figura 1.

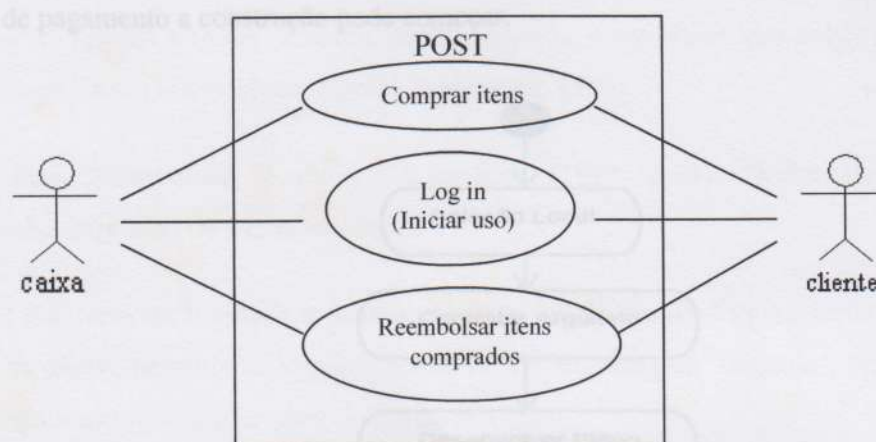


Figura 1: Diagrama parcial de caso de uso. (Larman, 2002, p.73).

Segundo Larman (2000, p.73), “Os casos de uso são ilustrados em ovais; os atores, como figuras de traço simples (palitos). Existem linhas de comunicação entre esses casos e os atores; as flechas podem indicar o fluxo de informação ou de estímulo.”

“A finalidade do diagrama de caso de uso é apresentar um tipo de diagrama de contexto, através do qual pode-se compreender rapidamente quais são os atores externos de um sistema e as maneiras principais, segundo as quais eles utilizam.” (Larman 2000, p.73).

2.1.2 Diagrama de Atividade

O diagrama de atividade mostra o fluxo sequencial das atividades, é normalmente utilizado para demonstrar as atividades executadas por uma operação específica do sistema. Consistem em estados de ação, que contém a especificação de uma atividade a ser desempenhada por uma operação do sistema. Decisões e condições, como execução paralela,

também podem ser mostradas na diagrama de atividade. O diagrama também pode conter especificações de mensagens enviadas e recebidas como partes de ações executadas. (Nogueira, 2005).

O diagrama de atividades mostra um fluxo de uma atividade para a outra, abrangendo uma visão dinâmica do sistema. Portanto, conforme mostra a figura 2, pode-se especificar o estado inicial (uma bola cheia) e o estado de parada (uma bola cheia num interior de um círculo). Considerando o fluxo de trabalho associado a construção de uma casa. Primeiro você seleciona o local. A seguir, contrata um arquiteto para projetar sua casa. Uma vez definida a planta, seu desenvolvedor determina os custos da casa. Após concordar com um preço e uma forma de pagamento a construção pode começar.



Figura 2: Exemplo de um diagrama de atividades.

No diagrama de atividade, o ponto inicial é representado por um pequeno círculo sólido e o ponto final é representado por um círculo maior com um círculo menor sólido dentro. As ações são desenhadas como retângulos com cantos arredondados e com a respectiva descrição na parte interna. Neste diagrama, um evento só é relacionado a uma transição do ponto inicial à primeira ação. As transições entre ações são representadas por uma flecha, onde freqüentemente nada é especificado indicando que a transição será disparada assim que todas as atividades nesta ação tenham sido executadas.

Figura 3: Elementos que compõem um diagrama de atividades.

2.2 Diagrama de Entidade e Relacionamento

“Constitui na técnica para representar a estrutura lógica de um banco de dados de modo pictórico. Com tal, fornece um meio simples e fácil de entender para comunicar os aspectos principais do projeto de qualquer banco de dados (“uma figura vale mil palavras”).”(Date, 2000, p.372).

O diagrama de Entidade-Relacionamento tem por base a percepção do mundo real, consistindo em um conjunto de objetos básicos chamados de entidades e relacionamentos.

Entidade é uma coisa ou objeto no mundo real que pode ser identificada de forma unívoca em relação a todos os outros objetos, ou seja, é um objeto que existe e é distinguível de outros objetos. (Silberschatz, Korth e Sudarshan, 1999)

Para Silberschatz, Korth e Sudarshan, (1999, p.24). “Relacionamento é uma associação entre uma ou varias entidades”.

O diagrama de Entidade e Relacionamento foi desenvolvido para facilitar o projeto de banco de dados, permitindo a especificação de um esquema de “negócio”, onde tal esquema representa a estrutura lógica geral do banco de dados.

A estrutura lógica geral de um banco de dados pode ser representada graficamente por um Diagrama de Entidade-Relacionamento (DE-R), como é mostrado na figura a baixo.

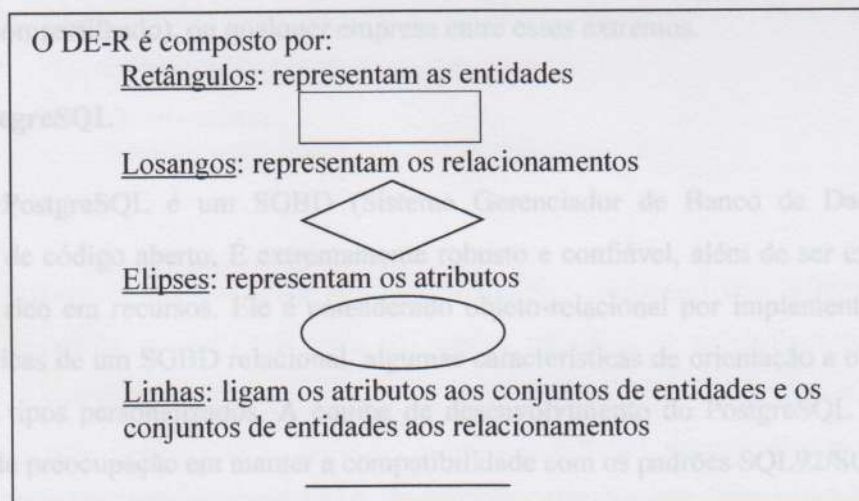


Figura 3: Elementos que compõem um diagrama de atividades.

Na figura 4, temos um exemplo de diagrama de entidade e relacionamento, composto pelas entidades “cliente” e “conta” juntamente com seus atributos fazendo uma associação através do relacionamento chamado “possui”.

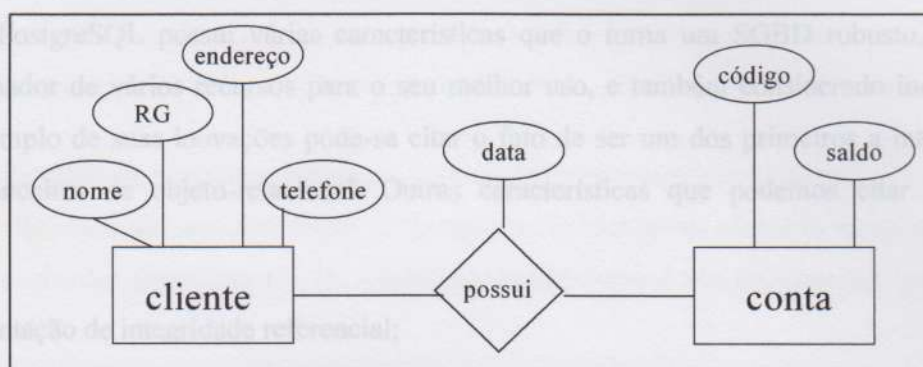


Figura 4: Exemplo de um diagrama de entidade e relacionamento.

2.3 Banco de Dados

Para Date (2000, p.9) “Um banco de dados é uma coleção de dados persistentes utilizada pelos sistemas de aplicação de uma determinada empresa”.

Nesse caso, o termo “empresa” é apenas um termo genérico conveniente para qualquer organização comercial, científica, técnica ou outra organização razoavelmente autônoma. Uma empresa poderia ser um único indivíduo (com um pequeno banco de dados pessoal), ou uma grande empresa completa ou corporação de grande porte similar (com um enorme banco de dados compartilhado), ou qualquer empresa entre esses extremos.

2.3.1 PostgreSQL

O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados. A equipe de desenvolvimento do PostgreSQL sempre teve uma grande preocupação em manter a compatibilidade com os padrões SQL92/SQL99.

Pela riqueza de recursos e conformidade com os padrões, ele é um SGBD muito adequado para o estudo universitário do modelo relacional, além de ser uma ótima opção para

empresas implementarem soluções de alta confiabilidade sem altos custos de licenciamento. É um programa distribuído sob a licença BSD, o que torna o seu código fonte disponível e o seu uso livre para aplicações comerciais ou não. (Biazus, 2005)

O PostgreSQL possui várias características que o torna um SGBD robusto, veloz, proporcionador de vários recursos para o seu melhor uso, e também considerado inovador. Como exemplo de suas inovações pode-se citar o fato de ser um dos primeiros a introduzir muitos conceitos de objeto-relacional. Outras características que podemos citar são as seguintes:

- Implementação de integridade referencial;
- Suporte a transações;
- Suporte aos padrões ANSI SQL 89, 92 e 99;
- Implementa o chamado MVCC (sistema de concorrência de múltiplas versões de dados), ou seja, resolve problemas relacionados a acessos concorrentes no banco de dados, o que poderia causar inconsistências nas informações armazenadas;
- A possibilidade de criar funções definidas pelo usuário e também a criação de gatilhos (triggers);
- Uso de herança de tabelas;
- Possibilidade de fazer backup on-line;
- Facilidades no uso dos objetos do banco;
- Possibilidade do usuário definir novos tipos de dados;
- Possuir suporte ou grande integração com várias linguagens como PHP, Java, Perl, Python, Delphi, C++, Visual Basic, etc.;
- Conexão por interfaces nativas.

2.4 Software Livre

É importante não confundir software livre (Free Software) com software grátis. A liberdade associada ao software livre de copiar, modificar e redistribuir independe de gratuidade. Existem programas que podem ser obtidos gratuitamente, mas que não podem ser modificados, nem redistribuídos. Software Livre é o software disponível com a permissão para qualquer um usá-lo, copiá-lo, e distribuí-lo, seja na sua forma original ou com modificações, seja gratuitamente ou com custo. Em especial, a possibilidade de modificações

implica em que o código fonte esteja disponível. Se um programa é livre, potencialmente ele pode ser incluído em um sistema operacional também livre. (Hewel, 2005)

De acordo com Lozano (2005): Software livre se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade, para os usuários do software:

- A liberdade de executar o programa, para qualquer propósito (liberdade nº. 0) .
- A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades (liberdade nº. 1). Acesso ao código-fonte é um pré-requisito para esta liberdade.
- A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade nº. 2).
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade nº. 3). Acesso ao código-fonte é um pré-requisito para esta liberdade.

Notem que os quatro itens acima não fazem nenhuma referência a custos ou preços. O fato de se cobrar ou não pela distribuição ou pela licença de uso do software não implica diretamente em ser o software livre ou não.

Você deve também ter a liberdade de fazer modificações e usá-las privativamente no seu trabalho ou lazer, sem nem mesmo mencionar que elas existem. Se você publicar as modificações, você não deve ser obrigado a avisar a ninguém em particular, ou de nenhum modo em especial. (Lozano, 2005).

A liberdade de utilizar um programa significa a liberdade para qualquer tipo de pessoa física ou jurídica utilizar o software em qualquer tipo de sistema computacional, para qualquer tipo de trabalho ou atividade, sem que seja necessário comunicar ao desenvolvedor ou a qualquer outra entidade em especial. (Lozano, 2005).

A liberdade de redistribuir cópias deve incluir formas binárias ou executáveis do programa, assim como o código-fonte, tanto para as versões originais quanto para as modificadas. Está ok se não for possível produzir uma forma binária ou executável (pois algumas linguagens de programação não suportam este recurso), mas deve ser concedida a

liberdade de redistribuir essas formas caso seja desenvolvido um meio de criá-las. (Lozano, 2005).

De modo que a liberdade de fazer modificações, e de publicar versões aperfeiçoadas, tenha algum significado, deve-se ter acesso ao código-fonte do programa. Portanto, acesso ao código-fonte é uma condição necessária ao software livre. (Lozano, 2005).

Para que essas liberdades sejam reais, elas têm que ser irrevogável desde que você não faça nada errado; caso o desenvolvedor do software tenha o poder de revogar a licença, mesmo que você não tenha dado motivo, o software não é livre. (Lozano, 2005).

Entretanto, certos tipos de regras sobre a maneira de distribuir software livre são aceitáveis, quando elas não entram em conflito com as liberdades principais. Por exemplo, copyleft (apresentado de forma bem simples) é a regra de que, quando redistribuindo um programa, você não pode adicionar restrições para negar para outras pessoas às liberdades principais. Esta regra não entra em conflito com as liberdades; na verdade, ela as protege. (Lozano, 2005).

Portanto, você pode ter pago para receber cópias do software GNU, ou você pode ter obtido cópias sem nenhum custo. Mas independente de como você obteve a sua cópia, você sempre tem a liberdade de copiar e modificar o software, ou mesmo de vender cópias. (Lozano, 2005).

"Software Livre" Não significa "não-comercial". Um programa livre deve estar disponível para uso comercial, desenvolvimento comercial, e distribuição comercial. O desenvolvimento comercial de software livre não é incomum; tais softwares livres comerciais são muito importantes. (Lozano, 2005).

2.5 HTML (HiperText Markup Language)

Para Marcondes(1998), HTML (Linguagem de Marcação de Hipertexto) é uma linguagem baseada em uma mais antiga e muito mais complexa denominada de SGML (Standart Generalized Markup Language). Uma das maiores características da linguagem HTML é não ter uma estrutura rígida e exata, a HTML apenas define a estrutura de uma página, definindo o que é título, texto, lista, subtítulo, etc.

Uma de suas principais características é a facilidade de manipulação, pois, a HTML é considerada a mais fácil linguagem de programação desenvolvida até hoje.

No núcleo de toda página da Web, encontramos a HyperText Markup Language, uma linguagem de diagramação baseada em marcadores, ou tags como é mais comumente conhecido. A HTML (HyperText Markup Language) é uma linguagem simples baseada em texto que podemos visualizar em qualquer plataforma. (Chicayban, 2000, p. 1).

A linguagem HTML tem por objetivos criar não apenas textos, mas criar hipertextos. Estes textos caracterizam-se por serem rápidos e pequenos, facilitando o acesso dos usuários da Web.

HTML é usada, para expressar o conteúdo e a formatação visual de uma página web.

2.5.1 Comandos HTML – Tag's

Os comandos em HTML são chamados de tag's que significa, em inglês, etiquetas, pois representam, na sua maioria comandos para estética de página, fonte, desenhos e linha.

Todas as tag's são apresentadas entre sinais de maior e menor "<...>" e sempre deve ser discriminada imediatamente antes da parte do documento que se deseja influenciar <tag> e, na maioria das vezes, deve finalizar a área com uma tag similar acompanhada por uma barra </tag> (Marcondes, 1998, p.24).

Na figura 5, apresenta exemplos de tag's como corpo de uma página web.

Exemplo:

```

1 <html>
2 <head>
3 <title>titulo da pagina, que se localizará no cabeçalho da página.</title>
4 </head>
5 <body>
6 Conteúdo da página
7 </body>
8 </html>
```

Figura 5:Exemplo de Tag's.

2.6 PHP (Personal Home Page)

A linguagem PHP foi concebida durante o outono de 1994 por Rasmus Lerdorf. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua home-page apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira

versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como "Personal Home Page Tools" (ferramentas para página pessoal). Era composta por um sistema bastante simples que interpretava alguns macros e alguns utilitários que rodavam "por trás" das home-pages: um livro de visitas, um contador e algumas outras coisas. (Htmlstaff, 2005).

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de PHP/FI, o "FI" veio de um outro pacote escrito por Rasmus que interpretava dados de formulários HTML (Form Interpreter). Ele combinou os scripts do pacote Personal Home Page Tools com o FI e adicionou suporte a mSQL, nascendo assim o PHP/FI, que cresceu bastante, e as pessoas passaram a contribuir com o projeto. Estima-se que em 1996 PHP/FI estava sendo usado por cerca de 15.000 sites pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000. Nessa época houve uma mudança no desenvolvimento do PHP. Ele deixou de ser um projeto de Rasmus com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por Zeev Suraski e Andi Gutmans, e esse novo interpretador foi a base para a versão 3. Atualmente o uso do PHP3 vem crescendo numa velocidade incrível, e já está sendo desenvolvida a versão 4 do PHP. (Htmlstaff, 2005).

Segundo Rocha (2003, p.7), "PHP é uma linguagem que permite criar sites dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros do URL e links".

É uma linguagem de criação de scripts do lado do servidor que foi projetada especificamente para web. Dentro de uma página HTML, você pode embutir código de PHP que será executado toda vez que a página for visitada. O código PHP é interpretado no servidor web e gera HTML ou outra saída que o visitante verá.

Uma página Web dinâmica é aquela que interage com o usuário de forma que cada usuário ao visitar a página verá informações personalizadas.

PHP é uma linguagem totalmente voltada à internet, possibilitando o desenvolvimento de site realmente dinâmicos. Dominando essa linguagem, pode-se transformar aqueles sites estáticos, feitos em HTML puro, em sites interativos, utilizando todas as técnicas de programação que essa linguagem oferece. (Niederauer, 2004)

Figura 6: Diagramas de requisição para o PHP (ROCHA, 2003, p.10)

Para Niederauer (2004, p.19), “O PHP é uma das linguagens mais avançadas da Web. A principal diferença entre as outras linguagens é a capacidade que o PHP tem de interagir com o mundo Web, transformando totalmente os websites que possuem paginas estáticas”.

Uma das vantagens do PHP é que, além de ser gratuito, é um software com código fonte aberto.

Basicamente, tudo que pode ser feito por algum programa CGI (Common Gateway Interface) pode também ser feito com PHP, como coletar dados de um formulário, gerar páginas dinâmicas ou enviar paginas e receber *cookies*.

O PHP também tem como uma das suas principais características o suporte a um grande número de banco de dados, como dBase, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL, MSSQL, Fontes ODBC e vários outros.

Em uma requisição para o PHP:

1. O Browser faz uma requisição ao servidor web;
2. O Servidor Web detecta que trata-se de uma página PHP e encaminha a requisição ao interpretador PHP;
3. O interpretador PHP faz os processamentos necessários, inclusive acesso a banco de dados e outros recursos, e devolve o HTML resultado para o Servidor Web;
4. O Servidor Web devolve a HTML ao Browser.

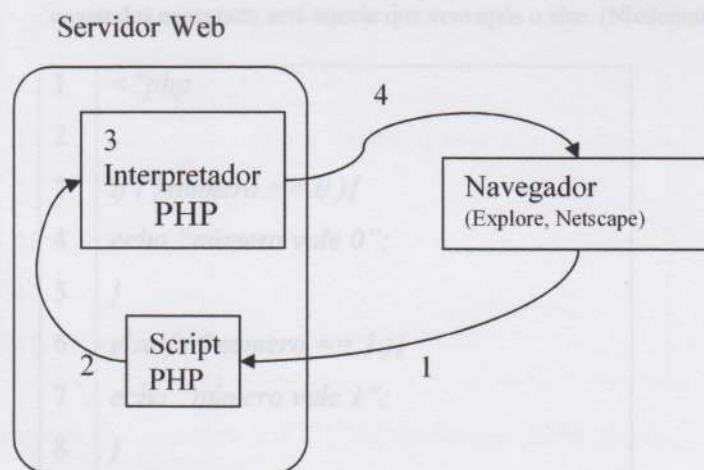


Figura 6: Diagramas de requisição para o PHP (ROCHA, 2003, p.10)

2.6.1 Estrutura de um programa PHP

Conforme Rocha(2003, p.10) “O código PHP fica embutido no próprio HTML. O interpretador identifica quando um código é PHP pelas seguintes tags: `<? php comandos ?>` ou `<script language="php"> comandos</script>` ou `<? comandos ?>` ou `<% comandos %>`”.

2.6.2 Estrutura de Controle em PHP

São os comandos que serão utilizados em a estrutura dos programas PHP. São comandos comuns à maioria das linguagens de programação, e o uso deles é fundamental para realizar decisões lógicas, testar se determinadas expressões são verdadeiras, repete um bloco de comandos por certo número de vezes ou até que uma condição seja atingida. (Niederauer 2003).

2.6.2.1 Comandos Condicionais

Esses comandos são usados sempre que há necessidades de uma tomada de decisão dentro de um programa. São eles if e switch.

IF – “comando que avalia uma expressão e, dependendo do resultado, é executado um conjunto diferente de instruções. O comando if pode possuir como complemento o elseif e/ ou o else”. (Niederauer 2004, p.60).

Se a condição avaliada no comando if for falsa (false), o bloco de comando seguinte não será executado, e será testada a expressão contida no primeiro elseif (se houver). Se todas as expressões avaliadas (do if e do elseif) forem falsas, o bloco de comandos executado será aquele que vem após o else. (Niederauer 2004, p.60).

1	<code><?php</code>
2	<code>...</code>
3	<code>if (\$numero == 0){</code>
4	<code>echo "número vale 0";</code>
5	<code>}</code>
6	<code>elseif (\$numero == 1){</code>
7	<code>echo "número vale 1";</code>
8	<code>}</code>
9	<code>elseif (\$numero == 2){</code>
10	<code>echo "número vale 2";</code>

11	}
12	...
13	?>

Figura 7: Comando IF (Niederauer 2004, p.64).

SWITCH – “Muito parecido com o IF, pois ambos avaliam o valor de uma expressão para escolher qual o bloco de instruções deve ser executado. Em algumas ocasiões você tem a uma mesma variável a ser testada com valores diferentes. Trabalha basicamente com o operador de igualdade”. (Niederauer 2004, p.63).

Para Niederauer (2004, p.60), “O uso do comando switch torna o código um pouco mais organizado que o if, pois esse comando utiliza apenas uma cláusula (case), enquanto if utiliza varias (if, elseif, else e as vezes endif)”.

1	<?php
2	...
3	Switch (\$numero)
4	{
5	case '0':
6	echo “número vale 0”;
7	break;
8	case '1':
9	echo “número vale 0”;
10	break;
11	case '2':
12	echo: “número vale 0”;
13	break;
14	}
15	...
16	?>

Figura 8: Comando SWITCH (Niederauer 2004, p.64).

2.6.2.2 Comandos de repetição

“São comandos utilizados para que um conjunto de instruções seja executado repetidamente por um número determinado de vezes, ou até que determinada função seja atingida”. (Niederauer, 2004)

WHILE – “O comando while é Composto por uma expressão e por um bloco de comandos, o comando avalia a expressão, e enquanto essa expressão retornar o valor verdadeiro, a execução do bloco de comandos em questão será repetida”. (Niederauer 2004, p.65).

1	<code><?php</code>
2	<code> \$cont = 1;</code>
3	<code> while (\$cont < 100)</code>
4	<code> {</code>
5	<code> Echo “O valor atual do contador é \$cont
”;</code>
6	<code> \$cont++;</code>
7	<code> }</code>
8	<code>?></code>

Figura 9: Comando WHILE (Niederauer 2004, p.66):

DO...WHILE – O do...while avaliada a expressão no final do laço. Portanto, você já pôde perceber que utilizando do..while o laço será executado pelo menos uma vez. (Niederauer 2004, p.67).

1	<code><?php</code>
2	<code> \$numero = 1;</code>
3	<code> Do</code>
4	<code> {</code>
5	<code> echo “O valor atual do número é \$numero
”;</code>
6	<code> \$numero++;</code>
7	<code> } while (\$numero < 4);</code>
8	<code>?></code>

Figura 10: Comando DO...WHILE (Niederauer 2004, p.67):

FOR – “Utilizado o comando for quando queremos executar um conjunto de instruções um número determinado de vezes. É um comando muito útil que pode ser usado, por exemplo para imprimir todos os elementos de um array”. (Niederauer 2004, p.68).

1	<code><?php</code>
2	<code>for (\$cont = 0 ; \$cont < 10 ; \$cont++)</code>
3	<code>{</code>
4	<code>echo "A variável \$cont vale \$cont";</code>
5	<code>echo "
";</code>
6	<code>}</code>
7	<code>?></code>

Figura 11: Comando FOR (Niederauer 2004, p.68).

2.7 Dreamweaver

Dreamweaver é um editor visual profissional que cria e gerencia sites e páginas da web. Com o Dreamweaver é fácil criar e editar páginas compatíveis com diversos navegadores e plataformas.

O dreamweaver é um editor de HTML que fornece aos usuários capacidades visuais de design e edição com a capacidade de trabalhar diretamente com o código. O dreamweaver ajuda a acelerar o tempo de produção de websites e fornece as ferramentas para o gerenciamento e a manutenção desses sites. (Pages, 2004, P. 6)

O Dreamweaver oferece ferramentas avançadas de desenho e layout, bem como facilita a utilização dos recursos do HTML dinâmico, como os comportamentos e camadas animadas, sem gravar uma linha de código. O Dreamweaver pode ser definido para limpar e reformatar o HTML quando você quiser. O Dreamweaver também pode ser totalmente personalizado.

2.8 Apache

O servidor Apache é o mais bem sucedido servidor web livre. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications), Universidade de Illinois. (Wikipédia, 2005).

Faz parte como tecnologia principal da Apache Software Foundation, responsável por mais de uma dezena de projetos envolvendo tecnologias de transmissão via web, processamento de dados e execução de aplicativos distribuídos. (Wikipédia, 2005).

O servidor é compatível com o protocolo HTTP versão 1.1 (especificado pelo documento RFC2616). Suas funcionalidades são mantidas através de uma estrutura de módulos, podendo inclusive o usuário escrever seus próprios módulos - utilizando a API do software. (Wikipédia, 2005).

É disponibilizado em versões para os sistemas Windows, Novell Netware, OS/2 e diversos outros do padrão POSIX (Unix, Linux, FreeBSD, etc). (Wikipédia, 2005).

Apache é um Servidor Web (programa que está sendo executado em uma máquina ligada à Rede, esperando por conexões para apresentar certos documentos solicitados por um browser). Para se comunicar, o servidor e o browser usam um método de comunicação conhecido como HTTP (Protocolo de Transferência de Hipertextos). O servidor mencionado possui Código aberto em constante desenvolvimento por colaboradores no mundo todo, além de ser *freeware*. (Apache, 2005).

Um dos mais populares programas para servidores de web, o Apache foi construído por um grupo de programadores de open-source e geralmente é usado por sua excelente desempenho, fortes características de segurança e o fato de que é de graça. (Midiaworks, 2005).

Apache é um sistema que funciona como servidor Web para Internet ou uma Intranet. Além disso, o Apache também é responsável por executar uma série de outras aplicações, como interpretar comandos PHP ou acessar um banco de dados como o PostgreSQL.

3. ANÁLISE E PROJETO DO SISTEMA DE CONTROLE DE VENDAS E ESTOQUE PARA PEQUENOS ESTABELECIMENTOS

Para o desenvolvimento, foi utilizada as técnicas de: diagramas de atividade, diagramas de caso de uso, expansão de casos de uso, diagramas de entidades e relacionamentos.

3.1 Diagrama de Caso de Uso

Temos dois diagramas de caso de uso, um para o administrador, figura 12 e o outro para o funcionário, figura 13.

Como mostrado na figura 12, o gerente primeiramente terá que ser cadastrado por um administrador do sistema, para isso ele deverá informar o seu login e sua senha, mas para que ele possa interagir com o caso de uso, depois de cadastrado ele terá que efetuar o login, para isso ele terá que informar o login e senha. O gerente terá privilégios de fazer vendas, visualizar vendas, visualizar estoque mínimo, controlar produtos e controlar acesso. O caso de uso controlar produto estende-se no cadastramento de novos produtos juntamente com seu cancelamento e sua alteração. O caso de uso controlar acesso prolonga-se no cadastramento de funcionários juntamente com seu cancelamento e sua alteração.

Figura 12: Diagramas de Caso de Uso que representa as ações do Gerente.

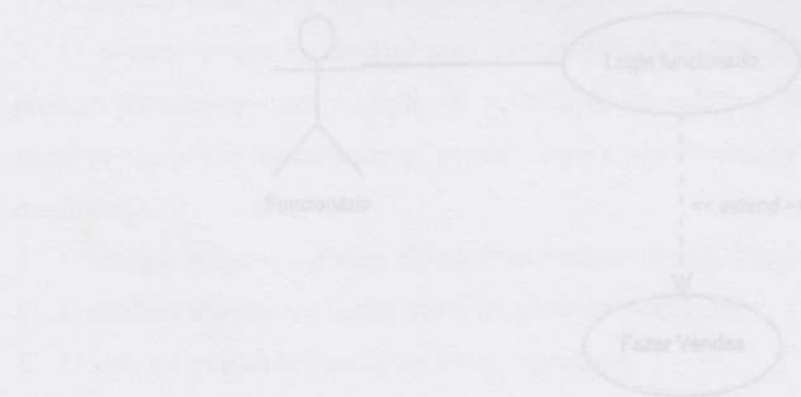


Figura 13: Diagramas de Caso de Uso que representa as ações do funcionário.

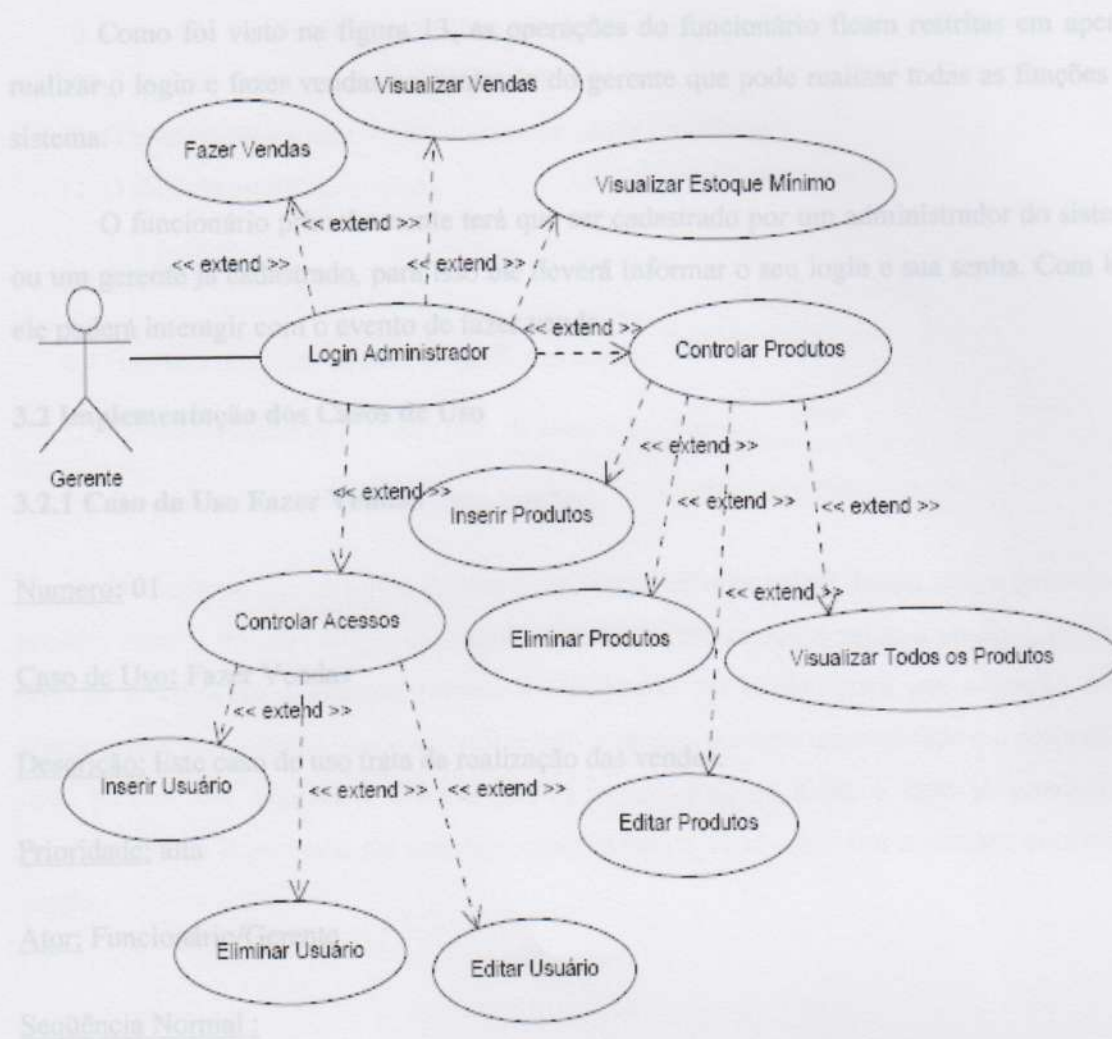


Figura 12: Diagramas de Caso de Uso que representa as ações do Gerente.

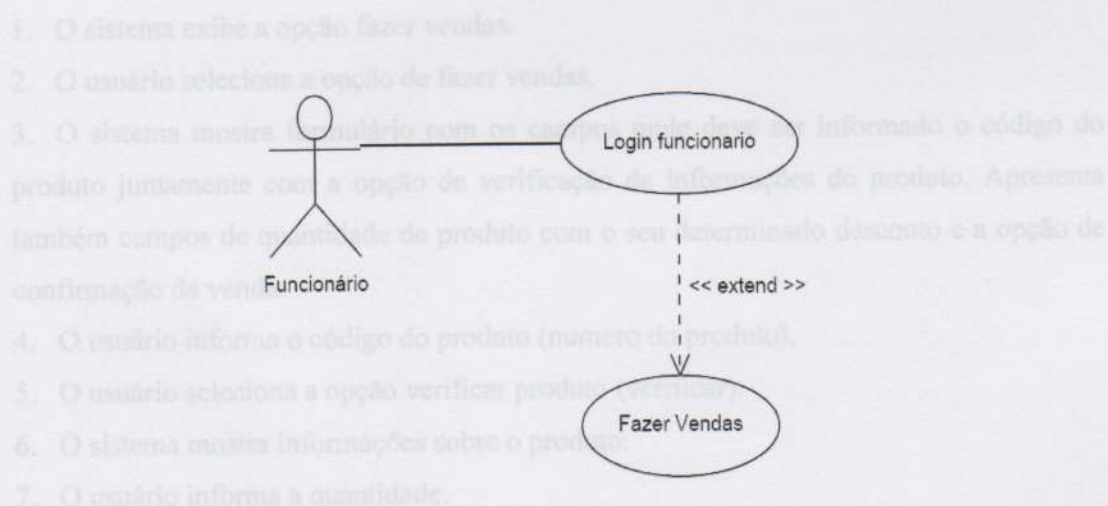


Figura 13: Diagramas de Caso de Uso que representa as ações do funcionário.

10. Como foi visto na figura 13, as operações do funcionário ficam restritas em apenas realizar o login e fazer vendas ao contrario do gerente que pode realizar todas as funções do sistema.

O funcionário primeiramente terá que ser cadastrado por um administrador do sistema ou um gerente já cadastrado, para isso ele deverá informar o seu login e sua senha. Com isso ele poderá interagir com o evento de fazer venda.

3.2 Implementação dos Casos de Uso

3.2.1 Caso de Uso Fazer Vendas

Numero: 01

Caso de Uso: Fazer Vendas

Descrição: Este caso de uso trata da realização das vendas.

Prioridade: alta

Ator: Funcionário/Gerente

Seqüência Normal :

0. O usuário inicia o sistema.
1. O sistema exibe a opção fazer vendas.
2. O usuário seleciona a opção de fazer vendas.
3. O sistema mostra formulário com os campos onde deve ser informado o código do produto juntamente com a opção de verificação de informações do produto. Apresenta também campos de quantidade de produto com o seu determinado desconto e a opção de confirmação da venda.
4. O usuário informa o código do produto (numero do produto).
5. O usuário seleciona a opção verificar produto (verificar).
6. O sistema mostra informações sobre o produto.
7. O usuário informa a quantidade.
8. O usuário informa o desconto.
9. O usuário adiciona o produto a venda (adicionar).

10. O usuário verifica se é necessário adicionar mais algum produto.
11. O usuário informa a forma de pagamento.
12. O usuário seleciona o botão confirmar venda (confirmar).
13. O sistema confirma a venda.

Seqüências Alternativas:

4. O usuário não preenche o campo corretamente.
5. O sistema mostra uma mensagem de erro.

3.2.1.1 Diagrama de Atividade Fazer Vendas

O diagrama que se segue mostra a realização de uma venda. Inicia com a pesquisa do produto através do seu código, em seguida mostra as informações sobre o produto. Havendo erro de codificação o sistema retorna a verificação do código para sua correção dando continuidade ao processo. Após a confirmação o usuário informa a quantidade e o desconto de cada produto em seguida o sistema guarda as informações sobre o item já selecionado, repetindo então o processo até que seja concretizada a venda, por fim o sistema confirma a venda.

Figura 14: Diagramas de atividade fazer vendas.

3.2.1.3 Telas e Códigos

As figuras 15, 16, 17, 18 representam o caso de uso fazer vendas, é apresentado a tela principal com seus principais códigos.

Na figura 15, é mostrado a tela do gerente onde possui a opção de fazer vendas. Após o gerente selecionar a opção fazer venda o sistema mostra um formulário com os campos onde deve ser informado o número do produto juntamente com a opção verificar informações do produto. Apresenta também os campo onde deve ser informado a quantidade de produto com o seu determinado desconto e a opção de confirmação da venda, como mostrado na figura 16.

Figura 15: Tela de login do gerente

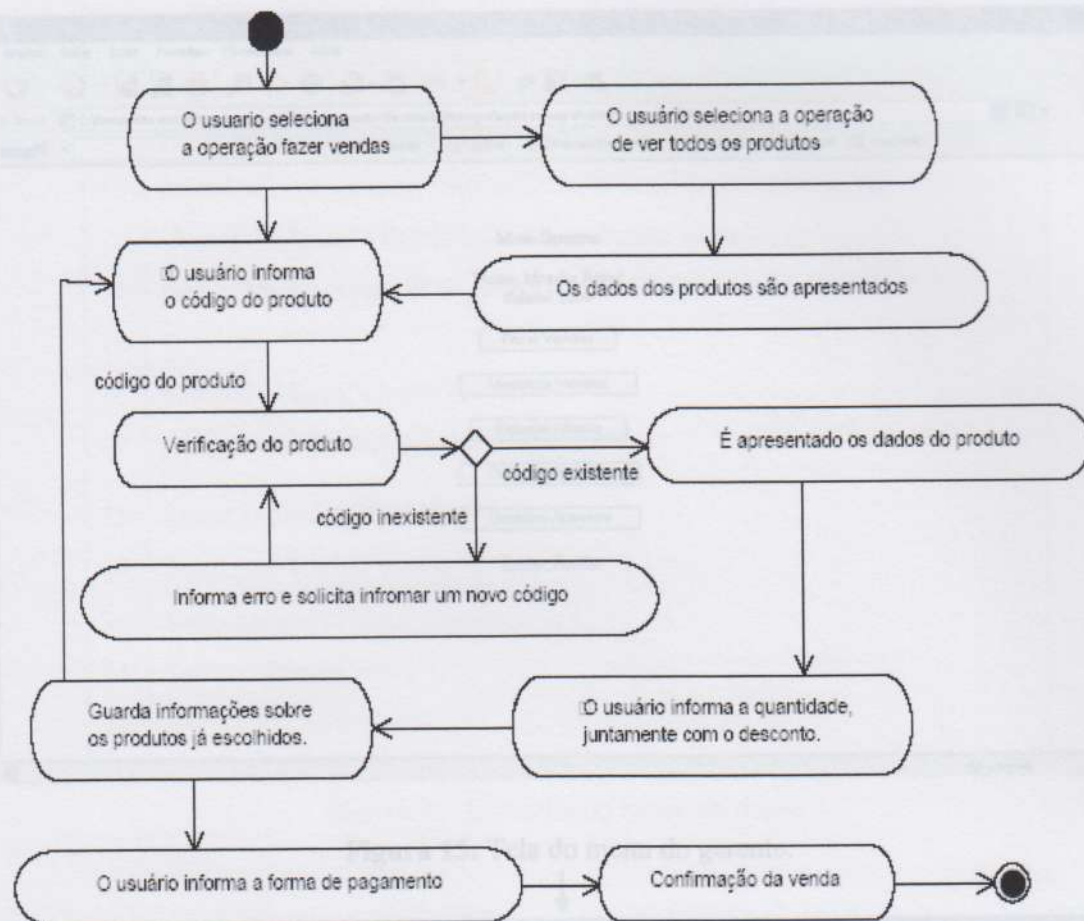


Figura 14: Diagramas de atividade fazer vendas.

3.2.1.2 Telas e Códigos

As figuras 15, 16, 17, 18 representam o caso de uso fazer vendas, é apresentado a tela principal com seus principais códigos.

Na figura 15, é mostrado a tela do gerente onde possui a opção de fazer vendas. Após o gerente selecionar a opção fazer venda o sistema mostra um formulário com os campos onde deve ser informado o número do produto juntamente com a opção verificar informações do produto. Apresenta também os campos onde deve ser informado a quantidade de produto com o seu determinado desconto e a opção de confirmação da venda, como mostrado na figura 16.

Figura 16: Tela de fazer vendas.

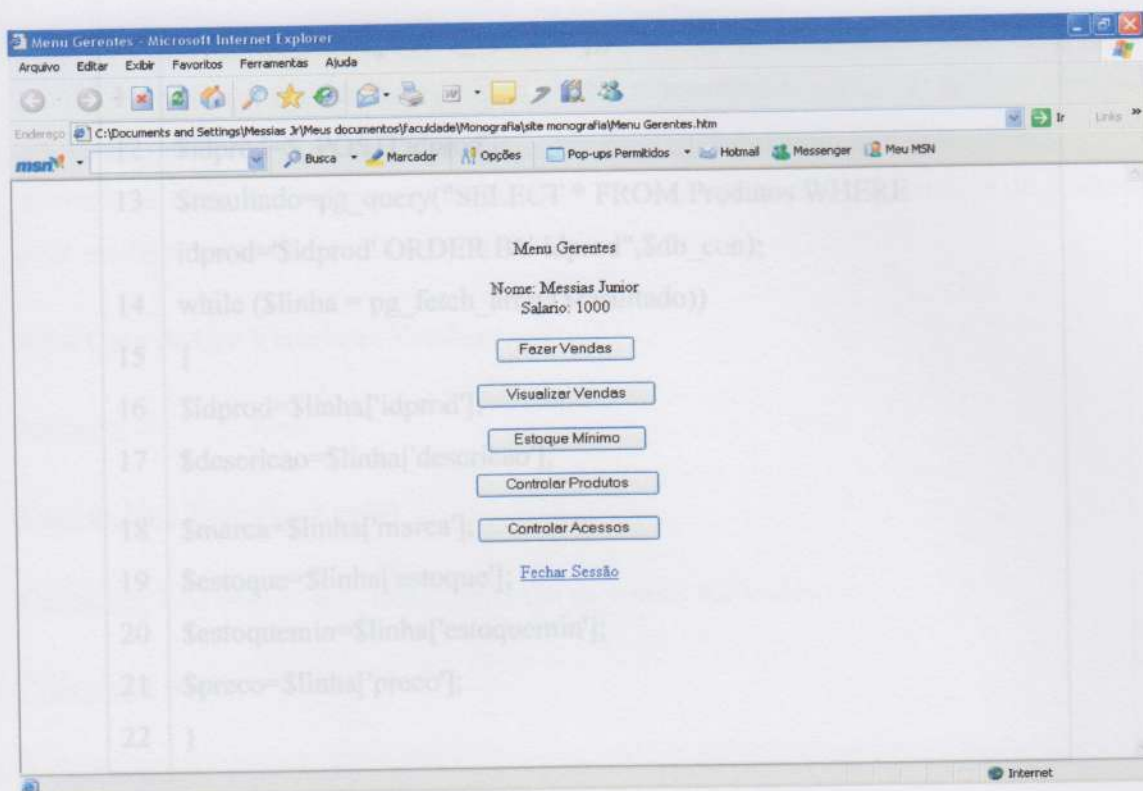


Figura 15: Consulta no banco de dados.

Figura 15: Tela do menu do gerente.

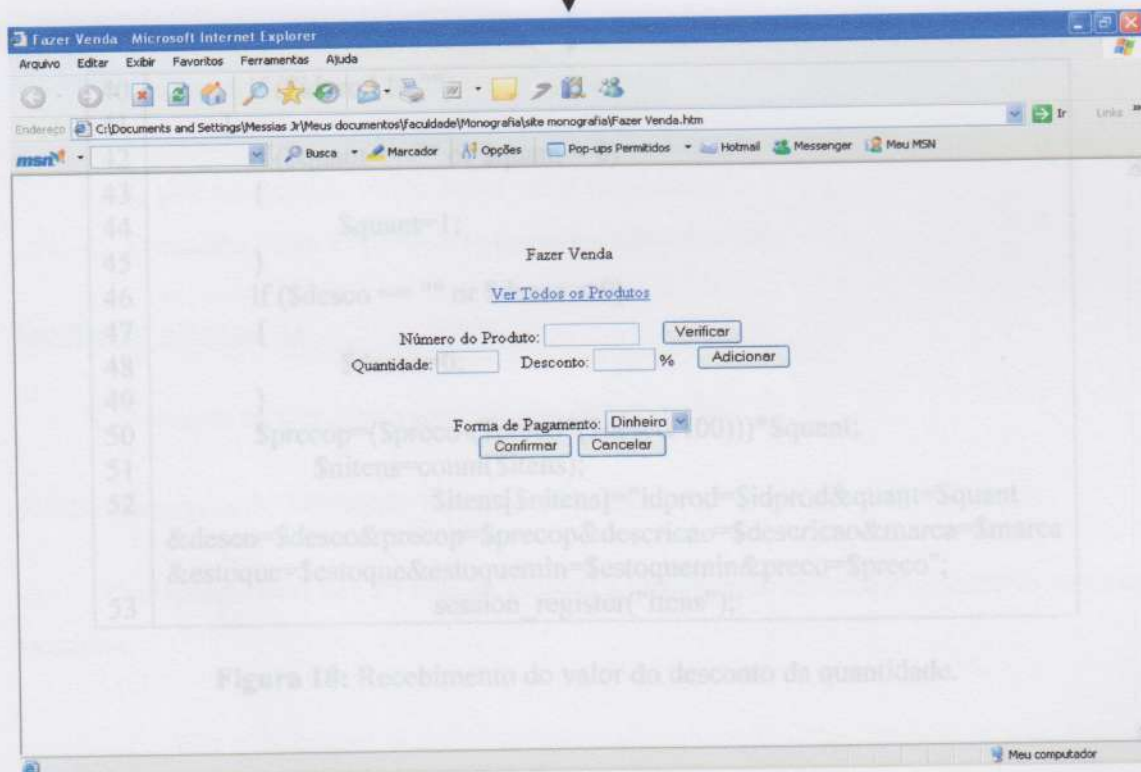


Figura 16: Recebimento do valor do desconto da quantidade.

Figura 16: Tela de fazer vendas.



```

10 if (isset($_POST['botao_verificar']))
11 {
12     $idprod=$_POST['idprod'];
13     $resultado=pg_query("SELECT * FROM Produtos WHERE
14     idprod='$idprod' ORDER BY idprod",$db_con);
15     while ($linha = pg_fetch_array($resultado))
16     {
17         $idprod=$linha['idprod'];
18         $descricao=$linha['descricao'];
19         $marca=$linha['marca'];
20         $estoque=$linha['estoque'];
21         $estoquemin=$linha['estoquemin'];
22         $preco=$linha['preco'];
23     }

```

Figura 17: Consulta no banco de dados.



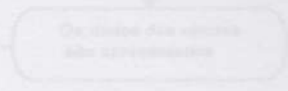
```

40 if ($idprod != "")
41 {
42     if ($quant == "" or $quant < 0)
43     {
44         $quant=1;
45     }
46     if ($desco == "" or $desco < 0)
47     {
48         $desco=0;
49     }
50     $precop=($preco-($preco*($desco/100)))*$quant;
51     $nitens=count($itens);
52     $itens[$nitens]="idprod=$idprod&quant=$quant
53     &desco=$desco&precop=$precop&descricao=$descricao&marca=$marca
54     &estoque=$estoque&estoquemin=$estoquemin&preco=$preco";
55     session_register("itens");

```

Figura 18: Recebimento do valor do desconto da quantidade.

Na figura 17 é mostrado o código que é responsável pelo pela identificação do produto através do select que faz a captura dos dados sobre o determinado código do produto através função pg_query que é responsável por executar a consulta no banco de dados, como é mostrado na linha 13. Na figura 18 é informado o código que recebe a quantidade de produtos a ser vendidos, juntamente com seu desconto.



3.2.2 Caso de Uso Visualizar Vendas

Numero: 02

Caso de Uso: Visualiza Vendas

Descrição: Este caso de uso trata de informar as vendas realizadas.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

0. O usuário inicia o sistema.
1. O sistema exibe a opção de visualizar vendas.
2. O usuário seleciona a opção.
3. O sistema exibe as informações sobre os pedidos.
4. O sistema exibe a opção de volta para a tela principal.

Seqüências Alternativas:

3. O sistema mostra mensagem que não existe vendas.

3.2.2.1 Diagrama de Atividade Visualizar Vendas

O diagrama da figura 19 trata de mostrar a visualização informações sobre os produtos vendidos.

4. O sistema mostra mensagem que não existe produtos com estoque mínimo.

3.2.3.1 Diagrama de Atividade Visualizar Estoque Mínimo

O diagrama a seguir ilustra o fluxo de atividades para visualizar o estoque mínimo no sistema.

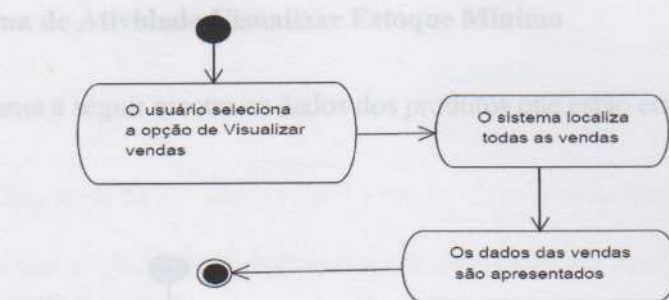


Figura 19: Diagrama de atividade visualizar vendas

3.2.3 Caso de Uso Visualizar Estoque Mínimo

Numero: 03

Caso de Uso: Visualiza Estoque Mínimo

Descrição: Este caso de uso informa todos os produtos que estão em quantidade mínima no estoque .

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

1. O usuário inicia o sistema.
2. O sistema exibe a opção de visualizar estoque mínimo.
3. O usuário seleciona a opção.
4. O sistema exibe as informações sobre os produtos que estão com a quantidade mínima no estoque.
5. O sistema exibe a opção de volta para a tela principal.

Seqüências Alternativas:

4. O sistema mostra mensagem que não existe produtos com estoque mínimo.
6. O usuário seleciona a opção de inserir novo código.
7. O sistema exibe as informações suficientes para o cadastro do produto.
8. O usuário preenche os campos e seleciona o botão inserir.

3.2.3.1 Diagrama de Atividade Visualizar Estoque Mínimo

O diagrama a seguir mostra os dados dos produtos que estão em quantidade mínima no sistema.

5. O usuário informa um código igual a um produto já cadastrado

5.1 O sistema mostra uma mensagem de que já existe um produto com o mesmo código que se deseja cadastrar

3.2.4.1 Diagrama

O diagrama a seguir trata-se de uma sequência de atividades que o usuário realiza no banco de dados, o sistema recebe o código e verifica se já existe um produto com o mesmo código, se já estiver cadastrado o sistema o sistema pode para ser inserido um novo código, se não o sistema

Figura 20: Diagrama de atividade visualizar estoque mínimo.

3.2.4 Caso de Uso Inserir Produtos

Numero: 04

Caso de Uso: Inserir Produtos.

Descrição: Este caso de uso armazena informações sobre os produtos no banco de dados.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal:

1. O usuário inicia o sistema.
2. O sistema exibe a opção de controlar produtos.
3. O usuário seleciona a opção.
4. O sistema exibe formulário com os campos onde deve ser informado o código do produto juntamente com a opção de inserção.
5. O usuário informa o código do produto (numero do produto)
6. O usuário seleciona a opção de inserir novo código.
7. O sistema exibe as informações suficientes para o cadastro do produto.
8. O usuário preenche os campos e seleciona o botão inserir.

9. O sistema valida o cadastro.

Seqüências Alternativas:

5. O usuário informa um código igual a um produto já cadastrado.

5.1 O sistema mostra uma mensagem de que já existe um produto com o mesmo código que se deseja cadastrar.

3.2.4.1 Diagrama de Atividade Inserir Produtos

O diagrama a seguir trata-se da inserção dos dados do produto no banco de dados, o sistema recebe um código e verifica se o determinado código já existe cadastrado no banco de dados, se já estiver cadastrado o sistema pede para ser inserido um novo código, se não o sistema recebe novas informações do produto e salva essas informações no banco de dados.

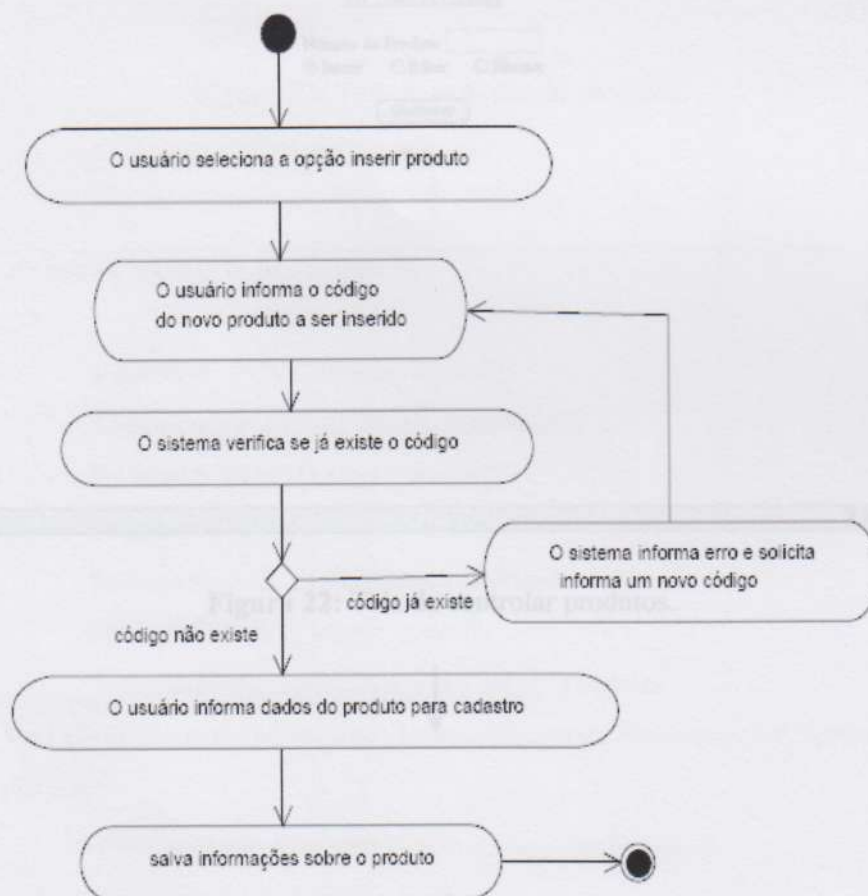


Figura 21: Diagrama de atividade inserir produto.

3.2.4.2 Telas e Códigos

As figuras 22, 23, 24 representam o caso de uso Inserir Produtos. É apresentada a tela principal de controlar produtos, juntamente com a tela de inserção de um novo produto com um trecho de código. Primeiramente o usuário seleciona a opção de controlar produto em seguida seleciona a opção de inserir novo produto. Informando os dados necessários para o cadastro do novo produto. Na figura 24 na linha 15, o código representa a inserção dos dados no banco de dados.

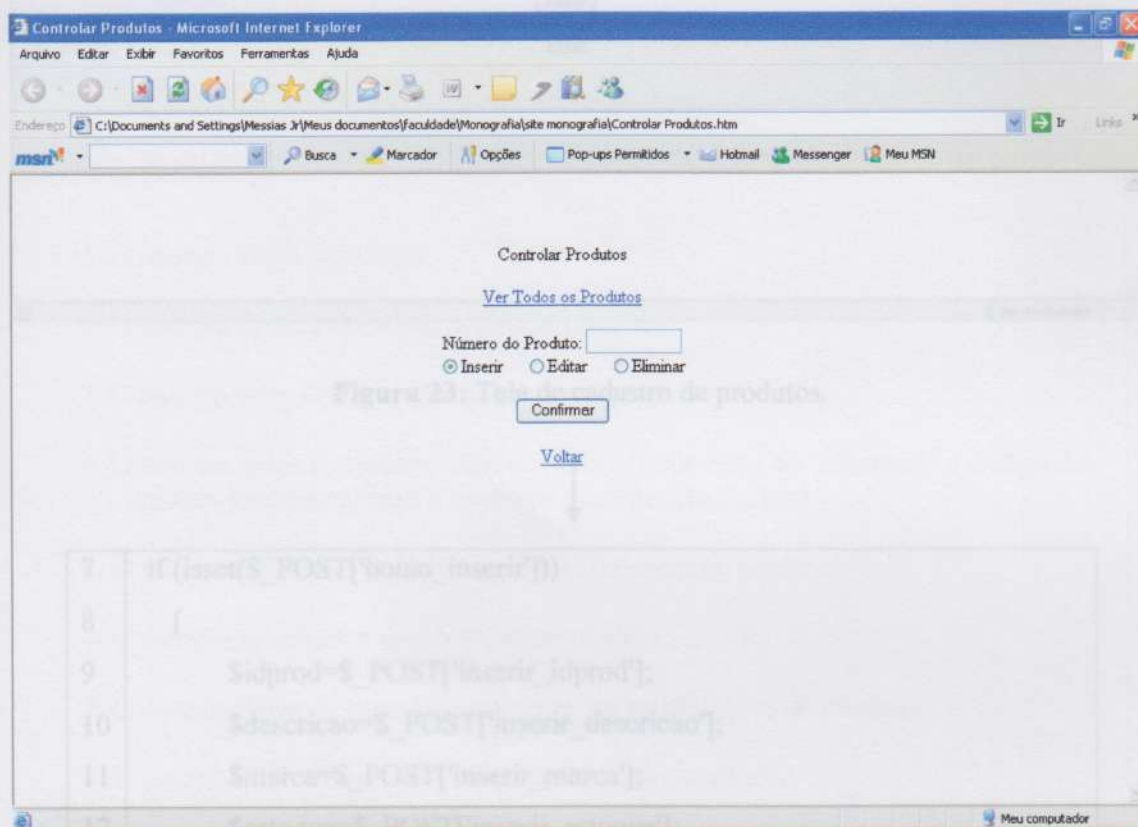


Figura 22: Tela de controlar produtos.

```
13 $estoque=$POST['estoque'];
14 $preco=$POST['preco'];
15 $resultado=pg_query("INSERT INTO Produtos
VALUES('$sidprod','$descricao','$marca','$estoque','$estoquembr','$preco') $
db_con);
16 header('location: produtos.php');
17
```

Figura 24: Inserção no banco de dados.

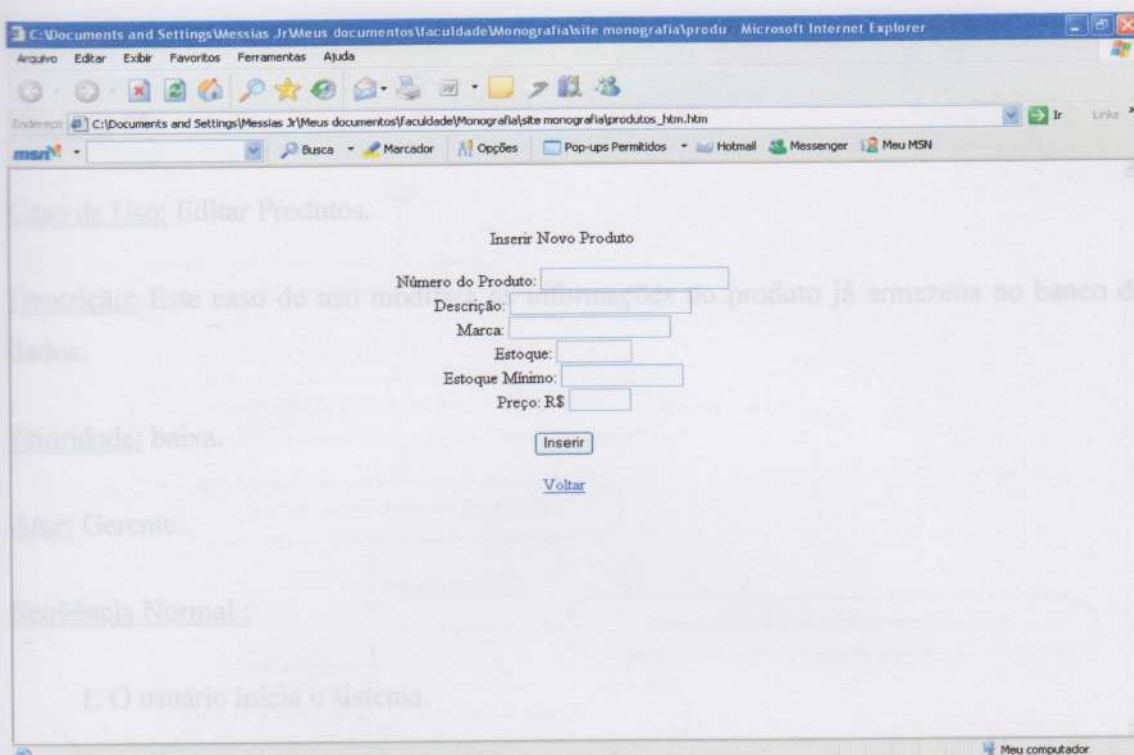


Figura 23: Tela de cadastro de produtos.

```

7  if (isset($_POST['botao_inserir']))
8  {
9      $idprod=$_POST['inserir_idprod'];
10     $descricao=$_POST['inserir_descricao'];
11     $marca=$_POST['inserir_marca'];
12     $estoque=$_POST['inserir_estoque'];
13     $estoquemín=$_POST['inserir_estoquemín'];
14     $preço=$_POST['inserir_preço'];
15     $resultado=pg_query("INSERT INTO Produtos
VALUES('$idprod','$descricao','$marca','$estoque','$estoquemín','$preço')",$
db_con);
16     header('location: produtos.php');
17 }

```

Figura 24: Inserção no banco de dados.

3.2.5 Caso de Uso Editar Produtos

Numero: 05

Caso de Uso: Editar Produtos.

Descrição: Este caso de uso modifica as informações do produto já armazenado no banco de dados.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

1. O usuário inicia o sistema.
2. O sistema exibe a opção de controlar produtos.
3. O usuário seleciona a opção de controlar produto.
4. O sistema exibe formulário com os campos onde deve ser informado o código do produto juntamente com a opção de modificação (editar).
5. O usuário informa o código do produto (numero do produto).
6. O usuário seleciona a opção de editar produto.
7. O sistema exibe os campos que podem ser modificados do produto.
8. O usuário edita os campos e seleciona o botão confirmar.
9. O sistema valida a mudança.

Seqüências Alternativas:

5. O usuário não preenche o campo corretamente.
6. O sistema exibe uma mensagem de erro.

3.2.5.1 Diagrama de Atividade Editar Produtos

O próximo diagrama trata-se da modificação de informações sobre determinado produto. Primeiro o sistema recebe o código do produto em seguida verifica se o produto esta

cadastrado no banco de dados, então o sistema mostra os campos das informações que podem ser modificadas.

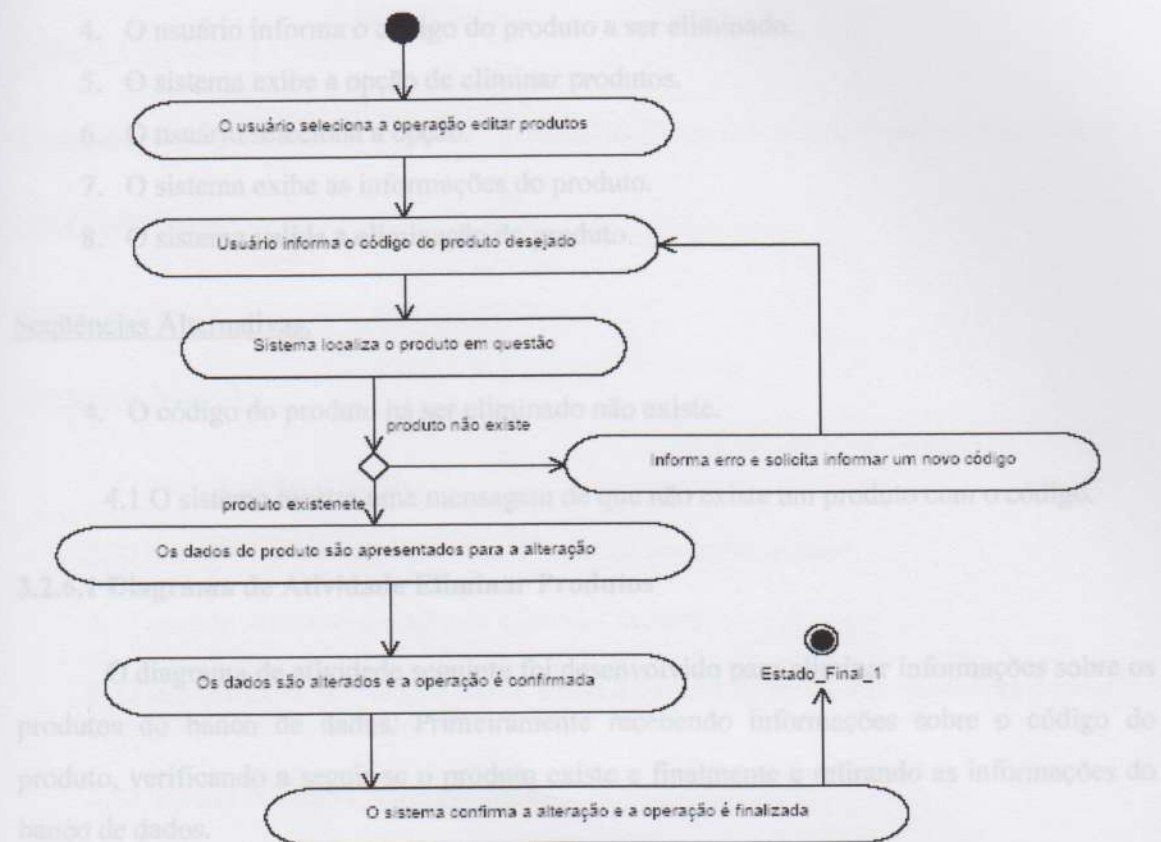


Figura 25: Diagrama de atividade editar produto

3.2.6 Caso de Uso Eliminar Produtos

Numero: 06

Caso de Uso: Eliminar Produtos.

Descrição: Este caso de uso elimina as informações sobre os produtos do banco de dados.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

Figura 26: Diagrama de atividade eliminar produto

1. O usuário inicia o sistema.
2. O sistema exibe a opção de controlar produtos.
3. O usuário seleciona a opção.
4. O usuário informa o código do produto a ser eliminado.
5. O sistema exibe a opção de eliminar produtos.
6. O usuário seleciona a opção.
7. O sistema exibe as informações do produto.
8. O sistema valida a eliminação do produto.

Seqüências Alternativas:

4. O código do produto há ser eliminado não existe.

4.1 O sistema mostra uma mensagem de que não existe um produto com o código.

3.2.6.1 Diagrama de Atividade Eliminar Produtos

O diagrama de atividade seguinte foi desenvolvido para eliminar informações sobre os produtos do banco de dados. Primeiramente recebendo informações sobre o código do produto, verificando a seguir se o produto existe e finalmente retirando as informações do banco de dados.

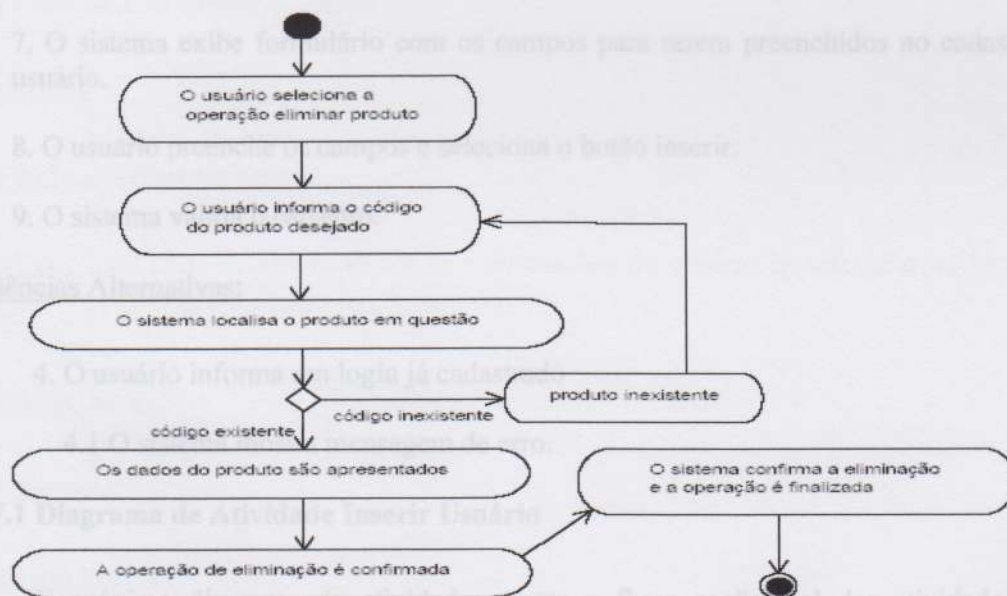


Figura 26: Diagrama de atividade eliminar produto

3.2.7 Caso de Uso Inserir Usuário

Numero: 07

Caso de Uso: Inserir Usuário.

Descrição: Este caso de uso armazena informações de um novo usuário no banco de dados.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

1. O usuário inicia o sistema.
2. O sistema exibe formulário com a opção de controlar usuário.
3. O usuário seleciona a opção de controlar usuário.
4. O sistema exibe formulário com a opção de inserir usuário juntamente com o campo do login para ser preenchido.
5. O usuário informa o login do usuário.
6. O usuário seleciona a opção inserir usuário.
7. O sistema exibe formulário com os campos para serem preenchidos no cadastro do usuário.
8. O usuário preenche os campos e seleciona o botão inserir.
9. O sistema valida o cadastro.

Seqüências Alternativas:

4. O usuário informa um login já cadastrado

- 4.1 O sistema mostra mensagem de erro.

3.2.7.1 Diagrama de Atividade Inserir Usuário

O próximo diagrama de atividade mostra o fluxo seqüencial das atividades para inserção de um novo usuário no sistema. O sistema recebe o login do usuário, verifica se o

login já existe, se não o sistema mostra formulário com campos para o armazenamento das informações necessárias para o cadastramento do novo usuário.

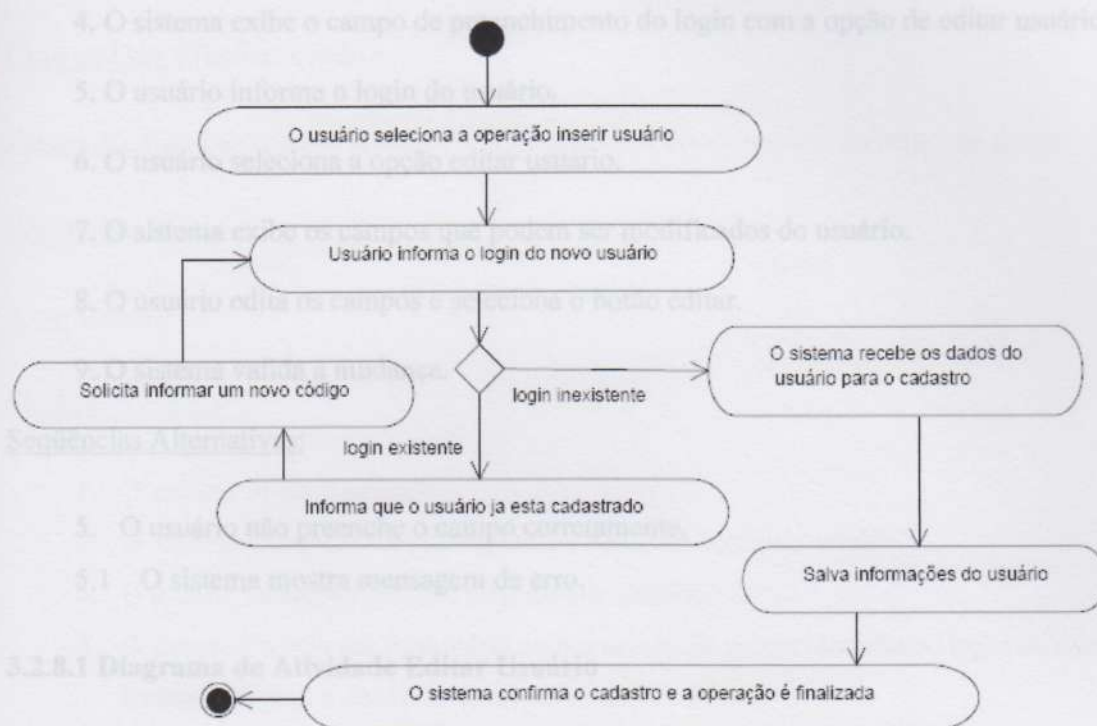


Figura 27: Diagrama de atividade inserir usuário.

3.2.8 Caso de Uso Editar Usuário

Numero: 08

Caso de Uso: Editar Usuário.

Descrição: Este caso de uso modifica as informações do usuário já armazena no banco de dados.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

1. O usuário inicia o sistema.

2. O sistema exibe com a opção de controlar usuário.
3. O usuário seleciona a opção de controlar usuário.
4. O sistema exibe o campo de preenchimento do login com a opção de editar usuário.
5. O usuário informa o login do usuário.
6. O usuário seleciona a opção editar usuário.
7. O sistema exibe os campos que podem ser modificados do usuário.
8. O usuário edita os campos e seleciona o botão editar.
9. O sistema valida a mudança.

Seqüências Alternativas:

5. O usuário não preenche o campo corretamente.
- 5.1 O sistema mostra mensagem de erro.

3.2.8.1 Diagrama de Atividade Editar Usuário

O diagrama a seguir é utilizado para mostrar o fluxo seqüencial das atividades para as modificações dos dados do funcionário no sistema. O sistema recebe o login do funcionário, verifica se o mesmo existe e recebe as novas modificações.

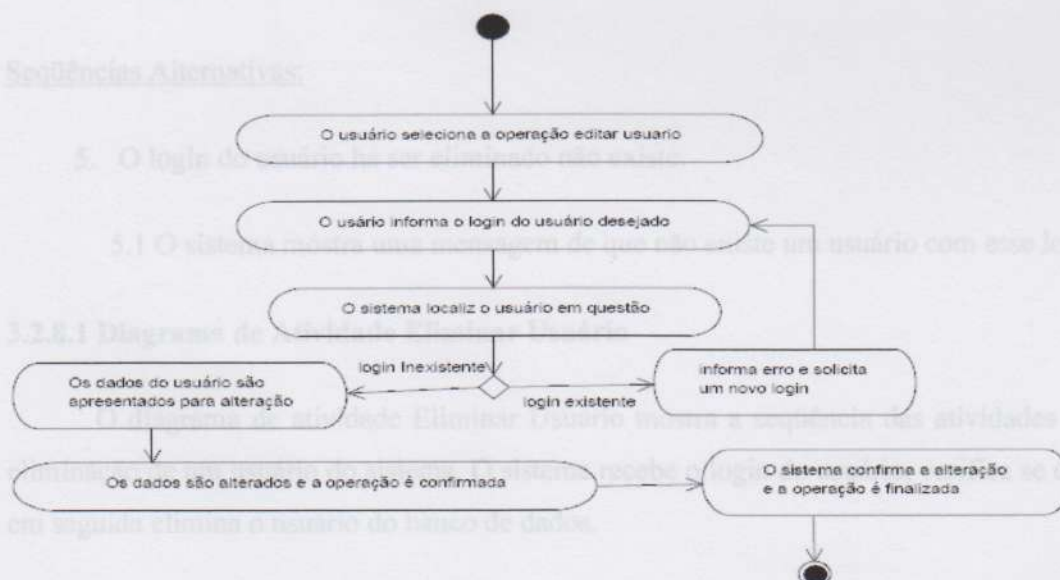


Figura 28: Diagrama de atividade editar usuário.

3.2.8 Caso de Uso Eliminar Usuário

Numero: 09

Caso de Uso: Eliminar Usuário.

Descrição: Este caso de uso elimina as informações sobre os usuário do banco de dados.

Prioridade: baixa.

Ator: Gerente

Seqüência Normal :

1. O usuário inicia o sistema.
2. O sistema exibe um formulário com a opção de controlar usuário.
3. O usuário seleciona a opção de controlar usuário.
4. O sistema mostra um formulário com o campo de preenchimento do login do usuário juntamente com a opção de eliminar usuário.
5. O usuário informa o login do usuário a ser eliminado.
6. O usuário seleciona a opção eliminar o usuário.
7. O sistema exibe as informações do usuário.
8. O sistema elimina o usuário do banco de dados.

Seqüências Alternativas:

5. O login do usuário há ser eliminado não existe.

5.1 O sistema mostra uma mensagem de que não existe um usuário com esse login.

3.2.8.1 Diagrama de Atividade Eliminar Usuário

O diagrama de atividade Eliminar Usuário mostra a seqüência das atividades para a eliminação de um usuário do sistema. O sistema recebe o login do usuário, verifica se existe e em seguida elimina o usuário do banco de dados.

3.3 Modelo de Dados (Banco de Dados)

As figuras abaixo apresentam os diagramas de atividade e de entidade e relacionamento do sistema de Controle de Vendas e Estoque para Pequenas Empresas Locais. É mostrado o relacionamento entre fornecedor e produtos com base no sistema.

Na figura 29, é apresentado o diagrama de atividade para a operação de eliminar usuário. O diagrama mostra a sequência de atividades para a remoção de um usuário do sistema.

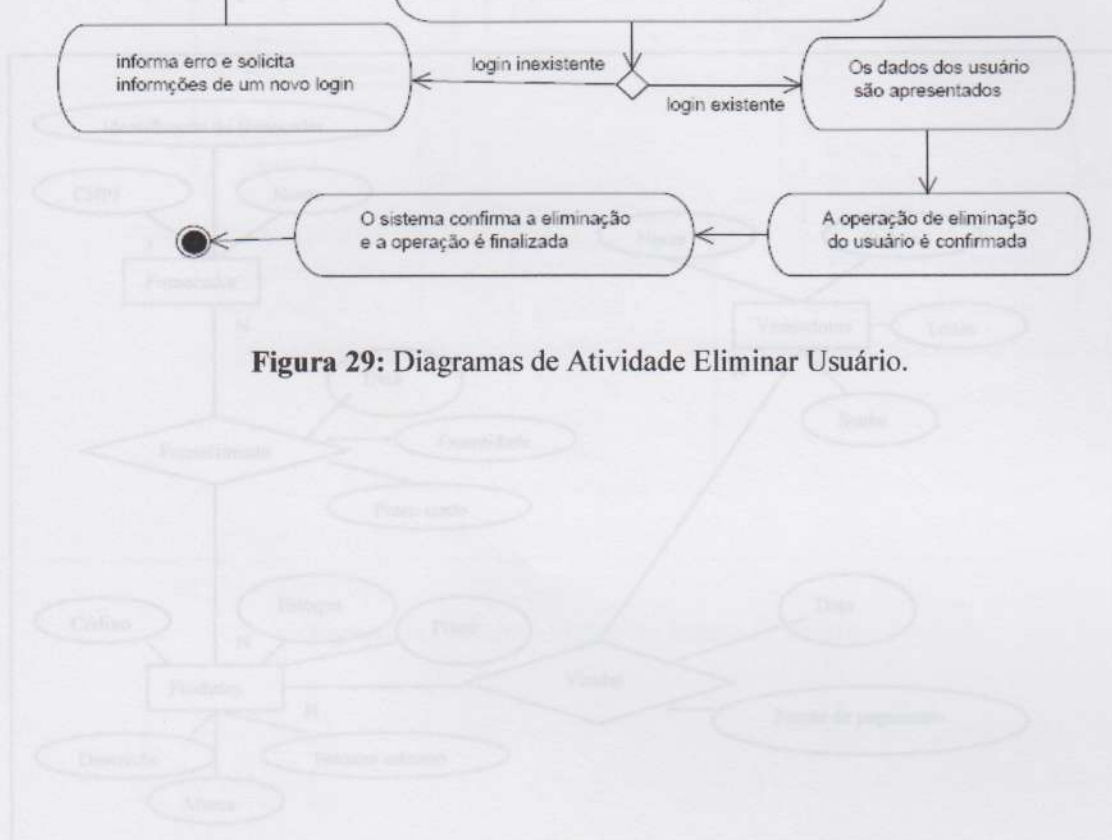


Figura 29: Diagramas de Atividade Eliminar Usuário.

Figura 30: Diagrama de entidade e relacionamento.

3.3 Modelo de Dados (Banco de Dados)

As figuras abaixo representam a estrutura do banco de dados com suas tabelas interligadas.

Na figura 30 é apresentado o diagrama de entidade e relacionamento do sistema de Controle de Vendas e Estoque para Pequenos Estabelecimentos. É mostrado o relacionamento entre fornecedor e produtos como também produtos com vendedores.

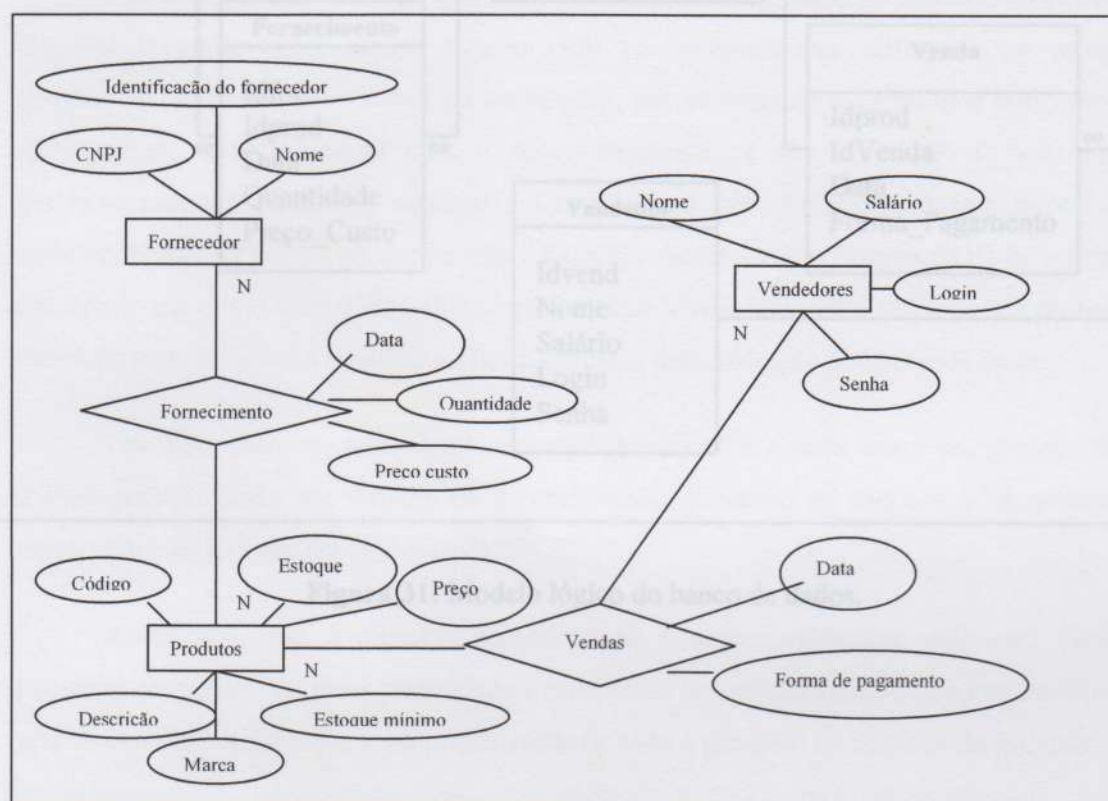


Figura 30: Diagrama de entidade e relacionamento.

A figura 31 representa o modelo lógico do banco de dados do sistema.

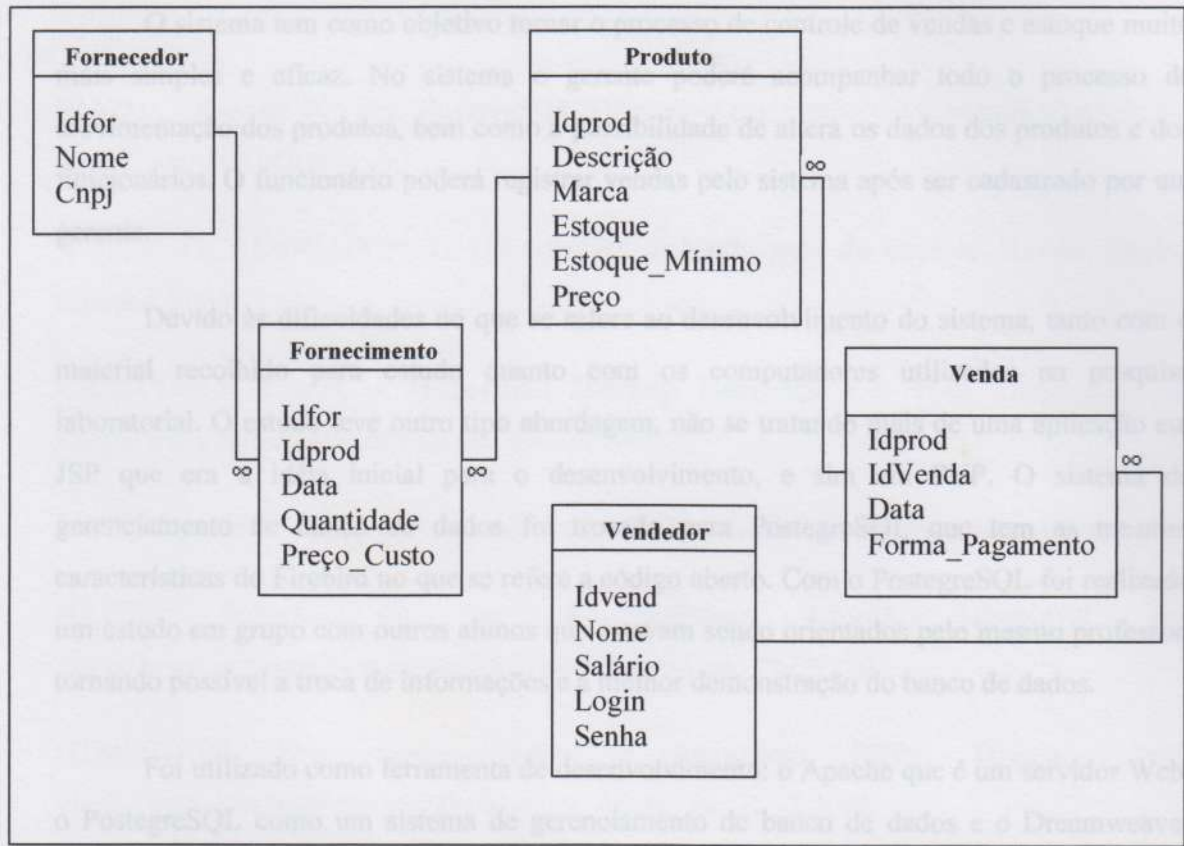


Figura 31: Modelo lógico do banco de dados.

4. CONCLUSÃO

O sistema tem como objetivo tornar o processo de controle de vendas e estoque muito mais simples e eficaz. No sistema o gerente poderá acompanhar todo o processo de movimentação dos produtos, bem como a possibilidade de alterar os dados dos produtos e dos funcionários. O funcionário poderá registrar vendas pelo sistema após ser cadastrado por um gerente.

Devido às dificuldades no que se refere ao desenvolvimento do sistema, tanto com o material recolhido para estudo quanto com os computadores utilizados na pesquisa laboratorial. O estudo teve outro tipo abordagem, não se tratando mais de uma aplicação em JSP que era a idéia inicial para o desenvolvimento, e sim em PHP. O sistema de gerenciamento de banco de dados foi trocado para PostgreSQL que tem as mesmas características do Firebird no que se refere a código aberto. Com o PostgreSQL foi realizado um estudo em grupo com outros alunos que estavam sendo orientados pelo mesmo professor, tornando possível a troca de informações e a melhor demonstração do banco de dados.

Foi utilizado como ferramenta de desenvolvimento: o Apache que é um servidor Web, o PostgreSQL como um sistema de gerenciamento de banco de dados e o Dreamweaver como editor de códigos em HTML e PHP.

Acredita-se que o sistema implementado é uma contribuição relevante para as pequenas empresas, devido à praticidade e custo zero no que diz respeito a software livre. O grande benefício do sistema é a automatização de todo o processo de controle de estoque.

Propõe-se como forma de continuação deste trabalho, uma versão com a devolução de produtos, também aliado com a implementação da linguagem java onde será utilizando os conceitos de orientação a objeto, como se tinha na idéia inicial desse projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

APACHE.org. Disponível em: <<http://www.apache.org/>>. Acesso em: 20/10/2005.

OLIVEIRA, Silva Luis de Tratamento de Metodologia Científica: Projeto de pesquisa.
 BIAZUS, Diogo de Oliveira. **PostgreSQL**. Disponível em: <<http://www.postgresql.org.br>>. Acesso em: 20/10/2005.

PAGE, Kristine. **DREAMWEAVER MX 2004**. Rio de Janeiro: Campus, 2004.
 BOOCH, G., RUMBAUGH, J., JACOBSON, I. **UML, guia do usuário**. Rio de Janeiro: Campus, 2000.

CHICAYBAN, Daniel. **A HyperText Markup Language e a Web**. Disponível em: <<http://www.tol.pro.br>>. Acesso em: 20 out. 2005.

DATE, C., J. **Introdução a Sistema de Banco de Dados**. Rio de Janeiro: Campus, 2000.

HEXEL, Roberto. **O que é Software Livre?**. Disponível em: <<http://www.softwarelivre.gov.br/SwLivre>>. Acesso em: 28/05/2005.

HTMLSTAFF. Org. **Como surgiu a linguagem PHP?**. Disponível em: <<http://www.htmlstaff.org/php/php01.php>>. Acesso em 21/12/2005.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos**. Porto Alegre: Bookman, 2000.

LOZANO, Fernando. **O que é o Software Livre?**. Disponível em: <www.gnu.org/philosophy/free-sw.pt.html>. Acesso em: 23/12/2005.

LÜDKE, Menga & ANDRÉ, Marli. **Pesquisa em educação: abordagens qualitativas**. São Paulo: EPU, 1986.

MARCONDES, Christian Alfin, **Programando em HTML 4.0**. São Paulo: Érica, 1998.

MIDIASWORKS. net. Disponível em: <<http://www.midiaworks.net/dicionario>>. Acesso em: 18/10/2005.

NIEDERAUER, Juliano. **Desenvolvendo Websites com PHP: Aprenda a criar Websites dinâmicos e interativos com PHP e banco de dados**. São Paulo: Novatec, 2004.

NOGUEIRA, Admilsom. **UML - Unified Modeling Language - Atores, Atividades e Componentes**. Disponível em: < www.linhadecodigo.com.br >. Acesso em : 25/11/2005.

OLIVEIRA, Silvo Luiz de. **Tratamento de Metodologia Científica: Projeto de pesquisa, TGI, TCC, Monografias, Dissertações e Tese**. São Paulo: Pioneira, 1997.

PAGE, Khristine. **DREAMWEAVER MX 2004**. Rio de Janeiro: Campus, 2004.

ROCHA, Cerli Antônio da. **Desenvolvendo Web Sites Dinâmicos: PHP, ASP e JSP**. Rio de Janeiro: Campus, 2003.

SILBERSHATZ, A., KORTH, HENRY F., SUDARSHAN, S. **Sistema de Banco de Dados**. São Paulo: MAKRON Books, 1999.

WIKIPÉDIA, a enciclopédia livre. **Servidor Apache**. Disponível em: <http://pt.wikipedia.org/wiki/Servidor_Apache> Acesso em: 20/12/2005.

ANEXO A

```
<?php
    $pag="login";
    include "config.php";
    if(isset($_POST['botao_entrar']))
    {
        $erro="Nenhum";
        $prioridade="Nenhum";
        if($_POST['escrever_login']!="" or $_POST['escrever_senha']!="")
        {
            $erro="O campo Login e Senha deve ser preenchido";
        }
        else
        {
            $login=$_POST['escrever_login'];
            $senha=$_POST['escrever_senha'];
            $resultado=mysqli_query($conexao,"SELECT *from login WHERE login='".$login.'" AND senha='".$senha.'"");
            while($linha=mysqli_fetch_array($resultado))
            {
                $destar_login=$linha['login'];
                $destar_senha=$linha['senha'];
                if($login!=$destar_login or $senha!=$destar_senha)
                {
                    $erro="Os dados Login ou Senha são invalidos";
                }
            }
            if($erro!="Nenhum")
            {
                unset($_SESSION['login']);
                unset($_SESSION['prioridade']);
            }
        }
    }
}
```

ANEXO A

```

<?php
    $pag="login";
    include "config.php";
    if (isset($_POST['botao_entrar']))
    {
        $erro="Nenhum";
        $prioridade="Nenhum";
        if ($_POST['escrever_login']=="" or $_POST['escrever_senha']=="")
        {
            $erro= "O campo Login e Senha tem que estar preenchido";
        }
        else
        {
            $login=$_POST['escrever_login'];
            $senha=$_POST['escrever_senha'];
            $resultado=pg_query("SELECT login,senha,prioridade FROM Vendedores
WHERE login='$login' AND senha='$senha'", $db_con);
            while ($linha = pg_fetch_array($resultado))
            {
                $testar_login=$linha['login'];
                $testar_senha=$linha['senha'];
                $prioridade=$linha['prioridade'];
            }
            if ($login!=$testar_login or $senha!=$testar_senha)
            {
                $erro="Os dados Login ou Senha são invalidos";
            }
        }
        if ($erro!="Nenhum")
        {
            unset($_SESSION['login']);
            unset($_SESSION['prioridade']);
        }
    }

```



```

</body>
<p align="center">Menu Gerentes
<p align="center">Nome: <?php echo $nome; ?> <br> Salario: <?php echo
</html>
<?
}
<p align="center">
?>
<form method="POST" action="vender.php">

```

Figura 32: Código PHP, Login.

ANEXO B

```

<?php
include "config.php";
if ($prioridade != "gerente")
{
    header('location: login.php');
}
$resultado=pg_query("SELECT nome,salario FROM Vendedores WHERE
login='$login'", $db_con);
while ($linha = pg_fetch_array($resultado))
{
    $nome=$linha['nome'];
    $salario=$linha['salario'];
}
?>
<html>
<head>
<meta http-equiv="Content-Language" content="pt">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Menu Gerentes</title>
</head>
<body>

```

```

<p align="center">Menu Gerentes
<p align="center">Nome: <?php echo $nome; ?><br>Salario: <?php echo
$salario; ?></p>

<p align="center">
<form method="POST" action="vender.php">
<input type="submit" value="Fazer Vendas" name="botao_vender">
</form>
</p>
<p align="center">
<form method="POST" action="vendas.php">
<input type="submit" value="Visualizar Vendas" name="botao_vendas">
</form>
</p>
<p align="center">
<form method="POST" action="estoque.php">
<input type="submit" value="Estoque Mínimo" name="botao_estoque">
</form>
</p>
<p align="center">
<form method="POST" action="produtos.php">
<input type="submit" value="Controlar Produtos" name="botao_prod">
</form>
</p>
<p align="center">
<form method="POST" action="acessos.php">
<input type="submit" value="Controlar Acessos" name="botao_acessos">
</form>
</p>
<p align="center"><a href="fecharsessao.php">Fechar Sessão</a></p>
</body>
</html>

```

Figura 33: Código PHP, Menu Gerente.

ANEXO C

```

<?php
    include "config.php";
    if ($prioridade != "funcionario")
    {
        header('location: login.php');
    }

    $resultado=pg_query("SELECT nome,salario FROM Vendedores WHERE
login='$login'", $db_con);
    while ($linha = pg_fetch_array($resultado))
    {
        $nome=$linha['nome'];
        $salario=$linha['salario'];
    }
?>

<html>

<head>
<meta http-equiv="Content-Language" content="pt">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Menu Funcionarios</title>
</head>

<body>
    <p align="center">Menu Funcionarios</p>
    <p align="center">Nome: <?php echo $nome; ?><br>Salario: <?php echo
$salario; ?></p>
    <form method="POST" action="vender.php">
        <p align="center"><input type="submit" value="Fazer Vendas"
name="botao_vender"></p>
    </form>
    <p align="center">
<form method="POST" action="estoque.php">

```

```

<input type="submit" value="Estoque Mínimo" name="botao_estoque">
</form>
</p>
<p align="center"><a href="fecharsessao.php">Fechar Sessão</a></p>
</body>

</html>

```

Figura 34: Código PHP, Menu Funcionário.

ANEXO D

```

<?php
    $pag="vender";
    include "config.php";
    if (isset($_POST['botao_cancelar']))
    {
        unset($_SESSION['itens']);
        unset($_SESSION['nitens']);
        header('location: '.$prioridade.'.php');
    }
    if (isset($_POST['botao_verificar']))
    {
        $idprod=$_POST['idprod'];
        $resultado=pg_query("SELECT * FROM Produtos WHERE idprod='$idprod'
ORDER BY idprod",$db_con);
        while ($linha = pg_fetch_array($resultado))
        {
            $idprod=$linha['idprod'];
            $descricao=$linha['descricao'];
            $marca=$linha['marca'];
            $estoque=$linha['estoque'];
            $estoquemin=$linha['estoquemin'];
            $preco=$linha['preco'];

```

```

    }

    $itens=$itens."idprod=$idprod&descricao=$descricao&marca=$marca&estoque=$estoque&estoquemin=$estoquemin&preco=$preco";

    }

    if (isset($_POST['botao_adicionar']))
    {
        if (isset($_POST['botao_continuar']))
        {
            $idprod=$_POST['idprod'];
            $quant=$_POST['quant'];
            $desco=$_POST['desco'];

            $resultado=pg_query("SELECT * FROM Produtos WHERE idprod='$idprod'
ORDER BY idprod",$db_con);
            while ($linha = pg_fetch_array($resultado))
            {
                $descricao=$linha['descricao'];
                $marca=$linha['marca'];
                $estoque=$linha['estoque'];
                $estoquemin=$linha['estoquemin'];
                $preco=$linha['preco'];
            }

            if ($idprod != "")
            {
                $resultado=pg_query("SELECT estoque FROM Produtos WHERE
idprod=$idprod");
                if ($quant == "" or $quant < 0)
                {
                    while ($linha = pg_fetch_array($resultado))
                    {
                        $quant=1;
                    }
                }
                if ($desco == "" or $desco < 0)
                {
                    $estoque=$estoque-$desco;
                    $desco=0;
                    $resultado=pg_query("UPDATE Produtos SET estoque=$estoque
WHERE idprod=$idprod",$db_con);
                }
                $precop=($preco-($preco*($desco/100)))*$quant;
                $nitens=count($itens);
                $itens=$itens."idprod=$idprod&descricao=$descricao&marca=$marca&estoque=$estoque&estoquemin=$estoquemin&preco=$precop&nitens=$nitens";
            }
        }
    }
}

```

```

        $itens[$nitens]="idprod=$idprod&quant=$quant&desco=$desco&precop=$precop&descricao=$descricao&marca=$marca&estoque=$estoque&estoquemini=$estoquemini&preco=$preco";
        session_register("itens");
    }
}

if (isset($_POST['botao_confirmar']))
{
    $data=date('Y/m/d');
    $pagamento=$_POST['pagamento'];
    $resultado=pg_query("SELECT MAX(idvenda) FROM Vendas",$db_con);
    while ($linha = pg_fetch_array($resultado))
    {
        $idvenda=$linha['MAX(idvenda)'];
    }
    $idvenda++;
    $resultado=pg_query("INSERT INTO Vendas
VALUES('$idvenda','$data','$pagamento')",$db_con);
    foreach($itens as $iten)
    {
        parse_str($iten);
        $resultado=pg_query("SELECT estoque FROM Produtos WHERE
idprod='$idprod'", $db_con);
        while ($linha = pg_fetch_array($resultado))
        {
            $estoque=$linha['estoque'];
        }
        $estoque=$estoque-$quant;
        $resultado=pg_query("UPDATE Produtos SET estoque='$estoque'
WHERE idprod='$idprod'", $db_con);
        $resultado=pg_query("INSERT INTO VendasItens
VALUES('$idvenda','$idprod','$quant','$desco','$precop')",$db_con);
    }
}

```



```

<option value="Dinheiro">Dinheiro</option>
<option value="Cheque">Cheque</option>
<option value="Cartão">Cartão</option>
</select><br>
<input type="submit" value="Confirmar" name="botao_confirmar">
<input type="submit" value="Cancelar" name="botao_cancelar">
</p>
</form>
</body>
</html>

```

Figura 35: Código PHP, Fazer Vendas.

ANEXO E

```

<?php
    include "config.php";
    if ($prioridade != "gerente")
    {
        header('location: login.php');
    }
?>
<head>
<meta http-equiv="Content-Language" content="pt">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Visualizar Vendas</title>
</head>
<body>
    <p align="center">Visualizar Vendas</p>
<?php
    $precototalgeral=0;
    $resultado=pg_query("SELECT * FROM Vendas ORDER BY

```

[illegible]


```

    }
    if (isset($_POST['botao_inserir']))
    {
        $idprod=$_POST['inserir_idprod'];
        $descricao=$_POST['inserir_descricao'];
        $marca=$_POST['inserir_marca'];
        $estoque=$_POST['inserir_estoque'];
        $estoquemin=$_POST['inserir_estoquemin'];
        $preco=$_POST['inserir_preco'];
        $resultado=pg_query("INSERT INTO Produtos
VALUES('$idprod','$descricao','$marca','$estoque','$estoquemin','$preco')",$db_con);
        header('location: produtos.php');
    }
    Else
    {
        if (isset($_POST['botao_editar']))
        {
            $idprod=$_POST['editar_idprod'];
            $descricao=$_POST['editar_descricao'];
            $marca=$_POST['editar_marca'];
            $estoque=$_POST['editar_estoque'];
            $estoquemin=$_POST['editar_estoquemin'];
            $preco=$_POST['editar_preco'];
            $resultado=pg_query("UPDATE Produtos SET idprod='$idprod',
descricao='$descricao', marca='$marca', estoque='$estoque',
estoquemin='$estoquemin', preco='$preco' WHERE idprod='$idprod'", $db_con);
            header('location: produtos.php');
        }
        Else
        {
            if (isset($_POST['botao_confirmar']))
            {
                $idprod=$_POST['escrever_idprod'];

```

```

$resultado=pg_query("SELECT * FROM Produtos WHERE
idprod='$idprod'", $db_con);
while ($linha = pg_fetch_array($resultado))
{
    $descricao=$linha['descricao'];
    $marca=$linha['marca'];
    $estoque=$linha['estoque'];
    $estoquemín=$linha['estoquemín'];
    $preço=$linha['preço'];
}
if ($_POST['escolher_accao'] == "Inserir")
{
    $resultado=pg_query("SELECT count(idprod) FROM Produtos
WHERE idprod='$idprod'", $db_con);
    while ($linha = pg_fetch_array($resultado))
    {
        $total_idprod=$linha['count(idprod)'];
    }
    if ($total_idprod != 0)
    {
        echo '<p align="center">Esse Número do Produto é
Invalido<br>Lembre-se que o Número de Produto apenas pode conter números e não
pode ser duplicado<br><a href="produtos.php">Voltar</a></p>';
    }
    Else
    {
        <p align="center">Inserir Novo Produto</p>
        <form method="POST" action="produtos.php">
        <p align="center">
            Número do Produto: <input type="text" name="inserir_idprod"
size="24" value="<?php echo $_POST['escrever_idprod']; ?>"><br>
            Descrição: <input type="text" name="inserir_descricao"

```

```

size="23"><br>
        Marca: <input type="text" name="inserir_marca"
size="20"><br>
        Estoque: <input type="text" name="inserir_estoque"
size="7"><br>
        Estoque Mínimo: <input type="text" name="inserir_estoquemin"
size="14"><br>
        Preço: R$ <input type="text" name="inserir_preco"
size="5"><br>
        <br>
        <input type="submit" value="Inserir"
name="botao_inserir"></p>
</form>
        <p align="center"><a href="produtos.php">Voltar</a></p>
        <?php
        }
    }
    if ($_POST['escolher_accao'] == "Editar")
    {
        <?>
        <p align="center">Editar Produto Número <?php echo $idprod;
?></p>
        <form method="POST" action="produtos.php">
        <p align="center">
            Número do Produto: <input type="text" name="editar_idprod"
size="24" value="<?php echo $idprod; ?>"><br>
            Descrição: <input type="text" name="editar_descricao" size="23"
value="<?php echo $descricao; ?>"><br>
            Marca: <input type="text" name="editar_marca" size="20"
value="<?php echo $marca; ?>"><br>
            Estoque: <input type="text" name="editar_estoque" size="7"
value="<?php echo $estoque; ?>"><br>
            Estoque Mínimo: <input type="text" name="editar_estoquemin"

```

```

size="14" value="<?php echo $estoquemin; ?>"><br>
    Preço: R$ <input type="text" name="editar_preco" size="5"
value="<?php echo $preco; ?>"><br>
    <br>
    <input type="submit" value="Editar" name="botao_editar"></p>
</form>
<p align="center"><a href="produtos.php">Voltar</a></p>
<?php
}
if ($_POST['escolher_accao'] == "Eliminar")
{
    $resultado=pg_query("DELETE FROM Produtos WHERE
idprod='$idprod'", $db_con);
    header('location: produtos.php');
}
}
Else
{
    ?>
</html>

<head>
<meta http-equiv="Content-Language" content="pt">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Controlar Produtos</title>
</head>
<body>
<p align="center">Controlar Produtos
<p align="center"><a target="_blank" href="verprodutos.php">Ver Todos os
Produtos</a></p>
<form method="POST" action="produtos.php">
<p align="center">

```

Figura 37: Código PHP, Controlar Produtos

ANEXO G

ANEXO H

```
?php
    include "config.php";
    if ($prioridade != "gerente")
    {
        header('location: login.php');
    }
    if (isset($_POST['botao_inserir']))
    {
        $vendedor=$_POST['inserir_login'];
        $senha=$_POST['inserir_senha'];
        $nome=$_POST['inserir_nome'];
        $salario=$_POST['inserir_salario'];
        $prioridade=$_POST['escolher_prioridade'];
        $resultado=pg_query("INSERT INTO Vendedores VALUES('$vendedor',
'$senha', '$nome', '$salario', '$prioridade')", $db_con);
        header('location: acessos.php');
    }
    Else
    {
        if (isset($_POST['botao_editar']))
        {
            $vendedor=$_POST['editar_login'];
            $senha=$_POST['editar_senha'];
            $nome=$_POST['editar_nome'];
            $salario=$_POST['editar_salario'];
            $prioridade=$_POST['escolher_prioridade'];
            $resultado=pg_query("UPDATE Vendedores SET login='$vendedor',
senha='$senha', nome='$nome', salario='$salario', prioridade='$prioridade' WHERE
login='$vendedor'", $db_con);
            header('location: acessos.php');
        }
        Else
```

```

{
    if (isset($_POST['botao_confirmar']))
    {
        $vendedor=$_POST['escrever_login'];
        $resultado=pg_query("SELECT * FROM Vendedores WHERE
        login='$vendedor'", $db_con);
        while ($linha = pg_fetch_array($resultado))
        {
            $nome=$linha['nome'];
            $senha=$linha['senha'];
            $salario=$linha['salario'];
            $prioridade=$linha['prioridade'];
        }
        if ($_POST['escolher_accao'] == "Inserir")
        {
            $resultado=pg_query("SELECT count(login) FROM Vendedores
            WHERE login='$vendedor'", $db_con);
            while ($linha = pg_fetch_array($resultado))
            {
                $total_login=$linha['count(login)'];
            }
            if ($total_login != 0)
            {
                echo '<p align="center">Esse Login já existe e não pode ser
                duplicado<br><a href="acessos.php">Voltar</a></p>';
            }
            Else
            {
                echo '<p align="center">Inserir Novo Vendedor</p>';
                <form method="POST" action="acessos.php">
                <p align="center">
                Login: <input type="text" name="inserir_login" size="24"

```

```

value="<?php echo $_POST['escrever_login']; ?>"><br>
value="<?php echo $nome; ?>"> Nome: <input type="text" name="inserir_nome"
size="23"><br>
echo $prioridade; ?> Prioridade: <input type="text" name="inserir_senha"
size="20"><br>
if ($prioridade) Salário: <input type="text" name="inserir_salario"
size="7"><br>
Prioridade: <select size="1" name="escolher_prioridade">
<option value="funcionario">Funcionário
value="funcionario">Funcionário
</option>
value="gerente">Gerente</option>
value="gerente">Gerente</option>
</select><br>
<input type="submit" value="Inserir"
name="botao_inserir"></p>
</form>
<p align="center"><a href="acessos.php">Voltar</a></p>
<?php
}
}
if ($_POST['escolher_accao'] == "Editar")
{
?>
<p align="center">Editar Login <?php echo $vendedor; ?></p>
<form method="POST" action="acessos.php">
<p align="center">
Login: <input type="text" name="editar_login" size="24" value="<?php
echo $vendedor; ?>"><br>
Nome: <input type="text" name="editar_nome" size="23"
value="<?php echo $nome; ?>"><br>
Senha: <input type="text" name="editar_senha" size="20"
value="<?php echo $senha; ?>"><br>

```

```

Salário: <input type="text" name="editar_salario" size="7"
value="<?php echo $salario; ?>"><br>
Prioridade: <select size="1" name="escolher_prioridade" value="<?php
echo $prioridade; ?>">

    <?php
    if ($prioridade == "gerente")
    {
        <option value="funcionario">Funcionário
    }
    <option selected
value="gerente">Gerente</option>
    <?php
    }
    Else
    {
        <option selected value="funcionario">Funcionário
    }
    <option
value="gerente">Gerente</option>

    <?php
    }
    }
    </select><br>
    <input type="submit" value="Editar" name="botao_editar"></p>
</form>
<p align="center"><a href="acessos.php">Voltar</a></p>
    <?php
    }
    if ($_POST['escolher_accao'] == "Eliminar")
    {
        $resultado=pg_query("DELETE FROM Vendedores WHERE

```

[illegible]

```

</html>
<?php
<div>
    <p align="center">Login:<?php echo $vendedor;
?><br><br><br> Prioridade:<?php echo $prioridade; ?>
?>
</div>Nome:<?php echo $nome; ?><br><br><br><br>
<div><?php echo $salario; ?>
</div>
</div>
<?php

```

Figura 39: Código PHP, Controlar Acesso.

ANEXO I

```

<?php
    include "config.php";
    if ($prioridade != "gerente")
    {
        header('location: login.php');
    }
?>
<head>
<meta http-equiv="Content-Language" content="pt">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Visualizar Acessos</title>
</head>
<body>
    <p align="center">Visualizar Acessos</p>
<?php
    $resultado=pg_query("SELECT * FROM Vendedores ORDER BY
login",$db_con);
    while ($linha = pg_fetch_array($resultado))
    {
        $vendedor=$linha['login'];
        $nome=$linha['nome'];
        $salario=$linha['salario'];
    }

```

Figura 40: Código PHP, Visualizar Acesso.

