UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

EMANUELE CARLA ROSSONI

PROTÓTIPO DE GERENCIAMENTO DE *PET SHOPS*

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2011

EMANUELE CARLA ROSSONI

PROTÓTIPO DE GERENCIAMENTO DE *PET SHOPS*

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profª. MSc. Rúbia E. O. Schultz Ascari.

PATO BRANCO
2011

A meus pais, meu padrasto, meus irmãos e familiares, amigos e professores.

A todos aqueles que, acreditam e me incentivam a correr atrás dos meus ideais.

ATA Nº: **185**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DA ALUNA **EMANUELE CARLA ROSSONI**.

Às 09:00 hrs do dia 7 de julho de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Rúbia E. de Oliveira Schultz Ascari (Orientadora), Eliane Maria de Bortoli Fávero (Convidada) e Soelaine Rodrigues Ascari (Convidada), para avaliar o Trabalho de Diplomação da aluna Emanuele Carla Rossoni, matrícula 749680, sob o título **Sistema para Gerenciamento de Pet Shops**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Coordenação de Informática. Após a apresentação a candidata foi entrevistada pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 09:45 hrs foi encerrada a sessão.

Profa. Rúbia E. de Oliveira Schultz Ascari, M.Sc.
Orientadora

Profa. Eliane Maria de Bortoli Fávero, M.Sc.
Convidada

Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

AGRADECIMENTOS

Agradeço primeiramente a Deus pelas oportunidades que me foram dadas na vida, principalmente por ele ter colocado pessoas maravilhosas em meu caminho para fazerem parte desta grande caminhada, mas também por ter vivido fases difíceis, que foram muito importantes para crescimento e grande aprendizado.

Não posso deixar de agradecer aos meus familiares, pelo apoio, principalmente da minha mãe Marinês que sempre me incentivou a prosseguir apesar das barreiras.

Aos professores e colegas de trabalho vai um muito obrigado por me ensinarem e enriquecerem meu conhecimento, além do incentivo para chegar até o final do trabalho. Vale destacar a empresa Sponte Informática que me deu a oportunidade de minha primeira experiência na área de TI e a universidade por me fornecerem a base para o meu conhecimento, que foram fundamentais para o desenvolvimento deste projeto.

Um agradecimento em especial à professora Soelaine Rodrigues Ascari pelo apoio, incentivos e ajuda quando eu mais precisei.

Um agradecimento em especial também à professora orientadora, Rúbia Eliza de Oliveira Schultz Ascari, pela sua atenção e paciência, me orientando em cada processo do trabalho, dando dicas e mostrando-se sempre disposta a ajudar.

E aos demais professores pelo conhecimento e dedicação.

A todos, que direta ou indiretamente contribuíram, para realização deste trabalho.

“Sonhar tudo o que quiser sonhar.
Essa é a beleza da mente.
Fazer o que quiser fazer.
Essa é a força da vontade humana.
Testar seus limites com confiança.
Essa é a coragem de alcançar a meta.”
(Bernard Edmonds)

RESUMO

ROSSONI, Emanuele Carla. **Protótipo de Gerenciamento de *Pet Shops***. 2011. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2011.

Nos dias atuais, o computador se tornou uma ferramenta de grande utilidade em todas as áreas comerciais e profissionais. A proposta deste trabalho consiste em analisar, modelar e desenvolver um protótipo de software para gerenciar um *Pet Shop,* em função da necessidade de se ter um controle dos processos realizados em empresas desta área, visando organizar e agilizar sua execução. O protótipo possui telas de cadastros, agenda de serviços e alguns relatórios de controle. Para o desenvolvimento do protótipo foi utilizada a linguagem de programação *Visual Basic.net* e o banco de dados *SQL Server*. O desenvolvimento deste trabalho contribuiu para aperfeiçoar técnicas e conhecimentos adquiridos no curso de graduação.

Palavras-chave: *Pet Shop*. Desenvolvimento de software. Protótipo de software. Sistemas de Informação. Controle de processos. Gerenciamento.

ABSTRACT

Nowadays, the computer has become a very useful tool in all areas of business and professionals. The purpose of this study is to analyze, model and develop a software prototype to manage a Pet Shop, in function of the need to exist a control of the processes carried out in this business area to organize and facilite their implementation. The prototype has forms records, services schedule and some controls reports. For the development of the prototype was used the programming language Visual Basic.net and database SQL Server. The development of this work has helped to improve skills and knowledge acquired in graduation course.

Key-words: Pet Shop. Software development. Software Prototype. Information Systems. Process control. Management.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CAD	*Computer-Aided Design*
CASE	*Computer-Aided Software Engineering*
DER	Diagrama de Entidade e Relacionamento
DFD	Diagrama de Fluxo de Dados
IDE	*Integrated Development Environment*
JUDE	*Java and UML Developers' Environment*
NT	*New Technology*
OMG	*Object Management Group*
OMT	*Object Modeling Technique*
OOSE	*Object-Oriented Software Engineering*
POO	Programação Orientada a Objeto
RF	Requisito Funcional
RNF	Requisito Não-funcional
SIG	Sistema de informação Gerencial
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
TI	Tecnologia de Informação

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, com uma visão geral do trabalho, os seus objetivos e a justificativa, bem como a organização do texto.

1.1 Considerações Iniciais

Pet Shop é o nome dado aos estabelecimentos especializados em comercializar filhotes de animais, alimentos e acessórios. Como o nome já diz – *pet* em inglês significa pequeno animal e *shop* área de compra - é destinado à área de compras para pequenos animais ou animais de estimação. Esses estabelecimentos oferecem também serviços de embelezamento como banho, tosa e perfumaria. Os animais mais comercializados neste ramo são cachorros, gatos e pássaros. Algumas lojas possuem ainda espécimes exóticos como esquilos, furões, lagartos, cobras, tartarugas e chinchilas.

Os *pets shops* surgiram no Brasil na década de 80, introduzidos no país por influência dos norte-americanos, povo altamente interessado no bem estar de seus animais. Com o passar dos anos, o setor de serviços médico veterinários, passou por um grande crescimento, fazendo com que as empresas especializadas neste ramo, cada vez mais busquem formas para agilizar a execução de seus processos, a fim de evitar a ocorrência de problemas durante a oferta de um serviço.

O mercado de produtos veterinários e afins no Brasil está crescendo cada vez mais, indícios que levam a essa conclusão não faltam: além do fenômeno de humanização dos animais de estimação, a cada dia os “bichinhos” ganham maior espaço nos lares brasileiros (BAHIA, 2005).

Atualmente a falta de tempo de seus donos em manter o animal em boas condições de higiene e saúde, faz com essas casas especializadas ganhem mais espaço no mercado, oferecendo serviços e produtos como, banho, tosa, hidratação do pêlo, venda de artigos especializados, entre outros.

Diante do grande crescimento desse ramo, é necessário que sejam criadas formas de agilizar o trabalho do empreendedor que nele atua. Um bom software pode auxiliar nesta questão. Em função disso, viu-se oportuno desenvolver um protótipo de software que terá as funcionalidades necessárias para permitir um

eficiente gerenciamento dos processos realizados em um *Pet Shop* visando disponibilizar uma solução alternativa para empresas que não necessitam de softwares tão completos como os já existentes no mercado, ou que busquem soluções de menor custo.

1.2 Objetivos

O objetivo geral apresenta a finalidade principal da realização do trabalho realizado e os objetivos específicos complementam o resultado obtido com o objetivo geral.

1.2.1 Objetivo Geral

Realizar a análise, modelagem e desenvolvimento de um protótipo de software para facilitar e melhor organizar processos executados em *Pet Shops*.

1.2.2 Objetivos Específicos

Para atender o objetivo geral deste trabalho, pretende-se:

- Realizar um levantamento de requisitos com empresários que atuem no ramo de *Pet Shops*, buscando informações sobre a atual situação do setor;
- Criar uma análise de dados identificando requisitos funcionais e não funcionais necessários ao desenvolvimento de um software para gerenciamento de *pet shops*;
- Criar a modelagem do banco de dados por meio de um Diagrama de Entidade e Relacionamento;
- Criar a modelagem estática e dinâmica do software utilizando diagramas da UML (*Unified Modeling Language* - Linguagem de Modelagem Unificada);
- Desenvolver um protótipo de sistema para gerenciamento de *Pet Shops* utilizando a ferramenta de desenvolvimento Visual Basic.net (VB .NET) e banco de dados SQL Server 2005.

1.3 Justificativa

Devido ao crescimento da demanda por lojas especializadas em oferecer produtos e serviços para animais domésticos de pequeno e médio porte, os P*et Shops* tem que se adaptar às mudanças que ocorrem continuamente no mercado. Acontecimentos sociais, ambientais e políticos, fazem com que os consumidores reajam de forma diferente em relação ao mercado.

Muitos *Pet Shops* modernos têm como lema crescer orientados e dirigidos para o consumidor. Prezando pela agilidade ao passar uma informação ao seu cliente, torna-se uma necessidade para as lojas deste ramo, a utilização de um sistema informatizado para gerenciar e apresentar estas informações. Mediante a necessidade de controlar os processos realizados pelas empresas em seu ambiente, viu-se oportuno o desenvolvimento de um protótipo de sistema para ajudar no gerenciamento das empresas de *pet shop*, visando agilizar e melhor organizar seus processos.

Este trabalho tem como objetivo principal desenvolver um protótipo de software que automatize o processo de gerenciamento de um *Pet Shop*, fundamentado nos resultados obtidos com o estudo dirigido realizado durante o trabalho de estágio. O desenvolvimento deste protótipo de sistema permitirá colocar em prática os conhecimentos adquiridos na graduação, com relação a Análise de Dados, Documentação de Sistemas, Desenvolvimento de Software e Manipulação de Banco de Dados.

1.4 Estrutura do Trabalho

Este trabalho está organizado em capítulos, dos quais este é o primeiro e apresenta as considerações iniciais sobre o trabalho, incluindo os objetivos e a justificativa.

No Capítulo 2 é apresentado o referencial teórico que compreende conceitos relacionados ao gerenciamento de *Pet Shop*, levantamento de requisitos, análise e projeto de sistemas orientados a objetos e análise estruturada e essencial.

No Capítulo 3 são apresentados os materiais utilizados para modelar e

implementar o sistema, incluindo as tecnologias, as ferramentas e os ambientes de desenvolvimento utilizados, e o método, que é a sequência geral de passos empregados para realizar o trabalho.

O Capítulo 4 contém o resultado do trabalho que é a descrição, modelagem e apresentação do protótipo de software desenvolvido para gerenciamento de Pet Shops, incluindo exemplificação de partes do código-fonte, utilizando a linguagem de programação adotada para desenvolvimento.

No Capítulo 5 está a conclusão do trabalho com as considerações finais, as dificuldades encontradas, vantagens e desvantagens do software, bem como suas soluções.

2. REFERENCIAL TEÓRICO

O desenvolvimento deste trabalho foi dividido em duas partes principais. A primeira tratou de realizar a fundamentação teórica e gerar a modelagem necessária para desenvolvimento do protótipo. Para isso, foi realizado estudo teórico, pesquisa de campo com empresas do ramo de *Pet Shops* e entrevistas com empreendedores do ramo. A realização desta etapa do trabalho permitiu analisar as vantagens em utilizar um *software* para controlar este tipo de negócio. A segunda parte correspondeu ao desenvolvimento de um protótipo para gerenciamento de *Pet Shop* utilizando práticas de codificação orientada a objetos, que visa uma reutilização de código e em consequência maior desempenho.

Esse capítulo explicará o motivo da utilização de cada prática empregada nas duas partes citadas, para posteriormente nos demais capítulos descrever a aplicação destas.

2.1 Gerenciamento de *Pet Shop*

Anos atrás, os cuidados com animais eram de responsabilidade de veterinários ou clínicas especializadas. Com o passar do tempo, surgiram os *Pet Shops*, empresas especializadas no tratamento de animais domésticos, que estão cada vez mais ocupando espaço no mercado.

Devido à grande procura dos serviços e produtos oferecidos pelo ramo de *Pet Shops,* o volume de trabalho aumentou significativamente e os empreendedores têm grande dificuldade para gerenciar a execução de seus processos. Em função disso, torna-se necessária a informatização dessas empresas, para melhor organizar suas atividades fundamentais, como por exemplo, manter sempre o seu estoque em dia. Muitas vezes o gerente tem capacidade, porém o fluxo de informação é tão grande que há a necessidade de um auxílio para controlar todos os processos. Como alternativa essas empresas podem contar com sistemas informatizados, como os denominados sistemas de informação.

De acordo com Batista (2004, p. 19), sistema de informação “é todo e qualquer sistema que possui dados ou informações de entrada que tenham por fim gerar informações de saída para suprir determinadas necessidades”.

Para controles de processos, as empresas têm como aliados os sistemas de informações gerenciais (SIG) oferecidos pela tecnologia da informação, proporcionando a geração de informações rápidas e precisas, facilitando a tomada de decisão e o registro de operações realizadas.

Segundo Cruz (1998, p. 20), a tecnologia da informação é todo e qualquer dispositivo que tenha capacidade para tratar dados e ou informações, tanto de forma sistêmica como esporádica, quer esteja aplicada ao produto, quer esteja aplicada no processo.

Devido à grande quantidade de informações que o empreendedor tem que gerenciar, surge a importância de informatizar os processos, utilizando um sistema de informação gerencial (SIG).

Para Batista (2004), sistema de informação gerencial:

> "É o conjunto de tecnologias que disponibilizam os meios necessários à operação do processamento dos dados disponíveis. É um sistema voltado para a coleta, armazenagem, recuperação e processamento de informações usadas ou desejadas por um ou mais executivos no desempenho de suas atividades. È o processo de transformação de dados em informações que são utilizadas na estrutura decisória da empresa proporcionam a sustentação administrativa para otimizar os resultados esperados" (BATISTA, 2004, p. 22).

Com a estrutura organizacional traçada, e a empresa tendo conhecimento sobre como utilizar os recursos oferecidos pela TI (tecnologia de informação), o sistema de informações gerenciais só agrega benefícios à gestão empresarial e a empresa estará melhor preparada para atender adequadamente seus clientes, e de maneira organizada possuir um controle interno eficiente e com menos falhas.

A finalidade ao desenvolver um protótipo para gerenciamento de um *pet shop* é a de manter o fluxo de informações da empresa, permitindo maior rapidez e disponibilidade dessas informações. Além disso, considera-se importante que as empresas de pequeno porte possam optar por sistemas menores e de fácil utilização, que facilitem o gerenciamento dos seus principais processos, e esse é o objetivo principal do protótipo proposto.

2.2 Levantamento de Requisitos

De acordo com Tonsig (2000, pg. 19) “A análise de sistemas consiste nos métodos e técnicas de investigação e especificação da solução de problemas, a partir dos requisitos levantados, para criação e implementação de software em algum meio que o suporte”.

O processo de desenvolvimento de software divide-se em quatro grandes fases: análise, projeto, implementação e testes. Na fase de análise do projeto do software será enfatizada a investigação do problema, para melhor compreensão do mesmo. Antes do início do desenvolvimento é necessário conhecer o que se deseja construir, ou seja, ter em mente as funções que o software irá realizar as restrições, a política de funcionamento da empresa onde o software será implantado e as interligações que existem no sistema.

O processo de desenvolvimento aplicado neste trabalho, baseado na obra do autor Raul Sidnei Wazlawick (WAZLAWICK, 2004), é o processo unificado de desenvolvimento de software, do qual serão abordadas as fases de concepção e elaboração.

> “A fase de concepção deve ser a primeira fase do processo de desenvolvimento de software, na qual se procura levantar os principais requisitos e compreender o sistema de forma abrangente. A elaboração é constituída da análise e projeto.” (WAZLAWICK, 2004, p. 23).

A seguir autor descreve como é o processo da fase de concepção:

> “a fase de concepção consiste em uma etapa na qual o analista vai buscar as primeiras informações sobre o sistema a ser desenvolvido. Nessa etapa assume-se pouco conhecimento do analista sobre o sistema e uma grande interação com o usuário e cliente” (WAZLAWICK, 2004, p. 32).

O levantamento de requisitos, organização dos requisitos e planejamento do desenvolvimento são as atividades que fazem parte da fase de concepção.

Ainda segundo Wazlawick (2004) a análise de requisitos é fundamental para o desenvolvimento de sistemas, pois trata justamente de descobrir o que o cliente quer com o sistema. Os requisitos podem ser:

- Funcionais – requisitos funcionais correspondem à listagem de todas as coisas que o sistema deve fazer. Como por exemplo: controlar entrada e saída, calcular multa, gerar relatórios.

- Não-funcionais – requisitos não-funcionais são restrições que se coloca sobre como o sistema deve realizar seus requisitos funcionais. Como por exemplo: De que forma? Quando, como, por quem?

A fase de levantamento de requisitos é considerada a primeira atividade técnica no desenvolvimento do software, é responsável por definir que serviços o sistema irá realizar, e diante de quais restrições fará isso. Os requisitos devem estabelecer o que o sistema fará e não como isto será feito. Nesta fase são produzidos alguns eventos como:

- Visão geral do sistema, que é a descrição das principais ideias do sistema.
- A definição dos requisitos funcionais e não-funcionais que indicam o que o sistema deve realizar e diante de que condições.

A UML *Unified Modeling Language* (Linguagem de Modelagem Unificada), e a modelagem auxiliam no processo de levantamento de requisitos. Processo este que não pode ser considerado uma tarefa fácil, pois o usuário e o analista muitas vezes não conseguem se entender, frequentemente o usuário tem vários problemas e não consegue descrevê-los ou o analista não consegue compreendê-los com clareza. Com isso é necessário gerar uma boa documentação do sistema, principalmente em casos de sistemas complexos, onde há muitos envolvidos no processo.

Para que o processo de documentação tenha sucesso é necessário que o analista colha o máximo de informações do usuário, e utilize ferramentas para auxiliar no projeto documentando o máximo possível para que nenhuma informação seja perdida.

Após, realizado o levantamento de todas as informações necessárias para o desenvolvimento do projeto, deve-se realizar um estudo do que foi levantado reunindo todos os envolvidos no projeto. Desta forma pode-se sanar dúvidas e realizar apresentações dos modelos ao usuário.

2.3 Análise e Projeto de Sistemas Orientados a Objetos

Os caminhos do desenvolvimento de software têm mudado significativamente desde a invenção do computador. A razão principal para a ocorrência dessas mudanças foi o aumento da complexidade dos programas, e consequente

dificuldade em dar suporte aos mesmos, sendo necessário buscar novas formas de desenvolvimento.

> "A análise de sistemas consiste nos métodos e técnicas de investigação e especificação da solução de problemas, a partir dos requisitos levantados, para criação e implementação de software em algum meio que o suporte" (TONSIG, 2000, pg.19).

Segundo Martin (1995, pg. 10) "A análise e projeto orientado a objetos tentam conseguir uma maciça reusabilidade de classes de objeto".

De maneira mais específica, Pressman (1995) explica que os métodos utilizados para fazer o levantamento dos requisitos, para o desenvolvimento do software orientado a objetos, permitem que o desenvolvedor modele um problema através de classes, objetos, atributos e operações.

Por meio da análise orientada a objetos, modela-se a realidade para os tipos de objetos e aquilo que ocorre a eles, levando a projetar e programar sistemas de forma orientada a objetos. Sua utilização oferece vários benefícios: os modelos gerados por meio da análise refletem o mundo real de maneira aproximada, proporcionando facilidade no entendimento: a realização de mudanças nos requisitos se torna mais fácil: permite a reutilização de código: a manutenção do protótipo se torna mais fácil e o projeto possui qualidade mais elevada, entre outros.

A seguir são descritos os principais conceitos relacionados à orientação a objetos.

2.3.1 Objetos

A seguir definição de objetos:

> "Cada conceito é uma idéia ou um entendimento pessoal que temos de nosso mundo. Os conceitos que adquirimos nos permitem dar sentido e raciocinar sobre as coisas de nosso mundo. Essas coisas às quais nossos conceitos se aplicam são denominados objetos" (MARTIN, 1995, pg.18).

O objeto é uma instância de uma classe, é uma variável do tipo de dados definida pela classe. Na análise e projeto orientado a objetos, objeto tem estado,

comportamento e identidade única. Ele possui atributos e métodos que atuam sobre ele.

O software é um conjunto de objetos que se comunicam por meio de mensagens informando uns aos outros, o que deve ser feito, ou seja, são chamadas as funções (métodos) que pertencem a um objeto em particular.

2.3.2 Atributos

Os atributos são as características do objeto. São a menor unidade que em si possui significância própria e inter-relacionada com o conceito lógico da classe à qual pertence (equivale aos campos de um registro). É uma propriedade nomeada de um tipo. Em síntese, armazena valores em células.

O conjunto de seus atributos, associados a seus valores correntes, definem o estado de um objeto.

2.3.3 Métodos

Às formas como os dados de um objeto são manipulados, chama-se de método. São declarados dentro de uma classe e representam as operações que os objetos pertencentes a esta classe podem executar. As operações executadas pelo objeto são rotinas de envio de mensagens que serão executadas por um objeto. Os métodos de um tipo de objeto fazem referencia somente as estruturas de dados desse tipo de objeto.

O comportamento de um objeto é dado pelo conjunto de serviços ou métodos que outros objetos, ditos clientes, podem requisitar.

2.3.4 Encapsulamento

É denominado encapsulamento o empacotamento ao mesmo tempo dos dados e métodos. Forma usada para ocultar dados de outros objetos, permitindo

que os dados só possam ser acessados por meio dos seus próprios métodos, protegendo assim os dados do objeto contra adulterações.

Os usuários compreendem quais as operações do objeto que podem ser solicitadas, mas não conhecem os detalhes de como as operações são executadas. O encapsulamento é importante porque separa a maneira como o objeto se comporta da maneira como ele é implementado. Ele restringe a visibilidade do objeto, mas facilita o reuso.

2.3.5 Mensagens

Para que um objeto execute alguma coisa, é necessário encaminhar uma solicitação. Esta solicitação que faz com que a operação seja executada.

De acordo com Martin (1995, p. 22), “Uma solicitação (*request*) pede que uma operação especificada seja invocada, usando um ou mais objetos como parâmetros”. Os objetos interagem por meio de mensagens, estas definem o nome do serviço requisitado, a informação necessária para execução do serviço e o nome do solicitante.

2.3.6 Classes

Uma classe descreve um conjunto de objetos semelhantes por meio de seus atributos e métodos. É o termo técnico utilizado em linguagens orientadas a objeto com uma estrutura modular completa que descreve as propriedades estáticas e dinâmicas dos elementos manipulados pelo programa.

Computacionalmente, pode-se dizer que uma classe é um tipo de dados definido pelo usuário, contendo alguns dados internos e alguns métodos, na forma de procedimentos ou funções.

Segundo Martin (1995, p.26) “uma classe é uma implementação de um tipo de objeto. Ela especifica uma estrutura de dados e os métodos operacionais permissíveis que se aplicam a cada um de seus objetos”. Para desenvolver um sistema orientado a objetos é essencial a utilização de classes.

2.3.7 Herança

A herança é um tipo de relacionamento entre as classes que permite a criação de uma hierarquia entre elas, possibilitando o compartilhamento de atributos e operações. De acordo com Martin (1995, p.26), “uma classe implementa o tipo de objeto. Uma subclasse herda as propriedades de sua classe-mãe; uma subsubclasse herda as propriedades das subclasses e assim por diante”.

Vale ressaltar que a herança é uma das responsáveis pela facilidade de reaproveitamento de código da POO (Programação Orientada a Objeto).

2.3.8 Polimorfismo

“O que possui várias formas”. Propriedade de uma ou mais classes responderem a mesma mensagem, cada uma de uma forma diferente. Utilizando polimorfismo, uma mensagem pode ser executada de acordo com as características do objeto que está recebendo o pedido de execução do serviço.

> “é o principio em que classes derivadas de uma mesma superclasse podem invocar operações que tem a mesma assinatura, mas comportamentos diferentes em cada subclasse, produzindo resultados diferentes, dependendo de como cada objeto implementa a operação. Em outras palavras é a capacidade de objetos diferentes possuírem operações com o mesmo nome e a mesma lista de argumentos, mas que executam tarefas de formas diferentes” (LIMA, 2005, p.26).

Existe polimorfismo de sobrecarga (parâmetros diferentes), e polimorfismo de sobrescrita (sobrescreve método herdado).

2.4 UML - *Unified Modeling Language* (Linguagem de Modelagem Unificada)

> "No período que precedeu o surgimento da UML, foram propostas dezenas de metodologias voltadas ao desenvolvimento de especificações de análise de projeto orientadas a objetos. Com vista a tornar a UML um padrão internacional a empresa Rational submeteu a versão 1.0 da UML ao OMG *(Object Management Group)* e em novembro de 1997 ela passou a ser o padrão de notação da entidade (SILVA, 2007, p.81)".

A *Unified Modeling Language* (UML) é uma linguagem para visualização, documentação, especificação e desenvolvimento de sistemas orientados a objetos. É considerada a linguagem mais expressiva para modelagem de sistemas orientados a objetos, pois sintetiza os principais métodos existentes.

> "A UML não é uma metodologia, mas sim uma linguagem de modelagem gráfica para descrever um projeto de software. A UML surgiu da união dos métodos Booch e OMT em 1994, quando seus autores (Grady Booch e James Rumbaugh) juntaram forças através da *Rational Corporation* e unificaram completamente seus trabalhos. (BOOCH, 2000)".

Foi nomeada inicialmente como OOSE (*Object-Oriented Software Engineering*), sendo posteriormente chamada de UML.

Por meio dos diagramas da UML é possível representar o sistema sob diversas perspectivas de visualização. A comunicação entre os envolvidos no processo de desenvolvimento do sistema pode ser facilitada com a UML, pois apresenta um vocabulário de fácil entendimento, sendo que um diagrama pode ser representado de várias formas, dependendo de quem irá interpretá-lo.

2.4.1 Diagramas da UML

Segundo Booch e Jacobson (2000, p. 89), "um diagrama é uma representação gráfica de um conjunto de elementos, geralmente representados como gráfico conectado de vértices (itens) e arcos (relacionamentos)".

A UML oferece vários diagramas, assim é importante escolher os diagramas mais adequados ao sistema para visualizá-lo sob diferentes perspectivas. No entanto, um sistema dificilmente pode ser entendido em sua totalidade a partir de uma única perspectiva.

Por meio de bons diagramas há maior facilidade para compreensão do sistema que está sendo desenvolvido, auxiliando na identificação de incompatibilidades, ajudando na tomada de decisões.

A Figura 1 apresenta a hierarquia dos diagramas UML, os quais são divididos em duas categorias: estrutural e estática. Partes estáticas de um sistema são visualizadas por meio dos diagramas de classes, objetos, componentes e implantação. Nesses diagramas é possível observar a representação da parte estável do sistema como classes, interfaces, componentes, nós, entre outros. Já as partes dinâmicas, ou comportamentais, podem ser representadas pelos diagramas de caso de uso, interação (sequência e colaboração), estados e atividades. Esses diagramas têm por função representar as partes do sistema que sofrem alterações, como por exemplo, o fluxo de mensagens ou a movimentação física de uma rede.

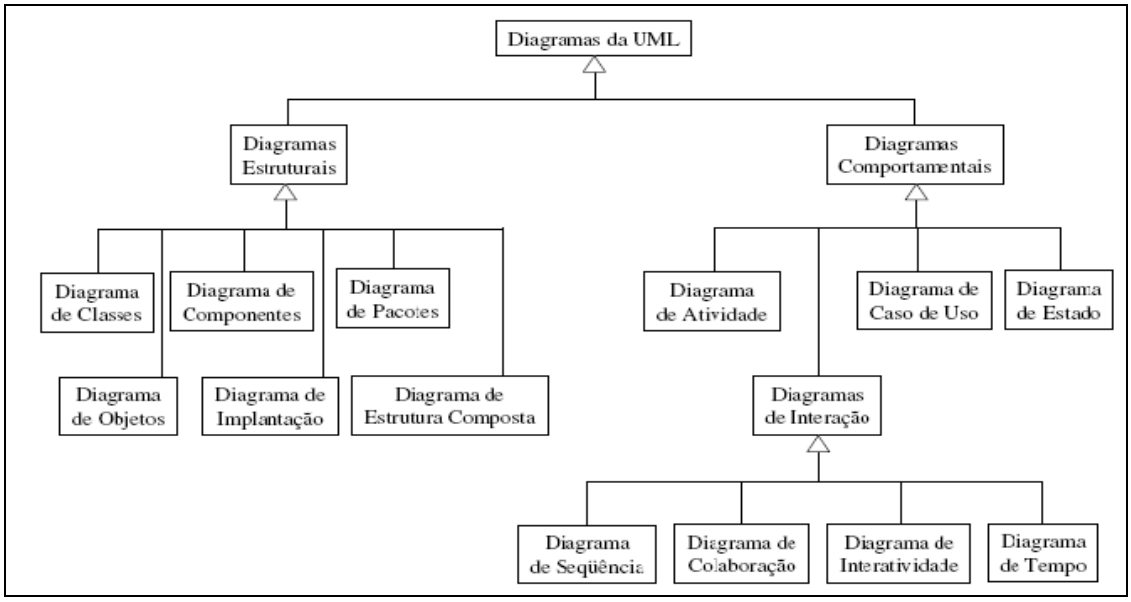

Figura 1 - Hierarquia dos Diagramas UML

Fonte: MAIA (2009)

A modelagem do projeto em questão será composta por diagramas de classes, de casos de uso e de sequência (interação), e em função, disso apenas estes diagramas da UML serão descritos.

2.4.1.1 Diagrama de Classes

Segundo Pender (2004, p. 42), “O diagrama de Classes (*Class*) está no núcleo do processo de modelagem de objetos. Ele modela as definições de recursos essenciais à operação correta do sistema”.

Um diagrama de classes é a representação da estrutura e das relações das classes que servem como modelo para os objetos, mostra o conjunto de classes, interfaces e colaborações que compõe um sistema, sua estrutura com informações sobre métodos, atributos, nome das funções e como serão relacionadas. As classes podem se relacionar umas com as outras de diversas maneiras, como por exemplo, por meio de associação (conectadas entre si), dependência (uma classe depende de outra), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes que são agrupadas por possuírem características similares).

A Figura 2 apresenta um exemplo de diagrama de classes com os relacionamentos das classes do sistema. Ou seja, um paciente possui um agendamento em um determinado horário para realizar uma consulta e um serviço. Os serviços oferecidos são limpeza/restauração ou exame.

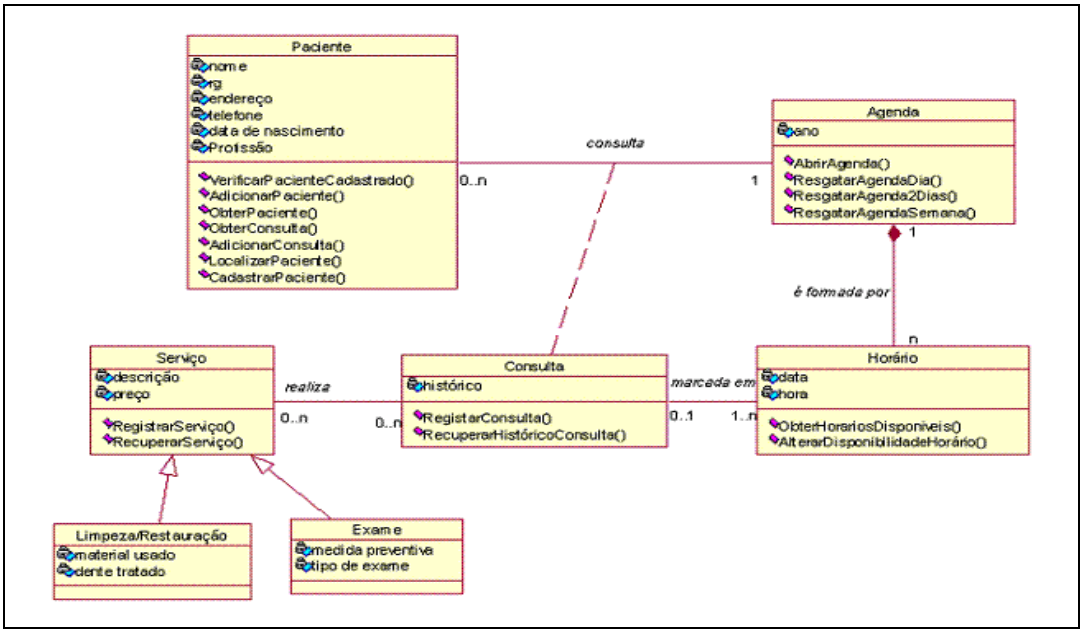

Figura 2 - Exemplo de Diagrama de Classes
Fonte: MACORATTI (2004)

2.4.1.2 Diagramas de Casos de Uso

Segundo Booch e Jacobson (2000, p. 95), “Um diagrama de caso de uso mostra um conjunto de casos de uso atores (um tipo especial de classe) e seus relacionamentos”.

O diagrama de casos de uso propicia uma visão externa do sistema, mostrando suas principais funcionalidades. Desta forma pode-se representar uma visão de alto nível de funcionalidades do sistema referente às requisições feitas pelo usuário.

De acordo com Silva (2007, p. 101) “A visão do sistema sob modelagem dada pelo diagrama é o que se chama de visão caixa-preta, isto é, observa-se o que o sistema faz, mas não sua estrutura interna”.

O objetivo do diagrama de casos de uso é identificar basicamente todos os recursos que os usuários esperam que o sistema ofereça, porém ele não revela qualquer detalhe sobre a implementação desses recursos.

A Figura 3 exemplifica o diagrama de casos de uso, que é voltado à modelagem do conjunto de funcionalidades de um software. Neste diagrama os atores(bonecos) são representados por Caixa, Gerente, Cliente e Usuário, e as elipses representam os casos de uso, ou seja, funcionalidades que podem ser executadas por estes atores, por exemplo, Consulta Saldo.

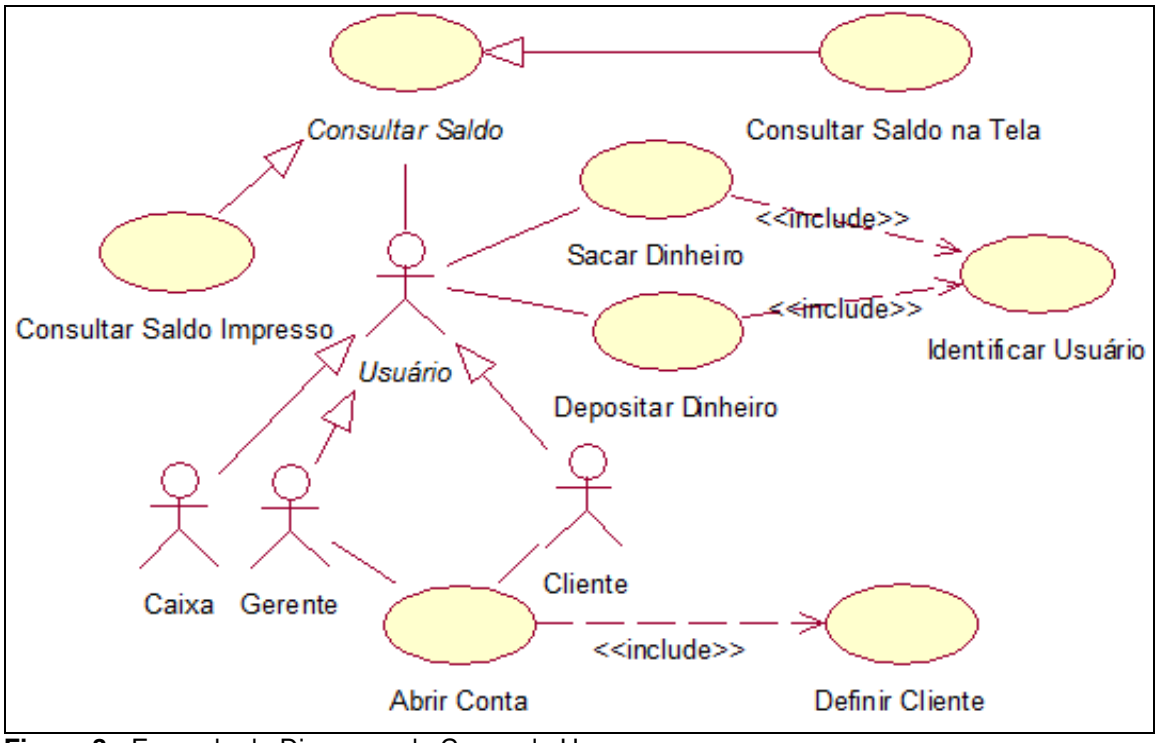

Figura 3 - Exemplo de Diagrama de Casos de Uso
Fonte: SOUZA (2009)

2.4.1.3 Diagrama de Sequência

O diagrama de sequência representa a sequência de mensagens passadas entre os objetos. O foco do diagrama está em identificar as interações entre os objetos com o passar do tempo.

Segundo Pender (2004, p. 192), “O diagrama de sequência utiliza uma visualização orientada para o tempo. Ele usa um conjunto de ícones de objeto e linhas de tempo associadas, chamadas linhas de tempo do objeto, para cada objeto”.

O diagrama de sequência enfatiza especialmente a ordem e os momentos em que as mensagens são enviadas para os objetos.

A Figura 4 ilustra o diagrama de sequência, com as trocas de mensagens entre objetos e entre atores e objetos.

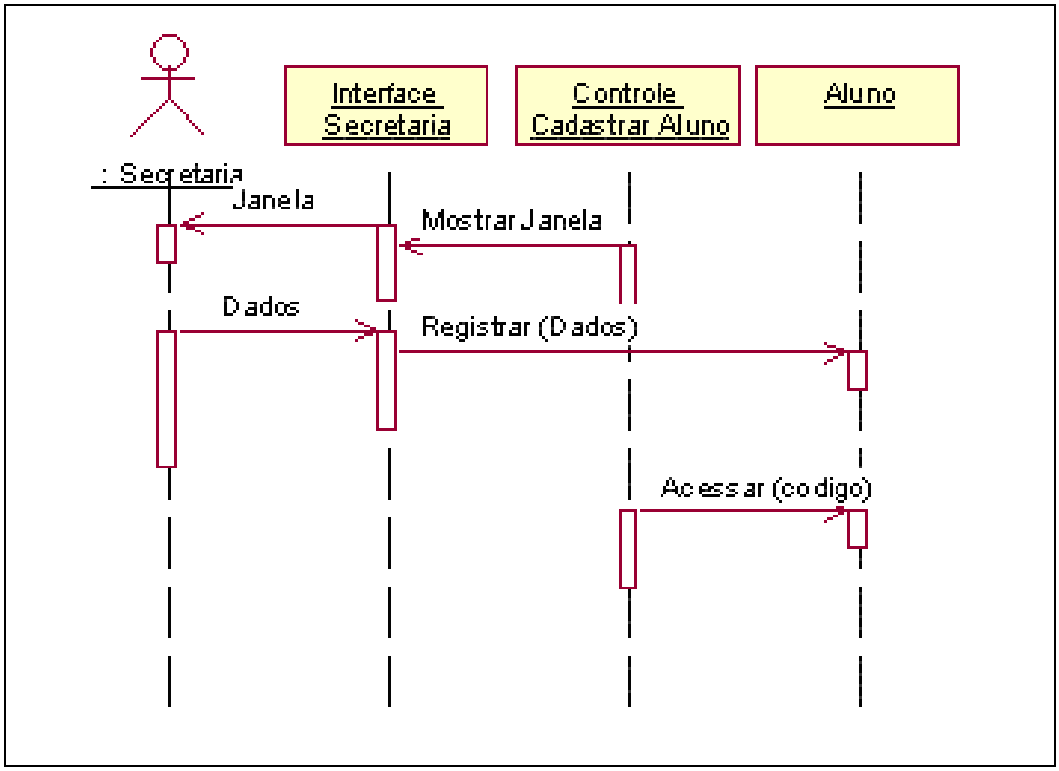

Figura 4 - Exemplo de Diagrama de Sequência
Fonte: MACORATTI (2005)

2.5 Análise Estruturada e Essencial

Antes do uso da análise estruturada, os sistemas eram indivisíveis, redundantes e de difícil manutenção.

A utilização da análise estruturada tornou possível quantificar alguns benefícios como: melhor produtividade, melhor uso do tempo de teste, maior facilidade para fazer qualquer modificação. Na fase da análise estruturada é feita a construção de modelos, retratando o fluxo e o conteúdo das informações utilizadas pelo sistema.

A análise estruturada tem por objetivo resolver dificuldades de comunicação entre usuário e responsável pelo desenvolvimento, por meio de uma abordagem sistemática, fase por fase. A especificação do sistema é o elo entre a análise e o projeto, fornecendo uma descrição dos requisitos do sistema a ser desenvolvido,

produzindo assim uma especificação do sistema que define a estrutura do problema a ser resolvido de acordo com a visão do usuário.

O conceito fundamental da análise estruturada é a construção de modelos lógicos do sistema, por meio de recursos técnicos que sejam capazes de representar para os usuários e desenvolvedores um modelo do sistema e como cada parte do sistema se encaixa. O objetivo é focalizar as mais importantes características, discutindo as alterações e correções, verificando se o desenvolvedor ou analista conhece corretamente o ambiente do usuário.

Dentro deste novo conceito dividiu-se então a análise na sua parte de estruturação e na parte essencial, onde a análise estruturada é responsável por propor os modelos lógicos e físicos, enquanto a análise essencial em criar os modelos essenciais e de implementação.

No entanto nota-se que a questão da documentação dos sistemas tornou-se de grande importância, tanto que ainda a análise essencial subdividiu-se em análise ambiental e comportamental. A primeira define a fronteira do sistema com o ambiente onde ele se situa, determinado o que é externo e interno a ele e a segunda descreve o comportamento das partes internas do sistema necessárias para interagir com o ambiente.

As Figuras 5 e 6 ilustram a divisão e subdivisão da análise essencial.

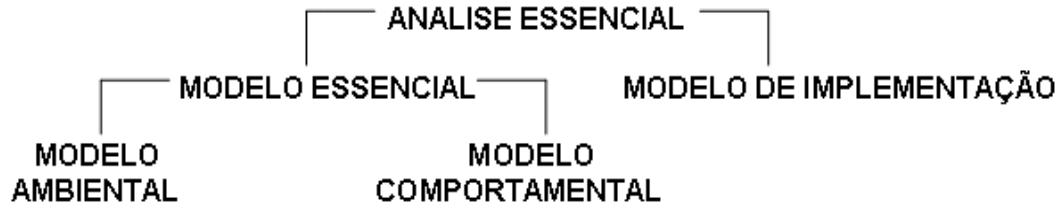

Figura 5 - Esquema da análise essencial

TÉCNICAS	ABORDAGENS	FERRAMENTAS
Análise Essencial	Funcional Dados Controle	Tabela de Eventos Diagrama de Fluxo de Dados (DFD) Diagrama de Entidade e Relacionamento (DER) Diagrama de Transição de Estados (DTE) Diagrama de Estrutura de Dados(DE) Especificações Funcionais Normalização Dicionário de Dados (DD)

Análise Estruturada	Funcional	Diagrama de Fluxos de Dados (DFD) Diagrama de Estrutura de Dados (DE) Especificações Funcionais Normalização Dicionário de Dados (DD)
Análise Tradicional	Funcional	Textos Fluxogramas

Figura 6 - Abordagens e Ferramentas utilizadas nas diferentes técnicas de análise

Dentre as ferramentas apresentadas na Figura 6, será abordado a seguir o Diagrama de Entidade e Relacionamento, utilizado neste trabalho para modelar a estrutura do banco de dados que foi criado.

2.5.1 Diagrama de Entidade e Relacionamento

Mesmo utilizando os conceitos de orientação a objetos citados anteriormente para modelar (diagramas da UML) e implementar o sistema, optou-se por modelar o banco de dados por meio de um diagrama de entidade e relacionamento, componente da análise estruturada. Esta escolha fundamenta-se no fato de que na programação orientada a objetos, pode-se trabalhar com classes, interfaces, ou outros elementos que nem sempre possuem dados para serem armazenados no banco de dados, e da mesma forma, uma ou mais tabelas, podem ser agrupadas em uma única classe, se necessário.

Desta forma, os diagramas da UML gerados com base na análise orientada a objetos apresentam a estrutura estática e dinâmica do sistema, enquanto o diagrama de entidade e relacionamento gerado com base na análise estruturada apresenta a estrutura do banco de dados relacional, destacando a forma como as tabelas se relacionam.

O modelo de entidade e relacionamento foi desenvolvido por Peter Chen, com a finalidade de representar as estruturas de dados de uma forma mais natural e mais próxima do mundo real.

Apesar de ter recebido algumas outras representações e abordagens diferentes por alguns outros estudiosos, o modelo entidade e relacionamento acabou se tornando o mais usado.

Os diagramas de entidade e relacionamento representam os relacionamentos entre objetos de dados e conduzem à modelagem de dados.

Este diagrama é composto por: entidades, atributos e relacionamentos.

As entidades representam tudo o que pode ser observado no mundo real com existência independente. Pode ser um objeto com existência física (ex. aluno), ou pode ser um objeto com existência conceitual (ex. curso).

Atributos são as propriedades que descrevem uma entidade, no caso de um aluno, o nome, e no caso de um curso, o seu código.

E os relacionamentos são as associações entre uma ou várias entidades. Por exemplo, um aluno se matricula em um curso, o ato de matricular é que relaciona o aluno ao curso.

No entanto, para identificar as chaves primárias e estrangeiras, que irão compor a estrutura das tabelas de um banco de dados, é necessário que se defina a cardinalidade dos relacionamentos. No exemplo descrito acima, um aluno pode se matricular em diversos cursos.

Há várias formas de representar a cardinalidade desses relacionamentos. No relacionamento de 1 para 1, por exemplo, um pedido só pode ter uma fatura. Já o relacionamento de 1 para muitos ou de muitos para 1, como mostra a Figura 7, um empregado tem vários dependentes e os dependentes tem vínculo com um empregado. E por último, no relacionamento de muitos para muitos, por exemplo, um aluno pode cursar várias disciplinas e a disciplina pode ter diversos alunos.

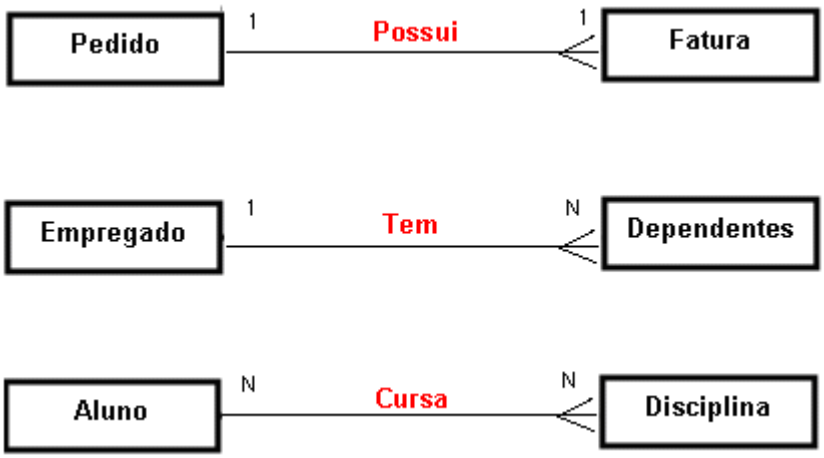

Figura 7 - Relacionamentos

A Figura 8 apresenta um exemplo de diagrama de entidade e relacionamento, conforme o que será desenvolvido neste trabalho para representar a estrutura dos dados.

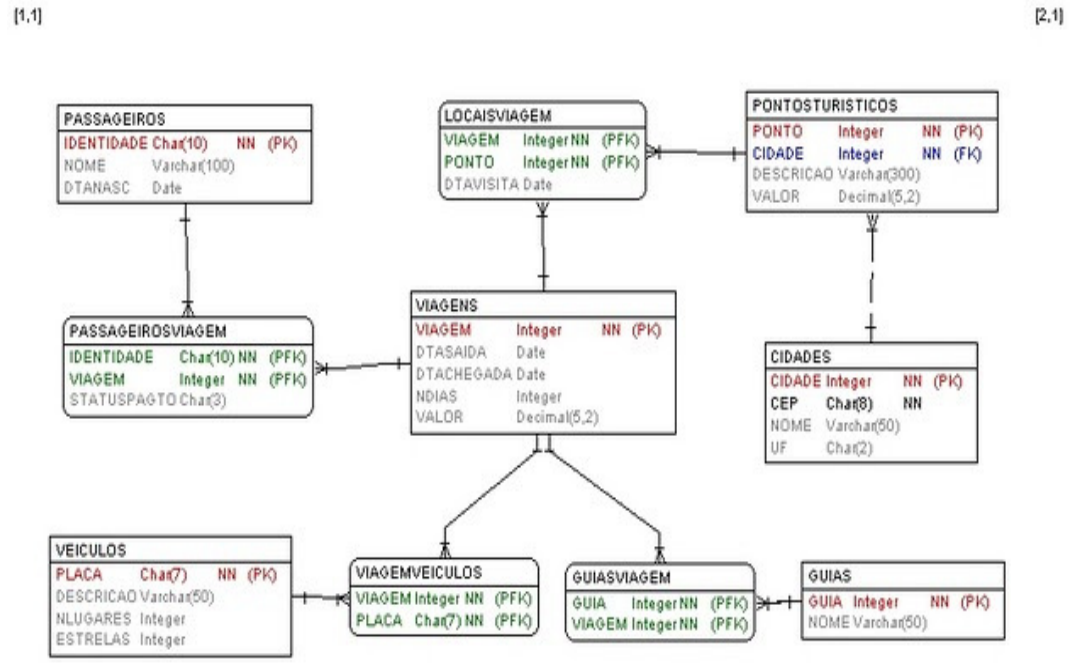

Figura 8 - Exemplo de Diagrama de Entidade e Relacionamento
Fonte: BOLÍVAR (2010)

3 MATERIAIS E MÉTODO

A ênfase deste capítulo está em reportar o que e como as atividades previstas foram realizadas para alcançar o objetivo do trabalho. Este capítulo apresenta as ferramentas utilizadas durante a análise, modelagem e desenvolvimento do protótipo de gerenciamento de *Pet Shops* e o método aplicado para elaboração do trabalho.

3.1 Materiais

Para análise e desenvolvimento do protótipo: primeiramente foi realizada uma entrevista com empresários do ramo de P*et Shop,* por meio da aplicação de um questionário (Apêndice A) para auxiliar no levantamento de requisitos. Em seguida,

com base nos requisitos identificados, foi realizada a modelagem estática e dinâmica do sistema por meio da ferramenta *Astah* para criação dos diagramas da UML e o DER (Diagrama de Entidade e Relacionamento) referente à modelagem do banco de dados foi criado com a ferramenta *CaseStudio*.

Para implementação do protótipo foi utilizada a linguagem de programação *Microsoft Visual Basic 2005 Express*, como sistema gerenciador de banco de dados o *Microsoft SQL Server 2005* e como ferramenta de gerenciamento *SQL Server Management Studio Express.*

Em seguida serão descritas as ferramentas que foram utilizadas para realizar a modelagem e implementação do protótipo proposto.

3.1.1 Ferramenta de Modelagem - *Astah Community*

O *Astah Community* (nova versão do software JUDE) é uma IDE (*Integrated Development Environment* ou ambiente integrado para desenvolvimento) para Modelagem de Dados (UML) criada em Java e de uso fácil e intuitivo. Com a IDE *Astah* é possível realizar uma modelagem de dados complexa, apresentando os dados para o usuário de forma clara.

A IDE possui uma árvore de estrutura com todos os dados à disposição do usuário para se criar diagramas, mapas entre outros. Dentre os recursos da ferramenta estão:

- Suporte a UML 2.1;
- Diagramas de Classe, Caso de Uso, Sequência, Atividade, Comunicação, Máquina de Estado, Componentes, Implantação, Estrutura de Composição, Objetos e Pacotes;
- Ajustes de alinhamento e tamanho dos diagramas;
- Impressão dos diagramas (com a marca d'água da ferramenta).

3.1.2 Ferramenta de Modelagem – *CASE Studio*

A ferramenta *CASE* (*Computer-Aided Software Engineering*) surgiu na década de 70, como uma variante de ferramentas *CAD* (*Computer-Aided Design*), utilizadas

até hoje no campo da engenharia. As ferramentas *CASE*, tinham como objetivo automatizar atividades manuais de pré-codificação, como criação de Diagramas de Entidade e Relacionamento (DER) e Diagramas de Fluxo de Dados (DFD) (BORTOLIN JUNIOR, 2005)

Atualmente, outros diagramas são criados com o auxílio destas ferramentas, mas de um modo geral, pode-se dizer que uma ferramenta *CASE* corresponde a um aplicativo que auxilia os profissionais envolvidos na tarefa de produzir sistemas.

Um dos componentes indispensáveis de uma ferramenta *CASE* é a modelagem visual, ou seja, a possibilidade de representar, por meio de modelos gráficos, o que está sendo definido. Neste trabalho, a ferramenta *CASE Studio* foi utilizada para criar modelos gráficos da análise orientada a objetos, baseando-se na linguagem UML.

Abaixo definição de autor:

> "*CASE Studio* é uma ferramenta para administrar e criar banco de dados de todos os estilos. Ela permite que se faça um mapa dos bancos de dados e seus vínculos, de uma forma muito intuitiva, ajudando muito quem tem que lidar com bancos de dados grandes e cheios de vínculos. Sua interface é arrojada, espaçosa; ele permite que se faça engenharia reversa de banco de dados existentes, e é compatível com todos os grandes formatos de Banco de Dados" (FUX, 2010).

3.1.3 SQL *Server Management Studio*

O SQL *Server Management Studio* é o ambiente gráfico que permite ao desenvolvedor realizar o gerenciamento do banco de dados, é a interface visual do banco de dados SQL *Server*. A partir dele é possível acessar, configurar, gerenciar, desenvolver e administrar todos os componentes do SQL Server, podendo manipular as bases visualmente ou por meio de comandos (*scripts*), esses últimos utilizando-se da linguagem padrão SQL e funções próprias do SQL *Server*. Essa ferramenta foi utilizada para criar as tabelas, *procedures*, e atualizações padrões nos dados do sistema.

O SQL *Server Management Studio* propicia ao desenvolvedor maior facilidade ao acessar os dados, pois combina ferramentas gráficas com sofisticadas

capacidades de comandos (*scripts*) para dar acesso ao SQL *Server* para usuários de todos os níveis de conhecimento e habilidade.

3.1.4 SQL *Server* 2005

O *SQL Server* é um dos gerenciadores de banco de dados mais conceituados do mundo e um dos mais utilizados por empresas de software.

> "Microsoft SQL *Server* é um banco de dados relacional que é executado no sistema operacional NT. A SQL é um padrão do setor amplamente aceito para definir, alterar e gerenciar dados e controlar como as mudanças no banco de dados são feitas usando-se tabelas, índices, chaves, linhas e colunas em dados armazenados" (COFFMAN, 2000, p.4).

O SQL *Server* possui uma interface de fácil utilização e entendimento, e muito interativa ao desenvolvedor. Possui múltiplas funções internas, utiliza-se da linguagem de consulta estruturada.

O SQL *Server* 2005 oferece uma solução integrada de gestão e análise de dados que pode ajudar as organizações, independentemente da sua dimensão, por intermédio de uma plataforma detalhada e confiável para execução das mais difíceis aplicações. Fornece a geração e restauração de backups, maior segurança e facilidade ao desenvolvedor para o gerenciamento dos dados.

Neste trabalho foi utilizada a versão *Express*, que é a versão gratuita. Essa versão é limitada em relação à versão completa tendo menos recursos e menor capacidade de armazenamento de dados. A versão gratuita é para pequenas empresas, mas geralmente usada para demonstração e distribuição a clientes de empresas que possuem uma base de dados pequena. A versão completa suporta banco de dados com grandes volumes de armazenamento mantendo o bom desempenho.

3.1.5 *Visual Basic* 2005

O *Visual Basic* é a linguagem de programação, em ambientes gráficos, com mais programadores a nível mundial, conforme afirma.

> "*Visual Basic* é a linguagem de programação, em ambientes gráficos, com mais programadores a nível mundial. Para quem se quer iniciar no

> fascinante e divertido mundo da programação, a versão *Visual Basic* 2005 *Express* é a mais eficaz de sempre, oferecendo, simultaneamente, uma extrema facilidade de programação para iniciados e ferramentas poderosas para programadores mais avançados. Nunca houve melhor altura para aprender a programar." (PEREIRA, 2006, p.21).

O *Visual Basic* 2005 oferece linguagem e ferramenta produtivas e de fácil uso para a criação rápida de aplicativos para o Microsoft Windows.

O *Visual Studio* é um kit de desenvolvimento de software criado pela Microsoft que auxilia programadores a integrarem ferramentas, editores, linguagens e diversas ferramentas já incluídas no pacote. É considerada uma ótima ferramenta para desenvolvimento *web* ou *desktop*, com interface amigável e relativamente simples. A versão utilizada nesse trabalho foi a Microsoft *Visual Basic* 2005 *Studio Express Edition*, que é gratuita e encontra-se disponível para download no próprio site da Microsoft.

3.2 Método

Como etapa inicial para o desenvolvimento deste trabalho, foram feitas entrevistas aplicando um questionário composto por questões objetivas e dissertativas (Apêndice A). Foram efetuadas também pesquisas na internet sobre o ramo de *Pet Shops,* além de um levantamento sobre as ferramentas utilizadas.

Para efetuar a captura dos requisitos podem ser utilizadas várias formas como aplicação de questionários, observação direta ou entrevistas.

O levantamento de requisitos deste trabalho foi feito por meio da aplicação de um questionário nas empresas do ramo estudado, tentando identificar as principais necessidades das empresas de *Pet Shop.* O questionário foi aplicado em duas empresas do ramo, porém somente uma retornou. Além disso, foi possível realizar observação direta por meio do acompanhamento das principais operações realizadas nestas empresas, identificando assim que os grandes problemas estão relacionados com o agendamento de serviços e a falta de informações dos clientes.

Os diagramas foram elaborados, utilizando obras de autores citados pelos professores nas disciplinas de análise e gerenciamento de projetos de sistemas.

A pesquisa realizada caracteriza-se como um estudo de caso por buscar identificar as principais necessidades do ramo de *Pet Shops*. É qualitativa à medida que procura melhorar os processos realizados nas empresas deste segmento, uma vez que propõe a informatização desses processos. É quantitativa quando busca por meio de entrevistas e aplicação de questionários fazer o reconhecimento da situação do problema, bem como quantificar dados a fim de identificar a atual situação dos empresários que trabalham em *Pet Shops* de pequeno porte.

4 RESULTADOS E DISCUSSÕES

Neste capítulo apresenta-se o protótipo desenvolvido, exibindo telas e partes do código, análise de requisitos e estrutura do banco de dados.

Primeiramente o protótipo é descrito exibindo suas funcionalidades e características. Em seguida são apresentada a análise e coleta de requisitos realizada, e a modelagem gerada por intermédio dos diagramas da UML: Diagramas de Caso de Uso, de Classes e de Sequência. E referente à modelagem do banco de dados, é apresentado o Diagrama de Entidade e Relacionamento criado.

Ao final é exibido o protótipo do sistema, sua interface visual e o código gerado, destacando as principais funcionalidades desenvolvidas utilizando conceitos de herança, polimorfismo e demais conceitos de orientação a objetos.

4.1 Descrição do Sistema

O protótipo desenvolvido permite ao usuário inserir informações básicas no sistema, possibilita acesso às mesmas de forma ágil, é prático e fácil de ser utilizado, mesmo para pessoas com pouco conhecimento em informática.

Foi desenvolvido especificamente para empresas de *Pet Shop*, visando otimizar os processos realizados nesses empreendimentos.

As telas geradas seguem um padrão de *layout* para facilitar a utilização.

Segue abaixo, como foi feita a modelagem do protótipo baseada na UML.

4.2 Modelagem do Sistema

Para o desenvolvimento de um protótipo é necessário primeiramente fazer a análise, que se inicia por meio do processo de levantamentos de requisitos. Neste processo, é possível identificar as principais necessidades e problemas do ramo escolhido, podendo assim identificar possibilidades de suprir as necessidades e sanar os problemas com a ajuda do protótipo proposto.

A modelagem do protótipo foi baseada na UML, criando diagramas de casos de uso, de classes e de sequência.

Esta seção apresentará os requisitos identificados e classificados com base na entrevista, aplicação do questionário (Apêndice A) e observação direta realizada. Foi possível identificar alguns dos requisitos funcionais e não-funcionais considerados necessários para o funcionamento adequado do software. Em seguida será apresentada a modelagem gerada para o banco de dados.

4.2.1 Requisitos Funcionais e Não-Funcionais

A coleta dos requisitos foi definida com base em um questionário (Apêndice A) encaminhado via e-mail aos empreendedores do ramo, por meio de observação direta dos processos e entrevistas.

Os requisitos funcionais identificados são apresentados na Tabela 1, sendo utilizada a abreviatura RFX para identificar cada requisito, sendo X um identificador numérico.

Tabela 1 - Requisitos funcionais do sistema

Requisitos Funcionais		
Código	**Nome**	**Prioridade**
RF1	Efetuar Login	Essencial
RF2	Cadastrar cliente	Essencial
RF3	Localizar Ciente	Necessário
RF4	Excluir Cliente	Essencial
RF5	Editar dados do Cliente	Essencial
RF6	Inserir consultas\serviços	Essencial

RF7	Excluir consultas\serviços	Essencial
RF8	Registrar\Controlar Contas	Essencial
RF9	Manter Ficha do Histórico do Animal	Necessário
RF10	Manter Produto	Essencial
RF11	Manter Fornecedor	Essencial
RF12	Controlar Vacinação	Essencial
RF13	Cadastrar valores dos produtos	Necessário
RF14	Emitir relatórios	Necessário
RF15	Manter Cliente	Essencial
RF16	Manter Animal	Essencial
RF17	Manter Funcionário	Essencial

Os requisitos não-funcionais, caracterizados como restrições ou complemento às funcionalidades que o software deve possuir são apresentados na Tabela 2. Para os requisitos não-funcionais utilizou-se a abreviatura RNFX para identificar cada requisito, sendo X um identificador numérico.

Tabela 2 - Requisitos não-funcionais do sistema

Requisitos não-funcionais			
Código	**Nome**	**Prioridade**	**Relação com Funcionais**
RNF1	Valor de venda dos produtos	Necessário	RF8
RNF2	Aviso data de validade dos produtos a vencer	Essencial	RF10
RNF3	Formas de pagamento	Essencial	RF8

4.2.2 Diagrama de Casos de Uso

O diagrama de casos de uso apresenta a visão externa do sistema, mostrando suas principais funcionalidades. A Figura 9 mostra o diagrama de casos de uso gerado para o protótipo.

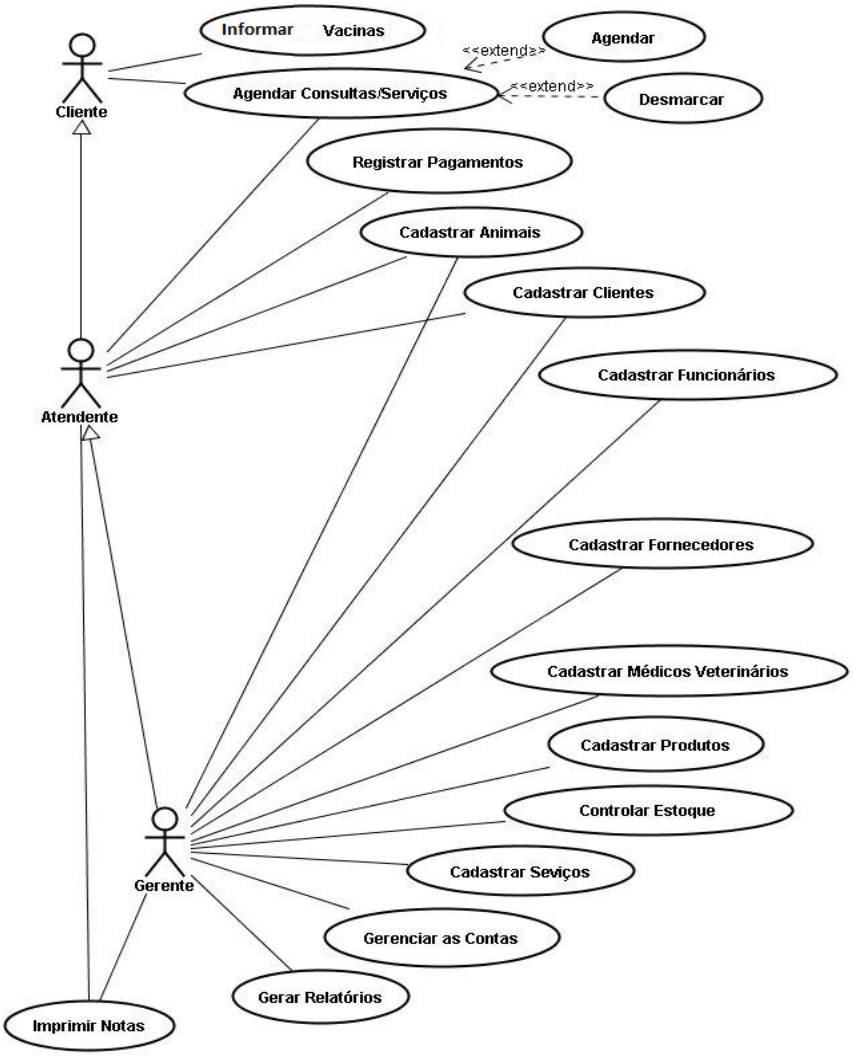

Figura 9 - Diagrama de Caso de Uso do Sistema de gerenciamento de *Pet Shop*

Os cenários dos casos de uso representam um conjunto de passos que são realizados durante a execução daquela funcionalidade. A seguir serão apresentados os cenários dos casos de uso do sistema proposto.

4.2.2.1 Informar Vacinas

Fluxo principal

1. O usuário irá acessar o sistema.
2. Cliente informa dados das vacinas para usuário do sistema.
3. O sistema registra os dados informados pelo usuário sobre as vacinas.

4.2.2.2 Agendar Consultas/Serviços

Fluxo principal

1. O cliente irá marcar uma consulta/serviço para seu animal.
2. O atendente irá acessar o sistema.
3. O atendente selecionará a opção de Agenda e em seguida localizará o cadastro do cliente e selecionará o animal.
4. O sistema registra os dados informados pelo atendente sobre o horário e data, incluindo o agendamento ou também poderá desmarcar.

4.2.2.3 Registrar Pagamentos

Fluxo principal

1. O atendente irá acessar o sistema.
2. O atendente selecionará a opção Financeiro e registrará o pagamento.
3. O sistema registra os dados informados pelo atendente sobre os dados do pagamento.

4.2.2.4 Cadastrar Animais

Fluxo principal

1. O usuário irá acessar o sistema.
2. O usuário selecionará na tela principal a opção Cadastro e em seguida a opção Animais.
3. O sistema registra os dados informados pelo usuário sobre os animais.

4.2.2.5 Cadastrar Clientes

Fluxo principal

1. O usuário irá acessar o sistema.

2. O usuário selecionará na tela principal a opção Cadastro e em seguida a opção Clientes.
3. O sistema registra os dados informados pelo usuário sobre os clientes.

4.2.2.6 Cadastrar Funcionários

Fluxo principal

1. O usuário irá acessar o sistema.
2. O usuário selecionará na tela principal a opção Cadastro e em seguida a opção Funcionários.
3. O sistema registra os dados informados pelo usuário sobre os funcionários.

4.2.2.7 Cadastrar Fornecedores

Fluxo principal

1. O usuário irá acessar o sistema.
2. O usuário selecionará na tela principal a opção Cadastro e em seguida a opção Fornecedores.
3. O sistema registra os dados informados pelo usuário sobre os fornecedores.

4.2.2.8 Cadastrar Médicos Veterinários

Fluxo principal

1. O gerente irá acessar o sistema.
2. O gerente selecionará a opção de Cadastro e selecionará em seguida a opção Médicos veterinários.
3. O sistema registra os dados informados pelo usuário sobre os médicos veterinários.

4.2.2.9 Cadastrar Produtos

Fluxo principal

1. O usuário irá acessar o sistema.
2. O usuário selecionará na tela principal a opção Cadastro e em seguida a opção Produtos.
3. O sistema registra os dados informados pelo usuário sobre os produtos.

4.2.2.10 Cadastrar Serviços

Fluxo principal

1. O usuário irá acessar o sistema.
2. O usuário selecionará na tela principal a opção Cadastro e em seguida a opção Serviços.
3. O sistema registra os dados informados pelo usuário sobre os serviços.

4.2.2.11 Gerenciar as Contas

Fluxo principal

1. O gerente irá acessar o sistema.
2. O gerente selecionará na tela principal a opção da tela de Financeiro.
3. O sistema registra os dados informados pelo usuário sobre as contas a receber.

4.2.2.12 Gerar Relatórios

Fluxo principal

1. O gerente irá acessar o sistema.
2. O gerente selecionará na tela principal a opção da tela de Gerar Relatórios.
3. O gerente selecionará qual(is) o(s) relatório(s) que deseja.

4. O gerente selecionará qual comando: para visualizar e após fazer a impressão ou para fazer a impressão direta.

4.2.2.13 Imprimir Notas

Fluxo principal

1. O gerente ou atendente irá acessar o sistema.
2. O gerente ou atendente selecionará na tela principal a opção Vendas.
3. O gerente ou atendente selecionará qual(is) a(s) venda(s) que deseja gerar a nota.
4. O gerente ou atendente selecionará qual comando: para visualizar e após fazer a impressão ou para fazer a impressão direta.

4.2.2.14 Controlar Estoque

Fluxo principal

1. O gerente ou atendente irá acessar o sistema.
2. O gerente ou atendente selecionará na tela principal a opção Controle de Estoque.
3. O gerente ou atendente selecionará qual(is) o(s) produtos(s) que deseja visualizar a quantidade contida em estoque.

4.3 Diagramas de Sequência

O diagrama de sequência descreve a maneira como os grupos de objetos interagem em algum comportamento ao longo do tempo, esboça a sequência dos acontecimentos no decorrer do processamento do sistema. Ele registra o comportamento de um único caso de uso e exibe os objetos e as mensagens passadas entre esses objetos no caso de uso. As Figuras 10 e 11 apresentam os diagramas de sequência criados para descrever como o sistema funciona internamente.

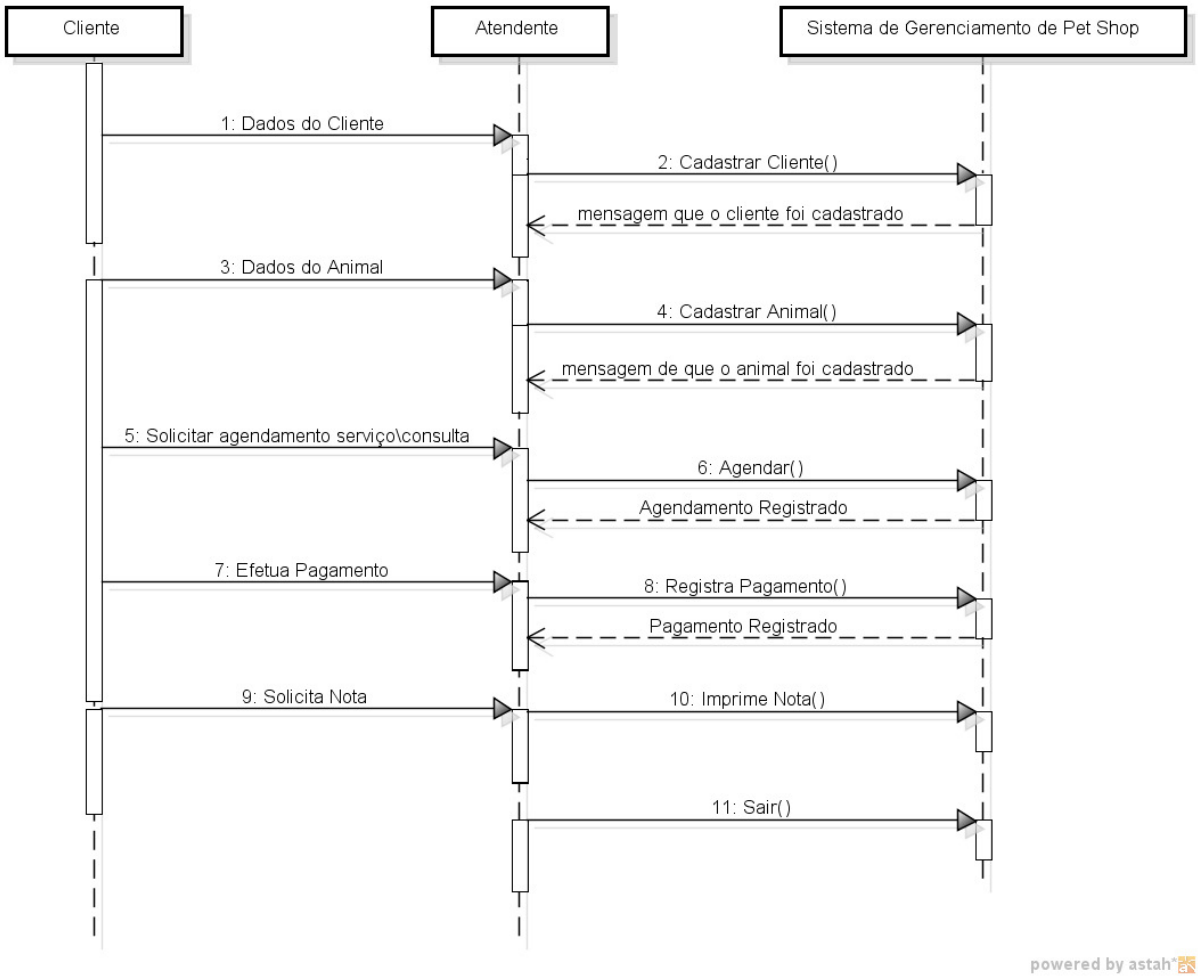

Figura 10 - Diagrama de Sequência 1 – Interação dos atores cliente e atendente com o sistema

Na Figura 10 tem-se a inclusão dos dados dos clientes e dos animais, inicialmente o atendente vai solicitar os dados ao cliente, e em seguida incluir os dados no sistema. Então para cada cliente que for ao *Pet Shop*, o atendente verificará o que ele deseja, e caso seja necessário, registrará os dados no sistema, agendará consultas/serviços, registrará o pagamento de serviços, consulta e produtos, e imprimirá notas. Este processo pode ser repetido por quantas vezes for necessário.

Na Figura 11 são representados os passos necessários para o gerente registrar funcionários, produtos, contas, relatórios e notas.

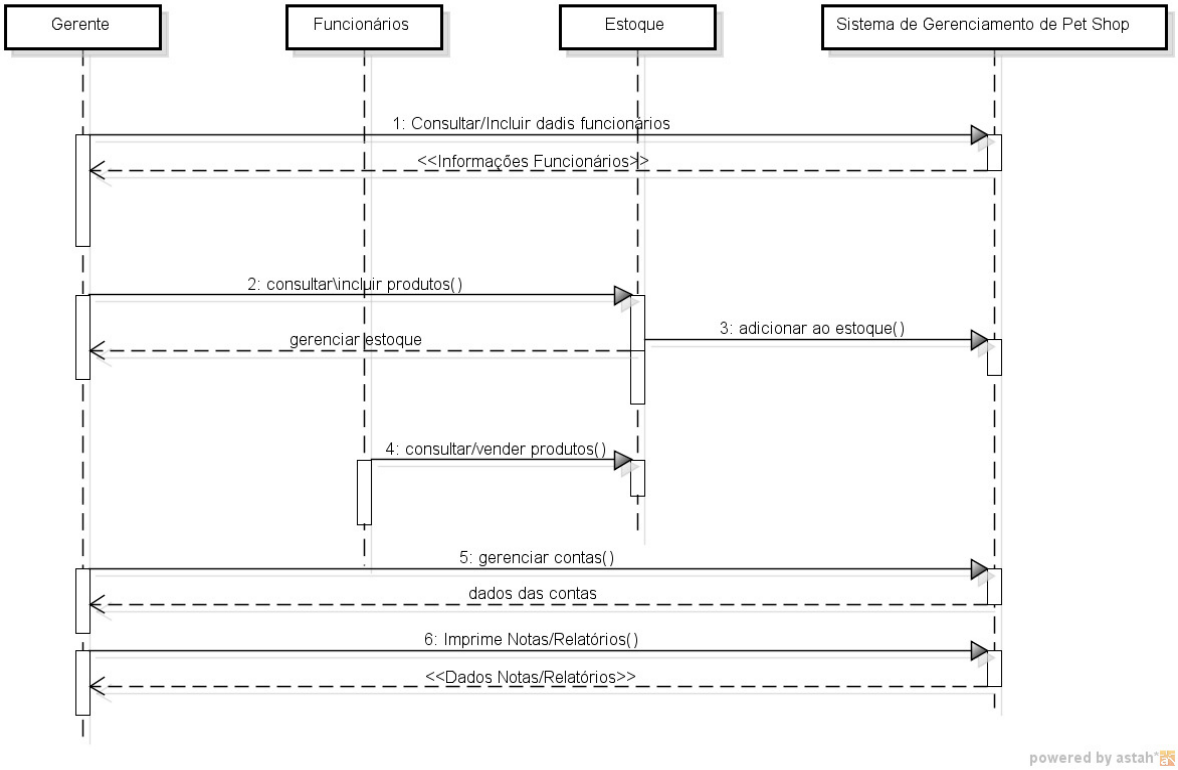

Figura 11 - Diagrama de Sequência 2 – Interação dos atores gerente e funcionário com o sistema

4.4 Diagramas de Classes

No desenvolvimento de sistemas, o diagrama de classes é a representação da estrutura e relações das classes que servem de modelo para objetos.

É uma modelagem muito útil para o sistema, define todas as classes que o sistema necessita possuir.

A Figura 12 mostra o diagrama de classes utilizado no sistema de gerenciamento de *Pet Shop*:

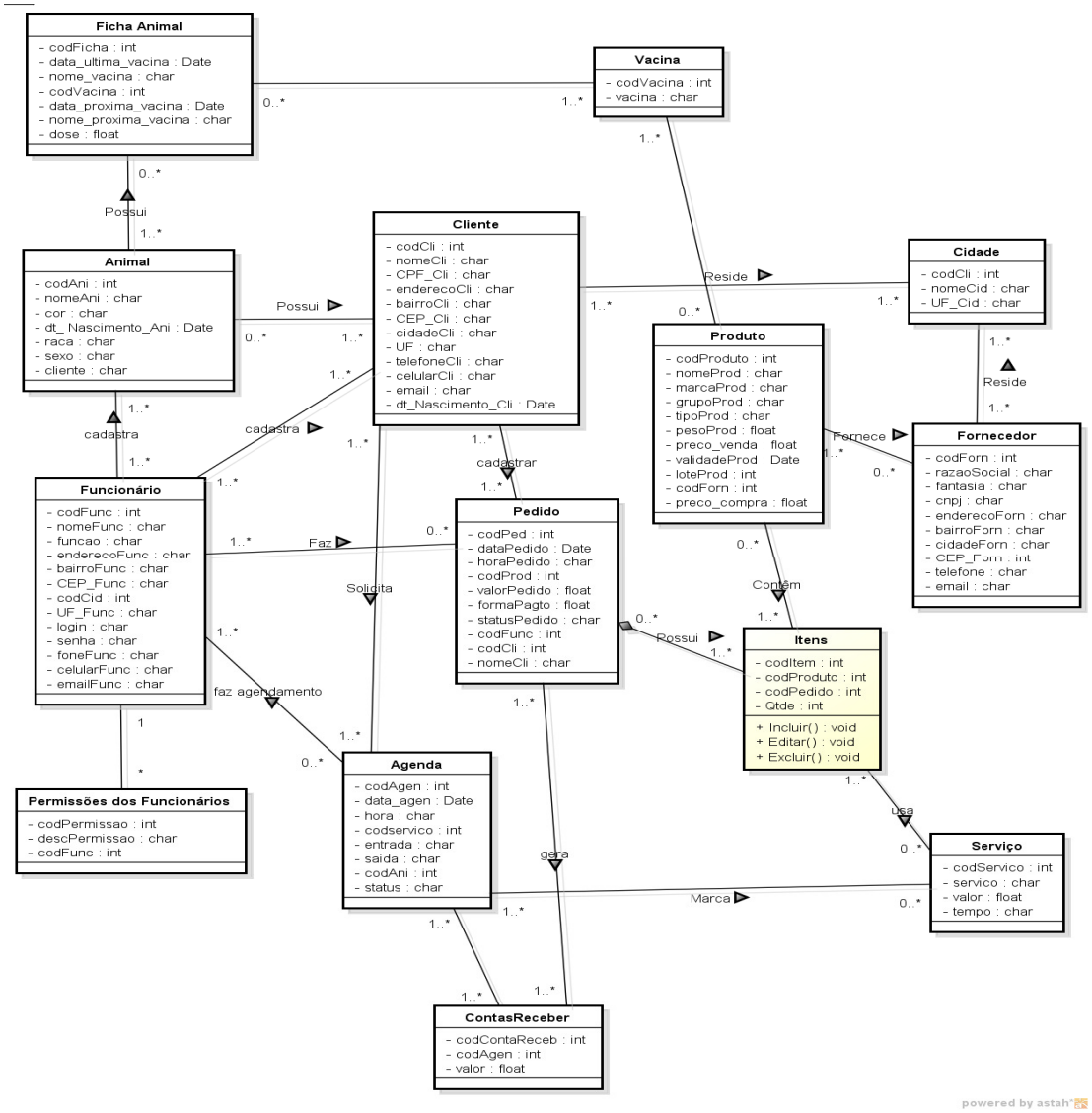

Figura 12 – Diagrama de Classes do Sistema de gerenciamento de Pet Shop

4.5 Diagrama de Entidade e Relacionamento

Todos os dados apresentados na modelagem até o momento estarão armazenados no banco de dados. Para isto definiu-se a estrutura do banco, que é demonstrada por meio do diagrama de entidade e relacionamento na Figura 13.

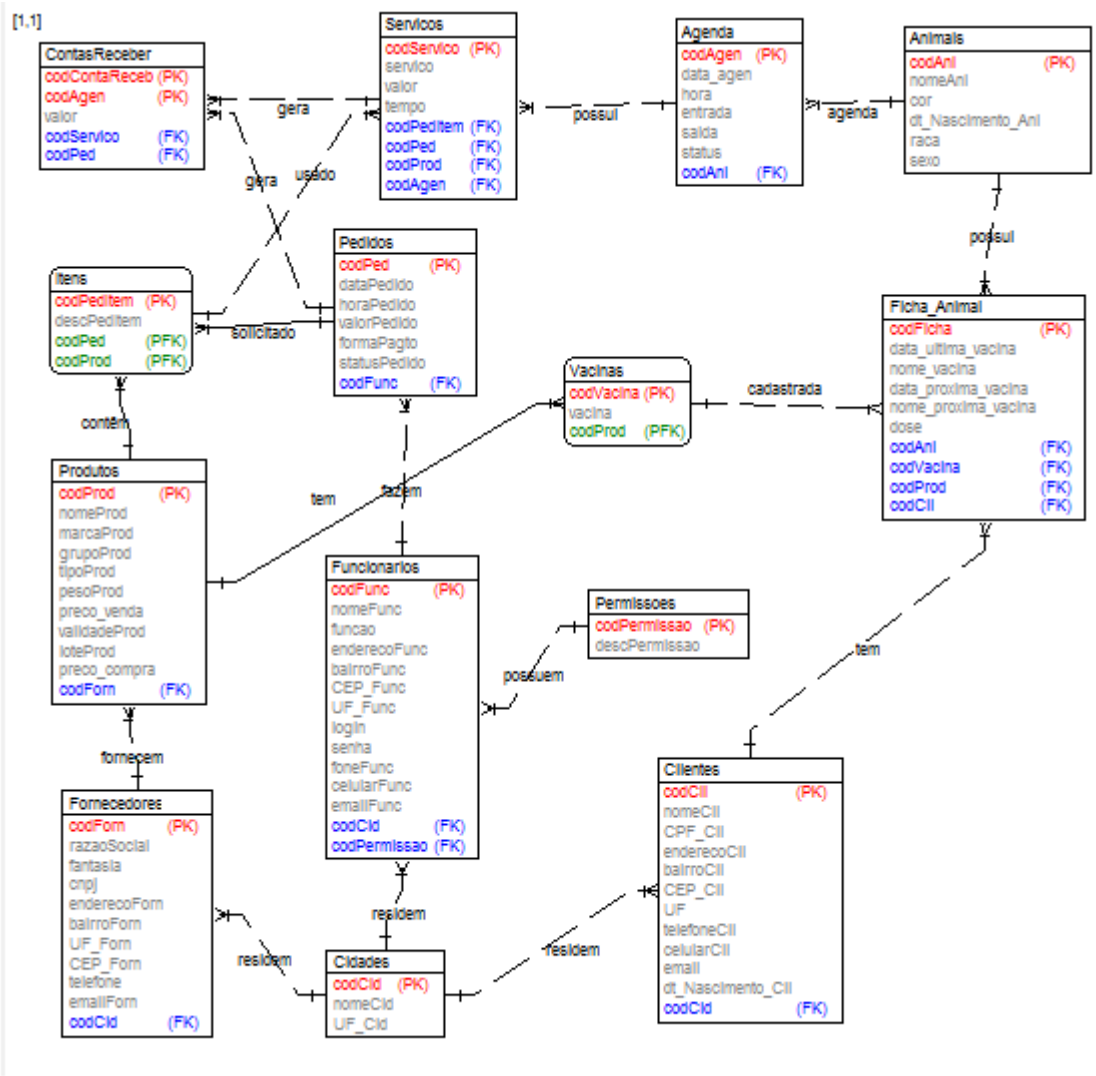

Figura 13 – Diagrama de Entidade e Relacionamento

4.6 Desenvolvimento e Codificação do Sistema

O protótipo desenvolvido possui 6 menus principais:

- Arquivo – possui o acesso ao cadastro de agendamentos de serviços e a função de saída do sistema, posteriormente em versões futuras conterá o sub-menu de permissões dos usuários.
- Cadastros - estão telas que permitem a entrada de dados, sendo telas dependentes uma das outras.
- Financeiro - são cadastradas as contas referentes aos serviços realizados.
- Relatórios - apresentam as informações dos lançamentos com a informação dos cadastros trazendo detalhadamente os resultados das

operações. Estes dados podem ser usadas pelo gerente para verificar lucros, lista de preços, contas, entre outros, visando principalmente uma melhor organização do estabelecimento.

- Sobre – Contém as informações da versão e do distribuidor do sistema.

Por meio das opções de menu descritas acima o usuário poderá visualizar informações precisas de forma rápida e com maior facilidade.

4.6.1 Tela Principal

Na Figura 14 observa-se a tela principal do protótipo. Na parte superior da tela, tem-se os menus principais, cada um deles composto por sub-menus. Também na parte superior da tela localiza-se a barra de botões de atalho, que tem por objetivo fornecer acesso rápido aquelas funcionalidades que o usuário utiliza com mais frequência, como por exemplo, Agendamentos, Ficha Animal, Clientes e Vacinas.

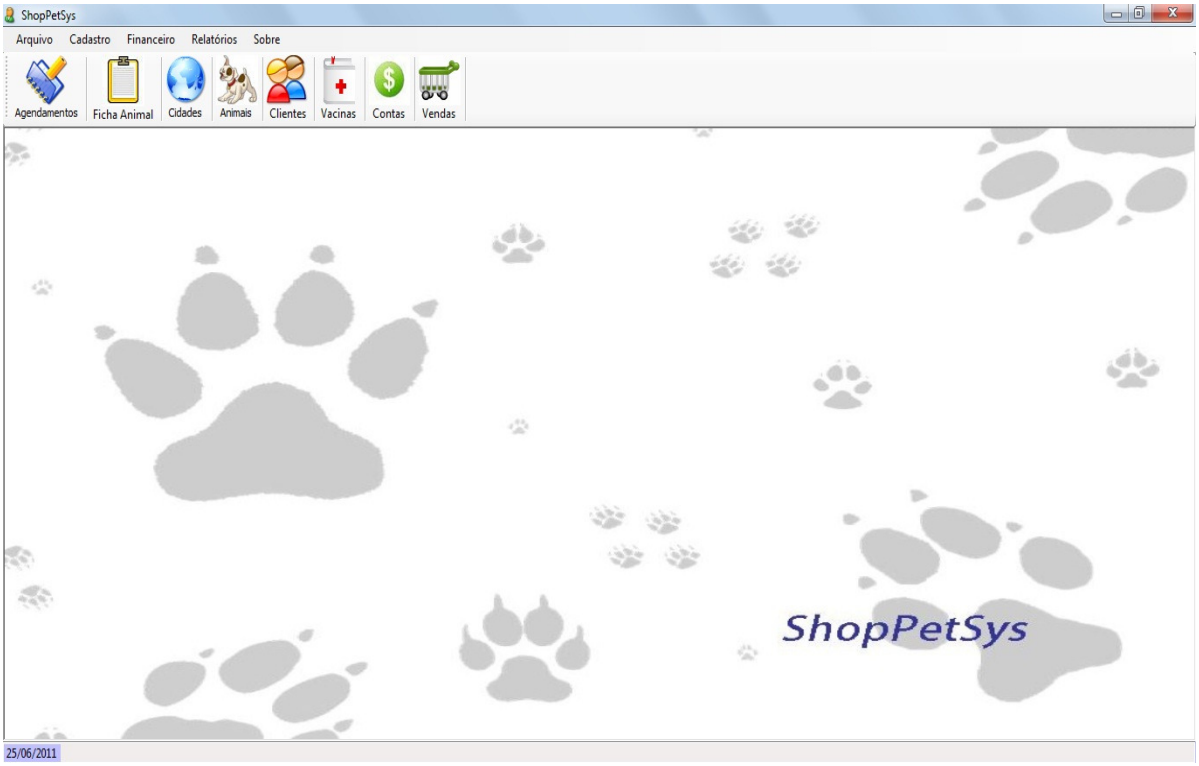

Figura 14 – Tela Principal

4.6.2 Classes

A principal classe do projeto foi codificada utilizando os conceitos de orientação a objeto estudados nas disciplinas da graduação.

No desenvolvimento por intermédio da ferramenta *Microsoft Visual Basic* 2005 *Express Edition*, os métodos e campos declarados no projeto podem ser *private* (privados) ou *public* (públicos).

Primeiramente foi feita a codificação da classe de conexão ao banco de dados. No Quadro 1 apresenta-se como foram declaradas as variáveis e métodos pertencentes à classe de conexão.

Quadro 1 – Código: Classe de Conexão com o Banco de Dados

```
Imports System.Data.SqlClient
Imports System.Data

Public Class ConexaoSQLServer
    Private ocn As SqlConnection
    Private sConexao As String

    Public Sub New()
        sConexao = "Data Source=.\SQLEXPRESS;Initial
Catalog=ShoPetSys;User Id=sa; password=asd123"
        ocn = New SqlConnection(sConexao)
    End Sub

    Public Property SqlConnection() As String
        Get
            Return sConexao
        End Get
        Set(ByVal value As String)
            sConexao = value
        End Set
    End Property

    Public Property ConexaoSQL() As SqlConnection
        Get
            Return ocn
        End Get
        Set(ByVal value As SqlConnection)
            ocn = value
        End Set
    End Property

    Public Sub Abrir()
        ocn.Open()
    End Sub

    Public Sub Fechar()
        ocn.Close()
    End Sub
End Class
```

refatoração, que permitirá a modificação do código do sistema para melhorar a

estrutura interna sem alterar o comportamento externo, por meio do encapsulamento de campos, ao gerar os métodos *get* e *set* para os atributos de uma classe, fazendo com que sejam atualizadas todas as referências dos atributos em código associados. Cantú (2003) diz que, para uma melhor abordagem orientada a objetos, os dados referentes aos atributos de uma classe, ou como eles são manipulados por ela, podem ser escondidos das outras classes, sob a forma de métodos de acesso.

O conceito nomeado como encapsulamento é utilizado na POO, por exemplo, para limitar o risco de gerar situações inválidas e permite ao autor da classe obter um código-fonte claro e de fácil manutenção, podendo modificar sua representação interna em uma versão futura.

Neste mesmo quadro são apresentadas as funções que possuem os métodos referentes à inserção, edição, exclusão e listagem das informações contidas no banco de dados. As demais classes do protótipo seguem o mesmo padrão da apresentada no Quadro 2.

Quadro 2 – Código: Classe Animais

```
Public Class Animais
    Private nCodigo As Integer
    Private sNome As String
    Private sCor As String
    Private dDt_Nasc_Ani As Date
    Private nCodRaca As Integer
    Private sSexo As String
    Private nCodCli As Integer
    Private nCodEspecie As Integer

    Public Property codAni() As Integer
        Get
            Return nCodigo
        End Get
        Set(ByVal value As Integer)
            nCodigo = value
        End Set
    End Property

    Public Property nomeAni() As String
        Get
            Return sNome
        End Get
        Set(ByVal value As String)
            sNome = value
        End Set
    End Property

    Public Property cor() As String
        Get
```

```
            Return sCor
        End Get
        Set(ByVal value As String)
            sCor = value
        End Set
    End Property

    Public Property dt_Nascimento_Ani() As Date
        Get
            Return dDt_Nasc_Ani
        End Get
        Set(ByVal value As Date)
            dDt_Nasc_Ani = value
        End Set
    End Property

    Public Property codRaca() As Integer
        Get
            Return nCodRaca
        End Get
        Set(ByVal value As Integer)
            nCodRaca = value
        End Set
    End Property

    Public Property sexo() As String
        Get
            Return sSexo
        End Get
        Set(ByVal value As String)
            sSexo = value
        End Set
    End Property

    Public Property codCli() As Integer
        Get
            Return nCodCli
        End Get
        Set(ByVal value As Integer)
            nCodCli = value
        End Set
    End Property

    Public Property codEspecie() As Integer
        Get
            Return nCodEspecie
        End Get
        Set(ByVal value As Integer)
            nCodEspecie = value
        End Set
    End Property


    Public Function Inserir _
        (ByVal pCN As SqlConnection) As Boolean

        Dim oCmd As New SqlCommand( _
        "Insert Into
Animais(nomeAni,cor,dt_Nascimento_Ani,codRaca,sexo,codCli,codEspecie)" & _
```

```
"Values(@nomeAni,@cor,@dt_Nascimento_Ani,@codRaca,@sexo,@codCli,@codEspecie)",
pCN)

        Dim p1 As New SqlParameter("nomeAni", sNome)
        Dim p2 As New SqlParameter("cor", sCor)
        Dim p3 As New SqlParameter("dt_Nascimento_Ani", Format(dDt_Nasc_Ani,
"dd/MM/yyyy"))
        Dim p4 As New SqlParameter("codRaca", nCodRaca)
        Dim p5 As New SqlParameter("sexo", sSexo)
        Dim p6 As New SqlParameter("codCli", nCodCli)
        Dim p7 As New SqlParameter("codEspecie", nCodEspecie)

        With oCmd.Parameters
            .Add(p1)
            .Add(p2)
            .Add(p3)
            .Add(p4)
            .Add(p5)
            .Add(p6)
            .Add(p7)
        End With

        If oCmd.ExecuteNonQuery() = 1 Then
            Return True
        Else
            Return False
        End If
    End Function


    Public Function Editar _
       (ByVal pCN As SqlConnection) As Boolean

        Dim oCmd As New SqlCommand( _
        "UPDATE Animais set nomeAni = @nomeAni, cor = @cor, " & _
          "dt_Nascimento_Ani = @dt_Nascimento_Ani, codRaca = @codRaca, sexo =
@sexo, codCli = @codCli, " & _
          "codEspecie = @codEspecie where codAni = @codAni", pCN)

        Dim p1 As New SqlParameter("nomeAni", sNome)
        Dim p2 As New SqlParameter("cor", sCor)
        Dim p3 As New SqlParameter("dt_Nascimento_Ani", Format(dDt_Nasc_Ani,
"dd/MM/yyyy"))
        Dim p4 As New SqlParameter("codRaca", nCodRaca)
        Dim p5 As New SqlParameter("sexo", sSexo)
        Dim p6 As New SqlParameter("codCli", nCodCli)
        Dim p7 As New SqlParameter("codEspecie", nCodEspecie)
        Dim p8 As New SqlParameter("codAni", nCodigo)

        With oCmd.Parameters
            .Add(p1)
            .Add(p2)
            .Add(p3)
            .Add(p4)
            .Add(p5)
            .Add(p6)
            .Add(p7)
            .Add(p8)
```

```
            If oCmd.ExecuteNonQuery() = 1 Then
                Return True
            Else
                Return False
            End If
        End With
    End Function

    Public Function Excluir _
        (ByVal pCN As SqlConnection) As Boolean

        Dim oCmd As New SqlCommand( _
            "Delete from Animais where codAni = @codAni", pCN)

        Dim p1 As New SqlParameter("codAni", nCodigo)

        With oCmd.Parameters
            .Add(p1)
        End With
        If oCmd.ExecuteNonQuery() = 1 Then
            Return True
        Else
            Return False
        End If
    End Function

    Public Function Listar _
        (ByVal pCN As SqlConnection) As DataTable
        Dim dt As New DataTable
        Dim da As New SqlDataAdapter( _
            "Select Animais.codAni, Animais.nomeAni, Animais.cor," & _
                   "Animais.dt_Nascimento_Ani, Animais.codRaca, Animais.sexo,"
& _
                   "Animais.codCli, Animais.codEspecie from " & _
                   "(Animais inner join Racas on Animais.codRaca =
Racas.codRaca) " & _
                   "inner join Clientes on Animais.codCli = Clientes.codCli",
pCN)

        da.Fill(dt)

        Return dt
    End Function

End Class
```

4.6.3 Cadastros

Cada formulário de cadastro possui sua classe do formulário de interface do banco de dados. A chamada das telas de cadastro é padrão, com cliques nas

opções de menu, além de permitir a inserção de novos registros, edição dos existentes, exclusão e geração de relatório, apresenta na tela os registros contidos no banco de dados na tabela respectiva do cadastro solicitado. Um exemplo de código da chamada de um desses cadastrados pode ser visualizada no Quadro 3.

Quadro 3 – Código: Carrega Cadastro de Animais

```
Imports System.Data.SqlClient

Public Class FormPai

Private Sub AbreCadastroFichaAnimal()
        Dim oForm As New frmCadastro_FichaAnimal
        oForm.MdiParent = Me

        oForm.Show()
    End Sub

    Private Sub btnFichaAnimal_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnFichaAnimal.Click
        AbreCadastroFichaAnimal()
    End Sub

    Private Sub FichaAnimalToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
FichaAnimalToolStripMenuItem.Click
        My.Forms.frmCadastro_FichaAnimal.MdiParent = Me
        My.Forms.frmCadastro_FichaAnimal.Show()
    End Sub
End Class
```

Na Figura 15 apresenta-se uma das telas de cadastro. A tela possui componentes de campo de texto (*TextBox*) e combos (*Combobox*) para inserção de informações, a grade (*DataGridView*) que apresenta e recebe as informações inseridas e os botões (*Button*), responsáveis pelas operações.

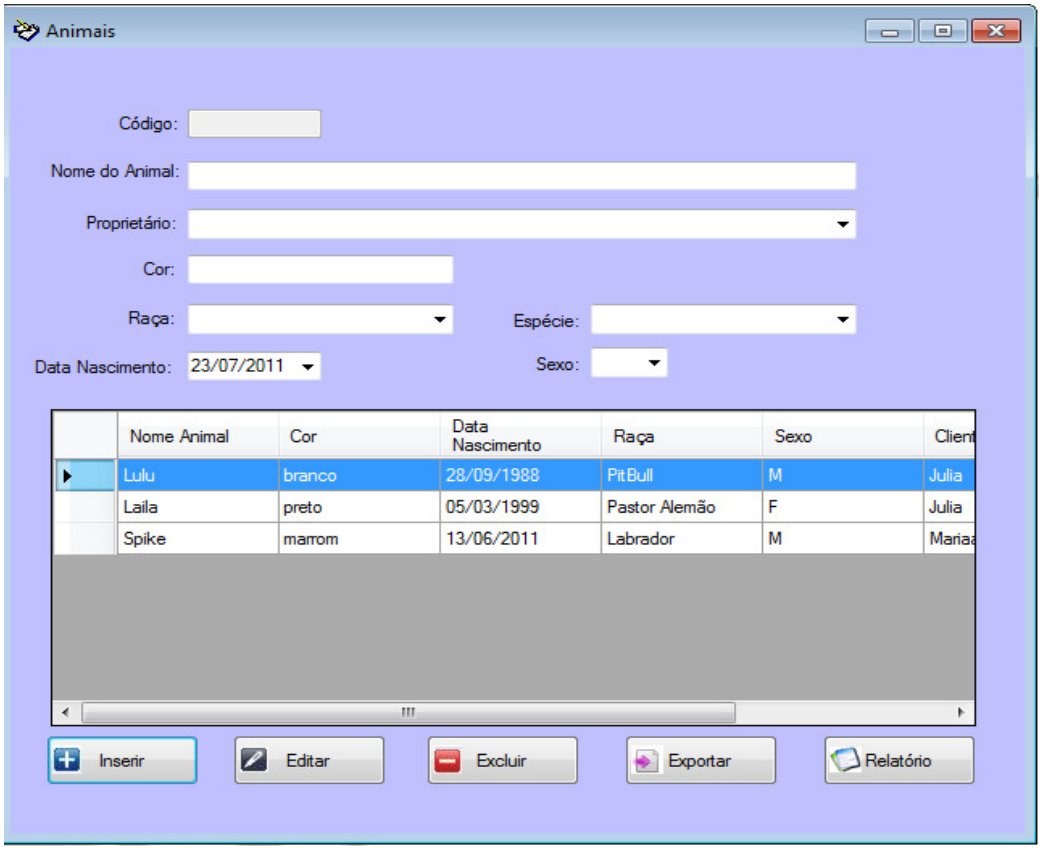

Figura 15 – Exemplo de Tela de Cadastro

Na parte inferior da tela de cadastro se encontram os botões de manipulação dos dados. A classe do formulário pode ser visualizada no Quadro 4.

Quadro 4 – Código: Classe de Manipulação de Informações

```
Imports System.IO
Imports System.Data.SqlClient

Public Class frmCadastro_Animais
    Dim nCodAniCorrente As Integer
    Dim sCaminho As String

    Private Sub AnimaisBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AnimaisBindingNavigatorSaveItem.Click
        Me.Validate()
        Me.AnimaisBindingSource.EndEdit()
        Me.AnimaisTableAdapter.Update(Me.ShoPetSysDataSet1.Animais)

    End Sub

    Private Sub CarregaCombo_Racas()
        Dim oConexao As New ConexaoSQLServer
        Dim oRacas As New Racas
```

```
        cmbRaca.DataSource = oRacas.Listar(oConexao.ConexaoSQL)

        cmbRaca.DisplayMember = "nomeRaca"

        cmbRaca.ValueMember = "codRaca"

    End Sub

    Private Sub CarregaCombo_Especies()
        Dim oConexao As New ConexaoSQLServer
        Dim oEspecies As New Especies

        cmbEspecie.DataSource = oEspecies.Listar(oConexao.ConexaoSQL)

        cmbEspecie.DisplayMember = "nomeEspecie"

        cmbEspecie.ValueMember = "codEspecie"

    End Sub

    Private Sub CarregaCombo_Clientes()
        Dim oConexao As New ConexaoSQLServer
        Dim oClientes As New Clientes

        cmbCodCli.DataSource = oClientes.Listar(oConexao.ConexaoSQL)

        cmbCodCli.DisplayMember = "nomeCli"

        cmbCodCli.ValueMember = "codCli"

    End Sub


    Private Sub frmCadastro_Animais_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        'TODO: This line of code loads data into the
'ShoPetSysDataSet1.Animais' table. You can move, or remove it, as needed.
        Me.AnimaisTableAdapter.Fill(Me.ShoPetSysDataSet1.Animais)
        CarregaCombo_Racas()
        CarregaCombo_Especies()
        CarregaCombo_Clientes()
        CarregaGridAnimais()
    End Sub

    Private Sub CarregaGridAnimais()
        Dim oConexao As New ConexaoSQLServer
        Dim oAnimais As New Animais

        AnimaisDgv.DataSource = _
        oAnimais.Listar(oConexao.ConexaoSQL)
        FormataGridAnimais()
        txtCodAni.Clear()
        txtNomeAni.Clear()
        txtCor.Clear()
        cmbRaca.SelectedValue = 0
        cmbSexo.Text = ""
        cmbCodCli.SelectedValue = 0
        cmbEspecie.SelectedValue = 0
```

```
    End Sub

    Private Sub CadastroAnimais()
        Dim oAnimais As New Animais
        Dim oConexao As New ConexaoSQLServer

        'EXECUTANDO METODO SET das PROPRIEDADES DO OBJETO
        oAnimais.nomeAni = txtNomeAni.Text
        oAnimais.cor = txtCor.Text
        oAnimais.dt_Nascimento_Ani = dtNascimento.Value
        oAnimais.codRaca = cmbRaca.SelectedValue
        oAnimais.sexo = cmbSexo.Text
        oAnimais.codCli = cmbCodCli.SelectedValue
        oAnimais.codEspecie = cmbEspecie.SelectedValue

        oConexao.Abrir()
        If oAnimais.Inserir(oConexao.ConexaoSQL) Then
            MsgBox("Animal cadastrado com sucesso!")
        End If
        oConexao.Fechar()
        CarregaGridAnimais()
    End Sub

    Private Sub FormataGridAnimais()
        AnimaisDgv.Columns(0).Visible = False
        AnimaisDgv.Columns(1).HeaderText = "Nome Animal"
        AnimaisDgv.Columns(2).HeaderText = "Cor"
        AnimaisDgv.Columns(3).HeaderText = "Data Nascimento"
        AnimaisDgv.Columns(4).HeaderText = "Raca"
        AnimaisDgv.Columns(5).HeaderText = "Sexo"
        AnimaisDgv.Columns(6).HeaderText = "Nome Proprietário"
        AnimaisDgv.Columns(7).HeaderText = "Espécie"
    End Sub

    Private Sub btnInserir_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnInserir.Click
        If nCodAniCorrente = 0 Then
            CadastroAnimais()
        Else
            EditaDadosAnimais()
        End If
        CarregaGridAnimais()
    End Sub

    Private Sub EditaDadosAnimais()
        Dim oAnimais As New Animais
        Dim oConexao As New ConexaoSQLServer

        oAnimais.nomeAni = txtNomeAni.Text
        oAnimais.cor = txtCor.Text
        oAnimais.dt_Nascimento_Ani = dtNascimento.Value
        oAnimais.codRaca = cmbRaca.SelectedValue
        oAnimais.sexo = cmbSexo.SelectedValue
        oAnimais.codCli = cmbCodCli.SelectedValue
        oAnimais.codEspecie = cmbEspecie.SelectedValue

        oConexao.Abrir()
        If oAnimais.Editar(oConexao.ConexaoSQL) Then
            MsgBox("Dados do Animal editado com sucesso!")
```

```
        End If
        oConexao.Fechar()
        CarregaGridAnimais()

        nCodAniCorrente = 0
        btnInserir.Text = "Inserir"
    End Sub

    Private Sub LimparDadosAnimais()
        If nCodAniCorrente = 0 Then
            txtCodAni.Clear()
            txtNomeAni.Clear()
            txtCor.Clear()
            cmbRaca.SelectedValue = 0
            cmbSexo.Text = ""
            cmbCodCli.SelectedValue = 0
            cmbEspecie.SelectedValue = 0
        End If
    End Sub

    Private Sub PreparaEdicao()
        'Jogar os dados do GRID para as TXTBoxes
        Dim nIndiceLinha As Integer
        Dim oConexao As New ConexaoSQLServer

        nIndiceLinha = AnimaisDgv.CurrentRow.Index

        With AnimaisDgv.Rows(nIndiceLinha)
            nCodAniCorrente = _
                 .Cells(0).Value

            txtCodAni.Text = _
                 .Cells(0).Value

            txtNomeAni.Text = _
                 .Cells(1).Value

            txtCor.Text = _
                 .Cells(2).Value

            dtNascimento.Value = _
                 .Cells(3).Value

            cmbRaca.SelectedValue = _
                 .Cells(4).Value

            cmbSexo.SelectedValue = _
                .Cells(5).Value

            cmbCodCli.SelectedValue = _
                .Cells(6).Value

            cmbEspecie.SelectedValue = _
                 .Cells(7).Value

            btnInserir.Text = "Salvar"
        End With
    End Sub
```

```
    Private Sub btnEditar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEditar.Click
        PreparaEdicao()
    End Sub

    Private Sub SalvarArquivoTexto()
        Dim file As FileStream 'Simboliza o trabalho com arquivos..
        Dim sw As StreamWriter
        Dim oAnimais As New Animais
        Dim oConexao As New ConexaoSQLServer
        Dim sNomeAni As String
        Dim sCor As String
        Dim sDataNasc As String
        Dim sRaca As String
        Dim sSexo As String
        Dim sNomeCli As String
        Dim sResultado As String


        'Representa .doc, dizendo que poderá ser aberto ou criado

        file = New FileStream(sCaminho, _
                              FileMode.OpenOrCreate, _
                              FileAccess.Write)
        'Objeto que ESCREVE caracteres no FILE (arquivo texto)
        sw = New StreamWriter(file)

        sw.BaseStream.Seek(0, SeekOrigin.End)

        sw.WriteLine("***************ANIMAIS***************")
        sw.WriteLine()


        For i As Integer = 0 To AnimaisDgv.Rows.Count - 1
            sNomeAni = "NOME ANIMAL: " &
AnimaisDgv.Rows(i).Cells(1).Value
            sCor = "COR: " & AnimaisDgv.Rows(i).Cells(2).Value
            sDataNasc = "DATA NASCIMENTO: " &
AnimaisDgv.Rows(i).Cells(3).Value
            sRaca = "RAÇA: " & AnimaisDgv.Rows(i).Cells(4).Value
            sSexo = "SEXO: " & AnimaisDgv.Rows(i).Cells(6).Value
            sNomeCli = "NOME PROPRIETÁRIO: " &
AnimaisDgv.Rows(i).Cells(7).Value
            sResultado = sNomeAni & vbCrLf & sCor & vbCrLf _
            & sDataNasc & vbCrLf & sRaca & vbCrLf & sSexo & vbCrLf &
sNomeCli
            sw.WriteLine(sResultado)
        Next

        'Para finalizar a operação é necessário FECHAR tanto o objeto que
ESCREVE, quanto o DOCUMENTO
        sw.Close()
        file.Close()

        MsgBox("Arquivo Criado Com sucesso!", _
                MsgBoxStyle.Information)

    End Sub
```

```
    Private Sub SalvarCaminho()
        'determina o filtro da caixa de diálogo que será apresentada...
        dlgSalvar.Filter = ".txt|*.txt"

        'Apresenta a caixa de diálogo e só salva caso o usuário
pressionar o botao de OK (Salvar)
        If dlgSalvar.ShowDialog() = Windows.Forms.DialogResult.OK Then
            sCaminho = dlgSalvar.FileName
            SalvarArquivoTexto()
        End If
    End Sub

    Private Sub btnExportar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnExportar.Click
        SalvarCaminho()
    End Sub

    Private Sub ExcluiAnimais()
        Dim oAnimais As New Animais
        Dim oConexao As New ConexaoSQLServer
        Dim nIndiceLinha As Integer

        nIndiceLinha = AnimaisDgv.CurrentRow.Index

        oAnimais.codAni = AnimaisDgv.Rows(nIndiceLinha).Cells(0).Value

        If oAnimais.codAni <= 0 Then Exit Sub


        oConexao.Abrir()
        If oAnimais.Excluir(oConexao.ConexaoSQL) Then
            MsgBox("Registro(s) excluído(s)!")
        End If
        oConexao.Fechar()
    End Sub

    Private Sub btnExcluir_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnExcluir.Click
        ExcluiAnimais()
        CarregaGridAnimais()
    End Sub


    'Chama formulário relatório
    Private Sub AbreFormularioRel()
        Dim oForm As New frmVisual_Relatorio_Animais

        oForm.Show()
    End Sub


      'Evento que abre tela do formulário
    Private Sub btnRelatorio_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnRelatorio.Click
        AbreFormularioRel()
    End Sub


    Private Sub AnimaisDgv_DoubleClick(ByVal sender As System.Object,
```

```
ByVal e As System.EventArgs)
        PreparaEdicao()
    End Sub

    Private Sub frmCadastro_Animais_Shown(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Me.Shown
        LimparDadosAnimais()

    End Sub

   End Class
```

4.6.4 Lançamentos

O formulário de lançamentos de vendas pode ser acessado tanto pelo menu Financeiro quanto pela barra de botões, que ficam na tela principal do protótipo. Este formulário possui sua classe para acesso às informações contidas no banco de dados. Abaixo, no Quadro 5, é apresentado o código da classe pertencente ao formulário de vendas.

Quadro 5 – Código: Classe do Formulário de Lançamentos de Vendas

```
Imports System.IO
Imports System.Data.SqlClient

Public Class frmCadastro_Vendas
    Dim nCodVendaCorrente As Integer
    Dim sCaminho As String


    Private Sub VendasBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
VendasBindingNavigatorSaveItem.Click
        Me.Validate()
        Me.VendasBindingSource.EndEdit()
        Me.VendasTableAdapter.Update(Me.ShoPetSysDataSet1.Vendas)

    End Sub

    Private Sub CarregaCombo_Produtos()
        Dim oConexao As New ConexaoSQLServer
        Dim oProdutos As New Produtos

        cmbProduto.DataSource = oProdutos.Listar(oConexao.ConexaoSQL)

        cmbProduto.DisplayMember = "nomeProd"

        cmbProduto.ValueMember = "codProd"

    End Sub
```

```
    Private Sub CarregaCombo_Clientes()
        Dim oConexao As New ConexaoSQLServer
        Dim oClientes As New Clientes

        cmbCliente.DataSource = oClientes.Listar(oConexao.ConexaoSQL)

        cmbCliente.DisplayMember = "nomeCli"

        cmbCliente.ValueMember = "codCli"

    End Sub

    Private Sub CarregaCombo_TipoPgtoVenda()
        Dim dt As New DataTable
        Dim dr As DataRow
        dt.Columns.Add("Nome", GetType(System.String))

        dr = dt.NewRow()
        dr("Nome") = "Dinheiro"
        dt.Rows.Add(dr)
        dr = dt.NewRow()
        dr("Nome") = "Cheque"
        dt.Rows.Add(dr)
        dr = dt.NewRow()
        dr("Nome") = "Cheque Pré-Datado"
        dt.Rows.Add(dr)
        dr = dt.NewRow()
        dr("Nome") = "Cartão"
        dt.Rows.Add(dr)

        With cmbTipoPgtoVenda
            .DataSource = dt
            .DisplayMember = "Nome"
            .ValueMember = "Nome"
            .SelectedIndex = 0
        End With

    End Sub

    Private Sub CarregaCombo_StatusVenda()
        Dim dt As New DataTable
        Dim dr As DataRow
        dt.Columns.Add("Nome", GetType(System.String))

        dr = dt.NewRow()
        dr("Nome") = "Recebida"
        dt.Rows.Add(dr)
        dr = dt.NewRow()
        dr("Nome") = "A Receber"
        dt.Rows.Add(dr)

        With cmbStatusVenda
            .DataSource = dt
            .DisplayMember = "Nome"
            .ValueMember = "Nome"
            .SelectedIndex = 0
        End With

    End Sub
```

```
    Private Sub frmCadastro_Vendas_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        'TODO: This line of code loads data into the
'ShoPetSysDataSet1.Vendas' table. You can move, or remove it, as needed.
        Me.VendasTableAdapter.Fill(Me.ShoPetSysDataSet1.Vendas)
        CarregaCombo_Produtos()
        CarregaCombo_Clientes()
        CarregaCombo_TipoPgtoVenda()
        CarregaCombo_StatusVenda()
        CarregaGridVendas()

    End Sub

    Private Sub CarregaGridVendas()
        Dim oConexao As New ConexaoSQLServer
        Dim oVendas As New Vendas

        dgvVendas.DataSource = oVendas.Listar(oConexao.ConexaoSQL)
        FormataGridVendas()
        txtCodVenda.Text = ""
        cmbProduto.SelectedValue = 0
        cmbCliente.SelectedValue = 0
        txtValorVenda.Text = ""
        cmbTipoPgtoVenda.SelectedValue = 0
        cmbStatusVenda.SelectedValue = 0

    End Sub

    Private Sub CadastroVendas()
        Dim oVendas As New Vendas
        Dim oConexao As New ConexaoSQLServer

        'EXECUTANDO METODO SET das PROPRIEDADES DO OBJETO
        oVendas.codProd = cmbProduto.SelectedValue
        oVendas.codCli = cmbCliente.SelectedValue
        oVendas.valorVenda = txtValorVenda.Text
        oVendas.dataVenda = dtDataVenda.Value
        oVendas.tipoPgtoVenda = cmbTipoPgtoVenda.SelectedValue
        oVendas.statusVenda = cmbStatusVenda.SelectedValue
        oVendas.codVenda = nCodVendaCorrente

        oConexao.Abrir()
        If oVendas.Inserir(oConexao.ConexaoSQL) Then
            MsgBox("Venda cadastrada com sucesso!")
        End If
        oConexao.Fechar()
        CarregaGridVendas()

    End Sub

    Private Sub FormataGridVendas()
        dgvVendas.Columns(0).Visible = False
        dgvVendas.Columns(1).Visible = False
        dgvVendas.Columns(2).HeaderText = "Produto"
        dgvVendas.Columns(3).Visible = False
        dgvVendas.Columns(4).HeaderText = "Cliente"
        dgvVendas.Columns(5).HeaderText = "Valor Venda"
        dgvVendas.Columns(5).DefaultCellStyle.Format = "###,##0.00"
```

```
        dgvVendas.Columns(6).HeaderText = "Data"
        dgvVendas.Columns(6).DefaultCellStyle.Format = "dd/MM/yyyy"
        dgvVendas.Columns(7).DefaultCellStyle.Format = "00,00"
        dgvVendas.Columns(7).HeaderText = "Tipo Pagamento"
        dgvVendas.Columns(8).HeaderText = "Status Venda"

    End Sub

    Private Sub btnInserir_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnInserir.Click
        If nCodVendaCorrente = 0 Then
            CadastroVendas()
        Else
            EditaDadosVendas()
        End If
        CarregaGridVendas()

        btnInserir.Text = "Inserir"
    End Sub

    Private Sub EditaDadosVendas()
        Dim oVendas As New Vendas
        Dim oConexao As New ConexaoSQLServer

        oVendas.codProd = cmbProduto.SelectedValue
        oVendas.codCli = cmbCliente.SelectedValue
        oVendas.valorVenda = txtValorVenda.Text
        oVendas.dataVenda = dtDataVenda.Value
        oVendas.tipoPgtoVenda = cmbTipoPgtoVenda.SelectedValue
        oVendas.statusVenda = cmbStatusVenda.SelectedValue
        oVendas.codVenda = nCodVendaCorrente

        oConexao.Abrir()
        If oVendas.Editar(oConexao.ConexaoSQL) Then
            MsgBox("Dado(s) da Venda editado(s) com sucesso!")
        End If
        oConexao.Fechar()
        CarregaGridVendas()

        nCodVendaCorrente = 0
        btnInserir.Text = "Inserir"
    End Sub

    Private Sub PreparaEdicao()
        'Jogar os dados do GRID para as TXTBoxes
        Dim nIndiceLinha As Integer

        nIndiceLinha = dgvVendas.CurrentRow.Index

        With dgvVendas.Rows(nIndiceLinha)
            nCodVendaCorrente = _
                 .Cells(0).Value

            txtCodVenda.Text = _
                 .Cells(0).Value

            cmbProduto.SelectedValue = _
                 .Cells(1).Value
```

```
            cmbCliente.SelectedValue = _
                 .Cells(3).Value

            txtValorVenda.Text = _
                 .Cells(5).Value

            dtDataVenda.Value = _
                 .Cells(6).Value

            cmbTipoPgtoVenda.SelectedValue = _
                 .Cells(7).Value

            cmbStatusVenda.SelectedValue = _
                 .Cells(8).Value

            btnInserir.Text = "Salvar"
        End With
    End Sub

    Private Sub dgvVendas_DoubleClick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles dgvVendas.DoubleClick
        PreparaEdicao()
    End Sub

    Private Sub btnEditar_Click_1(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnEditar.Click
        PreparaEdicao()
    End Sub

    Private Sub SalvarArquivoTexto()
        Dim file As FileStream 'Simboliza o trabalho com arquivos..
        Dim sw As StreamWriter
        Dim oVendas As New Vendas
        Dim oConexao As New ConexaoSQLServer
        Dim sProduto As String
        Dim sCliente As String
        Dim nValor As String
        Dim dtData As String
        Dim sTipoPgtoVenda As String
        Dim sStatusVenda As String
        Dim sResultado As String

        'Representa o .doc, dizendo que poderá ser aberto ou criado
        file = New FileStream(sCaminho, _
                              FileMode.OpenOrCreate, _
                              FileAccess.Write)
        'Objeto que ESCREVE caracteres no FILE (arquivo texto)
        sw = New StreamWriter(file)

        sw.BaseStream.Seek(0, SeekOrigin.End)

        sw.WriteLine("***************VENDAS***************")
        sw.WriteLine()


        For i As Integer = 0 To dgvVendas.Rows.Count - 1
            sProduto = "PRODUTO: " & dgvVendas.Rows(i).Cells(2).Value
            sCliente = "CLIENTE: " & dgvVendas.Rows(i).Cells(4).Value
            nValor = "VALOR VENDA: " & dgvVendas.Rows(i).Cells(5).Value
```

```
            'dgvVendas.Columns(5).DefaultCellStyle.Format = "###,##0.00
            dtData = "DATA: " & dgvVendas.Rows(i).Cells(6).Value
            ' dtData = "DATA: " &
dgvVendas.Rows(i).Cells(6).FormattedValue("dd/mm/yyyy")
            sTipoPgtoVenda = "TIPO PAGAMENTO: " &
dgvVendas.Rows(i).Cells(7).Value
            sStatusVenda = "STATUS VENDA: " &
dgvVendas.Rows(i).Cells(8).Value
            sResultado = sProduto & vbCrLf & sCliente & vbCrLf & nValor &
vbCrLf & dtData & vbCrLf _
            & sTipoPgtoVenda & vbCrLf & sStatusVenda & vbCrLf
            sw.WriteLine(sResultado)
        Next

        'Para finalizar a operação é necessário FECHAR tanto o objeto que
ESCREVE, quanto o DOCUMENTO
        sw.Close()
        file.Close()

        MsgBox("Arquivo Criado Com sucesso!", _
               MsgBoxStyle.Information)

    End Sub

    Private Sub SalvarCaminho()
        'determina o filtro da caixa de diálogo que será apresentada...
        dlgSalvar.Filter = ".doc|*.doc"

        'Apresenta a caixa de diálogo e só salva caso o usuário
pressionar o botao de OK (Salvar)
        If dlgSalvar.ShowDialog() = Windows.Forms.DialogResult.OK Then
            sCaminho = dlgSalvar.FileName
            SalvarArquivoTexto()
        End If
    End Sub

    Private Sub btnExportar_Click_1(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnExportar.Click
        SalvarCaminho()
    End Sub

    Private Sub ExcluiVenda()
        Dim oVendas As New Vendas
        Dim oConexao As New ConexaoSQLServer
        Dim nIndiceLinha As Integer

        nIndiceLinha = dgvVendas.CurrentRow.Index

        oVendas.codVenda = dgvVendas.Rows(nIndiceLinha).Cells(0).Value

        If oVendas.codVenda <= 0 Then Exit Sub

        oConexao.Abrir()
        If oVendas.Excluir(oConexao.ConexaoSQL) Then
            MsgBox("Registro(s) excluído(s)!")
        End If
        oConexao.Fechar()
    End Sub
```

```
    Private Sub btnExcluir_Click_1(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnExcluir.Click
        ExcluiVenda()
        CarregaGridVendas()
    End Sub

    'Chama formulário relatório
    Private Sub AbreFormularioRel()
        Dim oForm As New frmVisual_Relatorio_Vendas

        oForm.Show()
    End Sub

    'Evento que abre tela do formulário
    Private Sub btnRelatorio_Click_1(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnRelatorio.Click
        AbreFormularioRel()
    End Sub


    Private Sub LimparDadosVendas()
        If nCodVendaCorrente = 0 Then
            txtCodVenda.Clear()
            cmbProduto.Text = ""
            cmbCliente.Text = ""
            txtValorVenda.Clear()
            cmbTipoPgtoVenda.Text = ""
            cmbStatusVenda.Text = ""

        End If
    End Sub

    Private Sub frmCadastro_Vendas_Shown(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Me.Shown
        LimparDadosVendas()

    End Sub

    Private Sub CalculaSaldo()
        Dim SOMA As Decimal
        For Each coluna As DataGridViewRow In dgvVendas.Rows
            SOMA = SOMA + coluna.Cells(5).Value
        Next
        lblSaidaSaldo.Text = SOMA.ToString("R$ 0,0.00")

    End Sub

    Private Sub btnCarregaSaldo_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCarregaSaldo.Click
        CalculaSaldo()
    End Sub

End Class
```

A tela de Vendas apresenta as vendas já incluídas no banco de dados e permite ao usuário, a inserção de novas vendas, edição, exclusão e geração de relatórios das vendas realizadas. Abaixo, na Figura 16, é apresentada essa Tela.

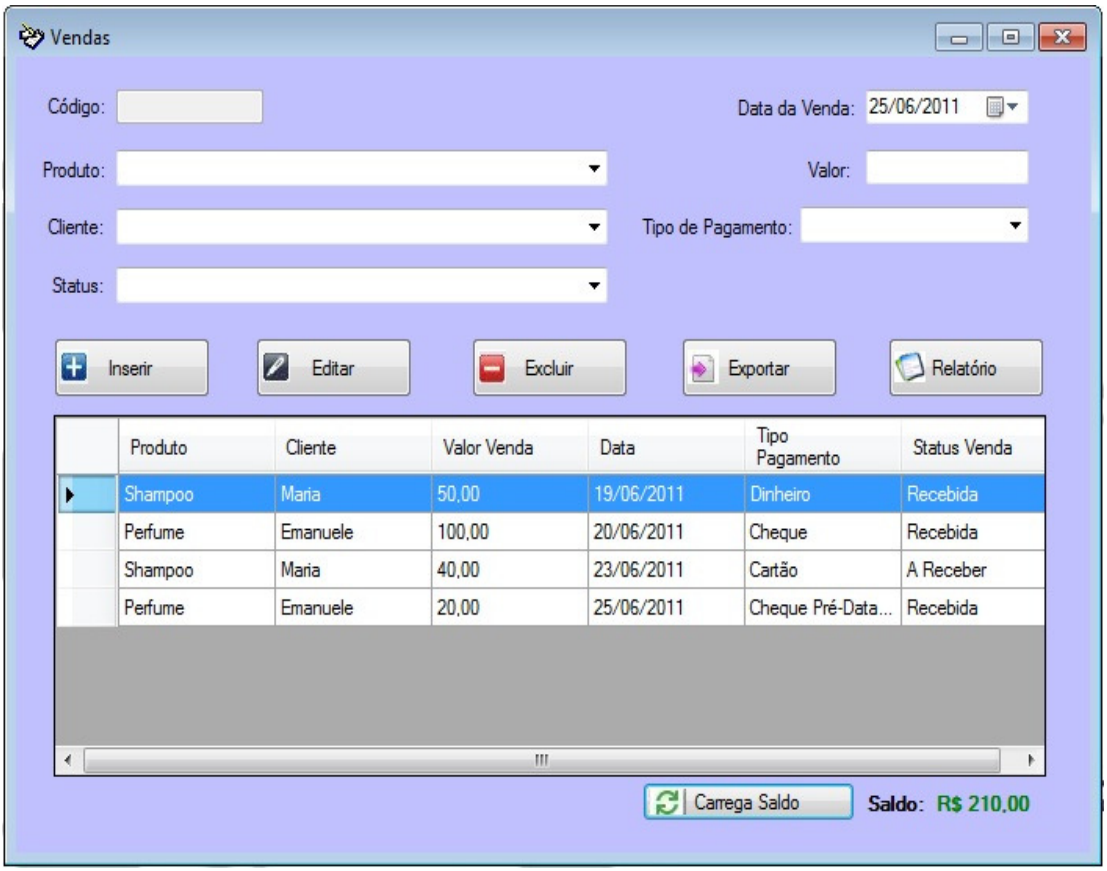

Figura 16 – Tela de Lançamentos das Vendas

4.6.5 Relatórios

O protótipo permite a geração de alguns relatórios. A visualização dos mesmos ocorre com o clique no botão “Relatório” das telas ou no menu relatórios. Abaixo, no Quadro 6, é apresentado o código da classe pertencente ao formulário de geração do relatório dos dados da tabela Cidades do banco de dados.

Quadro 6 – Código: Classe do Formulário de Relatórios

```
Public Class frmVisual_Relatorio_Cidades
    Dim dt As New DataTable
    Dim sCaminho As String

    Private Sub NumericUpDown1_ValueChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles NroZoom.ValueChanged
```

```
        Dim zoomno As Integer

        zoomno = Convert.ToInt32(NroZoom.Value)

        Imprimir.Size = New Size(zoomno * 10, zoomno * 10)

        Imprimir.AutoZoom = True

    End Sub

    Private Sub ImprimirDocumentos_PrintPage(ByVal sender As
System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs)
Handles ImprimirDocumentos.PrintPage
        Dim dt As New DataTable
        Dim oCN As New ConexaoSQLServer
        Dim oCidades As New Cidades
        Dim i As Integer = 1

        'Preenchendo o DATA TABLE
        dt = oCidades.Listar(oCN.ConexaoSQL)

        e.Graphics.DrawString("Dados de Cidades", New Font("arial", 40,
FontStyle.Regular), Brushes.Black, 160, 80)
        For Each dr As DataRow In dt.Rows

            e.Graphics.DrawString("Código : ", New Font("arial", 15,
FontStyle.Regular), Brushes.Black, 20, 200 + i)
            e.Graphics.DrawString(dr("codCid"), New Font("arial", 15,
FontStyle.Regular), Brushes.Black, 220, 200 + i)
            e.Graphics.DrawString("Nome : ", New Font("arial", 15,
FontStyle.Regular), Brushes.Black, 20, 230 + i)
            e.Graphics.DrawString(dr("nomeCid"), New Font("arial", 15,
FontStyle.Regular), Brushes.Black, 220, 230 + i)
            e.Graphics.DrawString("Estado : ", New Font("arial", 15,
FontStyle.Regular), Brushes.Black, 20, 260 + i)
            e.Graphics.DrawString(dr("UF_Cid"), New Font("arial", 15,
FontStyle.Regular), Brushes.Black, 220, 260 + i)

            i = i + 120
        Next

    End Sub

    Private Sub btnImprimir_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnImprimir.Click
        ImprimirDocumentos.Print()
    End Sub
End Class
```

Na tela de Relatório o usuário tem as opções de aumentar o zoom de visualização das informações geradas, opção de impressão do relatório e de saída da tela, em seguida é apresentada a tela do formulário de relatório (Figura 17).

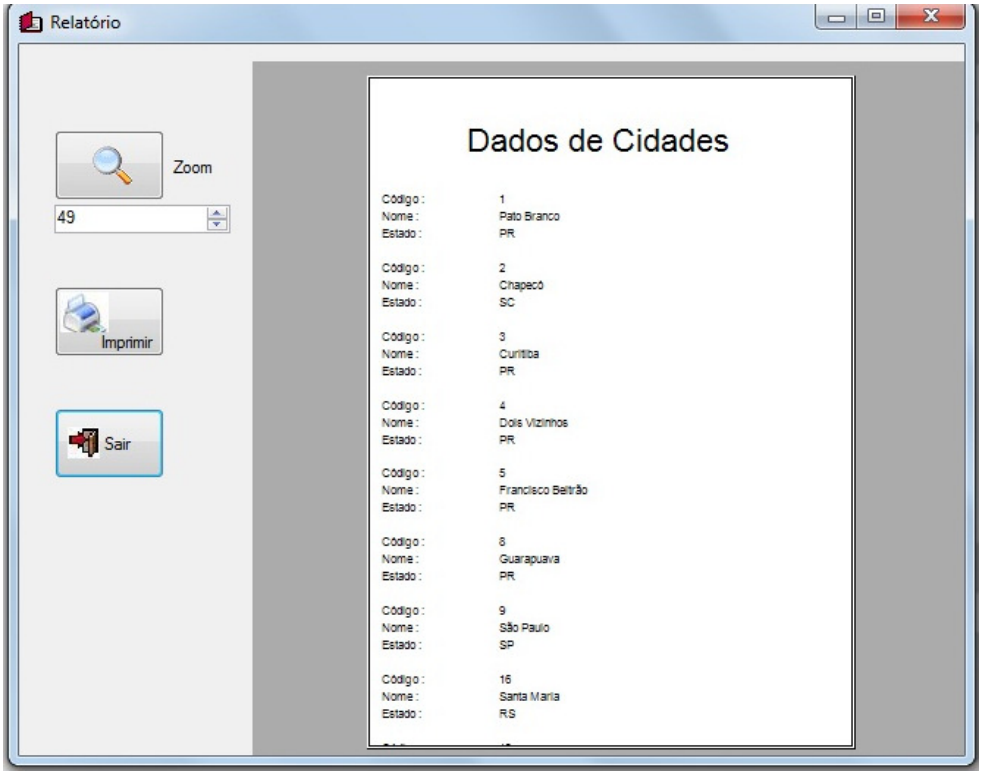

Figura 17 – Tela de Exemplo de Relatório

4.6.6 Utilização do Protótipo

Ao acessar a tela principal do protótipo o usuário primeiramente deve efetuar o cadastro de Espécies, que se encontra no menu Cadastro da tela Principal, em seguida efetuar o cadastro de Raças, Produtos, Vacinas, Cidades, Clientes, Animais, Fornecedores, Funcionários, Pedidos e Serviços. As telas possuem estrutura padrão, onde o usuário pode inserir, editar, exportar dados para arquivo em formato .doc, excluir e gerar relatórios.

5 CONSIDERAÇÕES FINAIS

No capítulo a seguir apresentam-se as dificuldades, vantagens e desvantagens observada durante a análise, documentação e desenvolvimento do protótipo de Gerenciamento para *Pet Shops*.

5.1 Dificuldades

A grande dificuldade na pesquisa e documentação do trabalho foi conseguir levantar os requisitos por meio da pesquisa exploratória, sendo que poucas pessoas mostraram interesse para estarem respondendo o questionário e algumas perguntas.

No desenvolvimento do protótipo a maior dificuldade foi na criação de algumas telas do sistema, pois foi utilizada a versão *Express* do *Visual Studio,* que não oferece muitas opções.

5.2 Vantagens do Protótipo

As principais vantagens do protótipo desenvolvido referem-se à possibilidade de realização de cadastros de clientes, animais, fornecedores, produtos, entre outros, de maneira simples e rápida. As informações armazenadas na base de dados poderão ser consultadas por meio de telas criadas e por intermédio da exportação dos dados para arquivos em formato .doc. O protótipo desenvolvido permite a inclusão de informações das vendas realizadas e das contas referentes aos serviços realizados.

Também permite uma melhor organização no armazenamento dos dados da empresa de *Pet Shop* como as informações de clientes, animais, produtos e lançamentos de vendas, evitando o uso de fichas de papel para salvar essas informações. Além disso, a utilização do protótipo desenvolvido permite um processamento informatizado das informações dos cadastros e lançamentos.

Outras vantagens são a maior facilidade ao acesso às informações cadastradas. Os pedidos de produtos podem ser feitos e acompanhados conforme

seu *status*. Todas as vantagens oferecidas visam agilidade e qualidade no atendimento ao cliente.

5.3 Desvantagens do Protótipo

O protótipo é indicado para pequenas lojas de *Pet Shop*. Para realização de processos básicos como cadastros, lançamentos de vendas e contas dos serviços realizados, suprindo as necessidades básicas do empreendimento. No entanto, o protótipo é bastante restrito, não possui as funcionalidades fiscal e contábil, não podendo ser utilizado por lojas que necessitem emitir nota fiscal dos produtos vendidos.

Os lançamentos e vendas feitos no protótipo permitem ainda a realização de pagamento em dinheiro, cheque, cheque pré-datado, cartão, mas com poucas informações de processamento dos mesmos, tendo como objetivo somente o informativo do tipo de pagamento realizado pelo cliente.

5.4 Soluções para a Documentação e Desenvolvimento do Protótipo

A solução para o desenvolvimento do protótipo e término da pesquisa foi buscar informações em livros, na internet, além de consultar orientadora e colegas de trabalho.

A partir disso foi possível fazer a modelagem baseada em UML que permite o desenvolvimento de sistemas que atendem as necessidades do usuário, permitindo futuras implementações e alterações no protótipo.

6 CONCLUSÃO

Este projeto foi desenvolvido com objetivo de oferecer um sistema simples e de baixo custo, permitindo informatizar as principais funções de uma loja de *Pet Shop* de pequeno porte.

O intuito do sistema é controlar internamente as operações, para melhorar o processo da empresa, considerando a quantidade de serviços que a empresa pode oferecer aos seus clientes e podendo controlar a quantidade de agendamentos no dia. Facilitando a oferta de serviços, podendo melhor atender os clientes que procuram os serviços oferecidos por empresas de *Pet Shops*.

Por intermédio das pesquisas realizadas, e a partir delas a realização da modelagem e desenvolvimento do protótipo, foi possível implementar uma ferramenta que atenda os pequenos empreendedores deste ramo.

Desta forma, o protótipo de gerenciamento desenvolvido é empregado para controle interno somente, tornando mais fácil a realização dos processos executados na empresa, e o entendimento dos envolvidos.

Para efetuar a modelagem e a implementação do protótipo, foram utilizadas ferramentas eficientes que forneceram uma interação fácil do usuário com o sistema. Ferramentas com uma aceitação de mercado muito grande, produzidas por empresas altamente conceituadas mundialmente.

No processo de desenvolvimento deste trabalho ocorreu um grande enriquecimento dos conhecimentos acadêmicos, principalmente em relação à modelagem utilizando a UML.

Enfim, realizar o desenvolvimento deste projeto permitiu adquirir maior conhecimento em relação à orientação a objetos, principalmente sobre utilização de classes, emprego de polimorfismo, herança e encapsulamento, que auxiliaram bastante na reutilização de código e telas.

De um modo geral a realização deste trabalho, permitiu ampliar conhecimentos adquiridos durante o decorrer do curso de graduação.

8 PERSPECTIVAS FUTURAS

Pretende-se futuramente dar continuidade ao desenvolvimento do protótipo, contemplando os seguintes itens considerados importantes:

- Desenvolvimento do controle de estoque de produtos.
- Desenvolvimento de telas de gerenciamento/controle de cheques. Devido ao grande nível de informações que ela necessita não foi possível concluí-la a tempo e de forma completa.
- Desenvolvimento dos relatórios detalhando os diversos resultados.
- Desenvolvimento dos pagamentos diversos, como de contas a pagar e controle de caixa da empresa.
- Melhoria no processo de pedidos e vendas/compras conforme solicitações dos usuários.
- Melhoria nas telas de cadastros
- Implementações referentes à impressão de cupom fiscal, recibos, envelopes e etiquetas.

REFERÊNCIAS

BAHIA. **Rede Bahia de Televisão -. O mercado de Pet Shop: Uma atividade econômica das mais relevantes**. 2005. Disponível em: < http://ibahia.globo.com/tvbahia/comercial/pdf/pet.pdf > Acesso em: 10/04/2011.

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

BOLÍVAR. **Criação de um sistema para Controle de Viagens no Delphi com Firebird**. 2009. Disponível em: < http://bolivarbutzke.blogspot.com/2009/10/sistema-controle-de-viagens-no-delphi.html > Acesso em: 10/04/2011.

BOOCH, Grady. **UML**: **guia do usuário**. Rio de Janeiro - RJ: Campus, 2000.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário.** Rio de Janeiro: Campus, 2000, 7ª edição.

BORTOLIN JÚNIOR, Sérgio Antônio Martini. **Ferramentas Case**. 2005. Disponível em: < http://www.al.urcamp.tche.br/infocamp/edicoes/nov05/Ferramentas%20Case.pdf > Acesso em: 10/04/2011.

CANTÚ, Marco. **Dominando o Delphi 7: a bíblia**. São Paulo: Makron Books, 2003.

COFFMAN, Gayle. **SQL server 7: completo e total guia de referência**. São Paulo: Makron Books, 2000.

CRUZ, Tadeu. **Sistemas de Informações Gerenciais**. São Paulo: Atlas, 1998.

FUX, David. **CASE Studio 2.** Disponível em:<http://ultradownloads.uol.com.br/download/CASE-Studio/ > Acesso em: 20 mai. 2011.

LIMA, Adilson da Silva. **UML 2.0: Do requisito a solução**. Editora Érica, 2005.

MACORATTI, José Carlos. **UML – Conceitos Básicos II.** Disponível em:< http://www.macoratti.net/vb_uml2.htm > Acesso em: 15 mai. 2011.

MAIA, Hugo. **A UML.** Disponível em:< http://hugohabbema.blogspot.com/2009/08/uml.html/ > Acesso em: 15 mai. 2011.

MARTIN, James. **Análise e projeto orientados a objeto.** São Paulo - SP: Makron Books, 1995.

PENDER, Tom. **UML a Bíblia.** Rio de Janeiro - RJ: Campus Books, 2004.

PEREIRA, Vitor Emanuel. **O Guia Prático do Visual Basic 2005 Express.** Lisboa – Portugal: Centro Atlântico, 2006.

PRESSMAN, Roger S. **Engenharia de Software.** São Paulo - SP: Makron, 1995.

SILVA, Ricardo Pereira. **UML2 em Modelagem Orientada a Objetos.** Florianópolis: Visual Books, 2007.

SOUZA, Vinicius Lourenço de. **Artigo – Artigo Engenharia de Software 2 – Desenvolvimento de Software Dirigido por Caso de Uso.** 2009. Disponível em:< http://www.devmedia.com.br/space.asp?id=176186/ > Acesso em: 15 mai. 2011.

TONSIG, Sergio Luiz. **Análise e Projeto de Sistemas. (2000).** Disponível em: <http://www.apostilando.com/download.php?cod=410&categoria=L%C3%B3gica%20de%20Programa%C3%A7%C3%A3o > Acesso em: 10/04/2011.

WAZLAWICK, Raul Sidnei. **Análise e Projeto de Sistemas de Informação Orientados a Objetos.** 5 ed. Rio de Janeiro: RJ: Elsevier, 2004.

APÊNDICES

APÊNDICE A

QUESTIONÁRIO DE COLETA DE REQUISITOS

1. Você possui computador em seu estabelecimento? E o utiliza para trabalho?
   _____
   _____
   _____

2. Há interesse em utilizar um controle informatizado dos processos?
   _____
   _____
   _____

3. Quais os principais Problemas no gerenciamento de sua empresa?
   _____
   _____
   _____
   _____
   _____

4. Possui algum software de gerenciamento em sua empresa?
   _____
   _____
   _____
   _____

5. Está satisfeito com o modo que gerencia as contas a receber e contas a pagar de seu estabelecimento? Por quê?
   _____
   _____
   _____

6. Quais tipos de pagamentos são realizados na sua empresa?
   [ ] Dinheiro
   [ ] Á Prazo
   [ ] Cheque a vista
   [ ] Cheque Pré-datado
   [ ] Cartão crédito
   [ ] Cartão Débito
   [ ] Outros,
   cite:_____
   _____
   _____
   _____

7. Quais outras informações da sua empresa você gostaria de ter informatizadas através de um sistema?

_____
_____
_____
_____
_____

Obrigado pela sua colaboração!