CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS – CAMPUS V

PetSystem – Desenvolvimento de um Sistema de Gerenciamento para Clínicas Veterinárias

Camila Lorrainy de Sousa Pereira

Daiane Conceição Gonçalves

Maria Vitória Pereira Vaz

Stéfano de Oliveira Mano

Divinópolis - MG 2014

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS – CAMPUS V

PetSystem -

Desenvolvimento de um Sistema de Gerenciamento para Clínicas Veterinárias

Camila Lorrainy de Sousa Pereira

Daiane Conceição Gonçalves

Maria Vitória Pereira Vaz

Stéfano de Oliveira Mano

Orientador: Jeneffer Ferreira Ribeiro

Co-orientador: Nestor Dias Oliveira Volpini

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Informática do Centro Federal de Educação Tecnológica de Minas Gerais – Campus V como requisito parcial para a obtenção do título de Técnico em Informática.

Divinópolis - MG

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS – CAMPUS V

Trabalho de Conclusão de Curso julgado adequado para obtenção do	título de
Técnico em Informática e aprovado pela banca composta pelos	seguintes
professores.	
Prof. Jeneffer Ferreira Ribeiro - CEFET-MG (Orientador)	_
	_
Prof. Nestor Dias Oliveira Volpini - CEFET-MG (Co-orientador)	
	_
Prof. Willyan Michel Ferreira - CEFET-MG	
Prof. Luís Augusto Mattos Mendes	
Coordenador do Curso Técnico em Informática	

Divinópolis, 11 de Dezembro de 2014.

Data de aprovação:

RESUMO

O PetSystem, sistema desenvolvido, é um software para desktop que atuará em

clínicas veterinárias de pequeno e médio porte substituindo as fichas de registros e históricos

médicos manuscritos dos animais, além de oferecer controle de funcionários e produtos. A

modelagem foi realizada por meio de diagramas, como o de caso de uso e o de classes,

modelo de entidade e relacionamento, dicionário de dados e entrevistas para levantamento de

requisitos. As linguagens utilizadas na implementação do programa foram Java, para o

desenvolvimento da aplicação, e SQL para a criação e manipulação do banco de dados. Uma

pesquisa foi realizada com veterinários e demais profissionais da área em clínicas locais,

tendo como resultado a percepção da necessidade da informatização dos processos

administrativos nesses estabelecimentos. Assim, o gerenciamento será mais eficiente, uma vez

que o sistema oferecerá padronização nos cadastros da empresa.

Palavras-chave: software; gerenciamento; clínicas veterinárias.

SUMÁRIO

1. Introdu	1ção	7
1.1.	Definição da Empresa	8
1.2.	Definição do Escopo	8
1.3.	Definição das Funcionalidades	8
1.4.	Referencial Teórico	
1.4.1.	Informatização de Clínicas Veterinárias	
1.4.2.	Entrevista	
1.4.3.	Softwares Veterinários	
1.4.3.	·	
1.4.3.		
1.4.3		
1.4.3.		
1.4.3. 1.4.4.	.5. Observações	
1.4.5.	Linguagens	
1.4.5.		
1.4.5.		
1.4.5.		
1.4.6.	PMBOK	14
1.4.7.	Padrão MVC – DAO	
2. Projeto	Conceitual	16
2.1.	Diagrama de Contexto UML	16
2.2.	Documentação dos Atores	17
2.2.1.	Ator 01 – Atendente	17
2.2.2.	Ator 02 – Veterinário	17
2.2.3.	Ator 03 – Gerente	17
2.3.	Descrição Detalhada das Funcionalidades	17
2.3.1.	Login	18
2.3.2.	Tela Inicial	18
2.3.3.	Cadastros	19
2.3.3.	.1. Proprietário	20
2.3.3.		
2.3.3.		
2.3.3. 2.3.3.	3	
2.3.3.		
2.3.3.		
2.3.4.	Pesquisar	27
2.3.4.	1	
2.3.4.	.2. Animal	30

	2.3.4.3. Produtos	37
	2.3.4.4. Funcionários	39
	2.3.4.5. Vacinas	
2	2.3.4.6. Fornecedores	
2	3.5. Agendar Horários	
	2.3.5.1. Inserir	
2	3.6. Gerar Relatórios	
	3.7. Ajuda	
	rojeto Físico	
3.1.	DER – Diagrama de Entidade e Relacionamento	51
3.2.	Diagrama de Classes	
3.3.	Dicionário de Dados	53
4. R	esultados	55
5. C	onclusão	56
6. C	ronograma	57
7. R	eferências	58
8. A	nexos	60
8.1.	Anexo A: DTR – Diagrama das Tabelas Relacionais	60
8.2.	Anexo B: Diagramas de Sequência	61
8.	2.1. Cadastros	61
8.	2.2. Consultas	65
8.	2.3. Edições	69
8.	2.4. Gerar Relatório	72
8.	2.5. Login do Usuário	73
8.	2.6. Remoções	73
8.3.	Anexo C: Classes detalhadas	
8.	3.1. View	77
	3.2. Model	
	3.3. Controller	
	3.4. DAO	
	3.5. Exception	
	3.6. Componentes	
	3.7. Default	
J.		

1. Introdução

A tecnologia consegue facilitar diversas necessidades presentes no dia a dia. Entretanto, algumas áreas ainda não adotaram a informatização, como nas clínicas veterinárias da região central de Divinópolis-MG, as quais utilizam fichas de cadastros e fichas médicas manuscritas. O projeto oferece um *software* que facilita o gerenciamento nesse ambiente.

O sistema desenvolvido denomina-se PetSystem, cujo logotipo pode ser visto na Figura 1, e é capaz de substituir trabalhos manuscritos exercidos na área de aplicação, de modo que não apresente carência nos recursos e atenda de maneira segura e prática às necessidades existentes.



FIGURA 1 - Logotipo do software PetSystem

1.1. Definição da Empresa

O sistema elaborado poderá ser implantado em clínicas veterinárias de pequeno ou médio porte, possibilitando o gerenciamento das mesmas de forma fácil e eficaz. Nesse ambiente são atendidos vários animais, sendo então examinados e medicados por um veterinário. Dessa forma, geram-se gastos com remédios, internações, vacinas e outros. Portanto, essa empresa precisa realizar cadastros, fazer diagnósticos, gerenciar produtos e funcionários, sendo que estas funcionalidades estão presentes no sistema.

1.2. Definição do Escopo

O PetSystem opera no ambiente *desktop* e é destinado à facilitar o armazenamento seguro de dados na sua área de atuação. Para isso, é necessária a autenticação do usuário, que consiste no preenchimento dos campos *login* e senha. Em seguida, serão atribuídas as permissões conforme a hierarquia do funcionário na empresa. No que diz respeito ao atendente, poderá agendar e excluir um horário de atendimento para um animal, além de cadastrar, buscar, editar e excluir os dados dos proprietários, animais, produtos, vacinas e fornecedores. Já ao veterinário é permitido, além das citadas, criar e consultar diagnósticos, assim como registrar produtos e vacinas aplicados durante o atendimento. Por fim, ao gerente são exibidas todas as funcionalidades, sendo as restantes: cadastrar, editar e excluir funcionários, além de gerar relatórios.

1.3. Definição das Funcionalidades

A seguir, serão descritas as funcionalidades necessárias para atender as demandas percebidas nas clínicas veterinárias:

- Inserção, exclusão e alteração de registros nas principais tabelas do banco de dados, sendo elas proprietário, animal, produto, funcionário, vacinas e fornecedor.
 - Realização de consultas aos dados das tabelas citadas anteriormente.

- Manutenção da agenda: relacionar horários, animais e serviços a serem prestados, assim como exclusão de agendamentos previamente realizados.
- Geração de relatórios: emitir a relação dos produtos em baixa e em alta com base em quantidades estabelecidas anteriormente, além de exibir os produtos e contatos de seus fornecedores.

1.4. Referencial Teórico

1.4.1. Informatização de Clínicas Veterinárias

Uma clínica veterinária necessita de uma gestão adequada para obter sucesso. Segundo a pesquisa realizada pela Sebrae – SP, 31% das empresas fracassam no primeiro ano de operação e 60% não atingem 5 anos de vida. Fatores em comum entre as mesmas, que se encontravam deficientes eram: planejamento prévio ou estruturação do negócio e gestão (MATOS; MELCHOR, 2005, p. 08). Considerando-se este contexto, o sistema em questão, que foi denominado PetSystem, tem por objetivo promover um gerenciamento que evite a deficiência dos fatores anteriormente mencionados.

A informática tem evoluído seu papel nas organizações, assumindo um alto grau de importância na empresa, pois independentemente do porte, possibilita um bom desempenho interno, que por sua vez garante a competitividade (SACILOTTI, 2011, p. 62).

"Ainda como máquina de armazenamento de dados o computador vem substituindo a utilização do papel e das fichas manuscritas utilizadas como arquivos nas clínicas veterinárias promovendo uma melhor organização dos setores clínicos de atendimento, tosa, cirurgia e internamento." (FÉLIX; MARTINS; MOURA, p.02)

Diante do cenário competitivo em que se encontram as empresas, no qual há um crescente avanço tecnológico, as mesmas precisam adotar uma postura favorável à inovação.

"A inovação pode se dar pelo emprego de uma tecnologia totalmente nova para a empresa e para o mercado, como também pela introdução de tecnologia utilizada em outro campo de atividade, porém, nova no campo de atuação da empresa. As empresas que possuem habilidade para aproveitar a tecnologia disponível e dar-lhes novas aplicações podem, por esta característica, alcançar vantagens competitivas." (DEITOS, 2002)

As clínicas veterinárias se enquadram nesse cenário, portanto, a implantação de um sistema de gerenciamento neste ambiente, trata-se de uma medida que lhes fornece vantagem competitiva.

1.4.2. Entrevista

Um estudo de viabilidade, desenvolvido através de uma entrevista realizada na Clínica Veterinária Animax¹, aborda problemas provenientes do gerenciamento não informatizado (MARTINS; RODRIGUES; SILVA; SOUZA, 2011). O principal deles é a dificuldade de controlar as vacinas aplicadas nos animais. Percebeu-se que devido à falta de aviso prévio aos clientes, muitos se esqueciam das datas de renovação das vacinas, deixando o animal propenso a diversas doenças. Outra complicação relatada foi o armazenamento dos dados dos clientes e seus animais em fichas, pois para consultá-las é necessária uma busca seguindo a ordem alfabética, entretanto, nem sempre estas estavam organizadas corretamente. Assim as fichas precisam ser reorganizadas de tempos em tempos, porém trata-se de um processo trabalhoso. Além disso, com o tempo é necessário criar uma nova ficha para registrar o histórico das consultas, por falta de espaço na atual.

A proprietária da clínica afirmou que havia utilizado um programa de gerenciamento, contudo este apresentava funcionalidades demais, dificultando que os objetivos fossem atingidos. Dessa forma, ela abandonou o *software* e voltou a registrar tudo no papel.

1.4.3. Softwares Veterinários

As sessões a seguir abordam quatro sistemas de gerenciamento de clínicas veterinárias para *desktop*, que se encontram disponíveis no mercado.

1.4.3.1. Doctor Vet

Doctor Vet² tem uma interface básica e suas principais funcionalidades são: cadastro e busca de clientes e animais; registro e venda de produtos; gerenciamento de finanças; envio

¹ A Clínica Veterinária Animax fica situada próxima ao terminal de Jardim Atlântico, em Olinda, Pernambuco.

² Maiores informações em http://www.xionce.com/doctorvet/pt/index.php

de *emails*; consulta de relatórios; agenda; ajustes para o sistema; e uma enciclopédia de armazenamento de informações.

Nesse sistema foi possível observar os seguintes contratempos:

- Para realizar o *login*, o usuário precisa digitar seu nome e apertar *enter* e fazer o mesmo com a senha, sendo este um processo que não é bem esclarecido a ele;
- A interface é de difícil manuseio e pouco intuitiva, devido à disposição e organização dos ícones e campos;
- Alguns serviços, vacinas e produtos vêm previamente cadastrados, de forma que se o usuário não possuir os mesmos em sua clínica, deverá removê-los do sistema;
- A página de movimentos, referente ao gerenciamento de produtos, inicia junto com a página principal e se o usuário sair da mesma, deve logar novamente no programa para ter acesso a ela.

1.4.3.2. NSVet

NSVet³ apresenta uma interface limpa, elegante e fácil de manusear. Suas funções mais importantes são cadastro e busca de clientes, animais e produtos; agendamentos; consultas; pré-vendas; caixa; finanças; estoque; e área gerencial.

Os inconvenientes encontrados no sistema foram:

- O agendamento pode ser realizado de duas maneiras: completo, se o cliente já estiver cadastrado, ou simplificado, no qual são solicitados dados sobre o animal e o cliente, que não podem ser reaproveitados para gerar um cadastro.
- Vários usuários, produtos, cadastros de clientes e animais, fichas clínicas, contas a pagar e a receber, entre outros, vêm pré-registrados, de forma que o usuário precisará excluir do sistema todos os dados que não se referem à sua clínica.

1.4.3.3. QVet

QVet⁴ disponibiliza a seus usuários as operações de: cadastros e buscas de clientes, animais, produtos e fornecedores; gestão de caixa; pedidos de compras; vendas; hospitalização; agenda; controle de serviços de banho e tosa; e *marketing* veterinário.

³ Maiores informações em http://nsconsultoria.com.br/downloads/

Nesse sistema foram observados os seguintes empecilhos:

- Apresenta uma interface carregada, cheia de ícones e opções.
- O usuário encontra dificuldades para manipular o sistema sendo necessário muito tempo de treinamento para entender todo o seu funcionamento.
- Possui muitas funcionalidades, sendo algumas delas pouco relevantes.
- As opções de cadastros de clientes, animais e produtos só estão disponíveis dentro do ícone de busca, ou seja, não existe um campo separado para esta operação.
- Vários cadastros de clientes, animais e produtos, vêm pré-registrados, de forma que o usuário precisará excluir do sistema todos os dados que não se referem à sua clínica.

1.4.3.4. VetSoft

VetSoft⁵ possui uma interface bem elaborada e suas funcionalidades estão relacionadas com ficha do proprietário, ficha do animal, histórico clínico, financeiro, farmácia/materiais, gestão de estoque, banho e tosa, e resultado de exames.

Percebeu-se que esse sistema possui muitos detalhes e funcionalidades que são pouco necessários, exigindo do usuário mais tempo para aprendizado.

1.4.3.5. Observações

Os inconvenientes mais presentes nos *softwares* analisados são a interface complicada, que dificulta a manipulação do sistema pelo usuário; e as informações previamente inseridas, pois precisam ser removidas quando não se referem à clínica onde o programa foi implantado.

1.4.4. PetSystem em relação aos outros softwares

O sistema PetSystem focará suas funcionalidades no gerenciamento das atividades prestadas por uma clínica veterinária e nos dados que esta precisa armazenar, e não abordará a parte financeira. A interface do programa será simples, limpa e intuitiva, para que o usuário aprenda facilmente a utilizá-lo.

⁵ Maiores informações em http://www.softwareveterinario.com.br/conheca/

⁴ Maiores informações em http://qvet.com.br/que-es-qvet.aspx

1.4.5. Linguagens

A seguir, serão abordadas as linguagens utilizadas para desenvolver o *software* PetSystem.

1.4.5.1. Java

A tecnologia Java é uma linguagem de programação e plataforma computacional que permite a criação de aplicações e serviços altamente personalizáveis. Aplicações móveis e incorporadas, jogos, conteúdo baseado na *web* e *softwares* corporativos adotaram-na como padrão em seu desenvolvimento e distribuição. As principais características dessa linguagem são sua sintaxe simples, permitindo ao usuário programar de forma clara e orientada a objetos; segurança; robustez, tendo por finalidade a criação de programas confiáveis, livres de situações de erro; e universalidade (Java, 2014).

Todo o sistema será implementado por meio do Java, pois os recursos oferecidos pelo mesmo permitirão construir um produto que atenda de maneira eficiente às necessidades de uma clínica veterinária.

1.4.5.2. SQL

A Structured Query Language (Linguagem Estruturada de Pesquisa) é uma forma de consulta e manipulação de dados, desenvolvida especialmente para o ambiente relacional. Devido à sua simplicidade e facilidade de uso, se tornou padrão nos sistemas de gerenciamento de banco de dados. As principais vantagens em se optar pela SQL são a portabilidade entre computadores, o fato de ser formada por um simples conjunto de comandos em inglês, oferecendo um entendimento rápido e fácil, e a consulta interativa, que provê um acesso rápido aos dados (ARANTES). A linguagem SQL será utilizada para criar e manipular o banco de dados do sistema, no qual serão armazenadas as informações antes registradas em papel nas clínicas veterinárias.

1.4.5.3. UML

A *Unified Modeling Language* (Linguagem de Modelagem Unificada) é uma linguagem que pode ser utilizada para analisar um processo quanto aos seus elementos ontológicos e o comportamento que estes apresentam. Através dela é possível modelar todas as etapas de um projeto de desenvolvimento de *software* e produzir os artefatos necessários para documentar essas fases. A linguagem UML é formal e visual, baseada em diferentes tipos de diagramas, sendo alguns deles diagrama de caso de uso, de classes e de sequência (GUDWIN, 2010).

Para auxiliar na elaboração do PetSystem, serão utilizados diagramas UML, pois estes especificam o projeto e esclarecem o funcionamento do sistema que está sendo projetado.

1.4.6. PMBOK

O livro A *Guide to the Project Management Body of Knowledge (PMBOK Guide)* fornece diretrizes para o gerenciamento de projetos. Consiste em um manual com uma padronização que define o gerenciamento e os conceitos relacionados, além de descrever o ciclo de vida do projeto e seus processos. As práticas de Gerenciamento de Projetos, abordadas no PMBOK, se aplicam a qualquer projeto, aumentando a probabilidade de sucesso do mesmo e reduzindo a ocorrência de surpresas negativas (Guia PMBOK).

Dessa forma, as ações necessárias ao longo do projeto, serão orientadas seguindo os conceitos do PMBOK, principalmente a parte de manipulação da Estrutura Analítica de Projeto (EAP), que regulamenta todas as etapas de desenvolvimento.

1.4.7. Padrão MVC – DAO

A elaboração do código do *software* PetSystem se deu seguindo os parâmetros dos padrões MVC (*Model View Controller*) e DAO (*Data Access Objects*). O primeiro é constituído por três conjuntos, sendo eles: *Model*, responsável pelo armazenamento dos dados; *View*, no qual são tratados os componentes da interface que interage com o usuário; e *Controller*, que coordena os componentes de processamento. O MVC foi adotado com o objetivo de facilitar o desenvolvimento, a manutenção e o reaproveitamento de código. O modelo DAO permite a construção de objetos com base nas informações do banco de dados.

O principal benefício de se optar pelo uso deste padrão é que o mesmo proporciona a separação do acesso ao banco do restante do código (CAETANO, 2012).

2. Projeto Conceitual

Nesta sessão serão apresentados os documentos conceituais desenvolvidos, a fim de ilustrar a arquitetura empregada e facilitar o entendimento do contexto do projeto.

2.1. Diagrama de Contexto UML

A seguir será apresentado na Figura 2 o diagrama de contexto UML, no qual estão contidos os atores e as permissões executadas por eles.

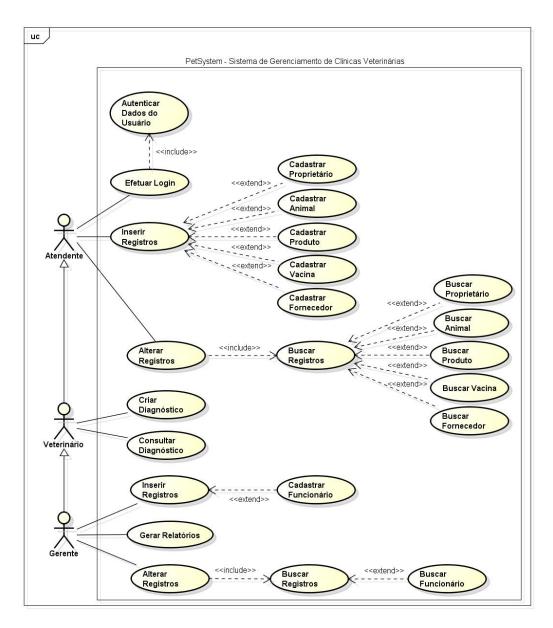


FIGURA 2 - Diagrama de Caso de Uso do PetSystem

2.2. Documentação dos Atores

2.2.1. Ator **01** – Atendente

O(a) secretário(a) da clínica representa o atendente, portanto, existem restrições de acesso para este ator. Assim, possui apenas algumas permissões, como cadastros, edições, exclusões e buscas de alguns módulos, ou seja, tarefas atribuídas aos recepcionistas em geral.

2.2.2. Ator 02 – Veterinário

O(a) veterinário(a) possui todas as permissões do atendente. Além disso, pode manipular dados coerentes à sua área de profissão, sendo eles criação e alteração dos diagnósticos relativos ao atendimento realizado.

2.2.3. Ator 03 – Gerente

Este ator representa o administrador do sistema, ou seja, detém todas as funcionalidades oferecidas. Somando às permissões já citadas, pode realizar cadastros, alterações, exclusões e buscas de funcionários, além da geração de relatórios, ou seja, partes referentes ao gerenciamento das questões internas da empresa.

2.3. Descrição Detalhada das Funcionalidades

Nesta seção, serão apresentadas as funcionalidades do sistema PetSystem de maneira detalhada. Consistem basicamente em, cadastros, buscas, edições e exclusões nas tabelas principais. Serão apresentadas todas as funções do sistema, entretanto essas são limitadas de acordo com o usuário logado.

2.3.1. Login

Na tela de *login*, que está ilustrada na Figura 3, existem apenas os componentes de informar *login* e senha. Optou-se por mudar a forma da janela de modo que esta fique mais atrativa.

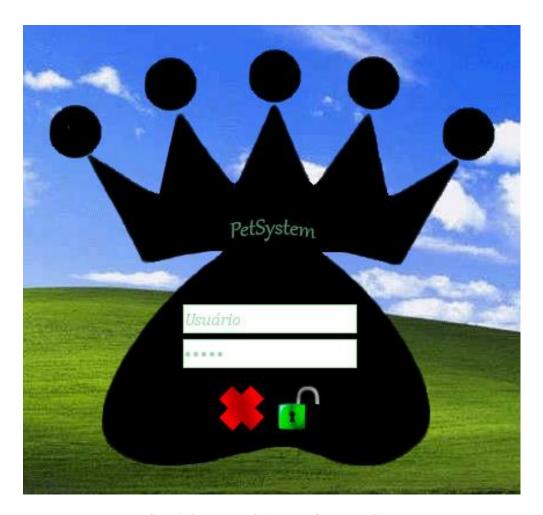


FIGURA 3 - Tela de login do software PetSystem

2.3.2. Tela Inicial

Na tela inicial, como pode ser visto na Figura 4, estão presentes os menus, o ícone e nome do usuário logado, além da imagem de apresentação do sistema, que consiste em uma animação de uma bolha contendo o nome do *software*.

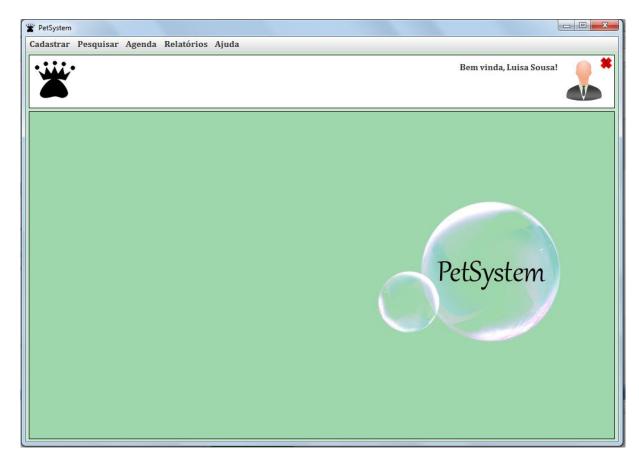


FIGURA 4 - Tela inicial do software PetSystem

2.3.3. Cadastros

Nesta seção, constará o menu de cadastros estendido, como pode ser percebido na Figura 5, e as respectivas telas de cadastros.

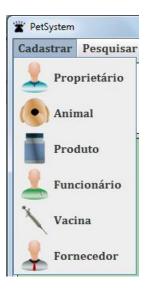


FIGURA 5 - Menu "Cadastrar" do software PetSystem

2.3.3.1. Proprietário

O cadastro do proprietário, ilustrado na Figura 6, exige campos de identificação, como nome e CPF, além do endereço e contatos.

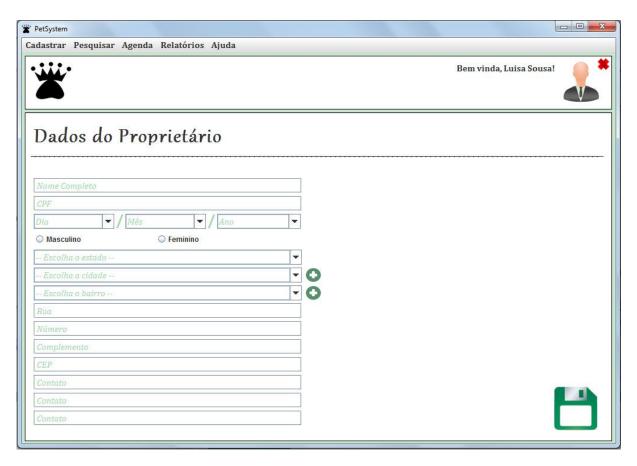


FIGURA 6 - Tela de cadastro de proprietário do software PetSystem

2.3.3.2. Animal

O cadastro do animal, como mostra a Figura 7, exige campos como nome, tipo e raça. Além do botão de vinculação com o proprietário a que pertence.

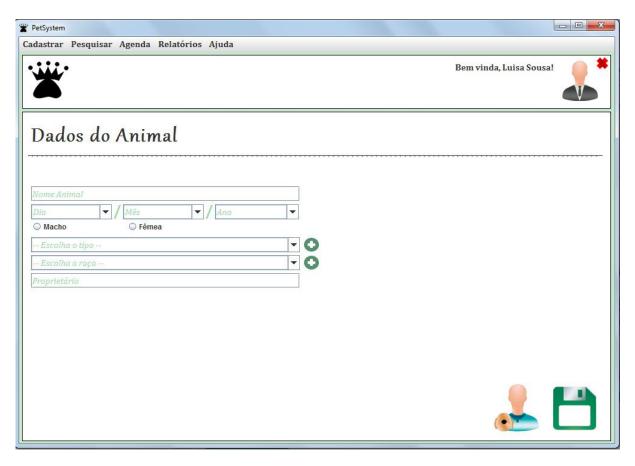


FIGURA 7 - Tela de cadastro de animal do software PetSystem

2.3.3.2.1. Vinculação Animal / Proprietário

A tela de vinculação, como pode ser percebido na Figura 8, é usada para relacionar um proprietário já cadastrado com um animal que está sendo registrado no momento.

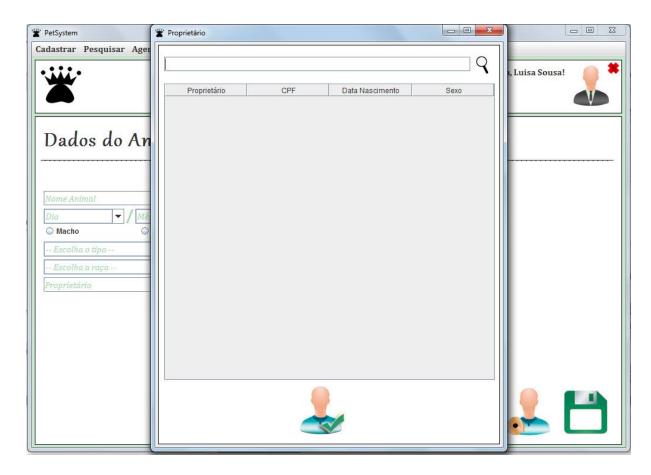


FIGURA 8 - Tela de vinculação entre animal e proprietário do software PetSystem

2.3.3.3. Produto

O cadastro de produto, que está ilustrado na Figura 9, exige campos como nome, quantidade, descrição e preço, além do fornecedor, vinculado através de um botão.

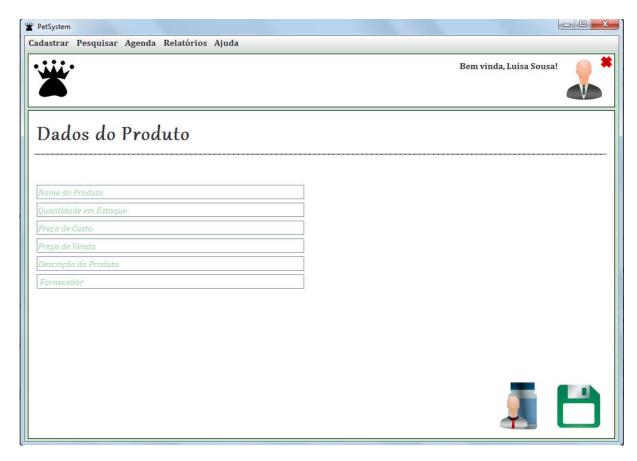


FIGURA 9 - Tela de cadastro de produto do software PetSystem

2.3.3.4. Vinculação Produto / Fornecedor

A tela de vinculação, como mostra a Figura 10, é usada para relacionar um fornecedor já cadastrado com um produto que está sendo registrado no momento.

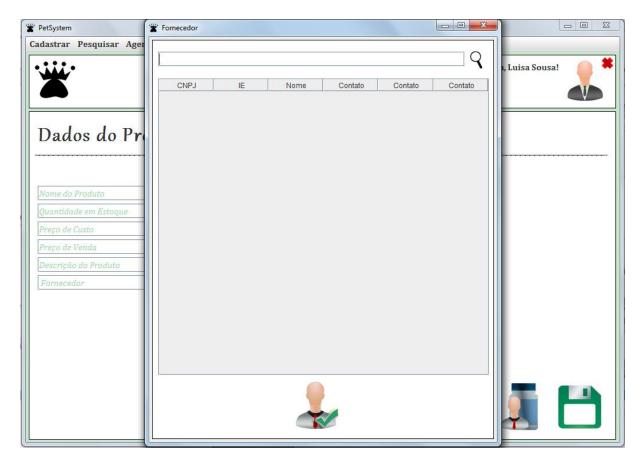


FIGURA 10 - Tela de vinculação entre produto e fornecedor do software PetSystem

2.3.3.5. Funcionário

O cadastro do funcionário ocorre de maneira análoga à do proprietário, como pode ser visto na Figura 11, porém exige outros campos, como categoria, salário, *login* e senha.

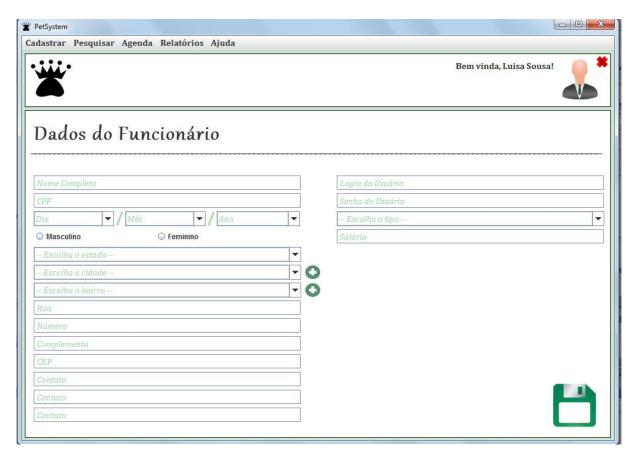


FIGURA 11 - Tela de cadastro de funcionário do software PetSystem

2.3.3.6. Vacina

O cadastro de vacina, que está ilustrado na Figura 12, exige os campos nome, preço, doença prevenida, melhor idade para aplicação e descrição.

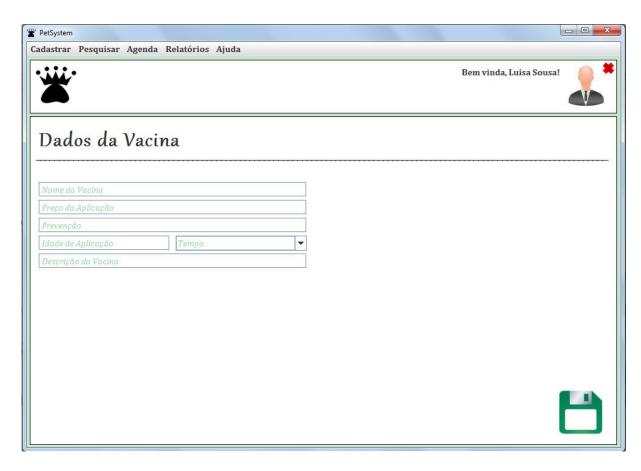


FIGURA 12 - Tela de cadastro de vacina do software PetSystem

2.3.3.7. Fornecedor

O cadastro de fornecedor, que pode ser visto na Figura 13, exige os campos de identificação como cnpj, inscrição estadual e nome, além do endereço e contatos.

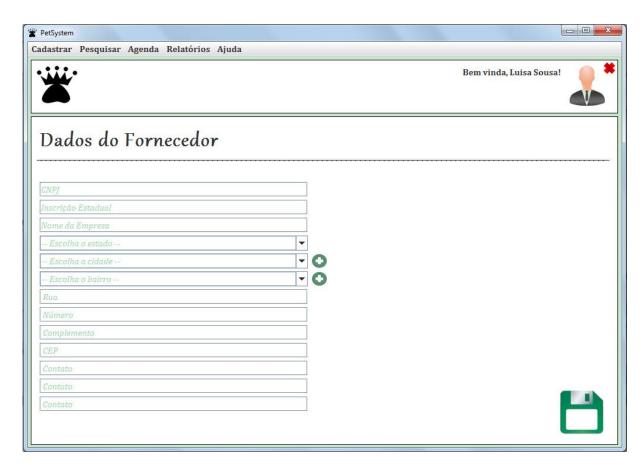


FIGURA 13 - Tela de cadastro de fornecedor do software PetSystem

2.3.4. Pesquisar

Nesta seção, constará o menu de pesquisas estendido, ilustrado na Figura 14, e as respectivas telas de buscas. É através desta funcionalidade que editamos e deletamos registros. No caso particular do animal, por meio desse módulo acessamos também sua ficha médica.



FIGURA 14 - Menu "Pesquisar" do software PetSystem

2.3.4.1. Proprietários

A busca de proprietários, como mostra a Figura 15, exibe uma tabela com alguns dados do proprietário buscado, além dos botões de alteração e exclusão.

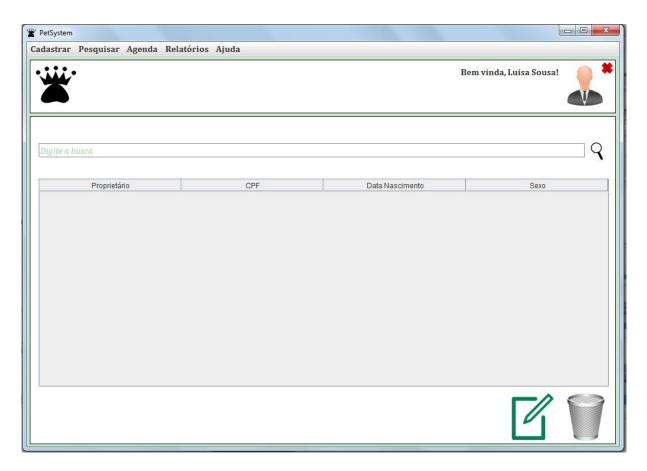


FIGURA 15 - Tela de busca de proprietários do software PetSystem

2.3.4.1.1. Edição Proprietário

A tela de edição de proprietários, ilustrada na Figura 16, é exibida após selecionar um proprietário na tabela de busca. É permitido fazer alterações em qualquer dado cadastrado anteriormente.

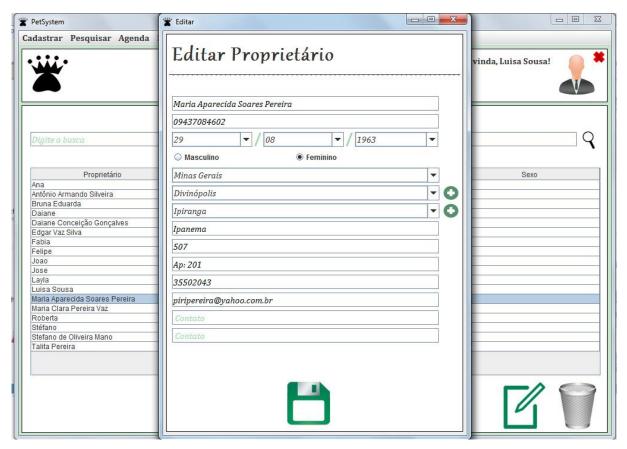


FIGURA 16 - Tela de edição de proprietário do software PetSystem

2.3.4.2. Animal

A busca de animais, como pode ser visto na Figura 17, exibe uma tabela com alguns dados do animal buscado, além dos botões de alteração, exclusão e consultar sua ficha médica.

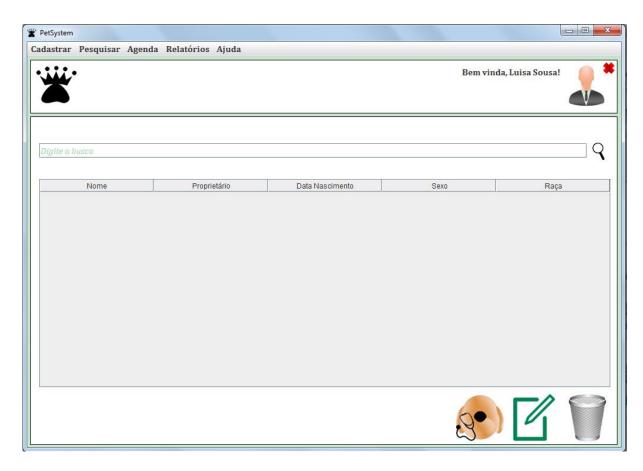


FIGURA 17 - Tela de busca de animais do software PetSystem

2.3.4.2.1. Edição Animal

A tela de edição de animais, que está ilustrada na Figura 18, é exibida após selecionar um animal na tabela de busca. É permitido fazer alterações em qualquer dado cadastrado anteriormente.

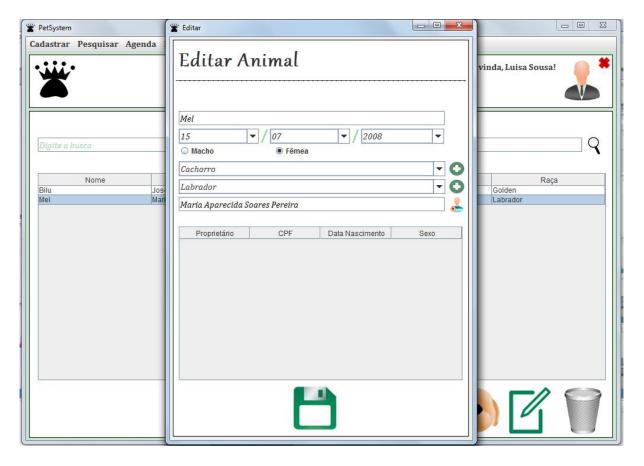


FIGURA 18 - Tela de edição de animal do software PetSystem

2.3.4.2.2. Ficha Médica Animal

A tela da ficha médica do animal é exibida após selecionar um animal na tabela de busca, e contém menus referentes ao atendimento a ser realizado, como pode ser percebido na Figura 19.

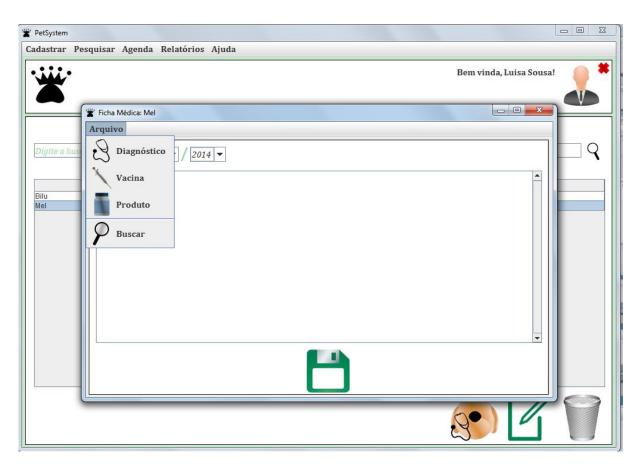


FIGURA 19 - Tela da ficha médica do animal com menu estendido do software PetSystem

2.3.4.2.2.1. Inserção Diagnóstico

Na inserção de um diagnóstico, como mostra a Figura 20, são apresentados campos para a descrição e para a data, sendo este iniciado com a data atual.

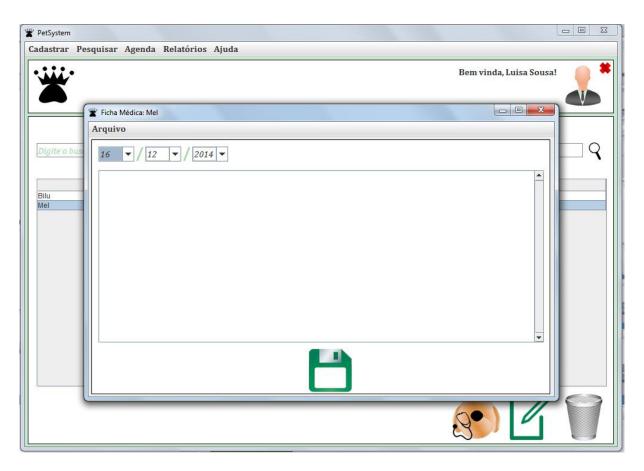


FIGURA 20 - Tela de inserção de diagnóstico do animal do software PetSystem

2.3.4.2.2.2. Vacinas Utilizadas

Na inserção de vacinas utilizadas, como pode ser visto na Figura 21, é apresentada uma tabela para buscas, além do campo para exibição das vacinas selecionadas.

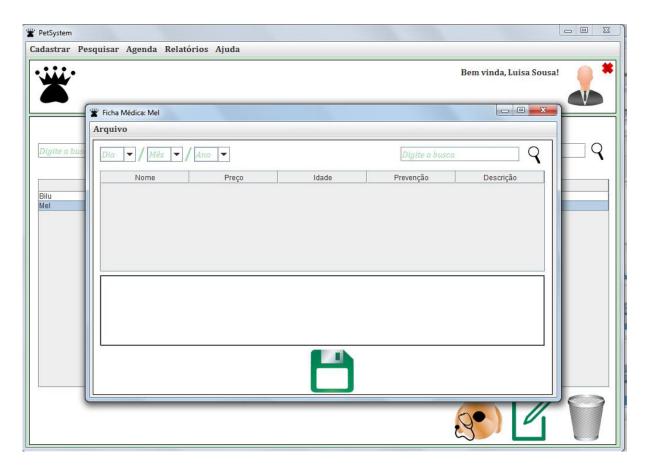


FIGURA 21 - Tela de inserção de vacinas aplicadas no animal do software PetSystem

2.3.4.2.2.3. Produtos Utilizados

Na inserção de produtos utilizados, que está ilustrada na Figura 22, é apresentada uma tabela para buscas, além do campo para exibição dos produtos selecionados.

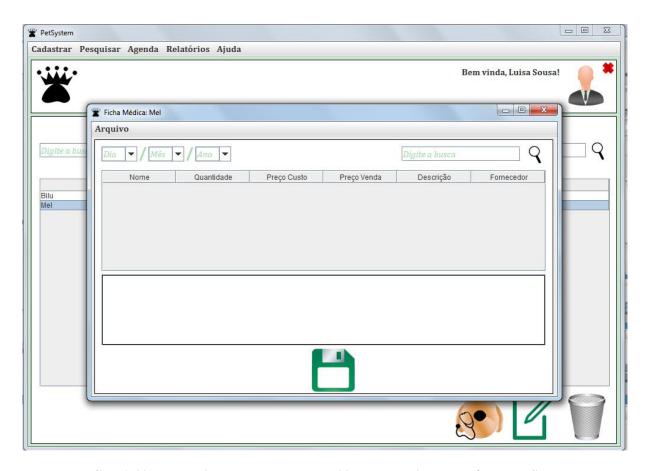


FIGURA 22 - Tela de inserção de produtos utilizados no animal do software PetSystem

2.3.4.2.2.4. Busca Ficha Médica

Na busca da ficha médica, como mostra a Figura 23, são oferecidas as opções de buscar diagnóstico inserindo a data na qual foi realizado, ou trafegar entre os registros presentes através dos botões: primeiro, anterior, próximo e último respectivamente.

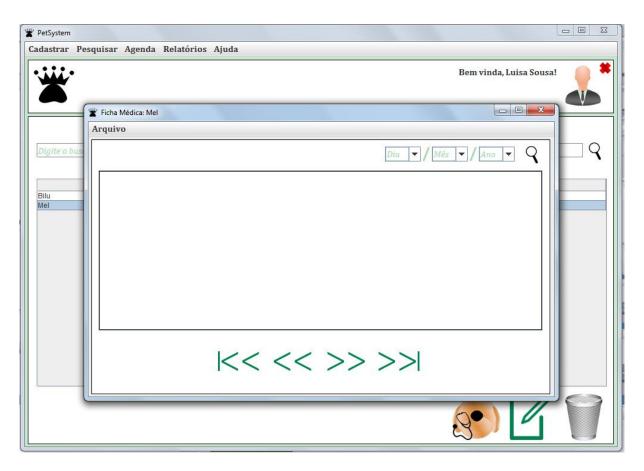


FIGURA 23 - Tela de busca da ficha médica do animal do software PetSystem

2.3.4.3. Produtos

A busca de produtos, ilustrada na Figura 24, exibe uma tabela com alguns dados do produto buscado, além dos botões de alteração e exclusão.

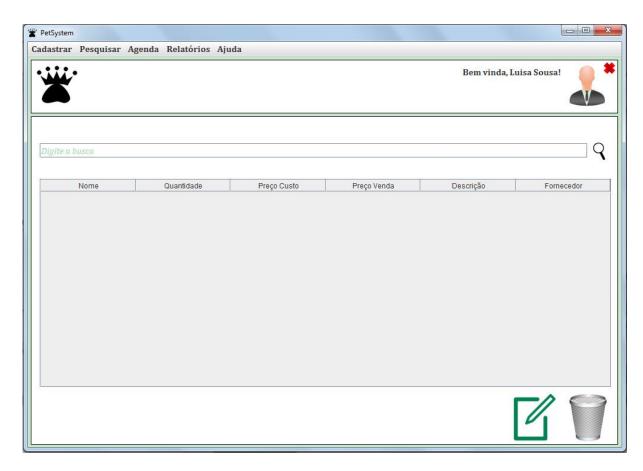


FIGURA 24 - Tela de busca de produtos do software PetSystem

2.3.4.3.1. Edição Produtos

A tela de edição de produtos é exibida após selecionar um produto na tabela de busca, como pode ser visto na Figura 25. É permitido fazer alterações em qualquer dado cadastrado anteriormente.

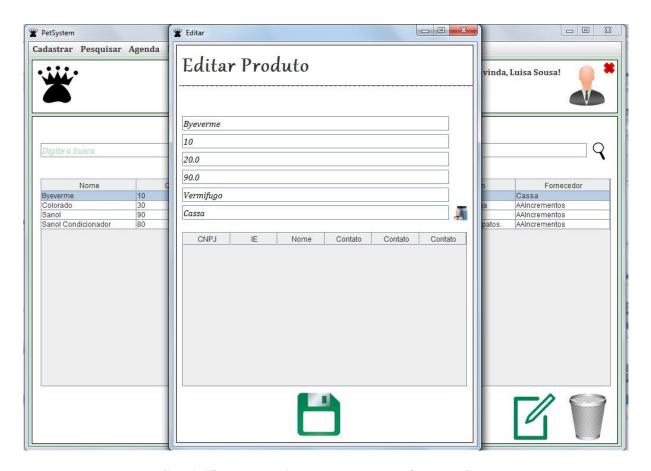


FIGURA 25 - Tela de edição de produto do software PetSystem

2.3.4.4. Funcionários

A busca de funcionários, que está ilustrada na Figura 26, exibe uma tabela com alguns dados do funcionário buscado, além dos botões de alteração e exclusão.

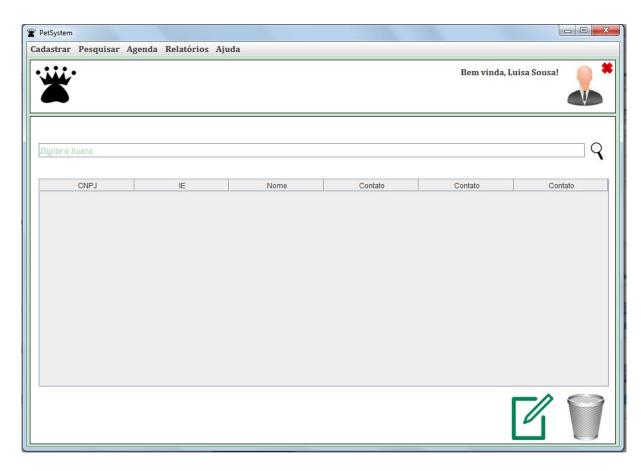


FIGURA 26 - Tela de busca de funcionários do software PetSystem

2.3.4.4.1. Edição Funcionário

A tela de edição de funcionários é exibida após selecionar um funcionário na tabela de busca, como mostra a Figura 27. É permitido fazer alterações em qualquer dado cadastrado anteriormente.

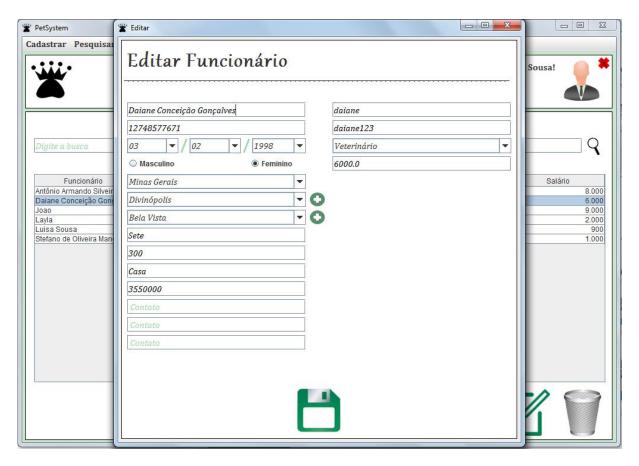


FIGURA 27 - Tela de edição de funcionário do software PetSystem

2.3.4.5. Vacinas

A busca de vacinas exibe uma tabela com alguns dados da vacina buscada, além dos botões de alteração e exclusão, como ilustrado na Figura 28.

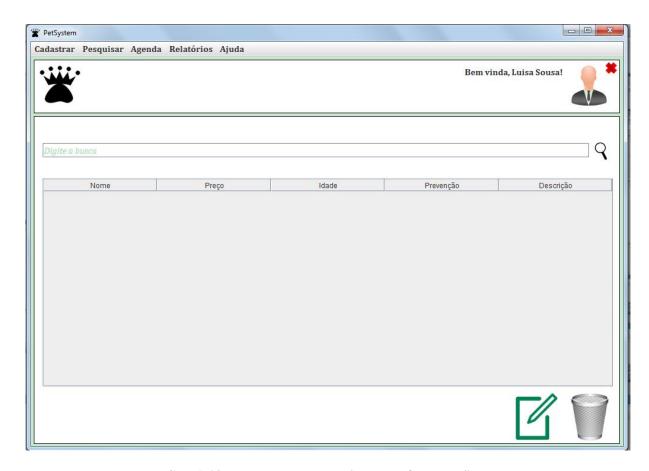


FIGURA 28 - Tela de busca de vacinas do software PetSystem

2.3.4.5.1. Edição Vacinas

A tela de edição de vacinas, que pode ser vista na Figura 29, é exibida após selecionar uma vacina na tabela de busca. É permitido fazer alterações em qualquer dado cadastrado anteriormente.

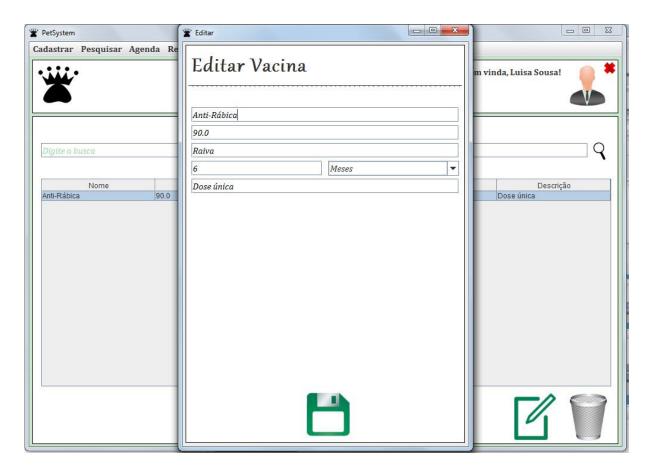


FIGURA 29 - Tela de edição de vacinas do software PetSystem

2.3.4.6. Fornecedores

A busca de fornecedores, que está ilustrada na Figura 30, exibe uma tabela com alguns dados do fornecedor buscado, além dos botões de alteração e exclusão.

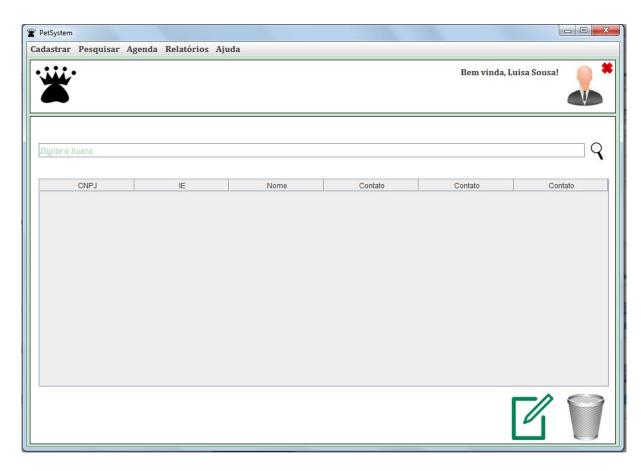


FIGURA 30 - Tela de busca de fornecedores do software PetSystem

2.3.4.6.1. Edição Fornecedores

A tela de edição de fornecedores, ilustrada na Figura 31, é exibida após selecionar um fornecedor na tabela de busca. É permitido fazer alterações em qualquer dado cadastrado anteriormente.

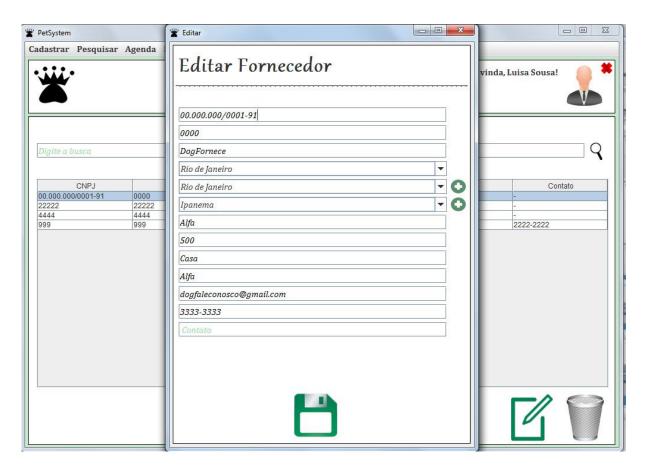


FIGURA 31 - Tela de edição de fornecedor do software PetSystem

2.3.5. Agendar Horários

Nesta seção, constará o menu da agenda estendido e as respectivas telas de agendamento e consulta, como pode ser visto na Figura 32.



FIGURA 32 - Menu "Agenda" do software PetSystem

2.3.5.1. Inserir

Na inserção de um novo agendamento, como mostra a Figura 33, são exigidas a data e a hora do atendimento. Além de serem apresentados os botões de vinculação com animal e produto oferecido.

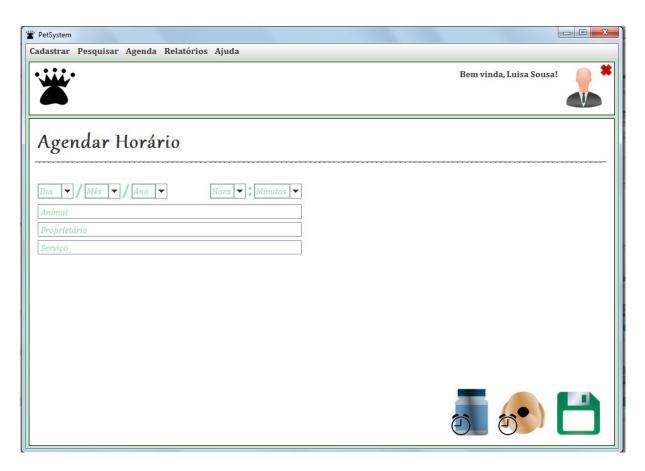


FIGURA 33 - Tela de agendamento de animais do software PetSystem

2.3.5.1.1. Vinculação Agendamento / Produto

A tela de vinculação, que está ilustrada na Figura 34, é usada para relacionar um produto já cadastrado com um agendamento que está sendo realizado no momento.

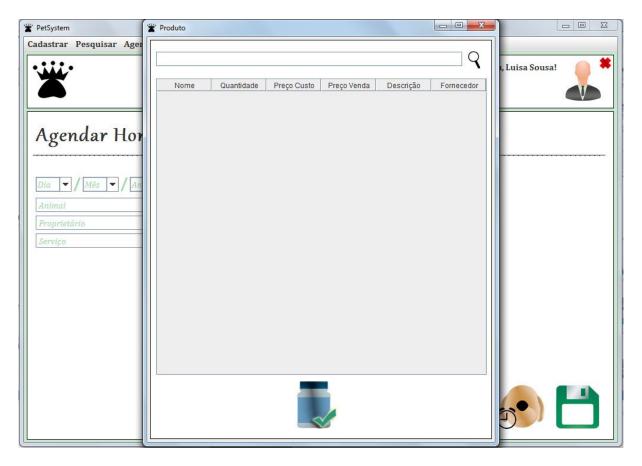


FIGURA 34 - Tela de vinculação entre agendamento e produto do software PetSystem

2.3.5.1.2. Vinculação Agendamento / Animal

A tela de vinculação é usada para relacionar um animal já cadastrado com um agendamento que está sendo realizado no momento, como pode ser percebido na Figura 35.

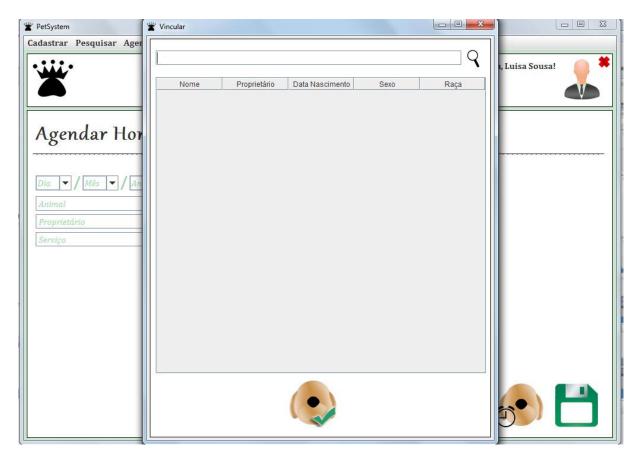


FIGURA 35 - Tela de vinculação entre agendamento e animal do software PetSystem

2.3.5.2. Consultar

Na consulta à agenda, como mostra a Figura 36, são exibidos inicialmente, os agendamentos referentes à data atual. Entretanto, é permitida a realização de busca pela data preferida. Além da possibilidade de exclusão de um horário ocupado.

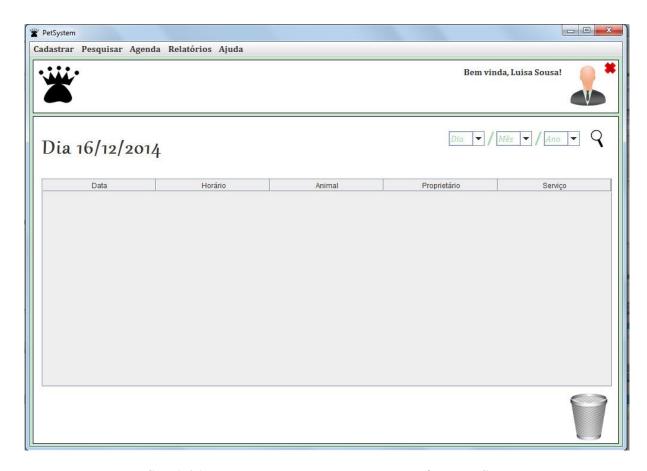


FIGURA 36 - Tela de busca de agendamentos do software PetSystem

2.3.6. Gerar Relatórios

Nesta seção, constará o menu de relatórios estendido, que está ilustrado na Figura 37, o qual oferece as opções de relações pra impressão.



FIGURA 37 - Menu "Relatórios" do software PetSystem

2.3.7. Ajuda

No menu "Ajuda", é apresentado o "Sobre" do sistema, como pode ser visto na Figura 38.

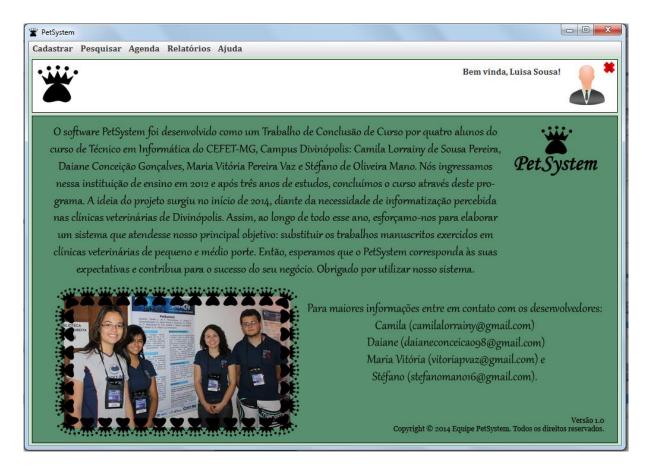


FIGURA 38 - Tela do "sobre" do software PetSystem

3. Projeto Físico

Nesta seção será apresentada a documentação das tabelas do banco de dados, cujo objetivo é organizar a parte física do sistema.

3.1. DER – Diagrama de Entidade e Relacionamento

O DER, diagrama de entidade e relacionamento, ilustra as entidades que irão compor o banco de dados e o tipo de relacionamento entre elas, como pode ser percebido na Figura 39. As tabelas do banco serão detalhadas no Anexo A.

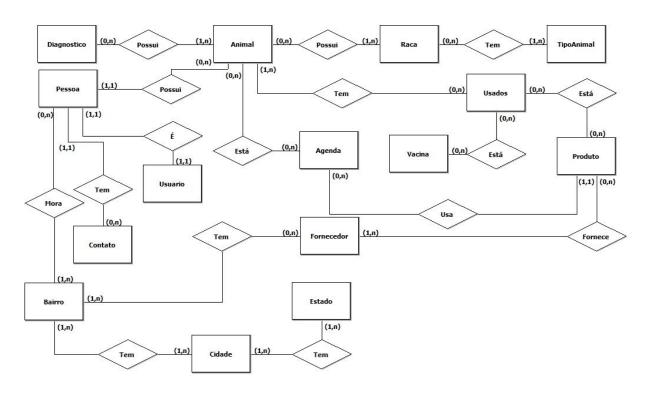


FIGURA 39 - DER do software PetSystem

3.2. Diagrama de Classes

O diagrama de classes, ilustrado na Figura 40, contém as principais classes do sistema, neste caso incluindo as referentes ao padrão MVC, e seus relacionamentos. No diagrama a seguir, estão contidos apenas os nomes das classes, estas serão detalhadas com seus atributos e métodos no Apêndice C.

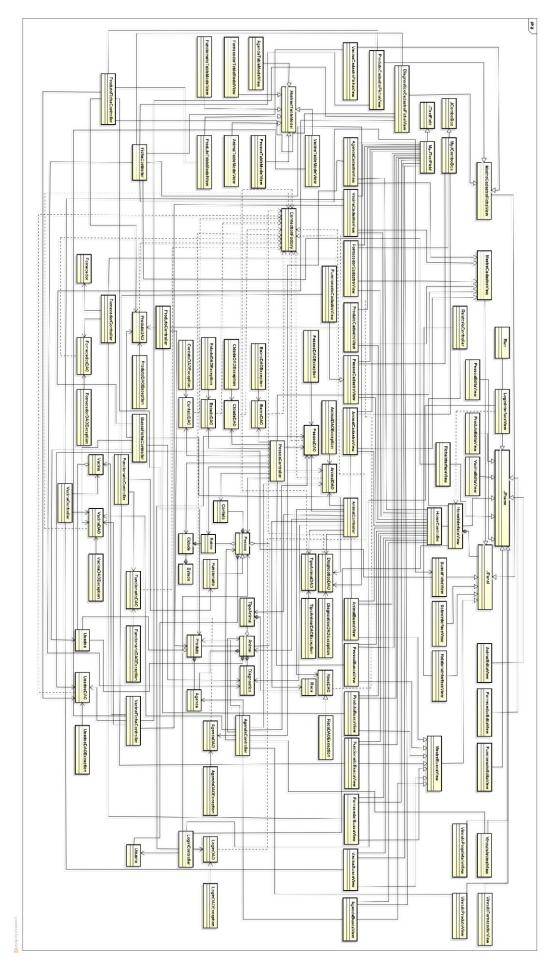


FIGURA 40 - Diagrama de Classes do software PetSystem

3.3. Dicionário de Dados

Nesta seção será exibido o dicionário de dados, que consiste no detalhamento das tabelas do banco de dados exibindo todos os atributos presentes nelas.

Agenda = *Agendamento de horários para prestação de serviços*

@idagenda + horario + dia + idanimal + idproduto

Animal = *Dados do animal*

@idanimal + nome + (datanascimento) + sexo + idraca + idpessoa

Bairro = *Bairro onde a pessoa mora*

@idbairro + bairro + idcidade

Cidade = *Cidade onde a pessoa mora*

@idcidade + cidade + idestado

Contato = *Contato da pessoa (cliente, fornecedor, funcionário, gerente, veterinário)*

@idcontato + contato +[idpessoa + (idfornecedor) / (idpessoa) + idfornecedor]

Diagnostico = *Diagnostico do animal*

@iddiagnostico + dia + descricao + idanimal

Estado = *Estado onde a pessoa mora*

@idestado + estado + uf

Fornecedor = *Vendedor de um produto*

@idfornecedor + (cnpj) + (inscricaoestadual) + nome + rua + numero + (complemento) + cep + idbairro

Pessoa = *Dados da pessoa (cliente, fornecedor, funcionário, gerente, veterinário)*

@idpessoa + nome + cpf + datanascimento + rua + (complemento) + cep + numero + sexo + idbairro

Produto = *Produto da clínica veterinária*

@idproduto + nome + quantidade + precocusto + descricao + idfornecedor + precovenda

Raca = *Raça do animal*

@idraca + nome + idtipoanimal

TipoAnimal = *Tipo do animal (gato, cachorro, cavalo, etc.)*

@idtipoanimal + tipo

Usados = *Produtos utilizados no animal*

@idusados + idanimal + (idproduto) + dia + (idvacina)

Usuario = *Funcionário que opera o sistema*

@idusuario + tipo + senha + login + salario + idpessoa

Vacina = *Vacina da clínica veterinária*

@idvacina + preco + nome + descricao + idade + prevencao

4. Resultados

Na fase inicial do projeto, focou-se no desenvolvimento dos diagramas e dos modelos para o banco de dados. Observou-se grande vantagem na prévia diagramação, pois esta permitiu a previsão e o esclarecimento de como seria elaborado o sistema. Os diagramas produzidos constam nesta documentação.

Já o período de codificação, apesar de extenso, foi facilitado pelo padrão MVC-DAO adotado, o qual melhorou a organização e o entendimento das classes e ações do sistema. No módulo *View* (Visualização) deste padrão, houveram atritos em relação a estética desenvolvida. Entretanto, após reuniões no *PetShop* escolhido como referencial, obteve-se a concordância sobre o melhor *design* para o *software*. Em consequência disso, houve atraso no período de implantação e testes do sistema nesta empresa.

Portanto, a estratégia de teste foi elaborada em duas partes, sendo elas: interno e externo. O primeiro, consistia na integração de todos os módulos (cadastro, busca, ficha médica, etc.) desenvolvidos separadamente, além da conferência de todas as validações necessárias. Já o segundo estado, referia-se à opinião dos usuários sobre o sistema.

Numa visão global, os resultados obtidos foram satisfatórios. Na empresa estudada, obteve-se a aprovação do programa, pois este atendeu as funcionalidades essenciais encontradas nesse ambiente de maneira enxuta através de uma interface simples e intuitiva.

5. Conclusão

Este trabalho teve como objetivo geral facilitar o gerenciamento de registros em clínicas veterinárias. Para isso, percebeu-se a necessidade de conhecimento na área, portanto, foram realizadas entrevistas nessas clínicas, na região central de Divinópolis-MG. Porém, constatou-se que alguns *PetShop's* funcionam integrando a parte de comércio e de atendimento médico.

Por consequência, encontrou-se a dificuldade de atender de maneira genérica todos os locais da pesquisa, resolvendo-se então especificar um referencial. O adotado foi um *PetShop* que ofereceu inteira disponibilidade para estudos de seus casos e testes no local. Dessa forma, foi possível aumentar a satisfação deste com o sistema desenvolvido.

Assim sendo, percebeu-se que o PetSystem cumpriu a proposta empregada incialmente, acrescentando o módulo de controle de produtos e fornecedores. Portanto, acredita-se que o projeto pode atender a clínicas de grande porte, caso sofra antes melhorias e incrementos que podem ser realizados em futuros trabalhos da instituição.

6. Cronograma

A seguir será apresentado na Figura 41, através da Estrutura Analítica do Projeto, EAP, o cronograma seguido para desenvolvimento do sistema.

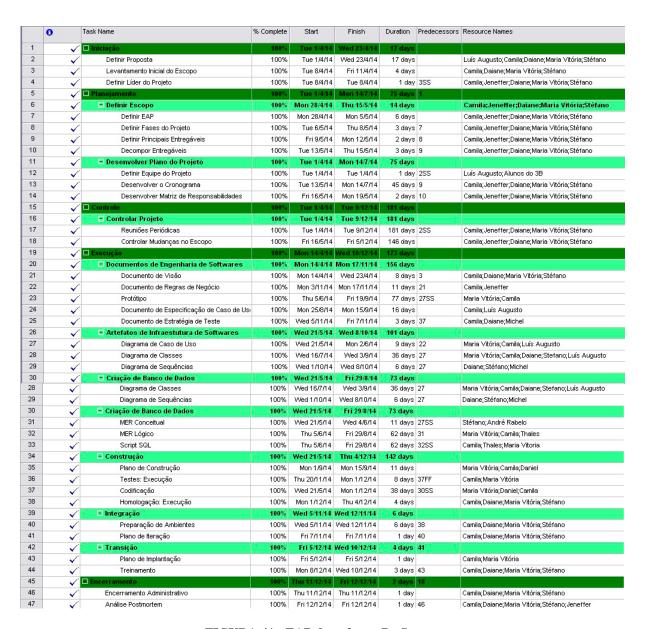


FIGURA 41 - EAP do softwate PetSystem

7. Referências

ARANTES, André L.. **SQL – Apostila.** Disponível em:

http://www.andrearantes.eti.br/icesp/bd/firebird/sql_apostila_conceitos.pdf>. Acesso em: 04 de setembro de 2014.

CAETANO, Daniel. **Programação Servidor em Sistemas Web - Padrões MVC e DAO**. 2012. Disponível em: http://www.caetano.eng.br/aulas/2012a/psw/psw_aula08.pdf>. Acesso em: 02 de setembro de 2014.

Comissão de Normalização de Trabalhos Acadêmicos. **Normas para elaboração de trabalhos acadêmicos.** Curitiba: Universidade Tecnológica Federal do Paraná; 2008.

DEITOS, Maria Lúcia Melo de Souza. **A gestão da tecnologia nas pequenas e médias empresas.** Cascavel: Edunioste; 2002.

Doctor Vet - Software de Gestão Clínica Veterinária. Disponível em:

http://www.xionce.com/doctorvet/pt/index.php>. Acesso em: 02 de setembro de 2014.

FÉLIX, Francisca H. Cavalcante; MARTINS, Jorge A. Matias; MOURA, Mauricio S. Rocha. A informática aplicada à Medicina Veterinária. Disponível em:

http://www.propgpq.uece.br/semana_universitaria/anais/anais2002/>. Acesso em: 30 de maio de 2014.

GUDWIN, Ricardo R.. **Introdução à Linguagem UML.** Departamento de Engenharia de Computação e Automação Industrial da Faculdade de Engenharia Elétrica e de Computação da UNICAMP; 2010.

Java. Disponível em: https://www.java.com/pt_BR/about/>. Acesso em: 04 de setembro de 2014.

MARTINS, Fernanda Nóbrega de Medeiros; SOUZA, Mayara Benicio de Barros; SILVA, Suzane Mendes da; RODRIGUES, Yane Wanderley dos Santos. **Estudo de Viabilidade.** Recife; 13 de setembro de 2011.

MATOS, Antonio Carlos de; MELCHOR, Paulo. **Comece certo Clínica Veterinária.** 2ª Edição. São Paulo: Sebrae-SP; 2005.

NS Consultoria. Disponível em: http://nsconsultoria.com.br/downloads/>. Acesso em: 02 de setembro de 2014.

Project Management Institute. Um guia do conhecimento em Gerenciamento de Projetos (Guia PMBOK ®). 4ª Edição. Pensilvânia – EUA: PMI; 2008.

QVet – Software de Gestão Integral para Clínicas Veterinárias. Disponível em: http://qvet.com.br/que-es-qvet.aspx. Acesso em: 02 de setembro de 2014.

SACILOTTI, Adaní Cusin. A importância da Tecnologia da Informação nas micro e pequenas empresas: um estudo exploratório na região de Jundiá. Campo Limpo Paulista; 2011.

VetSoft - Software Veterinário. Disponível em:

http://www.softwareveterinario.com.br/conheca/. Acesso em: 30 de maio de 2014.

8. Anexos

8.1. Anexo A: DTR - Diagrama das Tabelas Relacionais

Neste anexo, será apresentado o Diagrama das Tabelas Relacionais, DTR, ilustrado na Figura 42, que detalha as tabelas presentes no banco de dados com todos seus atributos e a relação entre elas.

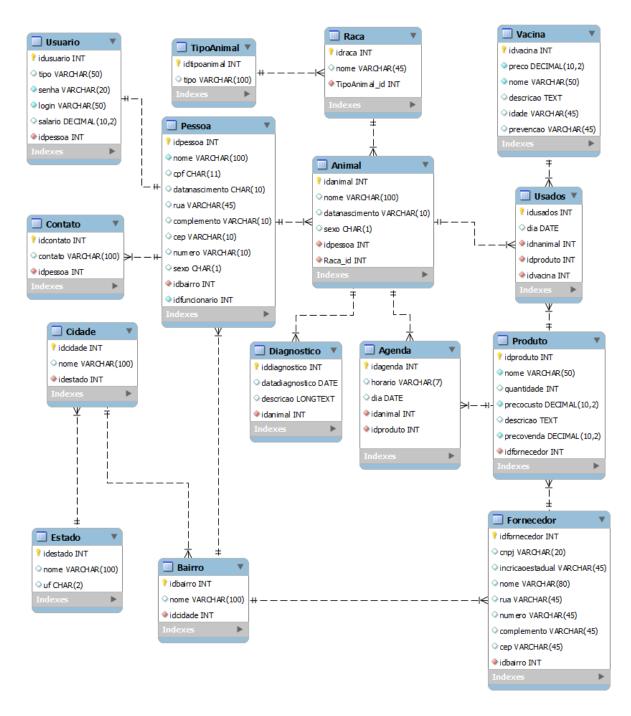


FIGURA 42 – DTR do software PetSystem

8.2. Anexo B: Diagramas de Sequência

Neste anexo, serão apresentados os diagramas de sequência, que podem ser vistos nas Figuras 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72 e 73 referentes a algumas ações do sistema. Este modelo de diagrama consiste em mostrar como as mensagens entre os objetos são trocadas no decorrer do tempo.

8.2.1. Cadastros

Nesta seção, serão apresentados, através das Figuras 43, 44, 45, 46, 47, 48, 49 e 50, os diagramas de sequência referentes aos cadastros que podem ser realizados no sistema.

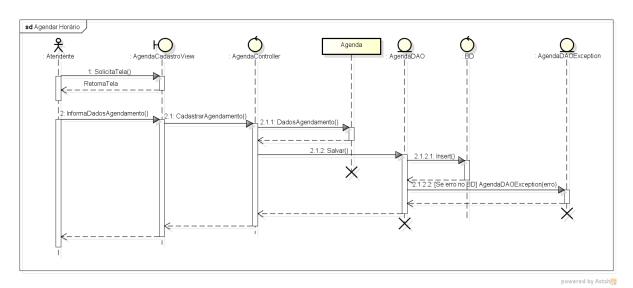


FIGURA 43 - Diagrama de Sequência Agendar Horário do software PetSystem

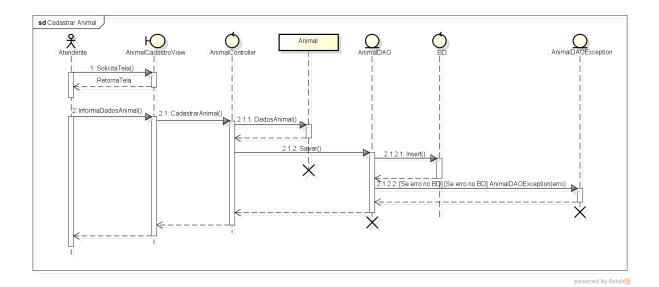


FIGURA 44 - Diagrama de Sequência Cadastrar Animal do software PetSystem

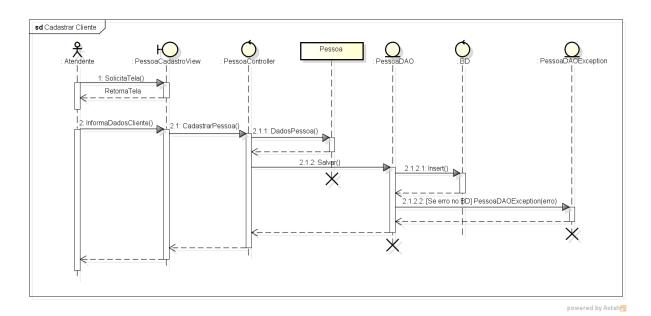


FIGURA 45 - Diagrama de Sequência Cadastrar Cliente do software PetSystem

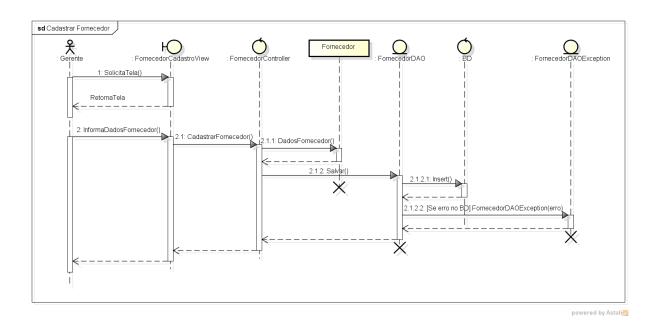


FIGURA 46 - Diagrama de Sequência Cadastrar Fornecedor do software PetSystem

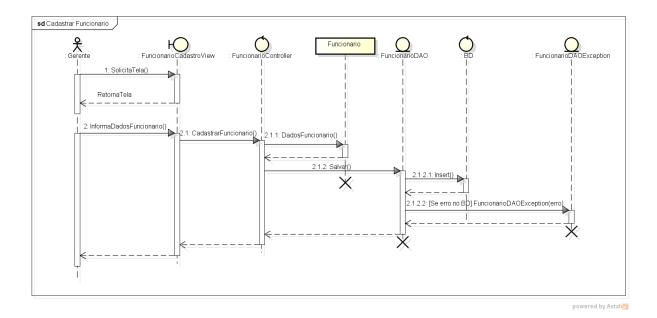


FIGURA 47 - Diagrama de Sequência Cadastrar Funcionário do software PetSystem

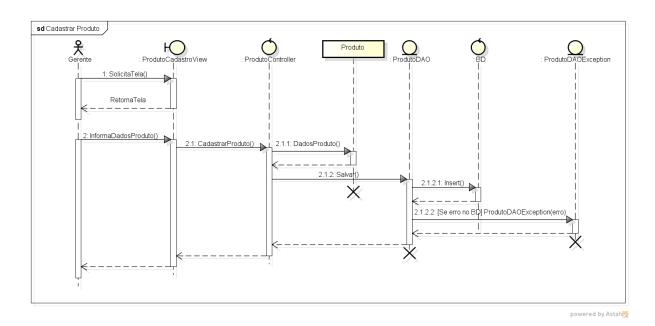


FIGURA 48 - Diagrama de Sequência Cadastrar Produto do software PetSystem

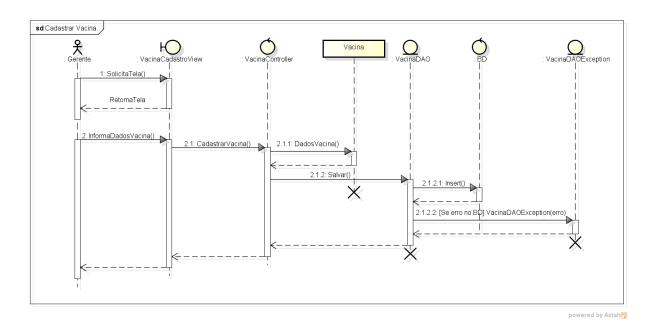


FIGURA 49 - Diagrama de Sequência Cadastrar Vacina do software PetSystem

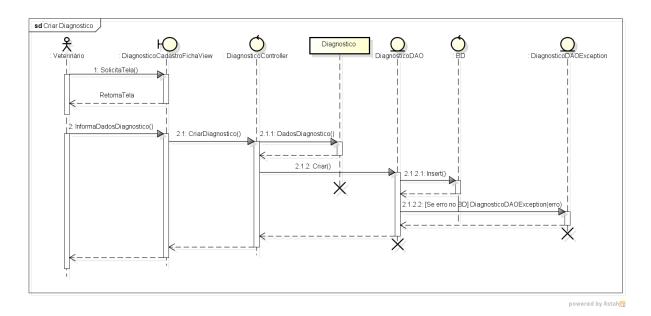


FIGURA 50 - Diagrama de Sequência Criar Diagnóstico do software PetSystem

8.2.2. Consultas

Nesta seção, serão apresentados os diagramas de sequência referentes às consultas que podem ser realizados no sistema, ilustrados nas Figuras 51, 52, 53, 54, 55, 56, 57 e 58.

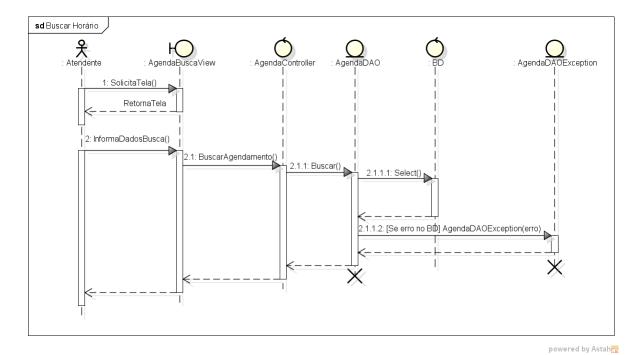


FIGURA 51 - Diagrama de Sequência Buscar Horário do software PetSystem

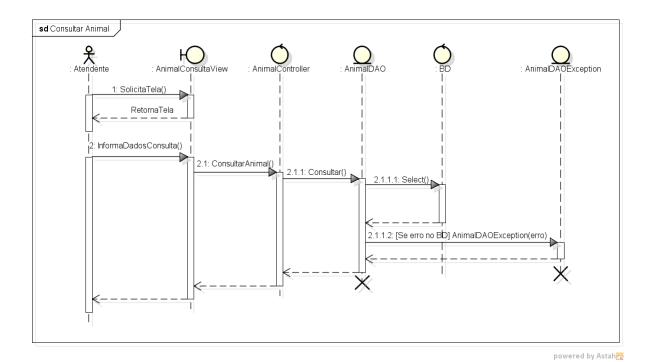


FIGURA 52 - Diagrama de Sequência Consultar Animal do software PetSystem

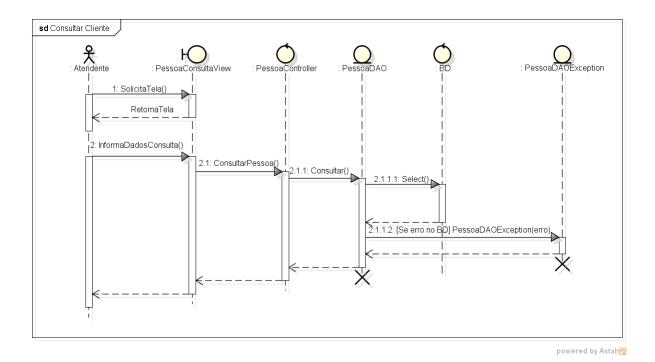


FIGURA 53 - Diagrama de Sequência Consultar Cliente do software PetSystem

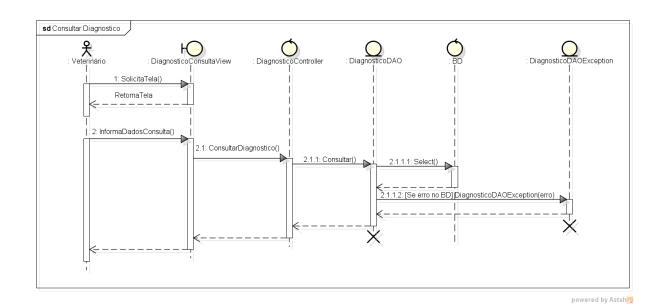


FIGURA 54 - Diagrama de Sequência Consultar Diagnóstico do software PetSystem

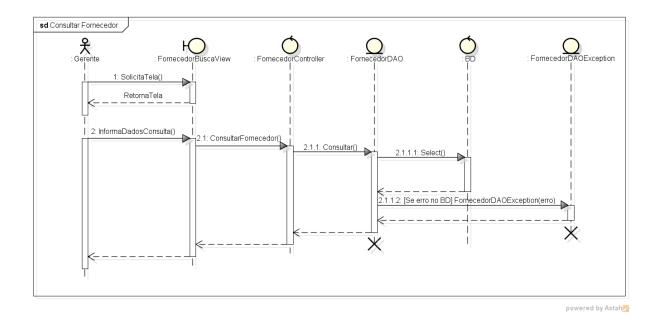


FIGURA 55 - Diagrama de Sequência Consultar Fornecedor do software PetSystem

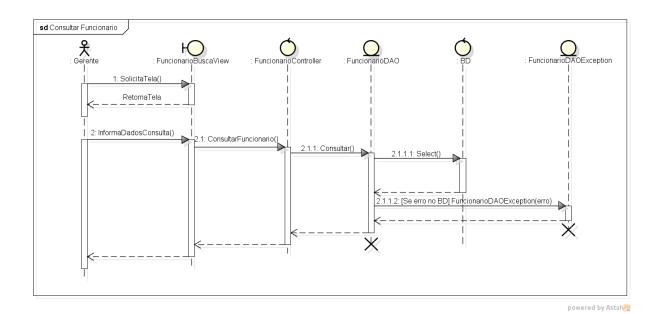


FIGURA 56 - Diagrama de Sequência Consultar Funcionário do software PetSystem

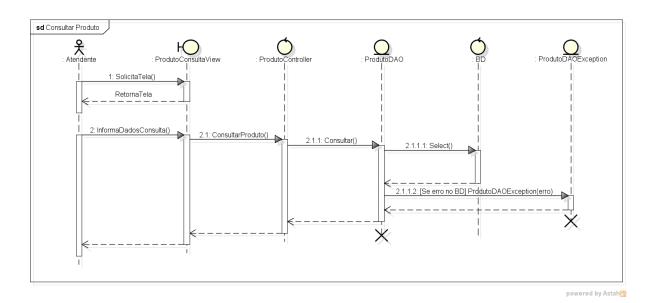


FIGURA 57 - Diagrama de Sequência Consultar Produto do software PetSystem

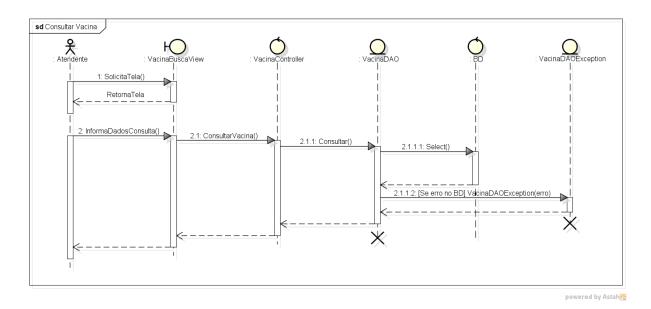


FIGURA 58 - Diagrama de Sequência Consultar Vacina do software PetSystem

8.2.3. Edições

Nesta seção, serão apresentados, por meio das Figuras 59, 60, 61, 62, 63 e 64, os diagramas de sequência referentes às edições que podem ser realizados no sistema.

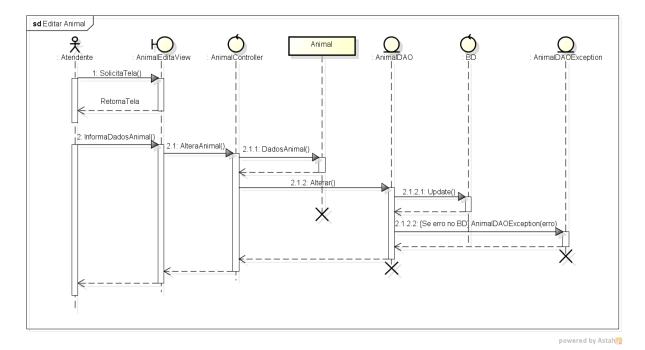


FIGURA 59 - Diagrama de Sequência Editar Animal do software PetSystem

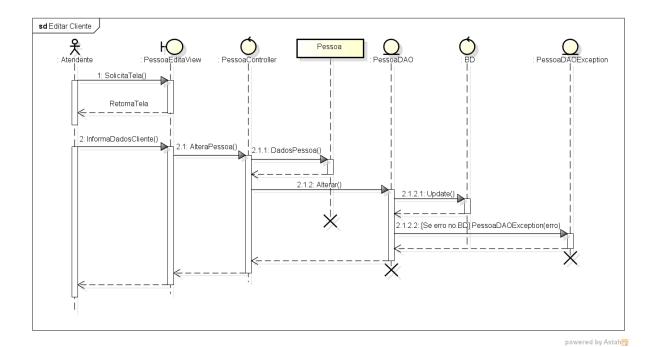


FIGURA 60 - Diagrama de Sequência Editar Cliente do software PetSystem

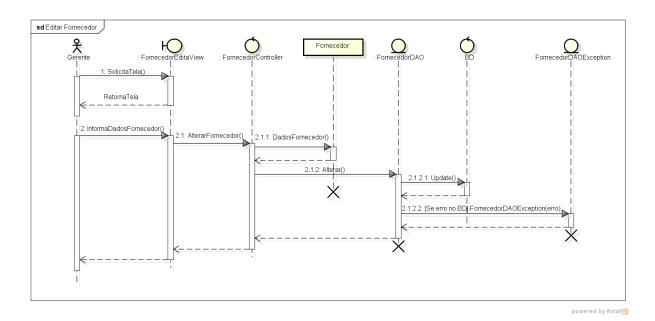


FIGURA 61 - Diagrama de Sequência Editar Fornecedor do software PetSystem

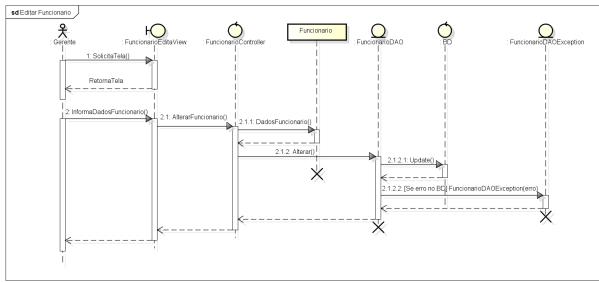


FIGURA 62 - Diagrama de Sequência Editar Funcionário do software PetSystem

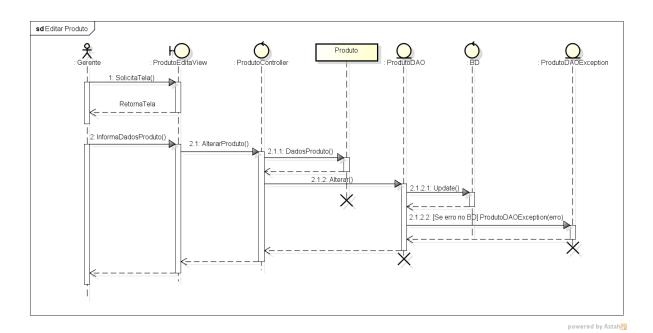


FIGURA 63 - Diagrama de Sequência Editar Produto do software PetSystem

powered by Astah

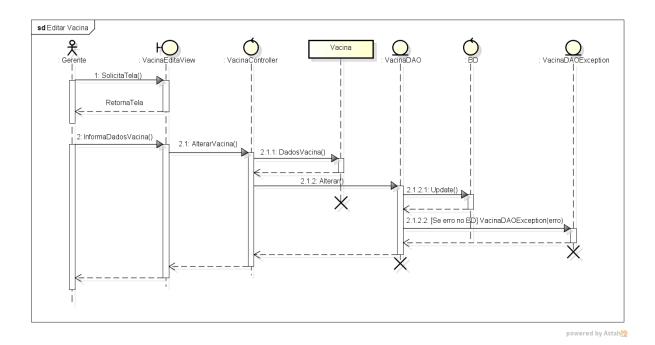


FIGURA 64 - Diagrama de Sequência Editar Vacina do software PetSystem

8.2.4. Gerar Relatório

Nesta seção, será apresentado o diagrama de sequência referente aos relatórios gerados pelo sistema, ilustrado na Figura 65.

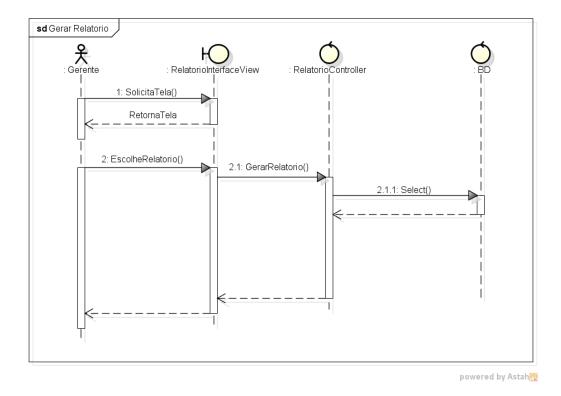


FIGURA 65 - Diagrama de Sequência Gerar Relatório do software PetSystem

8.2.5. Login do Usuário

Nesta seção, será apresentado através da Figura 66 o diagrama de sequência referente ao efetuar *login* do sistema.

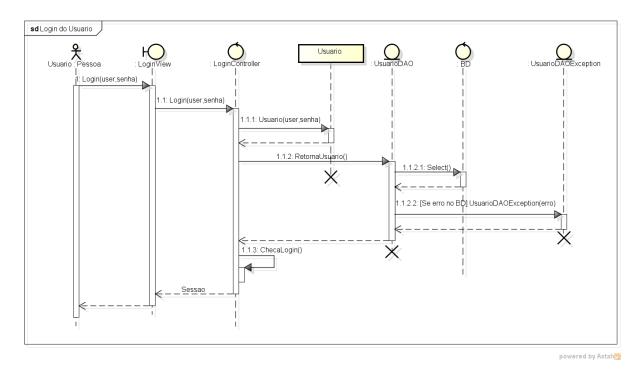
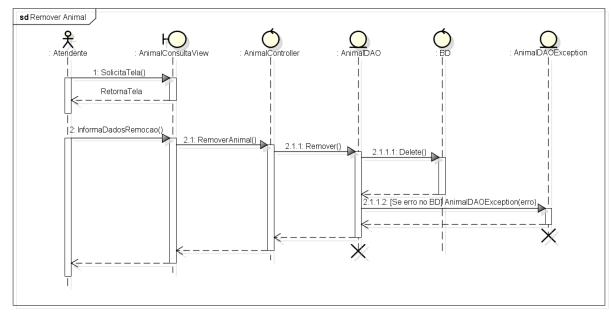


FIGURA 66 - Diagrama de Sequência Login do Usuário do software PetSystem

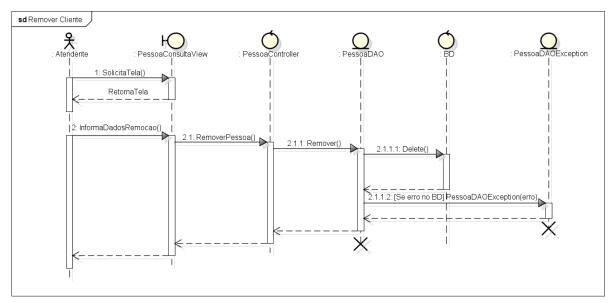
8.2.6. Remoções

Nesta seção, serão apresentados os diagramas de sequência referentes às remoções que podem ser realizados no sistema, sendo estes ilustrados nas Figuras 67, 68, 69, 70, 71, 72 e 73.



powered by Astah

FIGURA 67 - Diagrama de Sequência Remover Animal do software PetSystem



powered by Astah

FIGURA 68 - Diagrama de Sequência Remover Cliente do software PetSystem

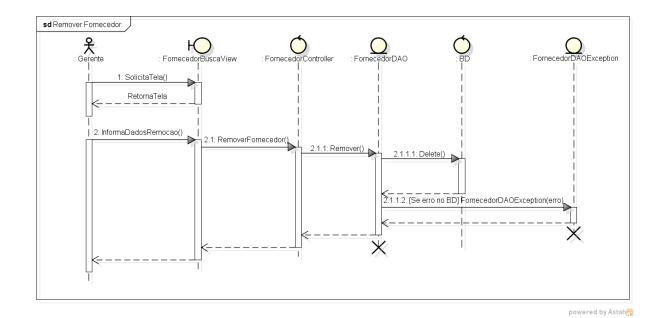


FIGURA 69 - Diagrama de Sequência Remover Fornecedor do software PetSystem

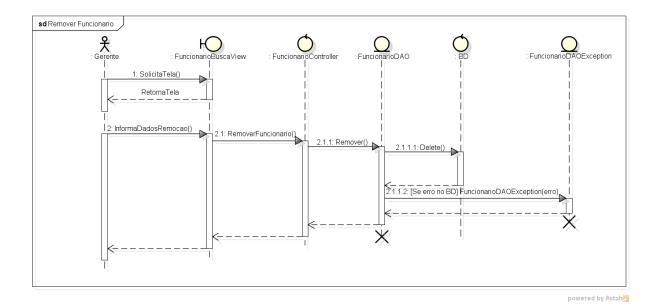


FIGURA 70 - Diagrama de Sequência Remover Funcionário do software PetSystem

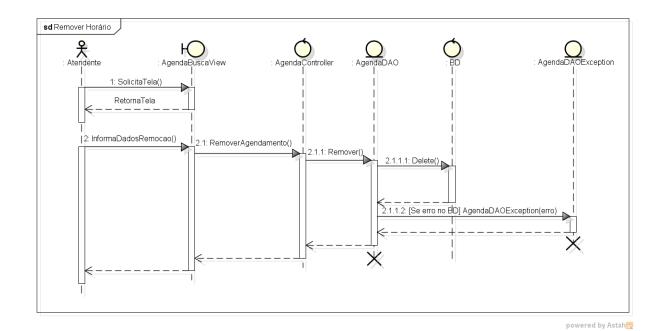


FIGURA 71 - Diagrama de Sequência Remover Horário do software PetSystem

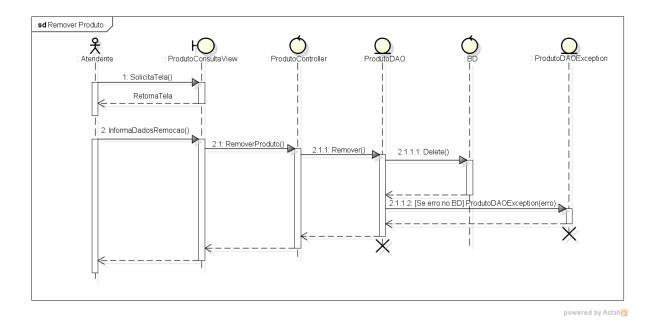


FIGURA 72 - Diagrama de Sequência Remover Produto do software PetSystem

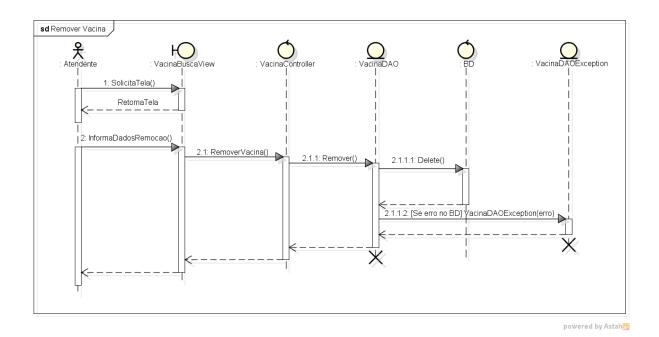


FIGURA 73 - Diagrama de Sequência Remover Vacina do software PetSystem

8.3. Anexo C: Classes detalhadas

Neste anexo, serão apresentadas com seus atributos e métodos as classes detalhadas, ilustradas nas Figuras 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183 e 184, citadas no diagrama de classes, que foi abordado anteriormente na Figura 40.

8.3.1. View

Nesta seção, serão apresentadas, através das Figuras 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116 e 117, as classes do *software* PetSystem relativas ao módulo *View* do modelo MVC-DAO adotado para desenvolvimento da codificação.

AnimalBuscaView |

- IAlterar : Imagelcon
- + AnimalBuscaView(largura:int, altura:int)
- + getModel(): AnimalTableModelView

FIGURA 74 - Classe AnimalBuscaView do software PetSystem

LoginInterfaceView

- largura : int
- altura : int
- cbmNome : JComboBox
- txtSenha: MyJTextField
- IblFundo : JLabel
- array: ArrayList
- btnEnviar : JButton
- ILogin: Imagelcon
- + LoginIterfaceView()
- + getSenha(): String
- + getCategoria(): String

FIGURA 75 - Classe LoginInterfaceView do software PetSystem

MestreBuscaView

- txtbusca : MyJTextField
- btnbusca : JButton
- IBusca : Imagelcon
- tblPesquisa : JTable
- scpResultado : JScrollPane
- + MestreBuscaView(largura:int, altura:int)
- + getTxtbusca(): String
- + getOcupado(): int

FIGURA 76 - Classe MestreBuscaView do software PetSystem

PessoaBuscaView

- IAlterar : Imagelcon
- model: PessoaTableModelView
- + PessoaBuscaView(largura:int, altura:int)
- + getModel(): PessoaTableModelView

FIGURA 77 - Classe PessoaBuscaView do software PetSystem

ProdutoBuscaView

- model: ProdutoTableModelView
- + ProdutoBuscaView(largura:int, altura:int)
- + getModel(): ProdutoTableModelView

FIGURA 78 - Classe ProdutoBuscaView do software PetSystem

HomeInterfaceView

- btnLogo : JButton - pnlCampos : JPanel - pnlTopo : JPanel

- pnlFundo : JPanel - ILogo: Imagelcon

- ICProprietario : Imagelcon

- ICAnimal : Imagelcon

- ICFuncionario : Imagelcon

- ICProduto : Imagelcon

- IPProprietario : Imagelcon

- IPAnimal : Imagelcon

- IPFuncionario : Imagelcon

- IPProduto : Imagelcon

- largura : int - altura : int

- IblCampos : JLabel

- menu : JMenuBar

- mnCadastro : JMenu

- mnPesquisa : JMenu

- mnCProp : JMenultem

- mnCAnimal : JMenultem

- mnRel : JMenultem

- mnPProp : JMenultem

- mnPAnimal: JMenultem

- mnPProd : JMenultem

- mnAgenda : JMenultem

- mnCaixa : JMenultem

- mnSobre : JMenultem

- mnCFunc : JMenultem

- mnPFunc : JMenultem

- ICampos : Imagelcon

+ HomeInterfaceView(title: String, left: int, top: int)

FIGURA 79 - Classe HomeInterfaceView do software PetSystem

RelatorioInterfaceView

btnProprietarios : JButtonbtnProdutosBaixa : JButton

btnSexo : JButtonbtnProdutos : JButtonbtnEstados : JButtonbtnContatos : JButton

btnGerar : JButtonbtnSexoValor : JButtonbtnPropAnimal : JButton

- IstRadios : JList

modeloLista : DefaultListModel

- IblTitulo : JLabel

txtProdutos : JTextFieldtxtSexo : JTextField

- txtPropAnimal: JTextField

IbIProd : JLabelIbISexo : JLabelIbIProp : JLabel

+ RelatorioInterfaceView(largura: int, altura: int)

FIGURA 80 - Classe RelatorioInterfaceView do software PetSystem

AnimalCadastroView

txtnome: MyJTextField
txtnascimento: MyJTextField
txtsexo: MyJTextField
cmbTipo: MyJComboBox
radMasc: JRadioButton
radFem: JRadioButton
grupo: ButtonGroup
lblSeparador: JLabel
lblTitulo: JLabel
btnVinculo: JButton
btnTipo: JButton
IAdicionar: Imagelcon
IVinculo: Imagelcon

+ AnimalCadastroView(title: String, left: int, top: int, altura: int, largura: int)

FIGURA 81 - Classe AnimalCadastroView do software PetSystem

MestreCadastroView | |

btnAdd : JButtonIAdd : ImagelconIDel : Imagelcon

+ MestreCadastroView(left: int, top: int, largura: int, altura: int)

FIGURA 82 - Classe MestreCadastroView do software PetSystem

PessoaCadastroView

cmbestado : MyJComboBoxcmbcidade : MyJComboBox

btnCidade : JButtonbtnBairro : JButtonbtnVinculo : JButton

- cmbbairro : MyJComboBox - txtnome : MyJTextField

txtcpf : MyJTextField

txtnascimento : MyJTextField

- txtrua : MyJTextField

txtcomplemento : MyJTextField

txtcep: MyJTextField
 txtnumero: MyJTextField
 txtemail: MyJTextField
 txttelefone: MyJTextField
 txttcelular: MyJTextField

radMasc : JRadioButtonradFem : JRadioButton

- grupo : ButtonGroup

+ public PessoaCadastroView(left: int, top: int, altura: int, largura: int)

FIGURA 83 - Classe PessoaCadastroView do software PetSystem

ProdutoCadastroView

- lblSeparador : JLabel

- IblTitulo : JLabel

txtPreco : MyJTextField
 txtDescricao : MyJTextField
 txtNome : MyJTextField
 txtQntde : MyJTextField

+ ProdutoCadastroView(left: int, top: int, largura: int, altura: int)

FIGURA 84 - Classe ProdutoCadastroView do software PetSystem

AgendaBuscaView

+ cmbAno : MyJComboBox

+ cmbMes : MyJComboBox

+ cmbDia : MyJComboBox

- Iblmeiom : JLabel

- Iblmeiod : JLabel

- model: AgendaTableModelView

- Ibldata : JLabel

- IBusca: Imagelcon

- data : Date

+ AgendaBuscaView()

+ AgendaTableModelView getModel()

FIGURA 85 - Classe AgendaBuscaView do software PetSystem

AgendaCadastroVlew

- + cmbHora : MyJComboBox
- + cmbMin : MyJComboBox
- + cmbDia : MyJComboBox
- + cmbMes : MyJComboBox
- + cmbAno : MyJComboBox
- Iblmeioh : JLabel
- Iblmeiom : JLabel
- Iblmeiod : JLabel
- + txtAnimal : MyJTextField
- + txtProp : MyJTextField
- + txtProd : MyJTextField
- btnVinculoA : JButton
- btnVinculoP : JButton
- IProd : Imagelcon
- IAnimal: Imagelcon
- + AgendaCadastroView()

FIGURA 86 - Classe AgendaCadastroView do software PetSystem

AgendaTableModelView

- linhas : List<Agenda>
- colunas : String[]
- + AgendaTableModelView()
- + AgendaTableModelView(lista : List<Agenda>)
- + getColumnCount(): int
- + getRowCount(): int
- + getColumnName(columnIndex:int): String
- + getColumnClass(columnIndex: int): Class<?>
- + getValueAt(rowIndex: int, columnIndex: int): Object
- + isCellEditable(rowlndex: int, columnlndex: int): boolean
- + getAgenda(indiceLinha:int): Agenda
- + addAgenda(agenda: Agenda): void
- + removeAgenda(indiceLinha: int): void
- + addListaDeAgendas(agenda: List<Agenda>): void
- + limpar(): void
- + isEmpty(): boolean

FIGURA 87 - Classe AgendaTableModelView do software PetSystem

AnimalEditaView - largura : int altura int. - idanimal: int - idpessoa : int - attribute4 : int + viewcampos : AnimalCadastroView - btnAdd : JButton - btnProp : JButton - IAdd : Imagelcon - IProp : Imagelcon - model: PessoaTableModelView + tblPesquisa : JTable scpResultado : JScrollPane + AnimalEditaView() + mudaComponentes(raca: String, proprietario: String, nomeanimal: String, datanascimento: String, sexo: String, tipo: String): void + setTipo(tipo : String) : void + getNovoNome(): String + getNovaData(): Date + getNovoSexo(): String + getNovoTipo(): String + getNovoRaca(): String + GuardalDAnimal(id:int):void + DevolvelDAnimal(): int + GuardalDPessoa(id:int):void + DevolvelDPessoa(): int + getModel(): PessoaTableModelView

FIGURA 88 - Classe AnimalEditaView do software PetSystem

```
AnimalTableModelView
- linhas : List<Animal>
colunas : String[]
+ AnimalTableModelView()
+ AnimalTableModelView(lista : List<Animal>)
+ getColumnCount(): int
+ getRowCount(): int
+ getColumnName(columnIndex:int): String
+ getColumnClass(columnIndex : int) : Class<?>
+ getValueAt(rowIndex: int, columnIndex: int): Object
+ setValueAt(aValue: Object, rowlndex: int, columnlndex: int): void
+ isCellEditable(rowlndex: int, columnlndex: int): boolean
+ getAnimal(indiceLinha: int): Animal
+ addAnimal(animal: Animal): void
+ removeAnimal(indiceLinha: int): void
+ addListaDeAnimais(animal: List<Animal>): void
+ limpar(): void
+ isEmpty(): boolean
```

FIGURA 89 - Classe AnimalTableModelView do software PetSystem

BuscaFichaView

+ Iblmeiod : JLabel

+ lblmeiom : JLabel

+ cmbDia : MyJComboBox

+ cmbMes : MyJComboBox

+ cmbAno : MyJComboBox

+ Istbusca : JList<String>

+ dftbusca : DefaultListModel<String>

- scplst: JScrollPane

- btnbusca : JButton

- btnprimeiro : JButton

- btnproximo : JButton

- btnanterior : JButton

- btnultimo : JButton

- IBusca : Imagelcon

- IPrimeiro : Imagelcon

- IProximo : Imagelcon

- IAnterior : Imagelcon

IUltimo : Imagelcon

+ BuscaFichaView()

FIGURA 90 - Classe BuscaFichaView do software PetSystem

DiagnosticoCadastroFichaView

+ idanimal: int

+ txtdiag : JTextArea

scrdiag: JScrollPane

+ DiagnosticoCadastroFichaView()

+ GuardalD(id:int):void

+ DevolveID(): int

FIGURA 91 - Classe DiagnosticoCadastroView do software PetSystem

FichalnterfaceView

- largura : int - altura : int - idanimal : int - menu : JMenuBar - mnarq : JMenu

mndiag : JMenultem
 mnvacina : JMenultem
 mnprod : JMenultem
 mnbusca : JMenultem

I : Imagelcon
IDiag : Imagelcon
IVac : Imagelcon
IProd : Imagelcon
IBusca : Imagelcon
pnlcampos : JPanel

+ FichalnterfaceView(animal: String)

FIGURA 92 - Classe FichaInterfaceView do software PetSystem

```
FuncionarioEditaView

- largura :int
- altura :int
- idpessoa : int
- idpessoa : int
- idpessoa : int
- vewcampos : FuncionarioCadastroView
- brh.dd : (Button
- I.dd : Imagelcon

- FuncionarioEditaView()
- setEstado(estado : String) : void
- GetNovoNome() : String
- yetNovoNome() : String
- yetN
```

FIGURA 93 - Classe FuncionarioEditaView do software PetSystem

FuncionarioTableModelView

- linhas : List<Funcionario>

colunas : String[]

- + FuncionarioTableModelView()
- + FuncionarioTableModelView(lista: List<Agenda>)
- + getColumnCount(): int
- + getRowCount(): int
- + getColumnName(columnIndex: int): String
- + getColumnClass(columnIndex:int): Class<?>
- + getValueAt(rowIndex: int, columnIndex: int): Object
- + isCellEditable(rowlndex: int, columnlndex: int): boolean
- + getFunc(indiceLinha: int): Funcionario
- + addFunc(func : Funcionario) : void
- + removeFuncionario(indiceLinha: int): void
- + addListaDeFunc(func : List<Funcionario>) : void
- + limpar(): void
- + isEmpty(): boolean

FIGURA 94 - Classe FuncionarioTableModelView do software PetSystem

MestreCadastroFichaView

- + Iblmeiod : JLabel
- + lblmeiom : JLabel
- + cmbDia : MyJComboBox
- + cmbMes : MyJComboBox
- + cmbAno : MyJComboBox
- + btnAdd : JButton
- IAdd : Imagelcon
- + MestreCadastroFichaView()

FIGURA 95 - Classe MestreCadastroFichaView do software PetSystem

```
PessoaEditaView

- largura : int - diura : int - dipessoa : int - dipessoa
```

FIGURA 96 - Classe PessoaEditaView do software PetSystem

PessoaTableModelView - linhas : List<Pessoa> colunas : String[] + PessoaTableModelView() + PessoaTableModelView(lista : List<Pessoa>) + getColumnCount(): int + getRowCount(): int + getColumnName(columnIndex:int): String + getColumnClass(columnIndex: int): Class<?> + getValueAt(rowIndex: int, columnIndex: int): Object + isCellEditable(rowlndex: int, columnlndex: int): boolean + getPessoa(indiceLinha: int): Pessoa + addPessoa(pessoa : Pessoa) : void + removePessoa(indiceLinha: int): void + addListaDePessoas(pessoa: List<Pessoa>): void + limpar(): void + isEmpty(): boolean

FIGURA 97 - Classe Pessoa Table Model View do software Pet System

ProdutoCadastroFichaView

+ Istprod : JList<String>

+ dftprod : boolean<String>

- model: ProdutoTableModelView

+ tblPesquisa : JTable

scpResultado : JScrollPane

- scplst: JScrollPane

+ txtbusca : MyJTextField

- btnbusca : JButton

- IBusca : Imagelcon+ array : ArrayList<Integer>

+ ProdutoCadastroFichaView()

+ ProdutoTableModelView getModel()

FIGURA 98 - Classe ProdutoCadastroFichaView do software PetSystem

ProdutoEditaView

- largura : int

- altura : int

- idprod : int

- idforn : int

+ viewcampos : ProdutoCadastroView

+ model : FornecedorTableModelView

+ tblPesquisa : JTable

- scpResultado : JScrollPane

- btnAdd : JButton

- btnForn : JButton

- IAdd : Imagelcon

- IForn: Imagelcon

+ ProdutoEditaView()

+ mudaComponentes(nomeprod: String, qntde: int, precoV: float, precoV: float, desc: String, forn: String): void

+ getNovoNome(): String

+ getNovaQuantidade(): int

+ getNovoPrecoC(): float

+ getNovoPrecoV(): float

+ getNovaDescricao(): String

+ GuardalDProduto(id:int):void

+ DevolvelDProduto(): int

+ GuardalDFornecedor(id:int):void

+ DevolvelDFornecedor(): int

+ getModel(): FornecedorTableModelView

FIGURA 99 - Classe ProdutoEditaView do software PetSystem

ProdutoTableModelView

linhas : List<Produto>colunas : String[]

- + ProdutoTableModelView()
- + ProdutoTableModelView(lista: List<Produto>)
- + getColumnCount(): int
- + getRowCount(): int
- + getColumnName(columnIndex: int): String
- + getColumnClass(columnIndex: int): Class<?>
- + getValueAt(rowIndex: int, columnIndex: int): Object
- + isCellEditable(rowlndex: int, columnlndex: int): boolean
- + getProduto(int indiceLinha: int): Produto
- + addProduto(Produto produto : int) : void
- + removeProduto(int indiceLinha: int): void
- + addListaDeProdutos(List<Produto> pessoa : int) : void
- + limpar(): void
- + isEmpty(): boolean

FIGURA 100 - Classe ProdutoTableModelView do software PetSystem

RelatorioInterfaceView

- btnanimal: JButton

- btnProdutosBaixa: JButton

btnSexo : JButtonbtnProdutos : JButtonbtnEstados : JButton

- btnContatos : JButton

- btnGerar : JButton

btnSexoValor : JButtonbtnPropAnimal : JButton

- IstRadios : JList<JRadioButton>

- modeloLista : DefaultListModel<JRadioButton>

- Ibltitulo : JLabel

+ txtProdutos : JTextField + txtSexo : JTextField

+ txtPropAnimal: JTextField

- IblProd : JLabel- IblSexo : JLabel- IblProp : JLabel

+ RelatorioInterfaceView(largura:int, altura:int)

FIGURA 101 - Classe Relatorio Interface
View do $\it software$ PetSystem

SobreInterfaceView

- I: Imagelcon

- Ibltitulo : JLabel

+ SobreInterfaceView()

FIGURA 102 - Classe SobreInterfaceView do software PetSystem

VacinaCadastroFichaView

- + Istvac : JList<String>
- + dftvac : DefaultListModel<String>
- model: VacinaTableModelView
- + tblPesquisa: JTable
- scpResultado : JScrollPane
- scplst: JScrollPane
- + txtbusca : MyJTextField
- btnBusca : JButton
- IBusca : Imagelcon+ array : ArrayList<Integer>
- + VacinaCadastroFichaView()
- + VacinaTableModelView getModel()

FIGURA 103 - Classe VacinaCadastroFichaView do software PetSystem

VacinaEditaView

- largura : int
- altura : int
- idvacina : int
- viewcampos : VacinaCadastroView
- btnAdd : JButton
- IAdd : Imagelcon
- + VacinaEditaView()
- + mudaComponentes(nomevacina: String, preco: Float, prevencao: String, tempo: String, idade: String, descricao: String): void
- + getNovoNome(): String
- + getNovoPreco(): float
- + getNovoPrevencao(): String
- + getNovaldade(): String
- + getNovaDescricao(): String
- + GuardalD(id : int) : void
- + DevolvelD(): int

FIGURA 104 - Classe VacinaEditaView do software PetSystem

VacinaTableModelView

linhas : List<Vacina>colunas : String[]

- + VacinaTableModelView()
- + VacinaTableModelView(lista: List<Vacina>)
- + getColumnCount(): int
- + getRowCount(): int
- + getColumnName(columnIndex:int): String
- + getColumnClass(columnIndex: int): Class<?>
- + getValueAt(rowIndex: int, columnIndex: int): Object
- + isCellEditable(rowlndex: int, columnlndex: int): boolean
- + getVacina(indiceLinha: int): Vacina
- + addVacina(vacina: Vacina): void
- + removeVacina(indiceLinha: int): void
- + addListaDeVacinas(vacina: List<Vacina>): void
- + limpar(): void
- + isEmpty(): boolean

FIGURA 105 - Classe VacinaTableModelView do software PetSystem

VinculoAnimalView

- altura : int

- largura : int

- btnBusca : JButton

- btnEscolher: JButton

+ txtBusca : MyJTextField

- pnlfundo : JPanel

- IBusca: Imagelcon

- IEscolher: Imagelcon

model: AnimalTableModelView

+ tblPesquisa: JTable

scpResultado : JScrollPane

+ VinculoAnimalView()

+ AnimalTableModelView getModel()

FIGURA 106 - Classe VinculoAnimalView do software PetSystem

VinculoFornecedorView

+ tblPesquisa : JTable

- largura : int - altura : int

scpResultado : JScrollPane

model: FornecedorTableModelView

+ txtProp : MyJTextField - btnBusca : JButton - btnEscolher : JButton

- IBusca : Imagelcon- IEscolher : Imagelcon

- pnlfundo : JPanel

+ VinculoFornecedorView()

+ getModel(): FornecedorTableModelView

FIGURA 107 - Classe VinculoFornecedorView do software PetSystem

VinculoProdutoView

+ tblPesquisa : JTable

- largura : int - altura : int

- scpResultado : JScrollPane

- model: ProdutoTableModelView

+ txtProp : MyJTextField

btnBusca : JButtonbtnEscolher : JButton

- IBusca : Imagelcon - IEscolher : Imagelcon

- pnlfundo : JPanel

+ VinculoProdutoView()

+ getModel(): ProdutoTableModelView

FIGURA 108 - Classe VinculoProdutoView do software PetSystem

VinculoProprietarioView

+ tblPesquisa : JTable

- largura : int - altura : int

- scpResultado : JScrollPane

model: PessoaTableModelView

+ txtProp : MyJTextField - btnBusca : JButton

btnEscolher : JButtonIBusca : Imagelcon

- IEscolher : Imagelcon
 - pnlfundo : JPanel

+ VinculoProprietarioView()

+ getModel(): PessoaTableModelView

FIGURA 109 - Classe VinculoProprietarioView do software PetSystem

FornecedorBuscaView

IAlterar : Imagelcon

model : FornecedorTableModelView

+ FornecedorBuscaView()

+ getModel(): FornecedorTableModelView

FIGURA 110 - Classe FornecedorBuscaView do software PetSystem

| Interest | Interest

FIGURA 111 - Classe FornecedorEditaView do software PetSystem

FornecedorTableModelView

- linhas : List<Fornecedor>

- colunas : String[]

- + FornecedorTableModelView()
- + FornecedorTableModelView(lista: List<Fornecedor>)
- + getColumnCount(): int
- + getRowCount(): int
- + getColumnName(columnIndex: int): String
- + getColumnClass(columnIndex: int): Class<?>
- + getValueAt(rowIndex: int, columnIndex: int): Object
- + isCellEditable(rowlndex: int, columnlndex: int): boolean
- + getFornecedor(indiceLinha: int): Fornecedor
- + addFornecedor(forn : Fornecedor) : void
- + removeFornecedor(indiceLinha: int): void
- + addListaDeFornecedores(forn : List<Fornecedor>) : void
- + limpar(): void
- + isEmpty(): boolean

FIGURA 112 - Classe FornecedorTableModelView do software PetSystem

FornecedorCadastroView

- txtCnpj : MyJTextField
- txtlnsc : MyJTextField
- txtNome : MyJTextField
- txtRua : MyJTextField
- txtComplemento : MyJTextField
- txtNumero : MyJTextField
- txtCep : MyJTextField
- txtEmail : MyJTextField
- txtTel: MyJTextField
- txtCel : MyJTextField
- cmbEstado : MyJComboBox
- cmbCidade : MyJComboBox
- cmbBairro : MvJComboBox
- btnCidade : JButton
- btnBairro : JButton
- IAdicionar : Imagelcon
- + FornecedorCadastroView()

FIGURA 113 - Classe Fornecedor Cadastro View do software Pet System

FuncionarioBuscaView

- IAlterar : Imagelcon
- model: FuncionarioTableModelView
- + FuncionarioBuscaView()
- + getModel(): FuncionarioTableModelView

FIGURA 114 - Classe FuncionarioBuscaView do software PetSystem

Funcionario Cadastro View

- cmbestado : MyJComboBox
- cmbcidade : MyJComboBox
- btnCidade : JButton
- btnBairro : JButton
- cmbbairro : MyJComboBox
- txtnome : MyJTextField
- txtcpf : MyJTextField
- txtnascimento : MyJTextField
- txtrua : MyJTextField
- txtcomplemento : MyJTextField
- txtcep: MyJTextField
- txtnumero : MyJTextField
- txtemail: MyJTextField
- txttelefone : MyJTextField
- txtcelular : MyJTextField
- txtlogin : MyJTextField
- txtsenha: MyJTextField
- txtsalario : MvJTextField
- cmbtipo : MyJComboBox
- radMasc : JRadioButton
- radFem : JRadioButton
- grupo : ButtonGroup
- IAdicionar : Imagelcon
- + FuncionarioCadastroView()

FIGURA 115 - Classe FuncionarioCadastroView do software PetSystem

VacinaBuscaView

- model: VacinaTableModelView
- + VacinaBuscaView()
- + getModel(): VacinaTableModelView

FIGURA 116 - Classe VacinaBuscaView do software PetSystem

VacinaCadastroView

- txtVacina: MyJTextField
- txtPreco: MyJTextField
- txtDes: MyJTextField
- txtldade : MyJTextField
- txtPrevencao : MyJTextField
- cmbTempo : MyJComboBox
- + VacinaCadastroView()

FIGURA 117 - Classe VacinaCadastroView do software PetSystem

8.3.2. Model

Nesta seção, serão apresentadas, por meio das Figuras 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132 e 133, as classes do *software* PetSystem relativas ao módulo Model do modelo MVC-DAO adotado para desenvolvimento da codificação.

Animal - Id_Animal: int - Id_Tipo : int - Id_Pessoa : int - Id_Diagnostico: int - Nome_Animal: String - Data_Nascimento : String - Sexo : String + Animal() + Animal(id_Animal: int, id_Tipo: int, id_Pessoa: int, nome_Animal: String, data_Nascimento: String, sexo: String) + getId_Animal(): int + getId_Tipo(): int + getId_Pessoa(): int + getId_Diagnostico(): int + setId_Diagnóstico(id_Diagnóstico: int): void + getSexo(): String + getNome_Animal(): String + getData_Nascimento(): String + setSexo(sexo : String) : void + setNome_Animal(nome_Animal: String): void + setData_Nascimento(data_Nascimento: String): void

FIGURA 118 - Classe Animal do software PetSystem

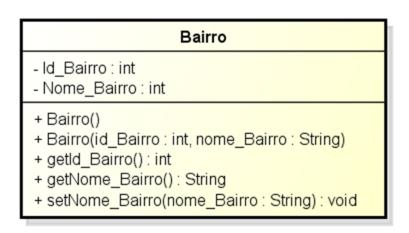


FIGURA 119 - Classe Busca do software PetSystem

Cidade

- Id_Cidade: intNome_Cidade: int
- _____

+ Cidade()

- + Cidade(id_Cidade:int, nome_Cidade: String)
- + getId_Cidade(): int
- + getNome Cidade(): String
- + setNome Cidade(nome Cidade: String): void

FIGURA 120 - Classe Cidade do software PetSystem

Contato

- Id_Contato : int
- Id_Pessoa : int
- Contato : String
- + Contato()
- + Contato(id_Contato:int,id_Pessoa:int,contato:String)
- + getId_Contato(): int
- + getId Pessoa(): int
- + getContato(): String
- + setContato(contato: String): void

FIGURA 121 - Classe Contato do software PetSystem

Diagnostico

- Id_Diagnostico: int
- Id_Produto: int
- Data_Diagnostico: String
- + Diagnostico()
- + Diagnostico(id_Diagnostico: int, id_Produto: int, data_Diagnostico: String)
- + getId_Diagnostico(): int
- + getId Produto(): int
- + getData_Diagnostico(): String
- + setData_Diagnostico(data_Diagnostico: String): void

FIGURA 122 - Classe Diagnóstico do software PetSystem

Estado

- Id_Estado : int

- Nome_Estado: String

- Uf : String

+ Estado()

+ Estado(id_Estado: int, nome_Estado: String, uf: String)

+ getId Estado(): int

+ getNome_Estado(): String

+ getUf(): String

+ setNome_Estado(nome_Estado: String): void

+ setUf(uf : String) : void

FIGURA 123 - Classe Estado do software PetSystem

```
Funcionario

- Salario : float
- Classificacao : String

+ Funcionario()
- Funcionario() - Pessoa : int, nome_Pessoa : String, cpf : String, data_Nascimento : String, rua : String, complemento : String, numero : String, sexo : char, salario : float, classificacao : String)
- getSalario() : float
- getClassificacao() : String
- setSalario(salario : float) : void
- setClassificacao(: String) : void
- setClassificacao : String) : void
```

FIGURA 124 - Classe Funcionario do software PetSystem

```
| Id_Pessoa int | Id_Pessoa String | Id_Pessoa Id_Pes
```

FIGURA 125 - Classe Pessoa do software PetSystem

Produto

- Id_Produto : intQuantidade : int
- Nome_Produto : String - Descricao : String
- Preco : float
- + Produto()
- + Produto(id_Produto: int, quantidade: int, nome_Produto: String, descricao: String, preco: float)
- + getId_Produto(): int + getQuantidade(): int
- + setQuantidade(quantidade: int): void
- + getNome_Produto(): String
- + setNome_Produto(nome_Produto: String): void
- + getDescricao(): String
- + setDescricao(descricao: String): void
- + getPreco(): float
- + setPreco(preco : float) : void

FIGURA 126 - Classe Produto do software PetSystem

TipoAnimal

- Id_TipoAnimal: int
- Nome_TipoAnimal: String
- + TipoAnimal()
- + TipoAnimal(id_TipoAnimal: int, nome_TipoAnimal: String)
- + getId_TipoAnimal(): int
- + getNome_TipoAnimal(): String
- + setNome_TipoAnimal(nome_TipoAnimal: String): void

FIGURA 127 - Classe TipoAnimal do software PetSystem

Usuario

- Id_Usuario : intCategoria : StringSenha : String
- + Usuario()
- + Usuario(id_Usuario: int, categoria: String, senha: String)
- + getId Usuario(): int
- + setId_Usuario(id_Usuario: int): void
- + getCategoria(): String
- + setCategoria(categoria: String): void
- + getSenha(): String
- + setSenha(senha : String) : void

FIGURA 128 - Classe Usuario do software PetSystem

Agenda - Id_Agenda; int: int - Horario : String - Data : Date - Id_Animal : int - Id_Produto : int + Agenda() + Agenda(id_Agenda: int, horario: String, data: Date, id_Animal: int, id_Produto: int) + Agenda(id_Agenda: int, horario: String, data: Date, animal: String, proprietario: String, produto: String, id_Animal: int, id_Produto: int) + getId_Agenda(): int + setId_Agenda(id_Agenda:int):void + getHorario(): String + setHorario(horario: String): void + getId_Animal(): int + setId_Animal(id_Animal: int): void + getId_Produto(): int + setId_Produto(id_Produto:int): void + getAnimal(): String + setAnimal(animal: String): void + getProprietario(): String + setProprietario(proprietario: String): void + getProduto(): String + setProduto(produto: String): void + getData(): Date + setData(data : Date) : void

FIGURA 129 - Classe Agenda do software PetSystem

Raca

- ld_Raca : int
- ld_Tipoanimal : int
- Nome : String
- + Raca()
- + Raca(id_Raca: int, id_Tipoanimal: int, nome: String)
- + getId Raca(): int
- + setId Raca(id Raca:int):void
- + getId Tipoanimal(): int
- + setId_Tipoanimal(id_Tipoanimal: int): void
- + getNome(): String
- + setNome(nome : String) : void

FIGURA 130 - Classe Raça do software PetSystem

Usados

- Data : Date
- Id Animal: int
- Id Produto : int
- Id_Vacina: int
- + Usados()
- + Usados(data: Date, id_Animal: int, id_Produto: int, id_Vacina: int)
- + getData(): Date
- + setData(data : Date) : void
- + getId_Animal(): int
- + setId_Animal(id_Animal:int):void
- + getId_Produto(): int
- + setId_Produto(id_Produto:int):void
- + getId_Vacina(): int
- + setId_Vacina(id_Vacina:int):void

FIGURA 131 - Classe Usados do software PetSystem

Vacina - Id_Vacina: int - Preco : Float - Nome : String - Descricao : String - Idade : String - Prevencao : String + Vacina() + Vacina(id_Vacina: int, preco: float, nome: String, descricao: String, idade: String, prevencao: String) + getPreco(): Float + setPreco(preco : float) : void + getNome(): String + setNome(nome : String) : void + getDescricao(): String + setDescricao(descricao: String): void + getId_Vacina(): int + setId_Vacina(id_Vacina: int): void + getIdade(): String + setIdade(idade : String) : void + getPrevencao(): String + setPrevencao(prevencao : String) : void

FIGURA 132 - Classe Vacina do software PetSystem

```
Fornecedor
  Id_Fornecedor:int
 - Id Bairro int
  Cnpj : String
 - InscricaoEstadual : String
- Nome : String
- Rua : String
- Numero : String
- Complemento : String
 - Cep : String
+ Fornecedor()
+ Fornecedor(id_Fornecedor: int, id_Bairro: int, cnpj: String, inscricaoEstadual: String, nome: String, numero: String, complemento: String, cep: String): void
+ Fornecedor(id_Fornecedor: int, cnpj: String, inscricaoEstadual: String, nome: String, email: String, tel: String, cel: String): void
+ getId_Fornecedor() : int
+ setId_Fornecedor(id_Fornecedor : int) : void
+ getId_Bairro() : int
+ setId_Bairro(id_Bairro : int) : void
+ getCnpj(): String
+ setCnpj(cnpj: String): void
+ getInscricaoEstadual() : String
+ setInscricaoEstadual(inscricaoEstadual : String) : void
+ getNome(): String
+ setNome(nome: String): void
+ getRua(): String
+ setRua(rua: String): void
+ getNumero(): String
+ setNumero(numero: String): void
+ getComplemento() : String
+ setComplemento(complemento : String) : void
+ setComplemento(complement
+ getCep(): String
+ setCep(cep: String): void
+ getEmail(): String
+ setEmail(email: String): void
+ getTel(): String
+ setTel(tel: String): void
 + getCel(): String
+ getCel(): String
+ setCel(cel: String): void
```

FIGURA 133 - Classe Fornecedor do software PetSystem

8.3.3. Controller

Nesta seção, serão apresentadas as classes do *software* PetSystem relativas ao módulo Controller do modelo MVC-DAO adotado para desenvolvimento da codificação, que estão ilustradas nas Figuras 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147 e 148.

AnimalController

- VisaoCadastro : AnimalCadastroView
- VisaoBusca: AnimalBuscaView
- VisaoVinculo: VinculoBuscaView
- largura : int
- altura : int
- idtipo : int
- Posicaobusca: int
- idpessoa : int
- array : ArrayList
- + AnimalController(visao : AnimalCadastroView)
- + AnimalController(visao : AnimalBuscaView)
- + AnimalController(visao : VinculoBuscaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 134 - Classe AnimalController do software PetSystem

HomeController

- Visao : HomeInterfaceView
- cadastroPessoa : PessoaCadastroView
- buscaPessoa : PessoaBuscaView
- cadastroAnimal: AnimalCadastroView
- buscaAnimal: AnimalBuscaView
- cadastroProduto : ProdutoCadastroView
- buscaProduto: ProdutoBuscaView
- relatorio : RelatorioInterfaceView
- array : ArrayList
- + HomeController(visao : HomeInterfaceView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 135 - Classe HomeController do software PetSystem

LoginController

- Visao : LoginInterface
- + LoginController(visao : LoginInterfaceView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 136 - Classe LoginController do software PetSystem

PessoaController

- VisaoCadastro : PessoaCadastroView
- VisaoBusca: PessoaBuscaView
- PosicaoBusca : int
- encontrado : boolean
- strbusca : StringlblExibe : JLabel
- + PessoaController(visao : PessoaCadastroView)
- + PessoaController(visao : PessoaBuscaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 137 - Classe PessoaController do software PetSystem

ProdutoController

- VisaoCadastro : ProdutoCadastroView
- VisaoBusca: ProdutoBuscaView
- + ProdutoController(visao : ProdutoCadastroView)
- + ProdutoController(visao: ProdutoBuscaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 138 - Classe ProdutoController do software PetSystem

RelatorioController

- Visao : RelatorioInterfaceView
- parametros : Map
- + RelatorioController(visao : RelatorioInterfaceView)
- + actionPerformed(arg 0 : ActionEvent) : void

FIGURA 139 - Classe RelatorioController do software PetSystem

AgendaController

- VisaoBusca: AgendaBuscaView
- VisaoCadastro : AgendaCadastroView
- VisaoVinculo: VinculoAnimalView
- vinculoProd : VinculoProdutoView
- + idanimal: int
- + idpessoa : int
- + idproduto : int
- + AgendaController(visao : AgendaBuscaView)
- + AgendaController(visao : AgendaCadastroView)
- + AgendaController(visao : VinculoAnimalView)
- + AgendaController(visao : VinculoProdutoView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 140 - Classe AgendaController do software PetSystem

BuscaFichaController

- VisaoBusca: BuscaFichaView
- + posicaobusca: int
- + BuscaFichaController(visao : BuscaFichaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 141 - Classe BuscaFichaController do software PetSystem

FichaController

- viewcadastro : DiagnosticoCadastroFichaView
- view : FichalnterfaceView
- cadastroprod : ProdutoCadastroFichaView
- cadastrovac : VacinaCadastroFichaView
- busca: BuscaFichaView
- + FichaController(visao : ProdutoCadastroFichaView)
- + FichaController(visao : DiagnosticoCadastroFichaView)
- + FichaController(visao : FichaInterfaceView)
- + FichaController(visao : BuscaFichaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 142 - Classe FichaController do software PetSystem

LoginController

- Visao : LoginInterfaceView
- home : HomeInterfaceView
- + LoginController(visao : LoginInterfaceView)
- + LoginController(visao : HomeInterfaceView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 143 - Classe LoginController do software PetSystem

ProdutoFichaController | ProdutoFichaController | ProductoFichaController | ProductoFichaControl

- cadastroprod : ProdutoCadastroFichaView
- + ProdutoFichaController(visao : ProdutoCadastroFichaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 144 - Classe ProdutoFichaController do software PetSystem

RelatorioController

- viewRelatorio : RelatorioInterfaceView
- visaohome : HomeInterfaceView
- + RelatorioController(visao : RelatorioInterfaceView)
- + RelatorioController(visao : HomeInterfaceView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 145 - Classe RelatorioController do software PetSystem

VacinaFichaController

- viewcadastro : VacinaCadastroFichaView
- + VacinaFichaController(visao : VacinaCadastroFichaView)
- + actionPerformed(arg0 : ActionEvent) : Void

FIGURA 146 - Classe VacinaFichaController do software PetSystem

FornecedorController

- VisaoCadastro : FornecedorCadastroView
- VisaoBusca: FornecedorBuscaView
- VisaoVinculo: VinculoFornecedorView
- VisaoEdita: FornecedorEditaView
- idforn : int
- e : String
- t : String
- c : String
- + FornecedorController(visao : FornecedorCadastroView)
- + FornecedorController(visao : FornecedorBuscaView)
- + FornecedorController(visao : FornecedorEditaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 147 - Classe FornecedorController do software PetSystem

Funcionario Controller

- VisaoCadastro : FuncionarioCadastroView
- VisaoBusca: FuncionarioBuscaView
- VisaoEdita: FuncionarioEditaView
- e : String
- t : String
- c : String
- + FuncionarioController(visao : FuncionarioCadastroView)
- + FuncionarioController(visao : FuncionarioBuscaView)
- + FuncionarioController(visao : FuncionarioEditaView)
- + actionPerformed(arg0 : ActionEvent) : void

FIGURA 148 - Classe FuncionarioController do software PetSystem

8.3.4. DAO

Nesta seção, serão apresentadas, através das Figuras 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164 e 165, as classes do *software* PetSystem relativas ao módulo DAO do modelo MVC-DAO, adotado para desenvolvimento da codificação.

AnimalDAO

- pstm : PreparedStatement
- + insert(animal: Animal): boolean
- + update(animal: Animal): boolean
- + delete(animal: Animal): boolean
- + retrieve All(): ArrayList

FIGURA 149 - Classe AnimalDAO do software PetSystem

BairroDAO

- pstm : PreparedStatement
- + retrieveBairros(fkcidade : int) : ArrayList
- + retrieveld(nome : String) : int
- + retriveNome(id:int): String
- + insert(novobairro : String, idcidade : int) : void

FIGURA 150 - Classe BairroDAO do software PetSystem

CidadeDAO

- pstm : PreparedStatement
- + retrieveCidades(fkestado:int): ArrayList
- + retrieveld(nome : String) : int
- + insert(novacidade : String, id : int) : void
- + retrieveNome(id:int): String

FIGURA 151 - Classe CidadeDAO do software PetSystem

ContatoDAO

- pstm : PreparedStatement
- + insert(contato : Contato) : boolean

FIGURA 152 - Classe ContatoDAO do software PetSystem

EstadoDAO

- pstm : PreparedStatement
- + retrieveAll() : ArrayList
- + Retrieveld(nome : String) : int
- + retrieveNome(id:int): String

FIGURA 153 - Classe EstadoDAO do software PetSystem

PessoaDAO

- pstm : PreparedStatement
- + insert(pessoa : Pessoa) : boolean
- + update(pessoa : Pessoa) : boolean
- + delete(idpessoa : int) : boolean
- + retrieve All(): ArrayList
- + retrieveld(nome : String) : int

FIGURA 154 - Classe PessoaDAO do software PetSystem

ProdutoDAO

- pstm : PreparedStatement
- + insert(produto : Produto) : boolean
- + update(produto : Produto) : Boolean
- + delete(produto : Produto) : boolean

FIGURA 155 - Classe ProdutoDAO do software PetSystem

TipoAnimalDA0

- pstm : PreparedStatement
- + retrieveTipos(): ArrayList
- + retrieveld(tipo: String): int

FIGURA 156 - Classe TipoAnimalDAO do software PetSystem

ConnectionFactory

- Con: Connection

- Pstm : PreparedStatement

- Rs : ResultSet

+ getConnection(): Connection

+ closeConnection(): void

+ closeConnection(pstm : PreparedStatement) : void

+ claseConnection(pstm: PreparedStatement, rs: ResultSet): void

FIGURA 157 - Classe ConnectionFactory do software PetSystem

AgendaDAO

pstm : PreparedStatement

+ insert(agenda : Agenda) : boolean

+ busca(data : Date) : ArrayList<Agenda>

+ buscaDia(dia: String): ArrayList<Agenda>

+ buscaMes(mes : String) : ArrayList<Agenda>

+ buscaAno(ano : String) : ArrayList<Agenda>

+ buscaDiaMes(dia: String, mes: String): ArrayList<Agenda>

+ buscaDiaAno(dia: String, ano: String): ArrayList<Agenda>

+ buscaMesAno(mes : String, ano : String) : ArrayList<Agenda>

+ busca(dia: String, mes: String, ano: String): ArrayList<Agenda>

FIGURA 158 - Classe AgendaDAO do software PetSystem

LoginDAO

- pstm : PreparedStatement

+ retrieveRacas(): ArrayList<String>

+ retrieveRacas(idtipo : int) : ArrayList<String>

+ retrieveld(tipo : String) : int

+ insert(novaraca: String, idtipo:int): void

+ retrieveldTipo(idraca: int): int

FIGURA 159 - Classe LoginDAO do software PetSystem

DiagnosticoDAO

- pstm : PreparedStatement
- + insert(diag : Diagnostico) : boolean
- + buscaMA(mes: String, ano: String, idanimal: int): ArrayList<Diagnostico>
- + buscaDMA(dia: String, mes: String, ano: String, idanimal: int): ArrayList<Diagnostico>
- + retrieveAllDatas(idanimal: int): ArrayList<String>
- + retrieve All Data (idanimal: int): ArrayList < Date >
- + retrieveDiag(d : Date, idanimal : int) : ArrayList<Diagnostico>
- + delete(idanimal:int): boolean

FIGURA 160 - Classe DiagnosticoDAO do software PetSystem

FornecedorDAO

- pstm : PreparedStatement
- + insert(forn : Fornecedor) : boolean
- + retrieveld(forn : Fornecedor) : int
- + retrieveBusca(busca: String): ArrayList<Fornecedor>
- + delete(idforn : int) : boolean
- + retrieveFornecedor(idforn : int) : Fornecedor
- + retrievelds(): ArrayList<Integer>
- + update(forn : Fornecedor, idforn : int) : boolean
- + retrieveAll(): ArrayList<Fornecedor>

FIGURA 161 - Classe FornecedorDAO do software PetSystem

Funcionario DAO

- pstm : PreparedStatement
- + insert(func : Funcionario) : boolean
- + retrieve All(): ArrayList<Funcionario>
- + retrieveBusca(busca : String) : ArrayList<Funcionario>
- + delete(idusuario : int) : boolean
- + retrieveFunc(idusuario : int) : Funcionario
- + update(func : Funcionario, idfunc : int) : boolean

FIGURA 162 - Classe FuncionarioDAO do software PetSystem

RacaDAO

- pstm : PreparedStatement
- + retrieveRacas(): ArrayList<String>
- + retrieveRacas(idtipo : int) : ArrayList<String>
- + retrieveld(tipo : String) : int
- + insert(novaraca: String, idtipo: int): void
- + retrieveldTipo(idraca:int):int

FIGURA 163 - Classe RacaDAO do software PetSystem

UsadosDAO

- pstm : PreparedStatement
- + insertP(idproduto : int, idanimal : int, data : Date) : boolean
- + insertV(idvacina : int, idanimal : int, data : Date) : boolean
- + PbuscaMA(mes: String, ano: String, idanimal: int): ArrayList<Usados>
- + PbuscaDMA(dia: String, mes: String, ano: String, idanimal: int): ArrayList<Usados>
- + VbuscaMA(mes: String, ano: String, idanimal: int): ArrayList<Usados>
- + VbuscaDMA(dia: String, mes: String, ano: String, idanimal: int): ArrayList<Usados>
- + retrieveUsadosP(d : Date, idanimal : int) : ArrayList<Usados>
- + retrieveUsadosV(d : Date, idanimal : int) : ArrayList<Usados>

FIGURA 164 - Classe UsadosDAO do software PetSystem

VacinaDA0

- pstm : PreparedStatement
- + insert(vacina : Vacina) : boolean
- + retrieveBusca(busca: String): ArrayList<Vacina>
- + retrieveAll(): ArrayList<Vacina>
- + delete(idvacina: int): boolean
- + retrieveVacina(idvacina: int): Vacina
- + update(vacina: Vacina, idvacina: int): boolean

FIGURA 165 - Classe VacinaDAO do software PetSystem

8.3.5. Exception

Nesta seção, serão apresentadas, por meio das Figuras 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180 e 181, as classes do *software* PetSystem relativas ao módulo *Exception* do modelo MVC-DAO, adotado para desenvolvimento da codificação.

AnimalDAOException

- + AnimalDAOException()
- + AnimalDAOException(message : String)

FIGURA 166 - Classe AnimalDAOException do software PetSystem

BairroDAOException

- + BairroDAOException()
- + BairroDAOException(message : String)

FIGURA 167 - Classe BairroDAOException do software PetSystem

CidadeDAOException

- + CidadeDAOException()
- + CidadeDAOException(message: String)

FIGURA 168 - Classe CidadeDAOException do software PetSystem

ContatoDAOException

- + ContatoDAOException()
- + ContatoDAOException(message: String)

FIGURA 169 - Classe ContatoDAOException do software PetSystem

DiagnosticoDAOException

- + DiagnosticoDAOException()
- + DiagnosticoDAOException(message : String)

FIGURA 170 - Classe DiagnosticoDAOException do software PetSystem

EstadoDAOException

- + EstadoDAOException()
- + EstadoDAOException(message : String)

FIGURA 171 - Classe EstadoDAOException do software PetSystem

Funcionario DAO Exception

- + FuncionarioDAOException()
- + FuncionarioDAOException(message: String)

FIGURA 172 - Classe FuncionarioDAOException do software PetSystem

PessoaDAOException

- + PessoaDAOException()
- + PessoaDAOException(message: String)

FIGURA 173 - Classe PessoaDAOException do software PetSystem

ProdutoDAOException

- + ProdutoDAOException()
- + ProdutoDAOException(message: String)

FIGURA 174 - Classe ProdutoDAOException do software PetSystem

TipoAnimalDAOException

- + TipoAnimalDAOException()
- + TipoAnimalDAOException(message: String)

FIGURA 175 - Classe TipoAnimalDAOException do software PetSystem

UsuarioDAOException

- + UsuarioDAOException()
- + UsuarioDAOException(message: String)

FIGURA 176 - Classe UsuarioDAOException do software PetSystem

FornecedorDAOException

- + FornecedorDAOException()
- + FornecedorDAOException(message: String)

FIGURA 177 - Classe FornecedorDAOException do software PetSystem

LoginDAOException

- + LoginDAOException()
- + LoginDAOException(message: String)

FIGURA 178 - Classe LoginDAOException do software PetSystem

RacaDAOException

- + RacaDAOException()
- + RacaDAOException(message: String)

FIGURA 179 - Classe RacaDAOException do software PetSystem

UsadosDAOException

- + UsadosDAOException()
- + UsadosDAOException(message : String)

FIGURA 180 - Classe UsadosDAOException do software PetSystem

VacinaDAOException

- + VacinaDAOException()
- + VacinaDAOException(message : String)

FIGURA 181 - Classe VacinaDAOException do software PetSystem

8.3.6. Componentes

Nesta seção, serão apresentadas as classes do *software* PetSystem relativas ao componentes desenvolvidos para o *design* do programa, que podem ser vistas nas Figuras 182 e 183.

MyJComboBox

- + MyJComboBox(primeiroitem : String, left : int, top : int)
- + MyJComboBox(left:int,top:int,tam:int)

FIGURA 182 - Classe MyJComboBox do software PetSystem

MyJTextField

- ocupado : int
- + MyJTextField(textointerno : String, left : int, top : int)
- + MyJTextField(textointerno : String, left : int, top : int, tam : int)

FIGURA 183 - Classe MyJTextField do software PetSystem

8.3.7. Default

Nesta seção, será apresentada a classe de execução do *software* PetSystem, que está ilustrada na Figura 184.

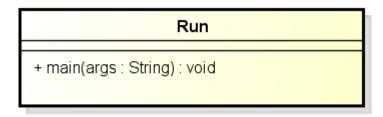


FIGURA 184 - Classe Run do software PetSystem