



UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ
FACULDADES LUIZ MENEGHEL



DANILO QUILICI

**SISTEMA DE CONTROLE DE ESTOQUE DE
LUBRIFICANTES AUTOMOTIVOS**

Bandeirantes

2009



UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ
FACULDADES LUIZ MENEGHEL



DANILO QUILICI

SISTEMA DE CONTROLE DE ESTOQUE DE LUBRIFICANTES AUTOMOTIVOS

Trabalho de Conclusão de Curso submetido às Faculdades Luiz Meneghel da Universidade Estadual do Norte do Paraná, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Msc. Ailton Sergio Bonifacio.

Bandeirantes

2009

DANILO QUILICI

**SISTEMA DE CONTROLE DE ESTOQUE DE
LUBRIFICANTES AUTOMOTIVOS**

Trabalho de Conclusão de Curso submetido às Faculdades Luiz Meneghel da Universidade Estadual do Norte do Paraná, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

Prof. Msc Ailton Sergio Bonifácio - FALM

Prof. Msc. Glauco Carlos Silva - FALM

Prof. Dra. Marília Abrahão Amaral - FALM

Bandeirantes, 10 de junho de 2009.

À minha família, por todo amor demonstrado e valores transmitidos. À vocês, dedico esta vitória.

AGRADECIMENTOS

Agradeço primeiramente à Deus, pois sem ele nada disso seria possível.

À toda minha família: meu irmão Fernando, minha avó Maria Célia, minhas tias Daniela, Fernanda e Ana, meu tio e grande amigo Junior, aos meus primos Felipe, João Vitor e Ana Clara, e especialmente aos meus querido pais, Fernando e Shirlei. Agradeço pela força e compreensão nos tantos momentos ao longo desta trajetória, em que foi necessário, apesar da saudade, estar distante de todos.

Agradeço à minha namorada, Thaís, por sempre estar ao meu lado, me ajudando e me incentivando a seguir em frente.

Ao Prof. Msc. Ailton Sergio Bonifácio, pela orientação na realização deste trabalho e durante toda a graduação.

À todos os colegas de trabalho (Fabio, Mateus, Rodrigo, Leonardo, Marcos, Anderson e Ricardo), pelo grande apoio e ajuda nos momentos de dificuldades.

Muito Obrigado!

RESUMO

O presente trabalho aborda o desenvolvimento de um sistema de controle de estoque de lubrificantes automotivos. Como características do mesmo têm-se o cadastro de clientes, funcionários, fornecedores, veículos, entre outros. O foco principal do sistema é o controle de entrada e saída de mercadorias. Para tal desenvolvimento foi empregado o uso de algumas tecnologias. Conceitos relacionados a estas tecnologias foram descritos de forma a comprovar sua usabilidade, e mostrar suas vantagens. O estudo teve por base o entendimento de conceitos de orientação objeto, padrões de projetos, sintaxe da linguagem de programação C#, framework de persistência NHibernate e da ferramenta de gerenciamento de banco de dados Microsoft SQL Server 2005. Ao final deste estudo, almejou-se ter um software estável e seguro para assim poder ter um maior controle de toda a movimentação da empresa que o utiliza.

Palavras-Chave: Estoque, lubrificantes, padrões de projeto.

ABSTRACT

This study presents the development of a stock control system for automotive lubricants. The characteristics of this system are the register of customers, employees, suppliers, vehicles, among others. The main function of the system is the control of input and output of goods. In order to develop this work, some technologies were used. Concepts related to these technologies were described to prove its usability and show its advantages. The study was based on the acknowledge of the object orientation, design patterns, syntax of the programming language C #, NHibernate persistence framework and management tool Database Microsoft SQL Server 2005. The purpose of this study is to provide a stable and secure software to the companies which use this system, allowing them a better control of their financial activities.

Keywords: Stock, lubricants, design patterns.

LISTA DE SIGLAS

API	Application Programming Interface
BI	Business Intelligence
BDR	Bancos de Dados Relacionais
CNPJ	Cadastro Nacional de Pessoa Jurídica
CLR	Common Language Runtime
COM	Common Object Model
DLLs	Dynamic-link library
DENATRAN	Departamento Nacional de Trânsito
GoF	Gang of Four
RENAVAM	Registro Nacional de Veículos Automotores
IDE	Integrated Development Environment
LINQ	Language-Integrated Query
MVC	Modelo, Visualização e Controle
MSIL	Microsoft Intermediate Language
ORM	object/relational
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
UML	Unified Modeling Language
WEB	Web World Wide
XML	Extensible Markup Language

LISTA DE FIGURAS

Figura 1: Padrão Observer.....	20
Figura 2: Padrão strategy.....	21
Figura 3: Padrão singleton.	22
Figura 4: Padrão MVC.....	23
Figura 5: Diagrama de Classes.....	32
Figura 6: Diagrama de Caso de Uso.....	33
Figura 7: Diagrama de Sequência.....	34
Figura 8: Diagrama de Entidade Relacionamento.....	35
Figura 9: Método da interface ISubjetc.....	37
Figura 10: Método da interface IObject.....	37
Figura 11: Classe Singleton.....	38
Figura 12: Arquivo xml de conexão com o banco de dados.....	39
Figura 13: Classe Pessoa.....	40
Figura 14: Arquivo xml de mapeamento da classe pessoa.....	41
Figura 15: Tela de Login.....	42
Figura 16: Tela principal.....	43
Figura 17: Tela de dados do cliente.....	44
Figura 18: Tela de Cadastro de Clientes.....	45
Figura 19: Tela de Entrada de Notas.....	46
Figura 20: Tela de Vendas.....	47

LISTA DE TABELAS

Tabela 1: Ferramentas utilizadas no desenvolvimento do sistema de controle de estoque de lubrificantes automotivos e suas respectivas finalidades	31
---	----

SUMÁRIO

1 INTRODUÇÃO	12
2 OBJETIVOS.....	13
2.1 OBJETIVO GERAL	13
2.2 OBJETIVOS ESPECÍFICOS	13
3 JUSTIFICATIVA.....	14
4 FUNDAMENTAÇÃO TEÓRICA	15
4.1 ORIENTAÇÃO OBJETO	15
4.1.1 ATRIBUTOS OU PROPRIEDADES	16
4.1.2 MÉTODOS OU COMPORTAMENTOS	16
4.1.3 ENCAPSULAMENTO.....	17
4.1.4 HERANÇA	17
4.1.5 POLIMOFISMO	18
4.2 PADRÕES DE PROJETO.....	18
4.2.1 PADRÃO OBSERVER	20
4.2.2 PADRÃO STRATEGY	21
4.2.3 PADRÃO SINGLETON	22
4.2.4 MVC	23
4.3 FRAMEWORK .NET	24
4.4 C# (C SHARP).....	25
4.5 MICROSOFT SQL SERVER 2005 EXPRESS EDITION	27
4.6 NHIBERNATE.....	27
4.7 MICROSOFT VISUAL STUDIO 2008	28
5 SISTEMA PROPOSTO	29
5.1 LEVANTAMENTO DE REQUISITOS	30
5.2 FERRAMENTAS UTILIZADAS	31
5.3 MODELAGENS DA FERRAMENTA	32
5.3.1 DIAGRAMAS DE CLASSES	32
5.3.2 DIAGRAMA DE CASOS DE USO.....	33
5.3.3 DIAGRAMA DE SEQUÊNCIA	34

5.3.4 DIAGRAMA DE ENTIDADE RELACIONAMENTO	35
5.4 IMPLEMENTAÇÃO DO SISTEMA	36
5.5 INTERFACE GRÁFICA	42
6 CONSIDERAÇÕES FINAIS	48
REFERÊNCIAS	49

1 INTRODUÇÃO

Tanto o homem quanto o automóvel são frutos de um processo evolutivo. Desde 1765, quando Nicolas-Joseph Cugnot criou o primeiro auto-veículo do mundo (STEINBRUCH, 2005), iniciaram-se sucessivas aproximações e adaptações tecnológicas a fim de desenvolver um modelo que atingisse maior velocidade, comodidade e segurança (IROKAWA; CUNHA; CÂMARA, 2007).

Hoje, os automóveis não representam apenas um meio de transporte imprescindível à vida e um símbolo de status social, mas também uma extensão da casa ou do trabalho, já que grande parte do dia se passa dentro deste (IROKAWA; CUNHA; CÂMARA, 2007).

Segundo as estatísticas do Departamento Nacional de Trânsito (DENATRAN) através do Registro Nacional de Veículos Automotores (RENAVAM), a frota de veículos tem aumentado muito significativamente nos últimos 10 anos em todas as unidades federativas do Brasil, ou seja, ocorreu um aumento de 30.145.314 veículos entre os anos de 1998 e 2008 (DENATRAN 1998; DENATRAN 2008).

Na justificativa de sustentar todo este montante, existem cerca de 35.017 postos de abastecimento espalhados por todo o território nacional (ANP, 2007), além de um grande número de empresas relacionadas com a manutenção dos veículos, como casas de óleo e oficinas mecânicas. A fim de manter estas unidades com destaque no competitivo panorama brasileiro, softwares inovadores são necessários para a implementação de novas tecnologias e diferenciais na conquista de clientes.

O foco deste trabalho está em desenvolver uma ferramenta que seja capaz de garantir um controle exato das movimentações da empresa, pois esta mostra, através de relatórios, a situação atual da mesma, evitando assim perdas em geral e auxiliando o usuário nas tomadas de decisões. Para tanto, são utilizadas as técnicas de orientação objeto, padrões de projetos, sintaxe da linguagem de programação C#, framework de persistência (NHibernate) e da ferramenta de gerenciamento de banco de dados (Microsoft SQL Server 2005).

2 OBJETIVOS

2.1 Objetivo Geral

O propósito deste trabalho é o desenvolvimento de uma aplicação que auxilia postos de combustíveis e casas de lubrificantes no controle de entradas e saídas de suas mercadorias. Com isso, o usuário terá maior controle da movimentação dos produtos de sua empresa podendo tomar decisões futuras, como compra e venda de produtos, com maior facilidade e segurança.

2.2 Objetivos Específicos

Os objetivos específicos deste projeto são:

- Utilização das técnicas de programação orientada a objeto.
- Implementação baseada em padrões de projeto:
 - Observer.
 - Factory.
 - MVC (Modelo, Visualização e Controle)
 - Singleton.
- Utilização da tecnologia .Net juntamente com a linguagem C#.
- Utilização da ferramenta NHibernate auxiliando na persistência de dados.
- Utilização do banco de dados Structured Query Language (SQL) SERVER 2005.

3 JUSTIFICATIVA

Devido à grande quantidade de postos de combustíveis e casas de lubrificantes, verificou-se a necessidade de um diferencial como atrativo na conquista de clientes.

A utilização deste software, que informa a situação da empresa para o usuário, através de seus relatórios e controle de estoque, de forma simples e precisa, permite uma maior organização e praticidade do estabelecimento. Esta organização e eficiência fornecida pelo conjunto usuário-sistema torna o cliente cada vez mais satisfeito, refletindo na conquista de novos consumidores e conseqüentemente maior lucro para o estabelecimento.

4 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão relatados conceitos relacionados à área de pesquisa de projeto.

4.1 ORIENTAÇÃO OBJETO

A evolução das linguagens de programação é derivada do grande aumento dos sistemas tanto em tamanho e complexibilidade, sendo criadas novas técnicas e linguagens que as utilizam (BERG; FIGUEIRÓ, 1999).

No princípio, foi criada a linguagem de programação Assembly para desenvolver programas através de representações simbólicas das instruções de máquina. Com aumento dos programas, as linguagens de auto nível e a programação estruturada também foram criadas. O crescimento contínuo dos programas levou a criação da técnica de programação orientada a objetos, permitindo que um problema seja mais facilmente decomposto em subgrupos relacionados, podendo-se traduzir de unidade autocontidas chamadas objeto (BERG; FIGUEIRÓ, 1999).

O ser humano desde criança aprende a pensar de maneira orientada a objetos representando o seu conhecimento por meio de abstrações e classificações. Por exemplo, ao aprenderem os simples conceitos de pessoas, carro, casa, etc. e ao fazerem isso definem classes, grupos de objetos, onde cada objeto é um membro de um determinado grupo, assim possuindo as mesmas características e comportamentos de qualquer objeto do grupo em questão. Então qualquer coisa que tenha cabeça, tronco e membros se torna uma pessoa, como qualquer construção onde pessoas possam morar passa a ser uma casa (GUEDES, 2008).

O processo em questão exige um grande esforço de abstração devido as casas apresentarem diferentes formatos, cores e estilos. A partir desse momento é preciso abstrair o conceito de “casa” para chegar à conclusão de que é um termo que se refere a muitos objetos, que, no entanto, cada um possui características semelhantes entre si (GUEDES, 2008).

Percebendo com clareza que “casa” é a denominação de um grupo, pode se dizer que foi abstraído a classe “casa”. Assim quando perceber a presença de um objeto com as características já determinadas, concluirá que o objeto é um exemplo de um grupo, ou seja, uma instância da classe casa (GUEDES, 2008).

Ao instanciar um novo objeto de uma determinada classe, é criado um novo item do conjunto representado pela mesma, obtendo as mesmas características e comportamentos dos outros objetos já instanciados (GUEDES, 2008).

Contudo, observar que apesar de possuírem os mesmos atributos, os objetos de uma mesma classe não são exatamente iguais, porém cada objeto armazena diferentes valores em seus atributos. Por exemplo, todos os objetos da classe casa contêm o atributo cor, mas podem existir objetos que o atributo cor terá como valor a cor amarela e outro a cor vermelha (GUEDES, 2008).

4.1.1 ATRIBUTOS OU PROPRIEDADES

Atributos ou propriedades são definidos na classe e representam as características de uma classe, ou seja, valores que costumam variar de um objeto e outro, como a cor de um objeto da classe casa, permitindo assim diferenciar objetos da mesma classe (GUEDES, 2008).

4.1.2 MÉTODOS OU COMPORTAMENTOS

Um método representa uma ação que um objeto de uma classe pode executar, e pode ser comparada a uma função (RAMOS, 2009). Essas funções podem ou não receber parâmetros e em geral, retornam valores que podem representar o resultado da operação executado ou simplesmente indicar se o processo foi concluído (GUEDES, 2008).

4.1.3 ENCAPSULAMENTO

Em programação orientada a objetos, encapsulamento refere-se à habilidade de um objeto esconder seus atributos e ao método da visão exterior, permitindo somente acesso à métodos públicos. Tal proteção consiste em utilizar modificadores mais restritos sobre os atributos da classe e fornecer métodos que alterem esses atributos de alguma forma. Porém, ajuda a prevenir a interferência externa indevida sobre os dados de um objeto (ARAÚJO, 2009).

Dessa forma, o encapsulamento evita a interdependência de um programa, fazendo com que as futuras alterações não causem grandes efeitos colaterais (ARAÚJO, 2009).

4.1.4 HERANÇA

Herança consiste em criar novas classes a partir das classes já existentes, assim, reaproveitando seus atributos e métodos, ou melhor, suas propriedades e comportamentos. (ARAÚJO, 2009).

Com a herança é possível trabalhar com os conceitos de superclasse e subclasse. Superclasse, chamada também de classe mãe, consiste em classes que possuem classes derivadas. Já as subclasses, chamadas de classes filha, são classes derivadas. As subclasses, por serem derivadas as superclasse, herdam todas as suas características (GUEDES, 2008).

Apresenta como vantagem o reaproveitamento de código, isto significa que, uma subclasse herda automaticamente os atributos e métodos de sua superclasse, não sendo necessário redeclará-los, conseqüentemente permitindo a reutilização do código pronto (ARAÚJO, 2009; GUEDES, 2008).

4.1.5 POLIMOFISMO

Polimorfismo está associado à herança e quer dizer várias formas. É a redeclaração de métodos herdados que diferem de alguma forma da implementação utilizada na superclasse, sendo necessário reimplementá-los na subclasse. (GUEDES, 2008).

Por exemplo, uma superclasse Pessoa tem um método chamando ValidaDocumento. O método da superclasse está implementado para validar CPF. Essa superclasse é herdada pela classe PessoaFisica e PessoaJuridica. A primeira classe poderá utilizar o método sem problema algum, já a outra classe terá que reimplementá-lo para que ele valide o cadastro nacional de pessoa jurídica (CNPJ) (ARAÚJO, 2009).

4.2 PADRÕES DE PROJETO

No final da década de 70, Christopher Alexander criou os padrões de projeto. Ele escreveu dois livros que especificavam e descreviam o uso dos padrões de projetos arquitetônicos (ALEXANDER, 1979), mais especificamente no planejamento de projetos de edifícios e cidades (GAMA, 2000). Alexander provou que os métodos de construção utilizados não estavam satisfazendo os indivíduos e a sociedade, então iniciou a busca pela melhor qualidade dos produtos, intuitivamente a satisfação da sociedade. Estabeleceu 253 padrões que descreviam em forma de catálogo, os problemas que se repetiam em diferentes locais ao mesmo tempo em que e criava soluções para os mesmo. Os padrões alcançavam a qualidade, pois Alexander busca a comunicação com as pessoas que participavam direta ou indiretamente do projeto (SALVIANO, 1997; LEA, 1993). Desta forma, pode-se reutilizar uma solução na construção de civil e obter resultados diferentes, otimizando tempo, sendo capaz satisfazer o usuário final (WELFER, 2005).

A idéia de utilizar padrões de projetos em desenvolvimento de software foi de Gamma (2000) e Buschmann (1996), gerando uma nova maneira de pensar em projetos de softwares (MAY; TAYLOR, 2003).

Essa nova abordagem junto com o paradigma de programação orientada a objetos levou a criação de 23 padrões de projeto conhecidos como padrões de Gangue dos Quatro (Gang of Four - "GoF") que se refere a Erich Gamma e seus colegas. Os mesmos são divididos em três categorias: padrões de criação , padrões estruturais e padrões comportamentais. O primeiro representa formas para criação ou instanciação de objetos. O segundo identifica os padrões que compõe as classes ou grupos de classes (WELFER, 2005). Já o terceiro e último, caracterizam a interação e responsabilidade entre os objetos, isto é, comunicação entre eles (COOPER, 1998). Assim, através desse catálogo de padrões de projeto, consegue-se solucionar problemas de projeto assegurando criar componentes adaptáveis e reutilizáveis (WELFER, 2005).

4.2.1 PADRÃO OBSERVER

O padrão observer (Figura 1) define uma dependência entre os objetos de um para muitos. Tal dependência serve para que quando um objeto mude de estado, todos os outros objetos dependentes sejam avisados e atualizados automaticamente (FREEMAN, 2007a).

Os objetos podem ter dois estados, o estado de observador (Subject) e o estado de observado (Observer). O subject fica observando se ocorre alguma alteração nos objetos, caso ocorra este atualiza automaticamente todos os seus observadores. Já os observers ficam esperando por atualizações (CARVALHO,2007).

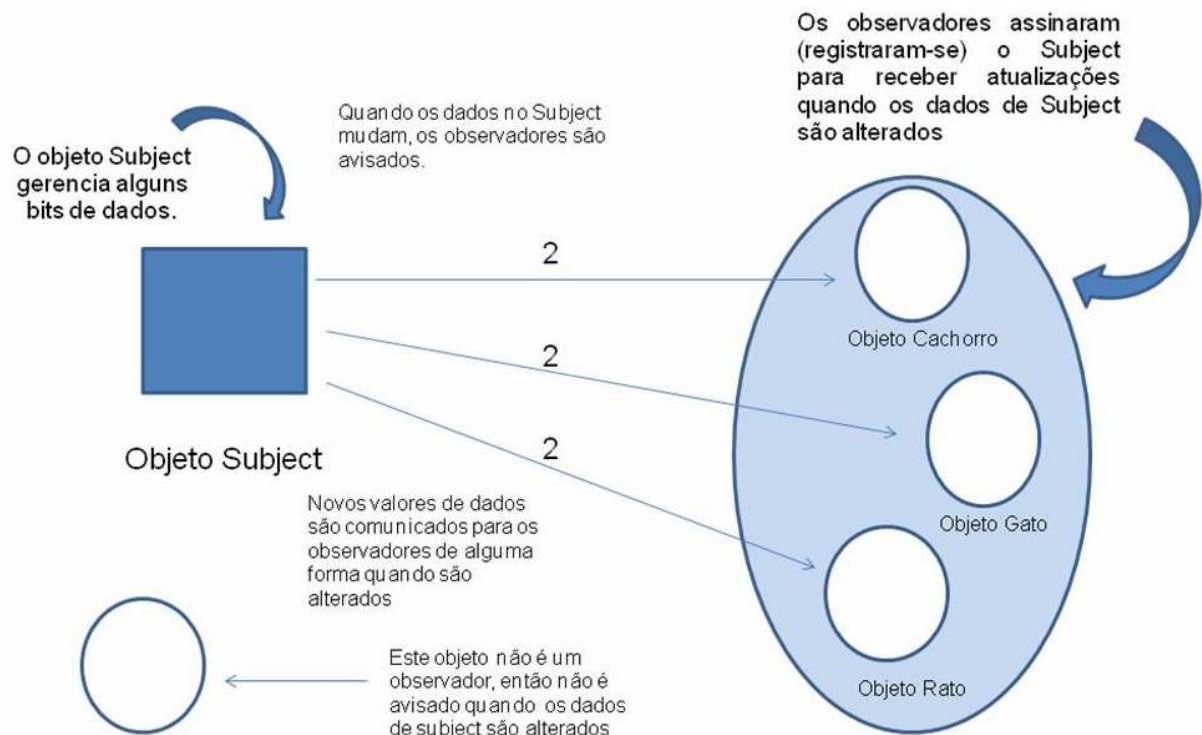


Figura 1: Padrão Observer.

Fonte: FREEMAN, 2007a

4.2.2 PADRÃO STRATEGY

O Padrão strategy (Figura 2) cria uma família de algoritmos, encapsulando cada um deles e assim os tornando intercambiáveis. Desta forma, os algoritmos podem variar independente dos clientes que o utilizam (FREEMAN, 2007b; GARCIA, 2009).

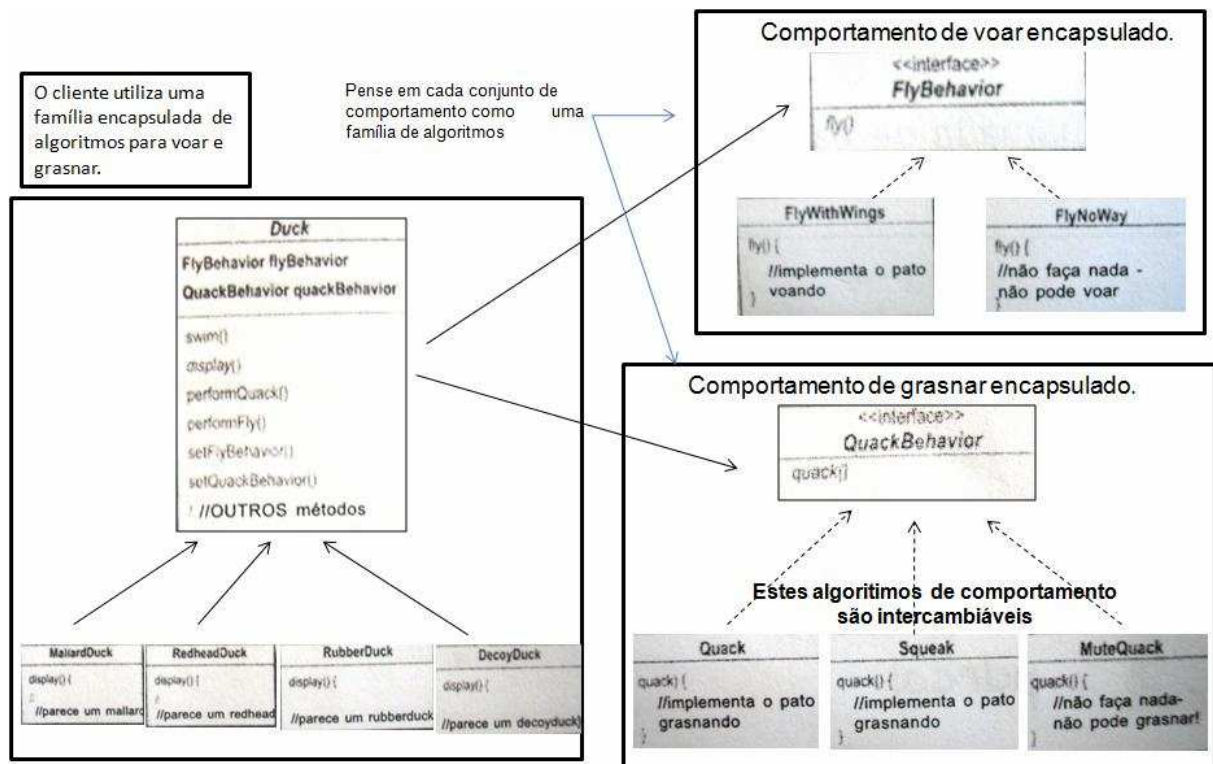


Figura 2: Padrão strategy.

Fonte: FREEMAN, 2007b.

4.2.3 PADRÃO SINGLETON

O padrão singleton (Figura 3) tem como objetivo garantir uma única instância acessível de forma global e uniforme por toda classe que implemente este padrão (FREEMAN, 2007c; PAULA, 2006).

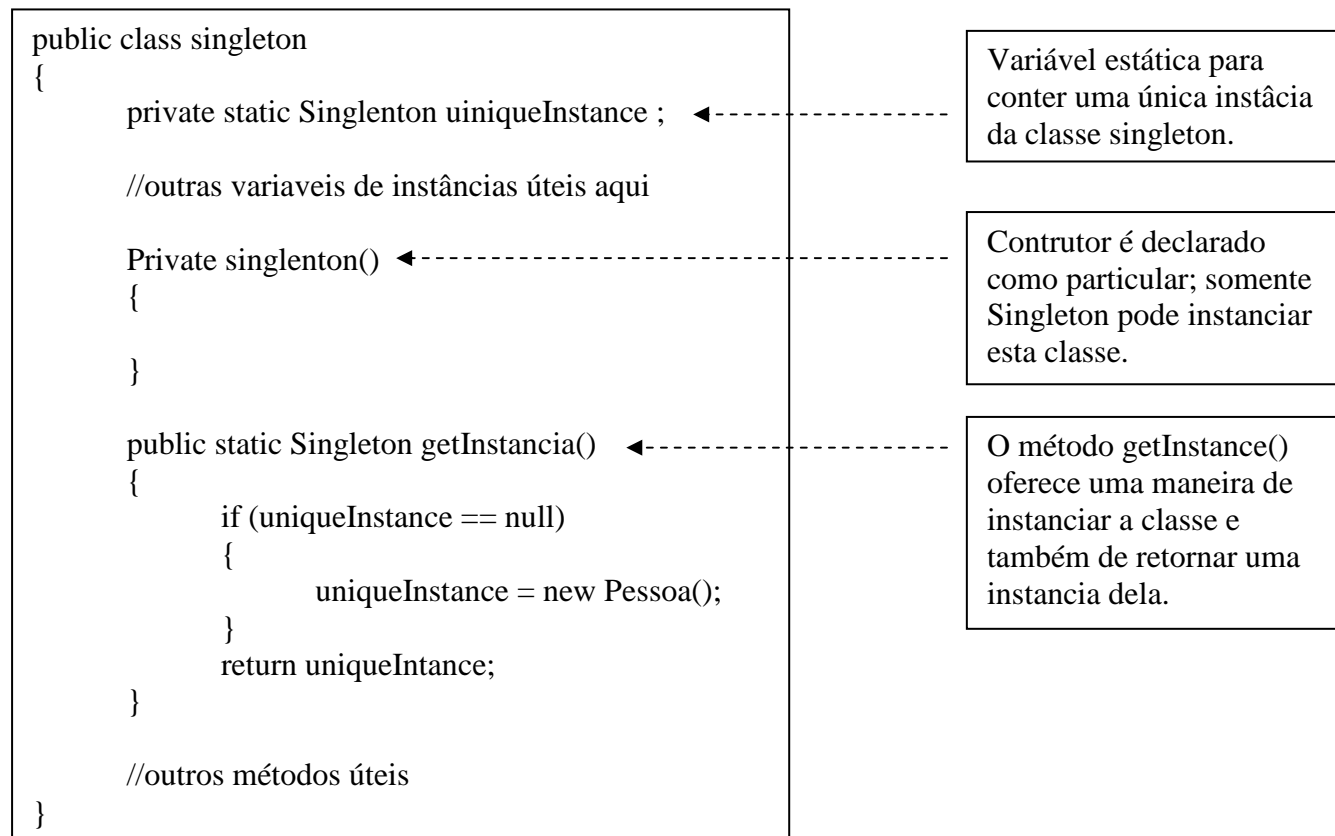


Figura 3: Padrão singleton.

Fonte: FREEMAN, 2007c

4.2.4 MVC

O padrão MVC (Figura 4) tem como objetivo a separação de responsabilidades entre os componentes. Por exemplo, todo código referente à regra de negócio deve ficar em uma classe modelo, já a camada de comunicação com o usuário, a view, deve conter somente código referente a ela. E a classe de controle faz a comunicação entre o modelo e a view (FREEMAN, 2007d; NUNES, 2005).

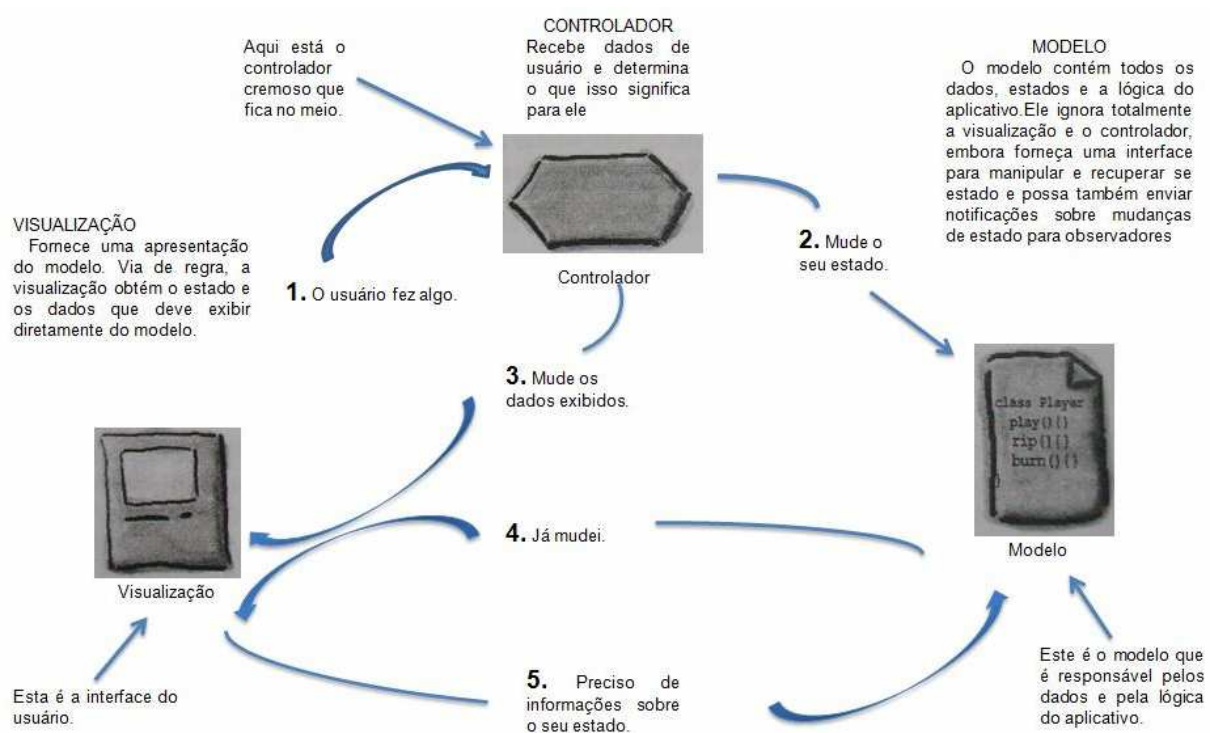


Figura 4: Padrão MVC

Fonte: FREEMAN, 2007d.

4.3 FRAMEWORK .NET

Considerado como um componente integral do Windows responsável por fornecer os serviços necessários para construção e carregamento de aplicações para Windows e Web (ANJOS, 2009), o .NET foi lançado para o mundo em 2002 durante o Professional Developer Conference (PDC) (UCHÔA, 2006). Esta é uma iniciativa da Microsoft que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações (WIKPEDIA, 2009b)

Com idéia semelhante à plataforma Java, o .Net Freamwork consiste em abstrair a necessidade do desenvolvedor interagir com o sistema operacional (MIRANDA, 2008; UCHÔA, 2006) ou seja, todo e qualquer código gerado para o .Net pode ser executado em qualquer dispositivo ou plataforma que o possua (MIRANDA, 2008).

O .Net Freamwork é composto pelo Common Language Runtime (CLR), Microsoft Intermediate Language (MSIL) (UCHÔA, 2006) e Framework Class Library (Biblioteca de Classes). CRL é a parte principal da plataforma, que funciona semelhante à uma máquina virtual e se encarrega de providenciar a execução das aplicações para ela escritas. Além disso, oferece segurança, confiança (WIKPEDIA, 2009a), interoperabilidade (ANJOS, 2009; WIKPEDIA, 2009a), suporte de versões aprimorado e Garbage Collection (ANJOS, 2009). Já o MSIL é um recurso que age independente da CPU e possibilita gerar na hora da execução um novo compilador que converte para código nativo, isto é, um código específico para o processador da máquina onde a aplicação será executada (UCHÔA, 2006). Por último, a biblioteca de classes é uma reunião de todas as funções normalmente associadas ao sistema operacional, (UCHÔA, 2006) é a base de onde são implementados aplicações, componentes e controles (LOTAR, 2009).

4.4 C# (C SHARP)

Utilizado atualmente por centenas de milhares de programadores, o projeto C# iniciou há cerca de cinco anos (HEJLSBERG, 2004), através da Microsoft, mais especificamente por Anders Hejlsberg (mesmo criador do Turbo Pascal e do Delphi) (CORRÊA, 2005).

Esta linguagem de programação surgiu como uma evolução da linguagem “C” (ANJOS, 2009; HEJLSBERG, 2004), sendo considerada uma linguagem simples, moderna, segura (HEJLSBERG, 2004) e que utiliza o .NET Framework e os novos conceitos de Orientação a Objetos (ANJOS, 2009).

Os desenvolvedores que estão familiarizados com C, C++ ou Java são capazes de iniciar com facilidade a trabalhar de forma produtiva, dentro de um tempo muito curto. Sua sintaxe é simples, fácil de aprender, simplifica muita das complexidades de C++ e fornece recursos poderosos, como tipos de valor nulo, enumerações, representantes, lambda expressões e Direct Memory Access, que não são encontrados no Java. Como toda linguagem orientada a objeto, C# suporta os conceitos de encapsulamento, herança e polimorfismo. Todas as variáveis e métodos, incluindo o método Main, ponto de entrada do aplicativo, são encapsulados em definições de classe. Uma classe pode herdar diretamente de uma classe pai, mas ela pode implementar qualquer número de interfaces. Métodos que substituem métodos virtuais em uma classe pai exigem o override, palavra-chave como uma maneira de evitar a redefinição acidental (MICROSOFT® DEVELOPER NETWORK, 2009a).

No C#, uma estrutura é como uma classe simplificada, ele é um tipo alocado na pilha que pode implementar interfaces mas não tem suporte a herança (MICROSOFT® DEVELOPER NETWORK, 2009a).

Além disso, para esses princípios básicos de orientação a objeto, C# facilita a desenvolver componentes de software por vários inovadores, incluindo o seguinte:

- Método encapsulado que permitem notificações de evento de tipo seguro (MICROSOFT® DEVELOPER NETWORK, 2009a).

- Propriedades, que servem como acessório de variáveis membros privadas (MICROSOFT® DEVELOPER NETWORK, 2009a).
- Atributos, que fornecem metadados declarativos sobre tipos no tempo de execução (MICROSOFT® DEVELOPER NETWORK, 2009a).
- Comentários internos da documentação Extensible Markup Language (XML) (MICROSOFT® DEVELOPER NETWORK, 2009a).
- Language-Integrated Query (LINQ) que fornece recursos de consulta interna (MICROSOFT® DEVELOPER NETWORK, 2009a).

Apresenta um processo chamado “interop” que permite a interação com outros softwares Windows, como objetos Common Object Model (COM) ou Dynamic-link library (DLLs) Win32 nativo. Esta Interoperabilidade permite programas C# a fazer quase tudo que um aplicativo nativo C++ pode fazer. (MICROSOFT® DEVELOPER NETWORK, 2009a).

A compilação é simples comparado ao C e C++, e mais flexível do que em Java. Não há nenhum arquivo cabeçalho separado, e nenhum requisito de que métodos e tipos sejam declarados em uma ordem específica. Um arquivo fonte pode definir qualquer número de classes, estruturas, interfaces, e eventos (MICROSOFT® DEVELOPER NETWORK, 2009a).

O C# permite que uma variável faça referencie não somente há uma classe, mais também a um método dentro de uma classe. Isto é ele cria um tipo chamado de delegado, que quando é usando cria uma instancia do tipo delegado (STELLMAN; GREENE, 2008).

4.5 MICROSOFT SQL SERVER 2005 EXPRESS EDITION

O Microsoft SQL Server 2005 Express Edition é um sistema de gerenciamento de banco de dados (SGBD) que foi criado para ser um protótipo rápido, de fácil implantação e para se integrar perfeitamente com outros investimentos em infra-estrutura de servidor. É um banco de dados poderoso e confiável que oferece recursos robustos, proteção de dados, desempenho para clientes de aplicativos incorporados, aplicativos Web simples e armazenamentos de dados locais (CERQUEIRA, 2008)

O mecanismo de dados do SQL Server 2005 é a parte central dessa solução de gerenciamento de dados empresariais. Além disso, SGBD combina os melhores recursos de análise, geração de relatórios, integração e notificação. Isso permite que sua empresa crie e implante soluções de BI econômicas que ajudem sua equipe a distribuir os dados por todos os cantos da empresa, através de scorecards, painéis, serviços da Web e dispositivos móveis (MICROSOFT®, 2009).

4.6 NHIBERNATE

Imagine uma pessoa fazendo uso de um sistema para controlar suas finanças. Ela passa a fornecer, como dados de entrada, todos os seus gastos mensais para que a aplicação lhe gere gráficos nos quais ela possa avaliar seus gastos. Feito isso, o usuário resolve desligar o computador. Passado alguns dias o usuário teve novos gastos gostaria de realizar novas análises com todos os seus gastos acumulados. Caso a aplicação não armazenar, de alguma forma, os primeiros dados fornecidos, o usuário teria que informá-los novamente e em seguida, acrescentar os novos gastos para fazer a nova análise, causando grande trabalho (CABRAL, 2008).

A solução para esse tipo de problema seria armazenar (persistir) os dados fornecidos a cada vez pelo usuário em um banco de dados relacional, utilizando Structured Query Language (SQL) (CABRAL, 2008).

Foram criados na década de 80, os bancos de dados relacionais (BDR) que substituíram as bases de dados de arquivos. Para se trabalhar com tal base foi criada a linguagem SQL. A linguagem foi toda baseada na lógica relacional dessa forma contava com diversas otimizações. Com isso, o tempo gasto com as operações de persistência foi diminuído, entretanto proporcionando aos desenvolvedores um aumento na produtividade (CABRAL, 2008).

Na década de 90 surgiu um modelo de banco de dados baseando no conceito de orientação a objetos. Tal modelo procurava facilitar, para os desenvolvedores, a implementação da camada de persistência da aplicação, pois eles já estavam familiarizados com o paradigma de orientação a objetos. Esse modelo de dados não foi utilizado em grandes aplicações porque seu tempo de resposta era ineficiente. Ao contrário dos bancos de dados relacionais, eles não tinham um modelo matemático que facilitasse as suas operações. A solução foi criar uma ferramenta que para trabalhar da forma orientada a objeto com os bancos de dados relacionais. Uma dessas ferramentas é o framework NHibernate (CABRAL, 2008b).

O Framework NHibernate é uma versão do Hibernate do J2EE para o .NET (CARVALHO, 2007) que possibilita o mapeamento object/relational (ORM) para o ambiente .NET. (CARVALHO, 2007). O termo ORM refere uma técnica de mapeamento entre representação em modelo de objetos, e representação em relações através de um esquema baseado em SQL (CARVALHO, 2007).

4.7 MICROSOFT VISUAL STUDIO 2008

O Visual Studio 2008 é uma ferramenta de ambiente de desenvolvimento integrado (IDE) desenvolvida pela Microsoft com o intuito de ajudar na criação, documentação, execução e depuração de programas implementados em varias linguagens do .NET (DEITEL; DEITEL; LLISTIFIELD, 2003).

5 SISTEMA PROPOSTO

Como forma de comprovar a usabilidade e eficácia do estudo, uma ferramenta foi desenvolvida visando garantir um maior controle de entradas e saídas de produtos. Tal ferramenta contém cadastros em geral para que o usuário possa armazenar as informações conforme sua necessidade. Para uma maior visualização, o software emite diversos relatórios que informa de uma maneira simples a situação da empresa. A interação com o sistema é através de uma interface gráfica de fácil utilização que lhe proporciona uma maior comodidade e flexibilidade em seu uso.

A motivação para o desenvolvimento desta ferramenta deu-se através de observações quanto ao crescente número de postos de combustíveis e casas de lubrificantes. No entanto, fazer tal controle com papel e caneta levaria muito tempo, e, tempo é dinheiro.

Alguns métodos de desenvolvimento foram adotados otimizando o desenvolvimento resultando em uma maior qualidade dos resultados obtidos, sendo utilizado o modelo de processo de software incremental e unificação de paradigma adotando o padrão de orientação a objetos desde a fase de análise até a fase final de implementação.

O modelo incremental utilizado modularizou o desenvolvimento da ferramenta, dividindo o processo em quatro partes. sendo a primeira a implementação da interface gráfica (View). O segundo processo foi a implementação da classe de comunicação, o controle (Control). O terceiro, a implementação da classe de regra de negócio e ou persistência de dados, o modelo (Model). O último processo foi o de teste para averiguação de bom funcionamento. O teste aplicado no sistema foi o de caixa preta.

A notação OO foi escolhida devido a seu grande desempenho, oferecendo facilidades como reaproveitamento de código, e utilização de padrões de projeto como o *observer*, *strategy*, *singleton* e *mvc* adotados neste projeto para torná-lo mais manutenível.

Durante a fase de análise e modelagem da ferramenta foi utilizada a ferramenta Visual Paradigm for UML 6.3 Enterprise Edition, segue a notação UML (*Unified Modeling Language*), sendo modeladas as funcionalidades do sistema facilitando o processo de implementação das mesmas.

A fase de implementação da ferramenta foi desenvolvida utilizando a linguagem C#. O desenvolvimento utilizou o modelo de programação *desktop*, sendo manipulada através do uso de métodos contidos nas APIs inclusas na plataforma .Net. Para lidar com a manipulação da informação entre sistema e a base de dados, foi utilizado o framework NHibernate. Os dados foram armazenados na base de dados Microsoft SQL Server 2005.

5.1 LEVANTAMENTO DE REQUISITOS

1 Visão Geral.

Desenvolver um sistema que, de forma eficiente, simples e organizada, faça o controle de estoque de empresas do ramo de venda de lubrificantes automotivo.

2 Objetivos.

O sistema tem como objetivo ajudar no controle de estoque de produtos tornando o estabelecimento mais organizado, o que, de certa forma, pode se considerar um diferencial.

3 Requisitos funcionais.

- Cadastro de clientes.
- Cadastro de produtos.
- Cadastro de cidades.
- Cadastro de fornecedores.
- Cadastro de funcionários.
- Cadastro de veículos.
- Cadastro de grupo.
- Cadastro de subgrupo.
- Emissão de relatórios diversos.
- Controle de estoque.
- Venda de produtos.
- Compra de produtos.

5.2 FERRAMENTAS UTILIZADAS

A tabela abaixo descreve as ferramentas necessárias com suas respectivas finalidades para o desenvolvimento deste trabalho, proporcionando concluir os objetivos propostos.

Tabela 1: Ferramentas utilizadas no desenvolvimento do sistema de controle de estoque de lubrificantes automotivos e suas respectivas finalidades

Microsoft Visual Studio	Ferramenta de desenvolvimento de aplicações (Desktop/Web).
SQL Server Management Studio Express	Ferramenta gráfica de gerenciamento para SQL Server 2005 Express Edition.
Visual Paradigm for UML 6.3 Enterprise Edition	Ferramenta de diagramas UML.
Microsoft SQL Server 2005	Ferramenta de gerenciamento de dados.
NHibernate	Ferramenta ORM (Object-Relational Mapping).
Windows XP SP. 2	Sistema operacional base para o desenvolvimento da ferramenta.

5.3 MODELAGENS DA FERRAMENTA

O uso da ferramenta de modelagem representou um processo importante facilitando a implementação do sistema, sendo modelados os diagramas de casos de uso, classes e de seqüência.

5.3.1 DIAGRAMAS DE CLASSES

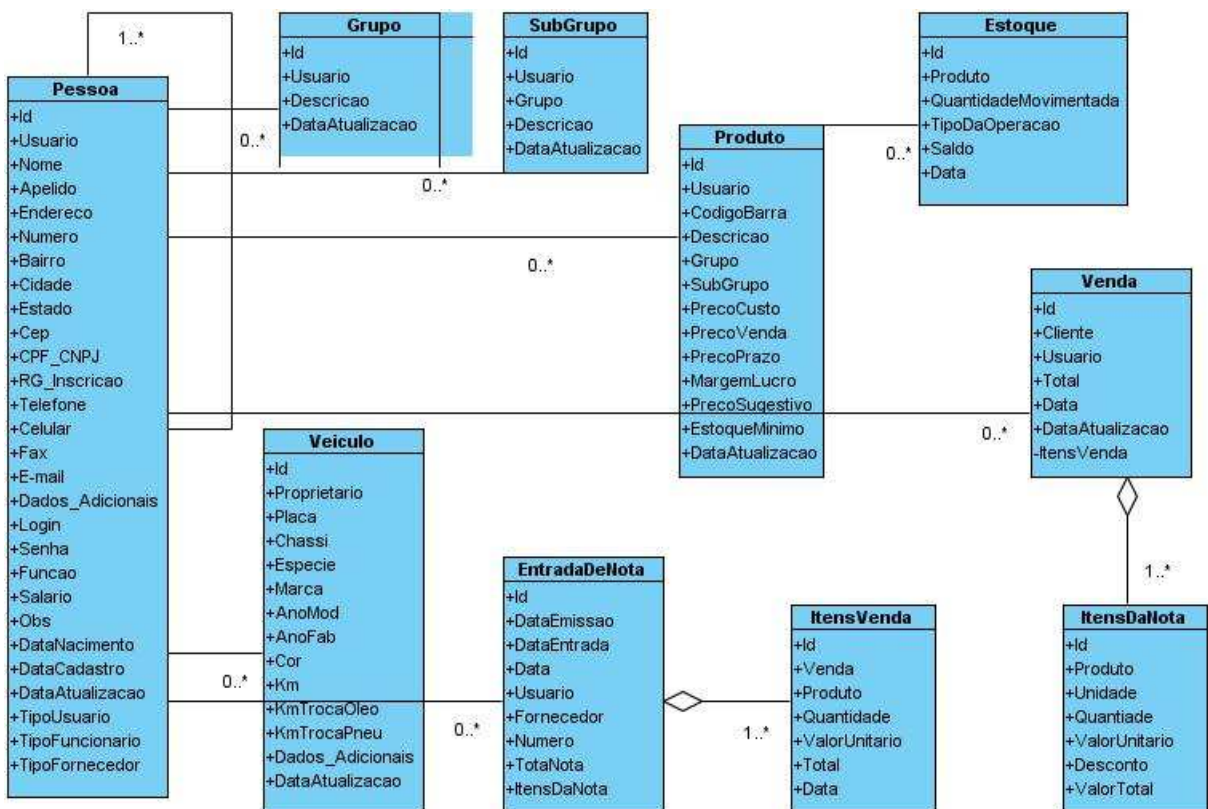


Figura 5: Diagrama de Classes

5.3.2 DIAGRAMA DE CASOS DE USO

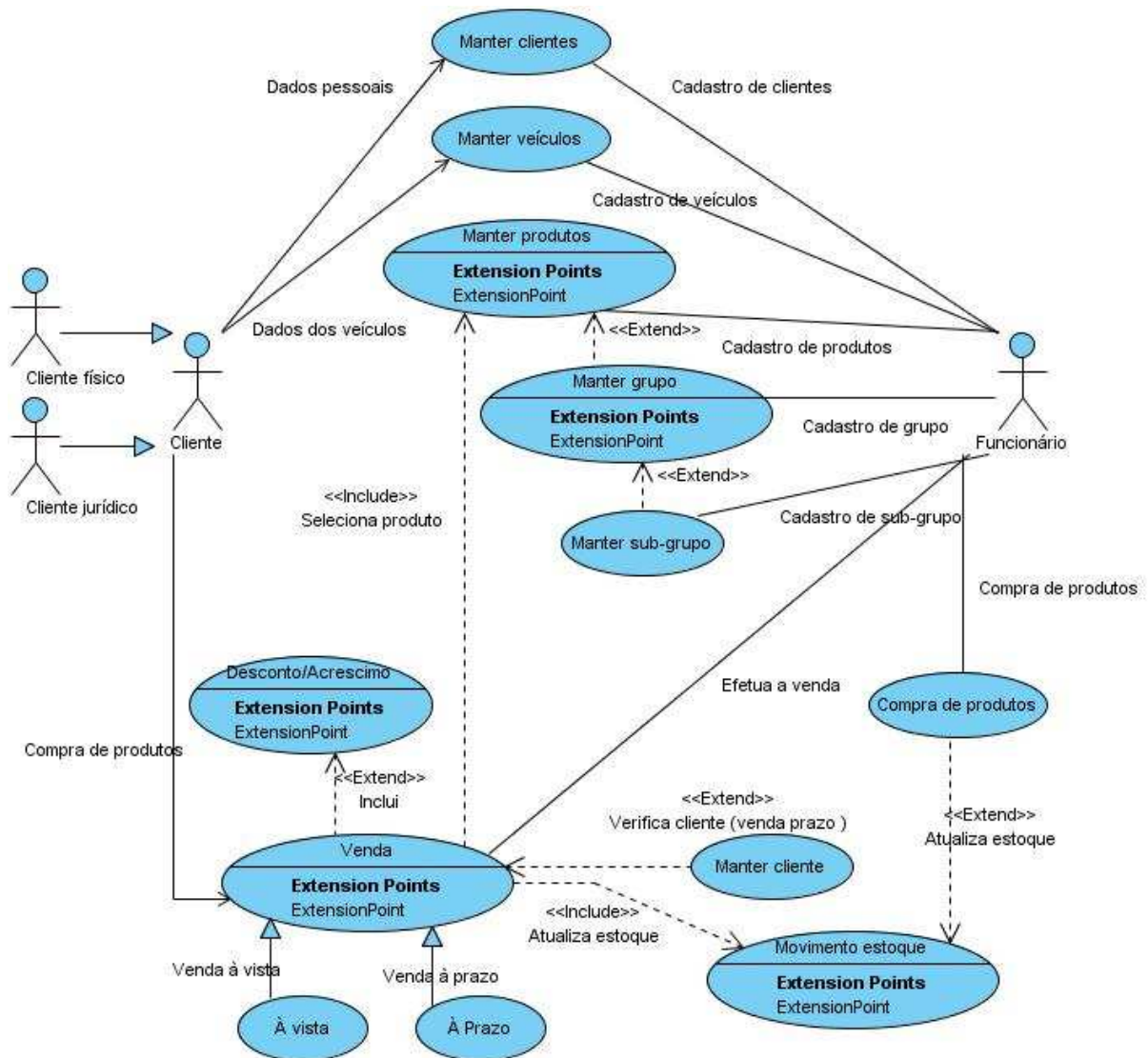


Figura 6: Diagrama de Caso de Uso

5.3.3 DIAGRAMA DE SEQUÊNCIA

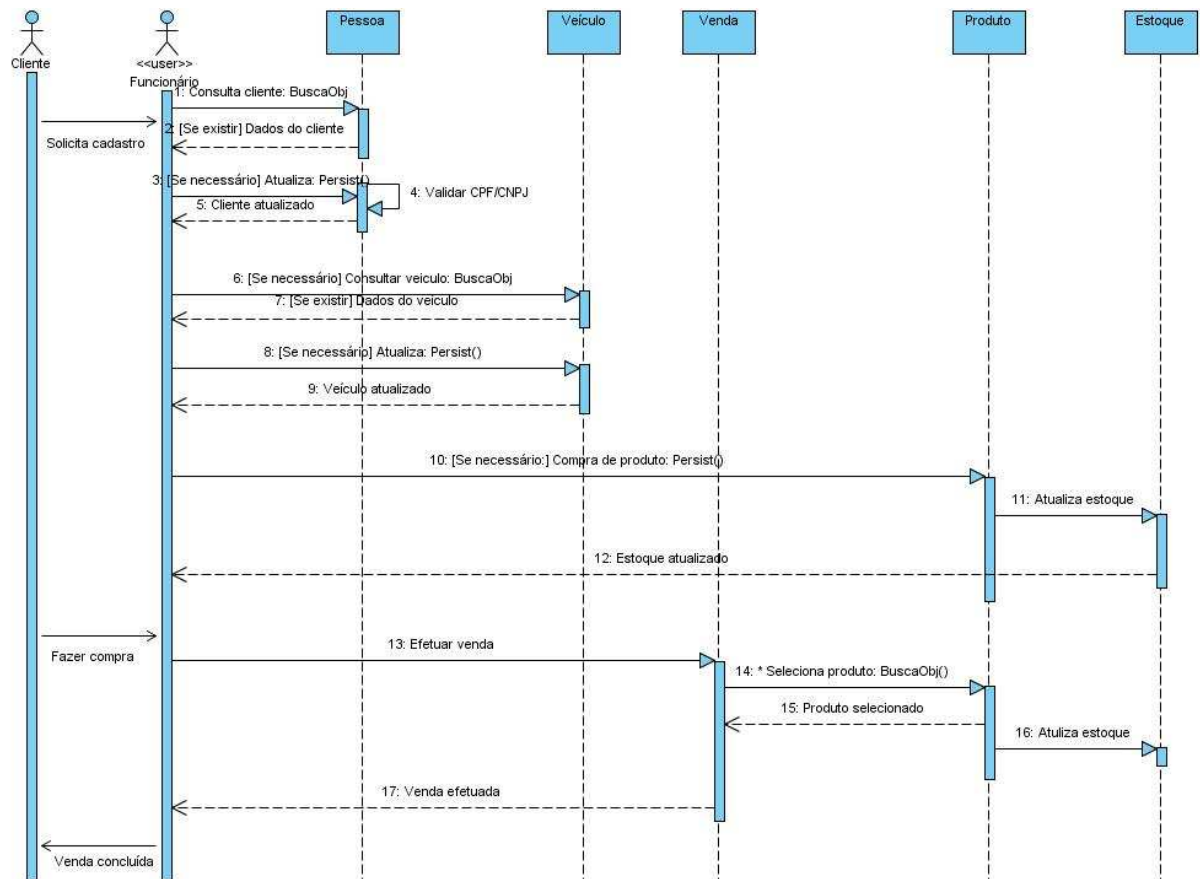


Figura 7: Diagrama de Sequência.

5.3.4 DIAGRAMA DE ENTIDADE RELACIONAMENTO

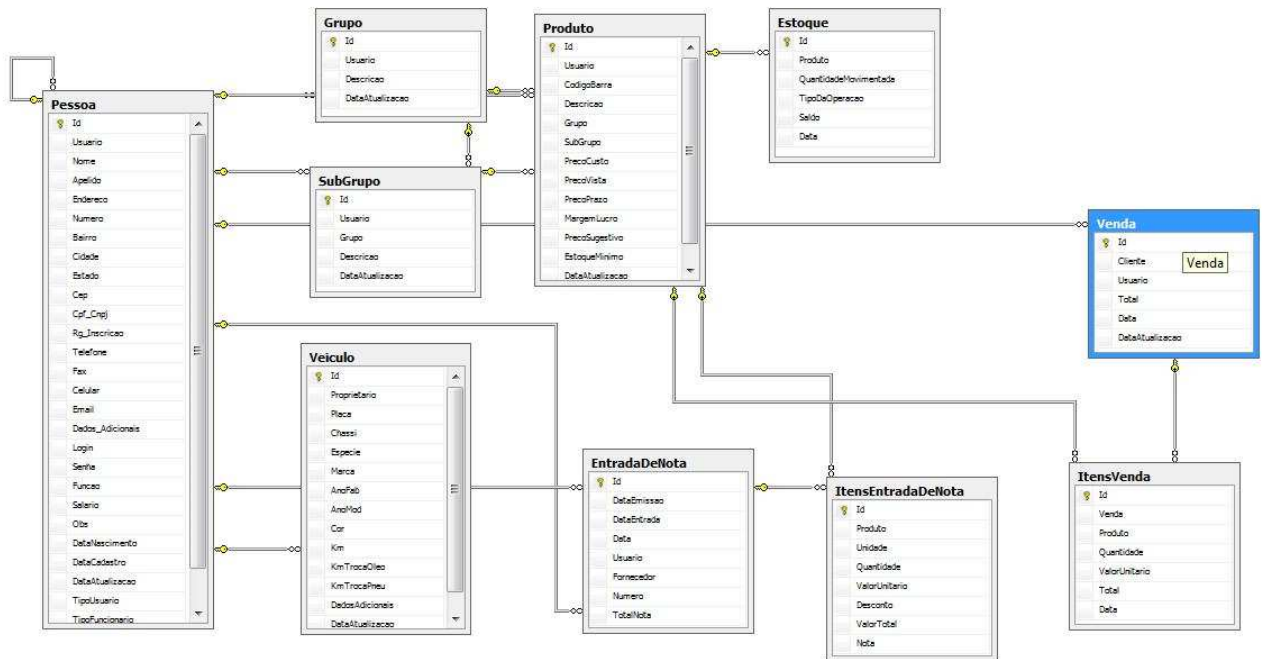


Figura 8: Diagrama de Entidade Relacionamento

5.4 IMPLEMENTAÇÃO DO SISTEMA

Esta seção objetiva justificar o estudo desenvolvido durante a fase de projeto, aplicando as metodologias descritas ao longo deste, de forma a deixar claro o processo de desenvolvimento e as funcionalidades da ferramenta proposta, atingindo o objetivo definido para este trabalho.

A implementação da ferramenta foi modularizada em classes para proporcionar uma melhor organização do trabalho, entendimento e facilitar a manutenção. Durante a finalização de cada processo foram aplicados testes para averiguação de possíveis erros.

Para uma melhor prática de orientação à objeto, foi-se empregado o uso de alguns padrões de projeto. Padrão Observer (figuras 9 e 10), os objetos podem ter dois estados: o estado de observador (Subject) e o estado de observado (Observer). Assim, o observador fica verificando se ocorre alguma alteração nos objetos e caso ocorra ele atualiza todos os observadores. Segue o código fonte abaixo:

```

#region METODOS DA INTERFACE ISUBJECT
public void RegisterObserver(IObserver obj)
{
    this.Observers.Add(obj);
}

public void RemoveObserver(IObserver obj)
{
    int i = this.Observers.IndexOf(obj);
    if (i >= 0)
    {
        this.Observers.Remove(obj);
    }
}

public void NotifyObserver(IList obj)
{
    for (int i = 0; i < this.Observers.Count; i++)
    {
        IObserver Observer = (IObserver)Observers[i];
        Observer.Update(obj, this.OperacaoEfetuada);
    }
}
#endregion

```

Figura 9: Método da interface ISubjetc

```

#region METODOS DA INTERFACE IOBSERVER
public void Update(IList obj, Boolean OperacaoEfetuada)
{
    if (OperacaoEfetuada)
    {
        this.View.GProduto.DataSource = obj;
    }
}
#endregion

```

Figura 10: Método da interface IObject

Outro padrão utilizado foi o Singleton (figura 11), o mesmo faz com que o objeto seja instanciado uma e única vez. Está sendo usado para controlar o usuário logado. Abaixo segue o código fonte.

```
public Pessoa pessoa { get; private set; }
private static Singleton uniqueInstance;

private Singleton(Pessoa pessoa)
{
    this.pessoa = pessoa;
}

public static Singleton getInstance(Pessoa pessoa)
{
    if (uniqueInstance == null)
    {
        uniqueInstance = new Singleton(pessoa);
    }
    return uniqueInstance;
}

public static Pessoa getInstance()
{
    return uniqueInstance.pessoa;
}
```

Figura 11: Classe Singleton

Para persistência de dados é utilizado o framework de persistência NHibernate (figuras 12, 13 e 14). Abaixo segue o código fonte de conexão com o banco de dados e mapeamento das classes.

```
<?xml version="1.0" ?>

<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
  <session-factory>
    <property
name="connection.provider">NHibernate.Connection.DriverConnectionProvider</
property>
    <property name="dialect">NHibernate.Dialect.MsSql2005Dialect</property>
    <property
name="connection.driver_class">NHibernate.Driver.SqlClientDriver</property>
    <property name="connection.connection_string">Data Source=DANILO-
LAPTOP\SQLEXPRESS;Initial Catalog=lubrificantes;Integrated
Security=True</property>
    <property name="connection.isolation">ReadCommitted</property>

    <!--<property name="hbm2ddl.auto">create</property-->
    <property name="show_sql">>true</property>

    <mapping assembly="Lubrificantes" />
  </session-factory>
</hibernate-configuration>
```

Figura 12: Arquivo xml de conexão com o banco de dados


```
public class Pessoa: ICrud
{
    #region METODOS DA CLASSE
    public virtual Int64? Id { get; set; }
    //ID DO USUÁRIO QUE FEZ O CADASTRO/ALTERAÇÃO
    public virtual Pessoa Usuario { get; set; }
    public virtual string Nome { get; set; }
    public virtual string Apelido { get; set; }
    public virtual string Endereco { get; set; }
    public virtual string Numero { get; set; }
    public virtual string Bairro { get; set; }
    public virtual string Cidade { get; set; }
    public virtual string Estado { get; set; }
    public virtual string Cep { get; set; }
    public virtual string Cpf_Cnpj { get; set; }
    public virtual string Rg_Inscricao { get; set; }
    public virtual string Telefone { get; set; }
    public virtual string Fax { get; set; }
    public virtual string Celular { get; set; }
    public virtual string Email { get; set; }
    public virtual string Dados_Adicionais { get; set; }
    //SOMENTE SE FOR USUÁRIO
    public virtual string Login { get; set; }
    public virtual string Senha { get; set; }
    //SOMENTE SE FOR FUNCIONÁRIO
    public virtual string Funcao { get; set; }
    public virtual double? Salario { get; set; }
    //SOMENTE SE FOR FORNECEDOR
    public virtual string Obs { get; set; }
    public virtual DateTime? DataNascimento { get; set; }
    public virtual DateTime? DataCadastro { get; set; }
    public virtual DateTime? DataAtualizacao { get; set; }
    public virtual Boolean? TipoUsuario { get; set; }
    public virtual Boolean? TipoFuncionario { get; set; }
    public virtual Boolean? TipoFornecedor { get; set; }
    #endregion
}
```

Figura 13: Classe Pessoa.

```

<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
assembly="Lubrificantes">
  <class name="Lubrificantes.Classes.Pessoa" dynamic-update="true">

    <id name="Id" column="Id">
      <generator class="native" />
    </id>

    <many-to-one name="Usuario" />
    <property name="Nome" not-null="true" />
    <property name="Apelido" />
    <property name="Endereco" />
    <property name="Numero" />
    <property name="Bairro" />
    <property name="Cidade" />
    <property name="Estado" />
    <property name="Cep" />
    <property name="Cpf_Cnpj" not-null="true" />
    <property name="Rg_Inscricao" />
    <property name="Telefone" />
    <property name="Fax" />
    <property name="Celular" />
    <property name="Email" />
    <property name="Dados_Adicionais" />
    <property name="Login" />
    <property name="Senha" />
    <property name="Funcao" />
    <property name="Salario" />
    <property name="Obs" />
    <property name="DataNascimento" />
    <property name="DataCadastro" />
    <property name="DataAtualizacao" />
    <property name="TipoUsuario" />
    <property name="TipoFuncionario" />
    <property name="TipoFornecedor" />

  </class>
</hibernate-mapping>

```

Figura 14: Arquivo xml de mapeamento da classe pessoa.

Durante a finalização de cada processo foram aplicados testes para averiguação de possíveis erros.

5.5 INTERFACE GRÁFICA

Para proporcionar uma melhor utilização da ferramenta, optou-se por fazer a interação entre esta e o usuário através de uma interface gráfica. A imagem abaixo (Figura 15) refere-se à tela de login.

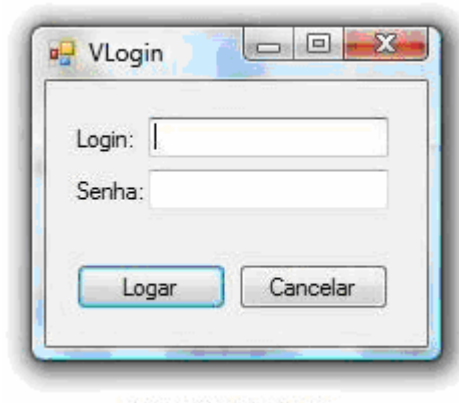


Figura 15: Tela de Login

Após o login efetuado com sucesso, o sistema apresentará a tela principal que apresenta todos os menus de funcionalidades (Figura 16).

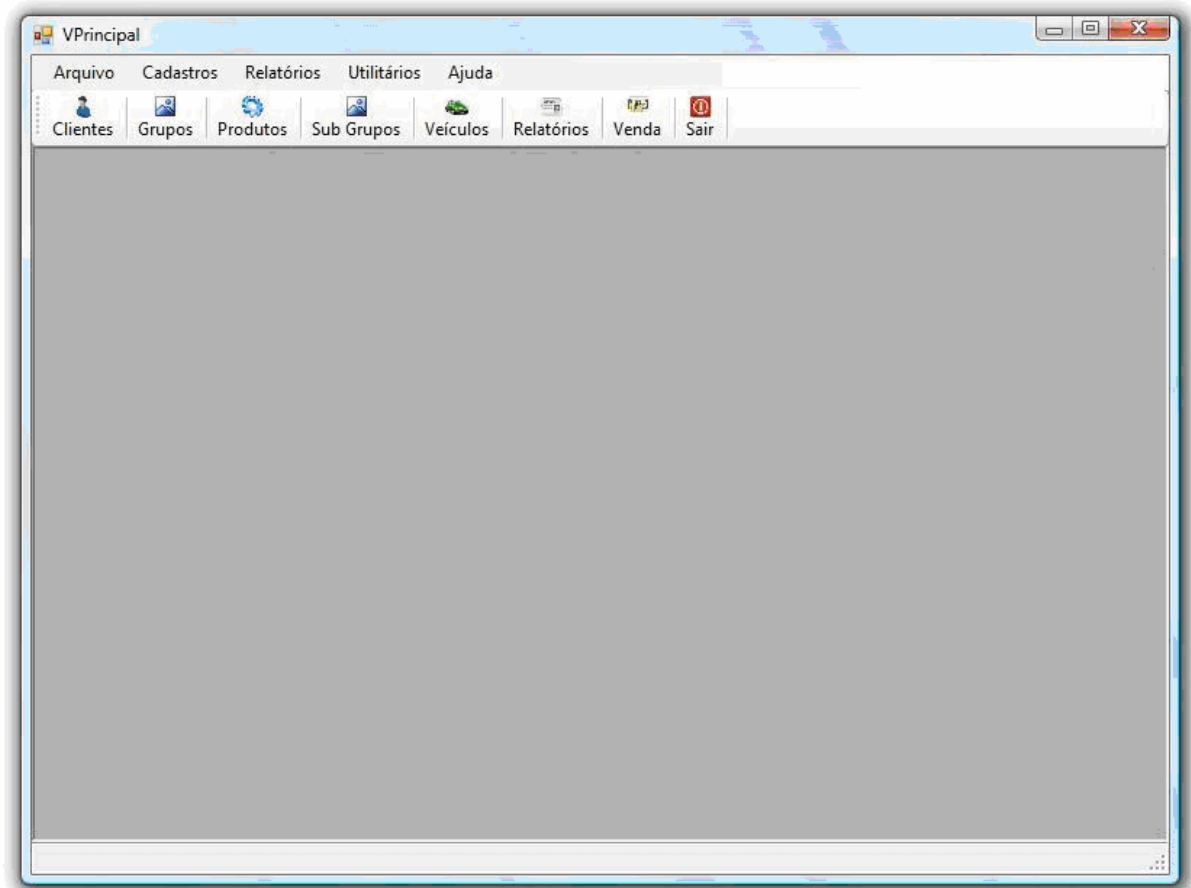


Figura 16: Tela principal

Ao clicar no menu cliente, a tela de dados do cliente será aberta dentro da tela principal (Figura 17). Isso proporciona uma maior organização para com o usuário.

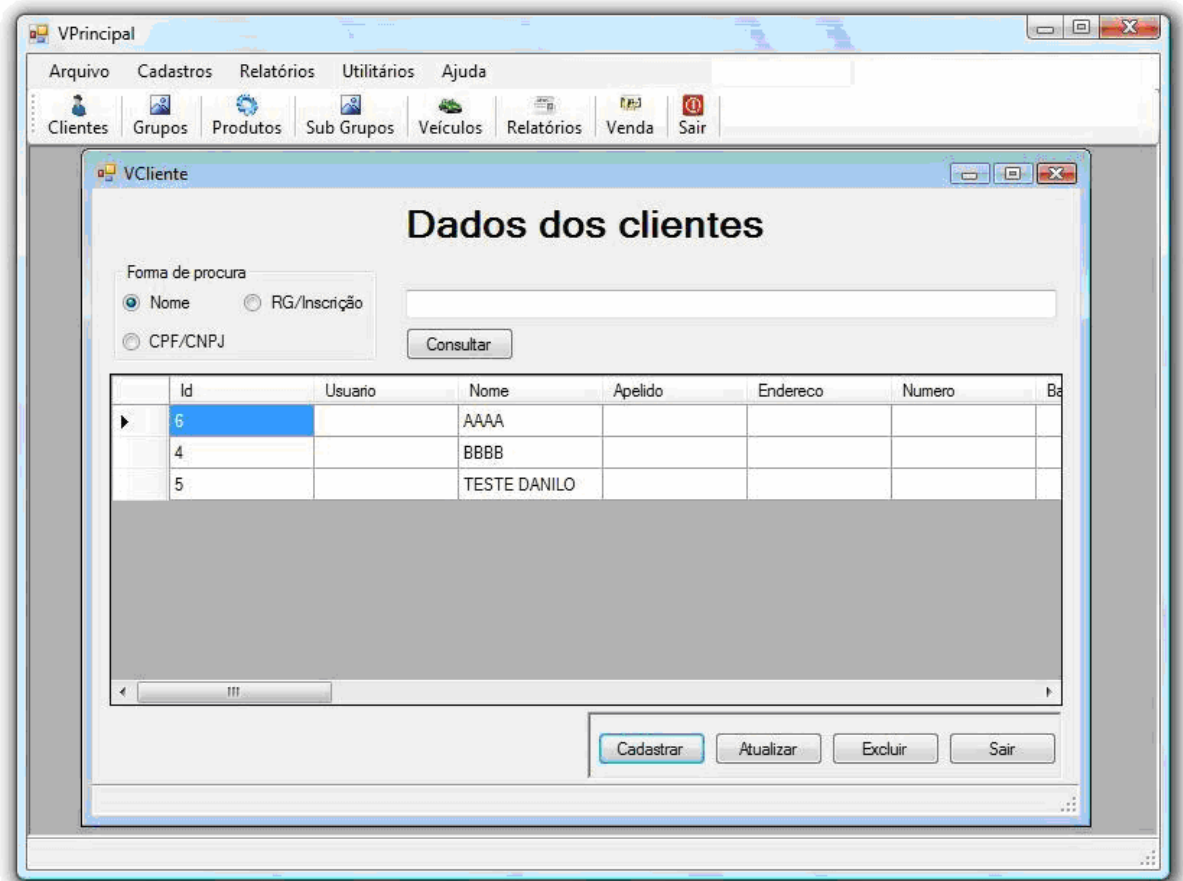
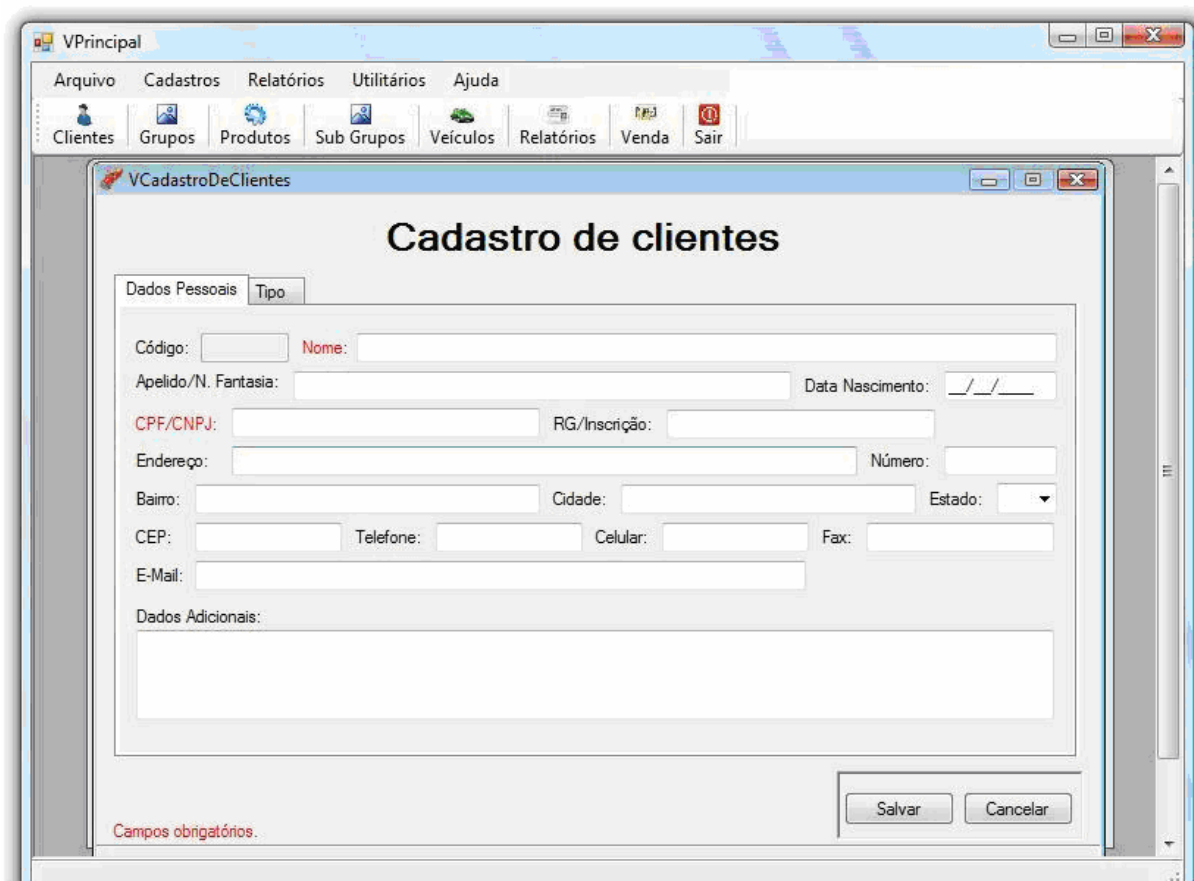


Figura 17: Tela de dados do cliente

Ao clicar em cadastrar, abrirá a tela de cadastro de clientes (Figura 18), onde o usuário entrará com os dados do cliente e o sistema armazenará na base de dados.



The image shows a screenshot of a software application window titled 'VPrincipal'. The main window is titled 'VCadastroDeClientes' and contains a form titled 'Cadastro de clientes'. The form is divided into two tabs: 'Dados Pessoais' (selected) and 'Tipo'. The 'Dados Pessoais' tab contains the following fields:

- Código:
- Nome:
- Apelido/N. Fantasia:
- Data Nascimento:
- CPF/CNPJ:
- RG/Inscrição:
- Endereço:
- Número:
- Bairro:
- Cidade:
- Estado:
- CEP:
- Telefone:
- Celular:
- Fax:
- E-Mail:
- Dados Adicionais:

At the bottom right of the form, there are two buttons: 'Salvar' and 'Cancelar'. A red text label 'Campos obrigatórios.' is located at the bottom left of the form area.

Figura 18: Tela de Cadastro de Clientes

Para alimentar o estoque, tem-se a tela de entrada de notas (Figura 19).

The screenshot shows a software interface for entering invoice data. The main window is titled "VPrincipal" and has a menu bar with "Arquivo", "Cadastros", "Relatórios", "Utilitários", and "Ajuda". Below the menu bar is a toolbar with icons for "Clientes", "Grupos", "Produtos", "Sub Grupos", "Veículos", "Notas", "Relatórios", "Venda", and "Sair". The main content area is titled "VEntradaDeNota" and contains a sub-window titled "VCadastroDeNotasDeEntrada" with the heading "Entrada de Notas". The form includes the following fields and controls:

- Numero: [text box]
- Data Emissão: [date picker]
- Data Entrada: [date picker]
- Fornecedor: [dropdown menu] [Cadastrar button]
- Total da Nota: [text box]
- Dados dos produtos da nota:
 - Inserir produtos: [dropdown menu] [Cadastrar button]
 - Produto: [dropdown menu]
 - Unidade: UN
 - Quantidade: [text box]
 - V. Unitário: [text box]
 - Desconto: [text box]
 - V. Total: [text box]
- Cesta de Compra: [text area] [Adicionar button]
- Salvar [button] Cancelar [button]

Figura 19: Tela de Entrada de Notas.

A tela a seguir corresponde à tela de venda do sistema na qual será registrada toda a venda efetuada no dia (figura 20).

VPrincipal

Arquivo Cadastros Relatórios Utilitários Ajuda

Clientes Grupos Produtos Sub Grupos Veículos Relatórios Venda Sair

Venda

VENDA

Codigo de barra:

Quantidade:

Descrição:

Grupo:

Sub-Grupo:

Preço de custo: R\$

Preço à vista: R\$

Preço à prazo: R\$

Margem de lucro: %

Preço sugestivo: R\$

Estoque mínimo:

Cesta de compras:

Legenda

Figura 20: Tela de Vendas.

6 CONSIDERAÇÕES FINAIS

Após a conclusão do estudo sobre orientação a objeto, padrões de projeto, sintaxe da linguagem de programação C#, framework NHibernate e Microsoft SQL Server Express Edition pôde-se chegar as conclusões abaixo descritas

A utilização das técnicas de orientação objeto nos proporciona um ganho de tempo nas implementações, devido ao conceito de herança, que permite o reaproveitamento de código. Esse código reaproveitado, também influencia na manutenção e ou alteração do sistema, portanto, a alteração feita no código herdado, refletirá em todos os objetos que o herdem.

Unindo as técnicas de orientação à objeto com os padrões de projetos proporciona uma implementação com o código separado, tornando também o código manutenível.

Devido ao fato de estarmos acostumados com o antigo SQL, a curva de aprendizado do NHibernate torna-se um pouco extensa. Depois de aprendido a forma que ele trabalha, a comunicação com a base de dados se torna mais simples e segura.

O modo como as tecnologias foram empregadas, deixou o sistema pronto para receber a implementação de futuros módulos, não interferindo em nada que está implementado.

REFERÊNCIAS

AGÊNCIA NACIONAL DO PETRÓLEO, GÁS NATURA E BIOCOMBUSTÍVEIS (ANP). **Quantidade de postos revendedores de combustíveis automotivos, por bandeira, Segundo Grandes Regiões e Unidades de Federação – 2007.** Disponível em: <http://www.anp.gov.br/conheca/anuario_2008.asp#secao_3>. Acesso em : 11/02/2009

ALEXANDER, C. **The timelles way of buiding.** [S.L.]: Oxorford University Press. 1979.

ANJOS, Thomas. **Curso Básico de C#.** Disponível em : <<http://www.csharpbr.com.br/>>. Acesso em: 02/05/2009

ARAÚJO, A. V. Conceitos Iniciais. In: _____. **Treinamento Avançado em .Net.** [S.L.]: Universo dos livros, 2009. p. 5-22. Disponível em : <http://books.google.com.br/books?id=Wrrw_mJ6Z64C&pg=PA17&dq=ENCAPSULAMENTO&lr=#PPA18,M1>. Acesso em : 17/04/2009

BERG, A.; FIGEIRÓ, J. P. Considerações finais. In:_____. **Lógica de Programação.** 3 ed. [S.L]: Unbra, 1999. p. 163. Disponível em : <<http://books.google.com.br/books?id=UKrDctoNzBMC&pg=PA163&dq=Orienta%C3%A7%C3%A3o+Objeto#PPP1,M1>>. Acesso em: 05/04/2009.

BUSCHMANN, F. et al. **Pattern – oriented Software Architecture:** a system of patterns. New York: John Wiley & Sons Ltd, v.1, 1996.

CABRAL, A. C. **NHibernate – Persistência de Dados em Plataforma .NET.** 2008. Disponível em: <<http://groups.google.com/group/nhibernate-br>>. Acesso em: 01/04/2009

CARVALHO, J. f. b. Questão 3 - Design Patterns E OAP. **Faculdade de Engenharia da Universidade do Porto,** 2007. Disponível em: <http://paginas.fe.up.pt/~ei03067/trabalhos/asso/artigo3.pdf>. Acesso em : 05/05/2009

CARVALHO, L. **Introdução ao NHibernate.** SQL Magazine, 30, paginas , 2007.

CERQUEIRA, A. C. **Sistema de Controle de Alunos em Estágio Curricular. 2008.** Trabalho de conclusão de curso (Bacharel em Engenharia da computação) – Universidade de Pernambuco, Recife, PE, BR.

COOPER, J. W. **The design patterns – java companion.** [S.L.]: Addison-Wesley, 1998.

CORRÊA, M. P. M. C. **Implementação de Padrões de Projeto Utilizando C# .net.** 2005. Trabalho de conclusão de curso (Formação específica em Engenharia do Software) – Centro Universitário UNA, Belo Horizonte, MG, BR.

DEITEL, H. M ., DEITEL, P. J., LLISTIFIELD, J., et al. Introdução ao IED do Visual studio .Net. In:_____. **C# Como Programar**. São Paulo: Pearson Makron Books, 2003. p.27-36.

DENATRAN. **Frota de veículos, por tipo e com placa, segundo as Grandes Regiões e Unidades da Federação**. 2008. Disponível em: <<http://201.24.24.73:8080/renaest/listaNoticiaPublicada.do?op=noticia.publicada.listaEstatistica>>. Acesso em: 11/02/2009.

DENATRAN. **Frota de veículos, por tipo e com placa, segundo as Grandes Regiões e Unidades da Federação**.1998. Disponível em: <<http://201.24.24.73:8080/renaest/listaNoticiaPublicada.do?op=noticia.publicada.listaEstatistica>>. Acesso em: 11/02/2009.

FREEMAN, E. et al. O Padrão Observer. In: _____. **Padrões de Projetos**. 2. Ed. Rio de Janeiro: Alta Books, 2007a, p. 51-80.

FREEMAN, E. et al. Introdução aos Padrões de Projeto. In: _____. **Padrões de Projetos**. 2. Ed. Rio de Janeiro: Alta Books, 2007b, p. 25-50.

FREEMAN, E. et al. O Padrão Singleton. In: _____. **Padrões de Projetos**. 2. Ed. Rio de Janeiro: Alta Books, 2007c, p. 150-166.

FREEMAN, E. et al. Padrões de Padrões. In: _____. **Padrões de Projetos**. 2. Ed. Rio de Janeiro: Alta Books, 2007d, p. 420-438.

GAMMA, E. et al. **Padrões de Projeto: Soluções reutilizáveis do software orientado a objetos**. Porto Alegre: Bookman, 2000.

GARCIA, I. F. **Disciplina de Engenharia do Software: Reusabilidade, Padrões de projeto (Design patterns), Frameworks**, 2009. Disponível em: www.garcia.pro.br. Acesso em: 03/05/2009

GUEDES, G. T. A. Orientação a objetos. In:_____. **UML Uma abordagem prática**. 3.Ed. São Paulo: Novatec, 2007, p. 39 – 45.

HEJLSBERG, Anders; WILTAMUTH, Scott; GOLDE, Peter .Part 1 C# 1.0. In: _____. **The C# Programming Language**. Canadá: Addison-Wesley, 2004. p. 1-43.

IROKAWA, E. S. F.; CUNHA, M. M. R.; CÂMARA, J. J. D. **A Importância e Desenvolvimento do Design de Interiores de Automóveis Relacionada aos Aspectos Sócio-Culturais**, 2007. Disponível em: <http://fido.palermo.edu/servicios_dyc/encuentro2007/02_auuspicios_publicaciones/actas_diseno/articulos_pdf/A4010.pdf >. Acesso em: 09/05/2009

LEA, D. **Christopher Alexander: an introduction for object-oriented designers**. Disponível em : <<http://gee.cs.oswego.edu/dl/ca/ca/ca.html>>. Acesso em: 11/05/2009

LOTAR, A. Introdução ao C# e . Net Framework. In: _____. **Como Programar com Asp.Net e C# - Dicas, Truques, Exemplos, Códigos, Conceitos, Tutoriais.** [S.L.]: Novatec, 2009. p. 24-71.

MAY, D.; TAYLOR, P. Knowledge Management with patterns. **Communications of the ACM**, v. 46, n.7, p. 94-99, 2003.

MICROSOFT ®. **Sql Server. O que é sql Server ?**. Disponível em: <<http://www.microsoft.com/brasil/servidores/sql/prodinfo/overview/what-is-sql-server.aspx>>. Acesso em: 21/01/2009.

MICROSOFT ® DEVELOPER NETWORK. **Introdução à linguagem C# e ao Framework .NET.** Disponível em: <<http://msdn.microsoft.com/pt-br/library/z1zx9t92.aspx>>. Acesso em: 20/01/2009a.

MICROSOFT® DEVELOPER NETWORK. **Visão geral conceitual do .NET Framework.** Disponível em: <<http://msdn.microsoft.com/pt-br/library/zw4w595w.aspx>>. Acesso em: 20/01/2009b.

MIRANDA, M. A. **Introdução ao .NET Framework.** 2008. Disponível em : <<http://www.concatenar.com/index.php?acao=artigo&cdtutorial=90&clique=1>>. Acesso em: 18/02/2009

NUNES, D. A. **HyperDE - um Framework e Ambiente de Desenvolvimento dirigido por Ontologias para Aplicações HiperMídia.** 2005. Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, BR.

PAULA, E. H. **Padrão de Projeto Singleton,** 2006. Disponível em: <http://elton.utfpr.net/Escola/Padr%E3o%20Singleton.pdf>. Acesso em: 21/04/2009

RAMOS, R. A. Introdução. In: _____. **Treinamento Prático em UML.** 1 ed. [SL.]: Universo dos Livros Ltda, 1999. p. 7-16. Disponível em: <<http://books.google.com.br/books?id=kVO6i9Ni1NsC&pg=PA12&dq=Atributos+ou+propriedades&lr=#PPA12,M1>>. Acesso em 13/04/2009.

SALVIANO, C. F. **Introdução à software patterns.** XI SBES, Fortaleza, Ceará- Brasil, p. 22, 1997

STELLMAN, A.; GREENE, J. Use a Cabeça C#. In: _____. **Eventos e Delegados.** Rio de Janeiro: Alta Books, 2009. p. 411-438.

STEINBRUCH, F.. **Alguns Aspectos da História do Automóvel no Brasil.** 1 ed. São Paulo: Conteúdo, 2005, 407 p.

THOMAZ, J. A. **Introdução ao NHibernate.** Persistência de objetos e mapeamento O/R no .NETv. .net magazine, 44, 40-47, 2007

UCHÔA, J. P. **.NET Framework** – Introdução. 2006. Disponível em: <http://www.linhadecodigo.com.br/ARTIGO.ASPX?ID=1149>. Acesso em: 18/02.2008

WIKPEDIA, A ENCICLOPÉDIA LIVRE. **Common Language Runtime**. Disponível em: < http://pt.wikipedia.org/wiki/Common_Language_Runtime>. Acesso em: 20/02/2009a.

WIKPEDIA, A ENCICLOPÉDIA LIVRE. **Microsoft .Net**. Disponível em: < http://pt.wikipedia.org/wiki/Microsoft_.NET>. Acesso em: 20/02/2009b.

WELFER, D. **Padrões de Projeto no Desenvolvimento de Sistemas de Processamento de Imagens. 2005**. Dissertação (Mestrado em Engenharia de Produção) - Universidade Federal de Santa Maria, Santa Maria, RS, BR.