GABRIEL FERNANDES RIOS

DESENVOLVIMENTO DE SOFTWARE PARA CLÍNICA VETERINÁRIA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação.

Orientador: Fernando Cesar de Lima

Área de Concentração: _____

Assis

2010

FICHA CATALOGRÁFICA

RIOS, Gabriel Fernandes

Desenvolvimento de Software para Clínica Veterinária / Gabriel Fernandes Rios. Fundação Educacional do Municipio de Assis – FEMA – Assis, 2010.

(Quantidade de páginas) p.

Orientador: Fernandes Cesar de Lima

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis

1.Software. 2. Clínica Veterinária.

CDD: 001.61

Biblioteca FEMA

DESENVOLVIMENTO DE SOFTWARE PARA CLÍNICA VETERINÁRIA

Gabriel Fernandes Rios

Trabalho de Conclusão de Curso aprensentado ao Instituto Minicipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Fernando Cesar de Lima

Analisador (1): Alex Sandro Romeo de Souza Poletto

Assis
2010

AGRADECIMENTOS

Aos familiares pelo apoio durante esses três anos de curso.

A minha namorada Patrícia Porto Nakamoto, pela paciência e incentivo.

Ao professor Fernando Cesar de Lima, pela orientação e pelas oportunidades.

A todos os meus amigos, principalmente Ivan Camolesi e Tiago Almeida Bungenstab (Roseta) pela ajuda neste trabalho.

RESUMO

Neste projeto será desenvolvido um sistema informatizado que auxiliará o médico veterinário na avaliação dos animais, com a finalidade de guardar todas as informações que achar conveniente para o acompanhamento da saúde dos mesmos, por intermédio dos registros das consultas, vacinas, exames, aplicações. O sistema será desenvolvido em Java com foco na web, levando a empresa a necessitar apenas de um navegador comum para a utilização do sistema.

Palavra-chave: Sistema; Veterinário; Java; Web.

ABSTRACT

This project will develop a computerized system which will assist the veterinarian in animal evaluation in order to store all the information they find suitable for monitoring health of themselves, through the records of consultations, vaccinations, examinations, applications. The system will be developed in Java with focus on the web, leading company need only a web browser to use the system.

Keywords: System; Veterinarian; Java; Web.

LISTA DE ILUSTRAÇÕES

Figura 1. Modelo MVC 18
Figura 2. Estrutura de arquivos após descompactar o Eclipse 19
Figura 3. Caso de Uso – Visão Geral do Software 22
Figura 4. Caso de Uso - Relatórios 23
Figura 5. Diagrama de Classe 24
Figura 6. Diagrama Entidade-Relacionamento 25
Figura 7. Página de Login 26
Figura 8. Página Principal do Sistema 26
Figura 9. Página de Cadastro de Animal 27
Figura 10. Página de Agendamento 35
Figura 11. Página de Consulta Clínica 36
Figura 12. Página Meu Animal 36
Figura 13. Página Histórico do Animal 37
Figura 14. Cronograma 39

LISTA DE ABREVIATURAS E SIGLAS

IDE	Integrated Development Environment
OS	Operating System
JVM	Java Virtual Machine
JSP	JavaServer Pages
JSF	JavaServer Faces
HTML	HyperText Markup Language
XML	Extensible Markup Language
API	Application Programming Interface
MVC	Model, View, Controller
SQL	Structure Query Language
SWT	Standard Widget Toolkit
JEE	Java Enterprise Edition
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO 10
1.1 OBJETIVO 11
1.2 ESCOPO 11
1.3 PÚBLICO ALVO 11
1.4 JUSTIFICATIVA E MOTIVAÇÃO 11
1.5 ESTRUTURA DO TRABALHO 12
2 LEVANTAMENTO DE REQUISITOS 13
3 DESENVOLVIMENTO DO SISTEMA 13
3.1 PERSPECTIVA DO SISTEMA 13
3.2 PRIORIDADES NA IMPLEMENTAÇÃO 14
3.3 RESULTADOS ESPERADOS NA IMPLANTAÇÃO DO SISTEMA 14
4 MÉTODO DE DESENVOLVIMENTO 14
4.1 TÉCNOLOGIA JAVA 14
4.1.1 Linguagem de programação Java 15
4.1.2 JavaServer Pages (JSP) 15
4.1.3 JavaServer Faces (JSF) 15
4.1.4 Jasper Report 16
4.1.5 iReport 16
4.2 PADRÃO MVC 17
4.3 HIBERNATE 18
4.4 TOMCAT 18
4.5 ECLIPSE 19
4.6 HSQLDB 20
4.7 JUDE 20
5 ESTRUTURA DE DESENVOLVIMENTO DO SISTEMA 20
6 DIAGRAMAS 21
6.1 DIAGRAMAS DE CASO DE USO 21
6.1.1 Caso de Uso – Visão Geral do Software 22
6.1.2 Caso de Uso – Relatórios 23
6.2 DIAGRAMA DE CLASSE 24
6.3 DIAGRAMA ENTIDADE-RELACIONAMENTO 25
7 INTERFACE DO SOFTWARE - PÁGINAS 26
7.1 PÁGINA DE LOGIN 26
7.2 PÁGINAS DO SISTEMA 26
7.2.1 Página Principal 26
7.2.2 Página de Cadastro de Animal 27
7.2.3 Página de Agendamento 35
7.2.4 Página de Consulta Clínica 36
7.3 PÁGINAS DO CLIENTE WEB 36
7.3.1 Página Meu Animal 36
7.3.2 Página do Histórico do Animal 37
8 CONCLUSÃO 38
CRONOGRAMA 39
REFERÊNCIAS 40

1 INTRODUÇÃO

A medicina veterinária é uma ciência que se dedica à prevenção, controle, erradicação e tratamento de doenças, traumatismos ou qualquer outro agravo à saúde dos animais.

Para controlar a saúde e bem estar dos animais, é só armazenar todas as suas informações de modo que posteriormente as mesmas possam ser usadas pelo médico veterinário quando necessário. A partir do momento em que o fluxo de informação cresce, torna-se difícil armazená-las, organizá-las, selecioná-las e/ou consultá-las.

Por razão da grande quantidade de informações e a necessidade de algo que facilitasse o acesso dos veterinários a essas informações, surgiu então a informatização de todos os dados. Dessa maneira, pretende-se desenvolver um software que organize e auxilie o veterinário a realizar consultas, além de guardar todos os dados sobre os animais para melhor acompanhamento veterinário.

Este software será implementado na empresa Agromotta Veterinária[1], que oferece a seus clientes as mais avançadas tecnologias em procedimentos e tratamentos disponíveis em medicina veterinária para os mais variados diagnósticos. Para tais procedimentos, a mesma dispõe de três médicos veterinários e oito funcionários que auxiliam para melhor atendimento.

1 Situada na cidade de Cândido Mota-SP

1.1 OBJETIVO

O software a ser desenvolvido tem como principal objetivo auxiliar o médico veterinário em suas consultas facilitando assim o acesso aos dados sobre o animal para melhor atendê-lo.

Este software conta também com uma página na web para que o dono/cliente possa visualizar os dados de seu(s) animal (is) tais como: características físicas e comportamentais, peso, vacinas, diagnósticos e consultas realizadas.

1.2 ESCOPO

Características do software:

- Permitir que o Usuário/Proprietário acesse o software de acordo com o seu login e senha, já cadastrado pelo Administrador;
- Permitir que o Usuário/Proprietário visualize os dados de seu(s) animal(is), como seus agendamentos e consultas;
- Manter agendamentos e consultas dos médicos veterinários;
- Emitir relatórios das informações do software;

1.3 PÚBLICO ALVO

Este software tem como público alvo clínicas veterinárias de pequeno e médio porte que ainda não tenham um sistema de informatizado.

1.4 JUSTIFICATIVA E MOTIVAÇÃO

O software web proposto nesse projeto, desenvolvido para clínicas veterinárias, vem auxiliar e agilizar o trabalho feito nas empresas, de modo que os agendamentos de

consultas sejam feitos pelo próprio cliente direto do seu computador pessoal com a cesso a internet. Esse tipo de software ainda não existe na empresa Agromotta ou em empresas similares o que torna esse tipo de sistema muito atrativo.

Portanto, atuar na área de programação voltada para web utilizando o Java é, certamente, um dos mercados que atualmente mais oferece oportunidades e ofertas de trabalho.

1.5 ESTRUTURA DO TRABALHO

Este trabalho está divido em sete capítulos. No primeiro capitulo é mostrado a idéia principal do trabalho, seus objetivos, escopo, publico alvo e a justificativa para o mesmo.

No segundo capítulo é mostrado o levantamento de requisitos, ou seja, um questionário sobre como a empresa trabalha e o que espera do software.

No terceiro capítulo é mostrado o desenvolvimento do software, ou seja, as características, prioridades no desenvolvimento e as perspectivas na sua implantação.

No quarto capítulo é mostrado o método de desenvolvimento do software, as tecnologias, ferramentas e outros softwares usados.

No quinto capítulo é mostrada a estrutura do desenvolvimento do software desde o levantamento de requisitos até a sua conclusão.

No sexto capítulo são mostrados os diagramas utilizados para a análise do software, sendo eles, Diagrama de Caso de Uso e Diagrama de Classes

No sétimo capítulo é composto pelas considerações finais e conclusão do trabalho.

2 LEVANTAMENTO DE REQUISITOS

Para o levantamento de requisitos foi necessário a realização de um questionário com o responsável pela empresa a ser adotado o software.

Deseja implantar um sistema de informação na empresa?

R: Sim

Possui computador na empresa?

R: Sim

Já existe algum sistema na empresa?

R: Sim

Por que implantar outro sistema?

R: Pois o existente é antigo, e não há manutenção, sendo assim, não atendendo mais as necessidades.

O que espera com o sistema?

R: Espero que o sistema venha auxiliar a empresa com informações de clientes e animais, fazendo com que essas informações sejam utilizadas no controle da saúde dos animais bem como em suas consultas.

3 DESENVOLVIMENTO DO SISTEMA

3.1 PERSPECTIVA DO SISTEMA

O sistema será desenvolvido em Java, o que independe de um sistema operacional (OS). Desta maneira não é preciso investir nesse ponto para a utilização do mesmo, já que existem OS's grátis no mercado assim como o Ubuntu que faz parte da distribuição mais famosa do Linux. Este sistema será também focado no

desenvolvimento para web, necessitando apenas de um navegador para utilização. O sistema será capaz de realizar consultas, agendamentos e atendimentos via Web.

3.2 PRIORIDADES NA IMPLEMENTAÇÃO

Num primeiro momento, será implementado os módulos apenas da clínica veterinária, que são: Cadastros, Agendamentos, Consultas, Relatórios e Cliente Web. No futuro, se necessário, será integrado com o Petshop contendo: Compra, Venda, Fluxo de Caixa e Controle de Estoque.

3.3 RESULTADOS ESPERADOS NA IMPLANTAÇÃO DO SISTEMA

É esperado que o software atenda a todos os requisitos levantados pelo cliente, organizando todas as informações sobre os animais e agilizando os atendimentos na clínica

4 MÉTODO DE DESENVOLVIMENTO

4.1 TÉCNOLOGIA JAVA

A tecnologia Java é uma plataforma de computação inovadora lançada pela Sun Microsystems em 1995. Inicialmente denominada OAK, essa linguagem de programação foi rebatizada como Java em 1995. A tecnologia Java abre um amplo leque de fascinantes possibilidades para os consumidores. Ela permite executar praticamente todos os aplicativos como jogos, ferramentas, programas e serviços de informações na maioria dos computadores e dispositivos. Hoje a tecnologia Java pode ser encontrada em quase todos os dispositivos: de desktops a dispositivos móveis portáteis e telefones celulares. (www.java.com/pt_BR/)

4.1.1 Linguagem de programação Java

Java é uma linguagem de programação desenvolvida por James Gosling na Sun Microsystems (agora Oracle Corporation) e lançado em 1995. A linguagem tem uma sintaxe muito parecida com C e C++. As aplicações Java são compiladas para um "bytecode" podendo assim rodar em qualquer máquina virtual Java (JVM) independente da arquitetura do computador. Java é uma linguagem orientada a objetos, fortemente tipada, independente de plataforma, segura, robusta, bem estruturada. (http://www.infowester.com/lingjava.php; *Escrito por I. F. Silveira - Publicado em 30/06/2003*)

4.1.2 JavaServer Pages (JSP)

JavaServer Pages (JSP) é uma tecnologia utilizada no desenvolvimento de aplicações Web, desenvolvida pela Sun Microsystem. Esta tecnologia está fundamentada na arquitetura SSI (Server Side Includes) que são comandos extensivos à linguagem HTML os quais podem manter conteúdo estático (HTML) ou dinâmico (JSP). Esses comandos dinâmicos são processados pelo servidor Web antes da página HTML ser enviada. Por ser baseada na linguagem de programação Java, tem a vantagem da portabilidade da plataforma, que permite a sua execução em diversos sistemas operacionais. Esta tecnologia permite ao desenvolvedor Web produzir aplicações que acessem o banco de dados, manipulem arquivos no formato texto, capturem informações a partir de formulários. (http://javafree.uol.com.br/files_user/files/A/74/F8/Tutorial_JSP1.pdf, SANTOS; FREITAS; 2008).

4.1.3 JavaServer Faces (JSF)

JavaServer Faces (JSF ou simplesmente "Faces") faz com que o desenvolvimento para aplicações web se torne fácil, com suporte para poderosos e ricos componentes de interface para o mundo do desenvolvimento web.

Sua principal característica é a facilidade, fazendo com que a ligação entre a cama de apresentação e o código se torne simples. Essa tecnologia torna possível a

utilização do padrão MVC, fazendo com que cada desenvolvedor foque apenas na sua parte do processo. Tem seu modelo de interfaces baseado em eventos e oferece ganhos no desenvolvimento de aplicações WEB, pois permite que o desenvolvedor crie interfaces para o usuário através de um conjunto de componentes pré-definidos, reusa componentes da página. Apesar de o JSF utilizar tags JSP para representar componentes em uma página, por ser flexível, ele não se limita a nenhuma linguagem de marcação, como por exemplo, o HTML. Permite também a criação de componentes próprios. (MANN, 2005).

4.1.4 Jasper Report

O JasperReports é um poderoso framework open-source escrito em Java para geração de relatórios.Ele permite gerar dinamicamente relatórios em diversos formatos; entre eles: PDF, HTML, XML entre outros.

O design do relatório é definido em um arquivo XML, que obedece a estrutura declarada no arquivo jasperreports.dtd.

O arquivo XML é compilado gerando um arquivo .jasper, onde as expressões Java existentes dentro do XML serão verificadas em tempo de compilação. (http://www.guj.com.br/content/articles/reports/JasperReportsIReport.pdf, PAIXÃO, 2004).

4.1.5 iReport

Criar o design do relatório diretamente em XML pode ser uma tarefa custos, por isso fez-se necessário uma ferramenta que automatizasse esse processo, então surgiu o iReport. O iReport permite definir o design do relatório dentro de um ambiente gráfico, contendo todos os recursos que a biblioteca Jasper Report oferece. É possível definir relatórios com designs modernos e complexos sem a necessidade de escrever código em XML, ou seja, é todo gerado automaticamente. O ambiente ainda oferece atalho para tarefas de compilação e visualização do relatório, permitindo a realização de testes, acelerando assim o processo design do relatório.

(http://www.guj.com.br/content/articles/reports/JasperReportsIReport.pdf, PAIXÃO, 2004).

4.2 PADRÃO MVC

A arquitetura MVC (Modelo, Visualização e Controle) consiste em um modelo que separa a aplicação em três camadas distintas: model, view e controller.

Model (Modelo): o modelo representa os dados da aplicação e as regras do negócio que governam o acesso e a modificação dos dados. O modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo.

View (Visualização): Um componente de visualização renderiza o conteúdo de uma parte do modelo e envia para o controlador as ações do usuário que acessa também os dados do modelo via controlador e define como esses dados devem ser apresentados.

Controller (Controle): Um controlador define o comportamento da aplicação, é ele que interpreta as ações do usuário e as mapeia para as chamadas do modelo. Em um cliente de aplicações Web essas ações do usuário poderiam ser cliques de botões ou seleções de menus. As ações realizadas pelo modelo incluem ativar processos de negócio ou alterar o estado do modelo. Com base na ação do usuário e no resultado do processamento do modelo, o controlador seleciona uma visualização a ser exibida como parte da resposta a solicitação do usuário. Há normalmente um controlador para cada conjunto de funcionalidades relacionadas.

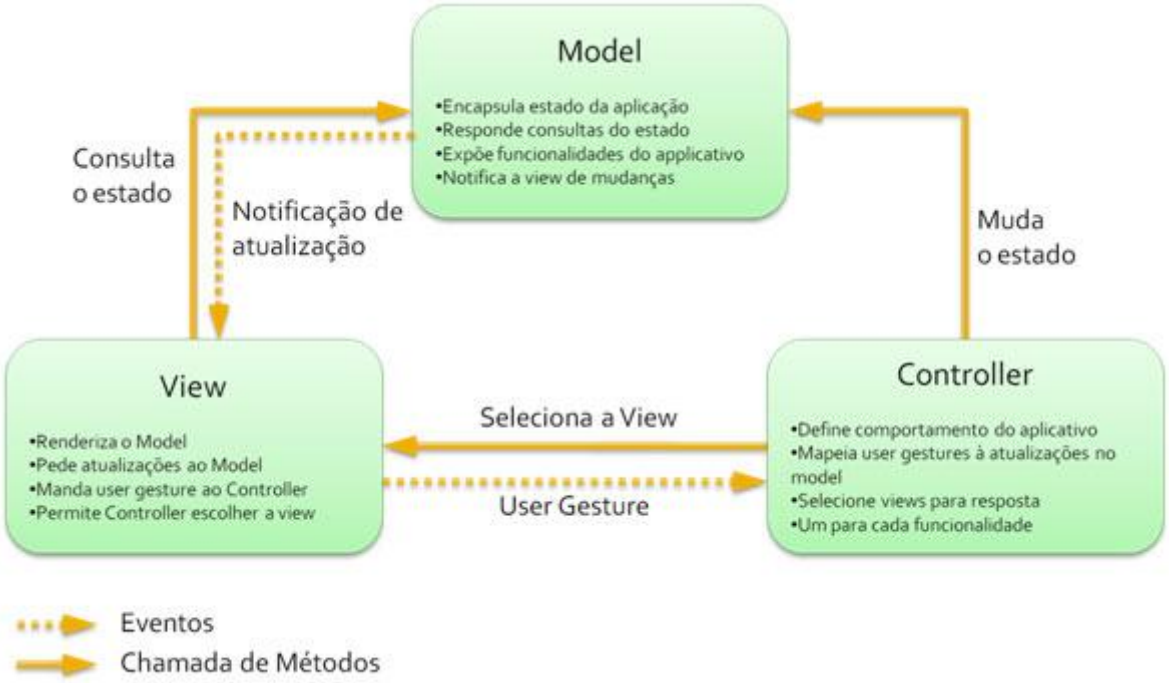

Figura 1. Modelo MVC

4.3 HIBERNATE

Hibernate é um framework para mapeamento objeto-relacional desenvolvido usando a linguagem de programação Java. Este framework facilita o mapeamento dos dados relacionais e o modelo de orientação a objeto.

O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseados em orientação a objeto, que precisam trabalhar com banco de dados relacionais. O Hibernate gera o código SQL, deixando o programador livre do trabalho de converter os dados, deixando o programa portável para qualquer banco de dados SQL. (www.guj.com.br, LINHARES, 2005).

4.4 TOMCAT

O Tomcat é um servidor de web Java. O Tomcat possui as características próprias de um servidor de aplicação, porém não pode ser considerado um servidor de aplicação por não preencher todos os requisitos necessários. Desenvolvido pela Apache Software Foundation, é distribuído como software livre. Ele tem a

capacidade de atuar também como servidor web, ou podendo funcionar integrado a um servidor web. Como servidor web ele prove um servidor web HTTP puramente em Java.

4.5 ECLIPSE

É possível desenvolver aplicações em Java através de vários ambientes de desenvolvimento integrado as chamadas IDEs. A IDE a ser utilizada será o Eclipse JEE Galileo SR2 desenvolvida em Java, grátis e de código aberto (open-source) . O Eclipse foi desenvolvido inicialmente pela IBM, que depois doou o software para a comunidade Java, sendo a mais utilizada para desenvolvimento com a linguagem. A principal característica dessa IDE é o uso da biblioteca gráfica SWT e o desenvolvimento com plugins para atender as necessidades dos programadores. Você pode fazer o download desse software gratuitamente através do site www.eclipse.org.

A versão Eclipse IDE for Java EE Developers (190 MB), é uma versão mais completa com ferramentas para o desenvolvimento web utilizando os recursos da plataforma JEE – Java Enterprise Edition. Uma das características do Eclipse, é que ele não é um software que será instalado em seu computador, ele consiste em um arquivo compactado, que após efetuar o download você deve descompactá-lo em um diretório de sua escolha. Após descompactar o arquivo obterá a seguinte estrutura:

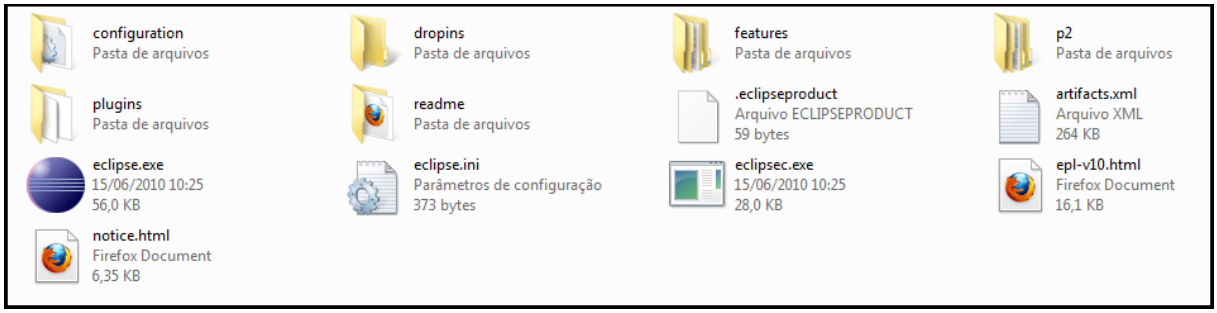

Figura 2. Estrutura de arquivos após descompactação

4.6 HSQLDB

O Hypersonic SQL Database (HSQLDB) é um projeto de banco de dados livre, escrito em Java, que permite a manipulação de banco de dados em uma arquitetura cliente-servidor, ou standalone. Uma grande vantagem de utilização do HSQLDB é a possibilidade de agregar o banco de dados ao pacote de nossas aplicações. O banco de dados é multiplataforma e ocupa um pequeno espaço em disco. Outra característica do banco de dados é a possibilidade de manipularmos bancos de dados em disco, memória ou em formato texto. Trata-se de uma tecnologia flexível e muito útil na construção de aplicações que manipulem banco de dados. (www.guj.com.br, SEVERO, 2004).

4.7 JUDE

JUDE é um software para modelagem UML. É desenvolvido na plataforma Java, o que garante sua portabilidade para qualquer plataforma que possui uma máquina virtual Java. O nome do programa é um acrônimo de ***J****ava and* ***U****ML* ***D****evelopers* ***E****nvironment* (Ambiente para Desenvolvedores UML e Java). A versão utilizada é a JUDE Community 5.5.2, que é grátis.

5 ESTRUTURA DE DESENVOLVIMENTO DO SISTEMA

- Introdução
- Levantamento de requisitos
- Análise do Sistema
- Elaboração da UML (*Unified Modeling Language)*
- Desenvolvimento do Sistema
- Testes
- Implantação do Sistema
- Conclusão

A estrutura a ser utilizada para o desenvolvimento será baseada em orientação à objeto. Orientação à objeto é um padrão a ser seguido na análise, projeto e desenvolvimento de um software.

6 DIAGRAMAS

6.1 DIAGRAMAS DE CASO DE USO

O caso de uso (ou use case) representa uma unidade funcional do sistema, subsistema. Pode ser representado por uma elipse contendo, internamente, o nome do caso de uso. Um caso de uso representa uma unidade de interação entre um usuário e o sistema. Um caso de uso é uma unidade de um trabalho significante. Por exemplo: o "login para o sistema", "manter clientes" e "criar pedidos" são todos casos de uso. Cada caso de uso tem uma descrição no qual descreve a funcionalidade no sistema proposto. Um caso de uso pode "incluir" outra funcionalidade de caso de uso ou "estender" outro caso de uso com seu próprio comportamento

6.1.1 Caso de Uso – Visão Geral do Software

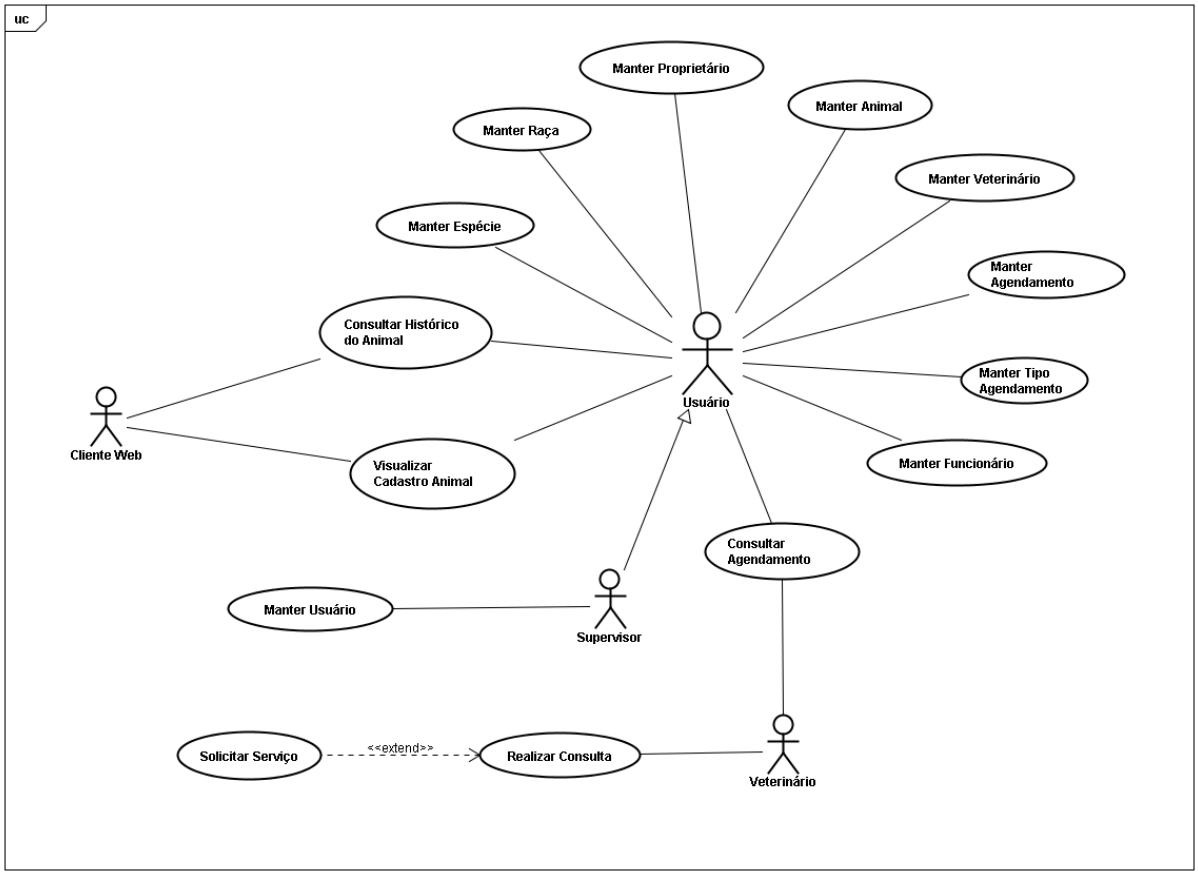

Figura 3. Caso de Uso – Visão Geral do Software

6.1.2 Caso de Uso – Relatórios

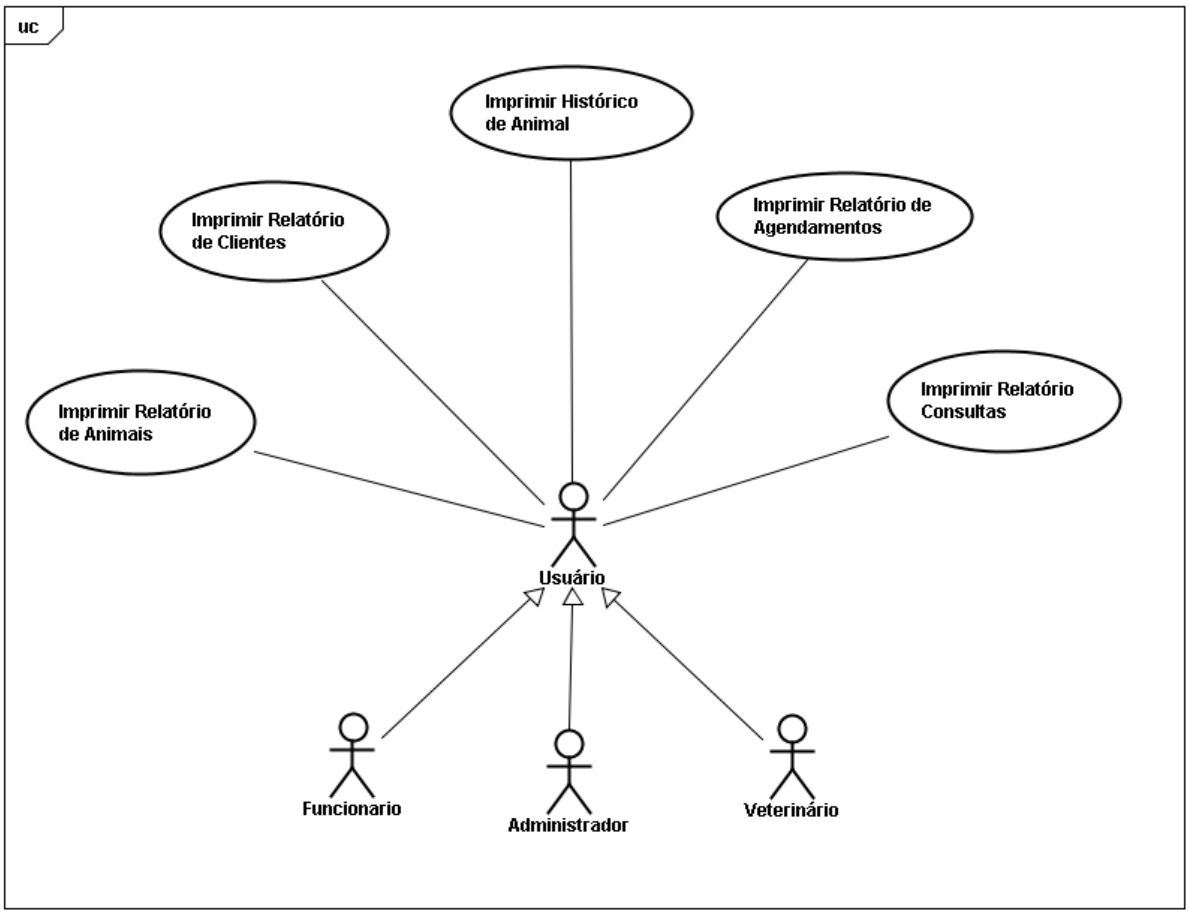

Figura 4. Caso de Uso - Relatórios

6.2 DIAGRAMA DE CLASSE

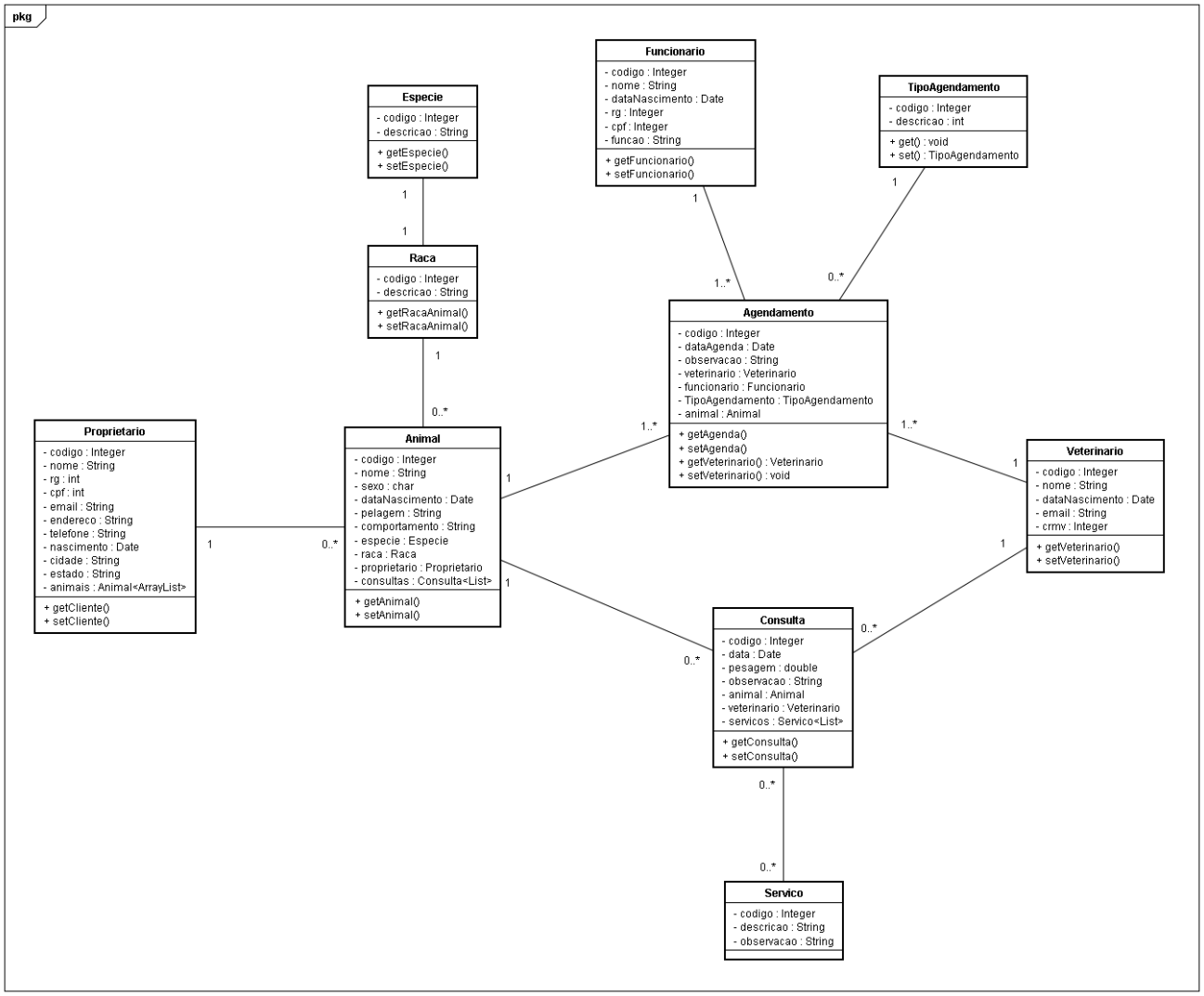

Figura 5. Diagrama de Classe

6.3 DIAGRAMA ENTIDADE-RELACIONAMENTO

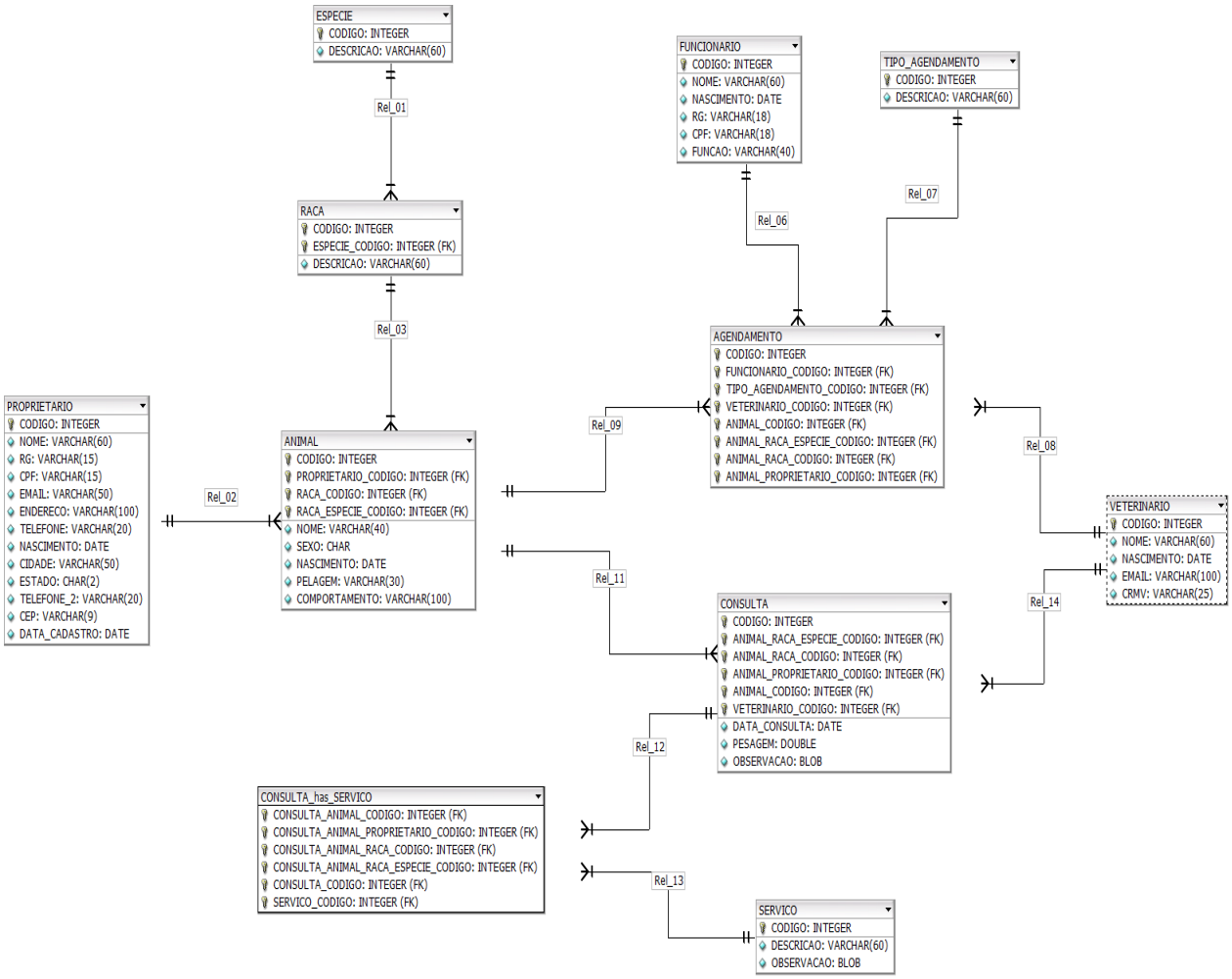

Figura 6. Diagrama Entidade-Relacionamento

7 INTERFACE DO SOFTWARE - PÁGINAS

7.1 PÁGINA DE LOGIN

Figura 7. Página de Login

7.2 PÁGINAS DO SISTEMA

7.2.1 Página Principal

Figura 8. Página Principal do Sistema

7.2.2 Página de Cadastro de Animal

Figura 9. Página de Cadastro de Animal

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="https://ajax4jsf.dev.java.net/ajax" prefix="a4j"%>
<%@ taglib prefix="f"  uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h"  uri="http://java.sun.com/jsf/html"%>
<%@ taglib prefix="t"  uri="http://myfaces.apache.org/tomahawk"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Cadastro Animal</title>
</head>
<body>
<f:view>
	<h:form id="form" styleClass="conteudoCentralizado">
		<t:saveState value="#{cadastrarAnimal}"></t:saveState>
		<t:saveState value="#{pesquisaProprietario}"></t:saveState>
		<jsp:include page="/paginas/Menu.jsp"></jsp:include>

		<fieldset>
		<table width="100%" cellspacing="2px">
			<tr align="center">
				<td colspan="4"
class="subCabecalhoFormulario">
					<h:outputLabel value="Cadastro do
Animal"></h:outputLabel>
				</td>
			</tr>
			<tr>
				<td width="10%"
class="linhaFormulario"><h:outputLabel
value="Proprietário:"></h:outputLabel></td>
```

```
                                <td width="42%" class="linhaFormulario">
                                        <h:inputText id="textProprietario"
style="width: 333px" value="#{cadastrarAnimal.animal.proprietario.nome}"
                                                required="#{not empty
param['form:botaoSalvar']}"  requiredMessage="*Obrigatório"
readonly="true">
                                        </h:inputText>
                                         
                                        <h:commandButton id="botaoPesquisar"
value="" action="#{pesquisaProprietario.prepararPesquisa}"
styleClass="botaoPesquisar"
                                                immediate="true"/>
                                        <h:message for="textProprietario"
styleClass="requerido"></h:message>
                                </td>
                                <td width="14%"
class="linhaFormulario"><h:outputLabel value="Nome do Animal:"/></td>
                                <td width="41%" class="linhaFormulario">
                                        <h:inputText id="textAnimal"
style="width: 212px" value="#{cadastrarAnimal.animal.nome}"
required="#{not empty param['form:botaoSalvar']}"
                                                requiredMessage="*Obrigatório"
maxlength="40">
                                        </h:inputText>  
                                        <h:message for="textAnimal"
styleClass="requerido"></h:message>
                                </td>
                        </tr>

                        <tr>
                                <td class="linhaFormulario"><h:outputLabel
value="Espécie:"></h:outputLabel></td>
                                <td class="linhaFormulario">
                                        <h:selectOneMenu id="comboEspecie"
value="#{cadastrarAnimal.especieSelecionada}" style="width: 214px"
                                                required="#{not empty
param['form:botaoSalvar']}" requiredMessage="*Obrigatório">
                                                <f:selectItems
value="#{cadastrarAnimal.especies}"/>
                                                <a4j:support ajaxSingle="true"
event="onchange" actionListener="#{cadastrarAnimal.filtraRaca}"
reRender="comboRaca"></a4j:support>
                                        </h:selectOneMenu>  
                                        <h:message for="comboEspecie"
styleClass="requerido"/>
                                </td>
                                <td class="linhaFormulario"><h:outputLabel
value="Raça:"></h:outputLabel></td>
                                <td class="linhaFormulario">
                                        <h:selectOneMenu id="comboRaca"
value="#{cadastrarAnimal.racaSelecionada}" style="width: 192px"
                                                required="#{not empty
param['form:botaoSalvar']}" requiredMessage="*Obrigatório">
                                                <f:selectItems
value="#{cadastrarAnimal.racas}"/>
                                        </h:selectOneMenu>  
                                        <h:message for="comboRaca"
styleClass="requerido"></h:message>
                                </td>
```

```
                        </tr>

                        <tr>
                                <td  class="linhaFormulario"
align="left"><h:outputLabel value="Nascimento:"></h:outputLabel></td>
                                <td class="linhaFormulario">
                                        <t:inputCalendar
value="#{cadastrarAnimal.animal.dataNascimento}" onkeypress="data(this,
event)" maxlength="10"
                                                renderPopupButtonAsImage="false"
popupDateFormat="dd/MM/yyyy" renderAsPopup="true"
popupButtonStyleClass="botaoCalendario">
                                        </t:inputCalendar>
                                </td>
                                <td class="linhaFormulario"><h:outputLabel
value="Sexo:"></h:outputLabel></td>
                                <td class="linhaFormulario">
                                        <h:selectOneMenu id="comboSexo"
value="#{cadastrarAnimal.sexoSelecionado}" style="width: 111px"
                                                required="#{not empty
param['form:botaoSalvar']}" requiredMessage="*Obrigatório">
                                                <f:selectItems
value="#{cadastrarAnimal.sexo}"/>
                                        </h:selectOneMenu>  
                                        <h:message for="comboSexo"
styleClass="requerido" ></h:message>
                                </td>
                        </tr>

                        <tr>
                                <td class="linhaFormulario"><h:outputLabel
value="Pelagem:"></h:outputLabel></td>
                                <td class="linhaFormulario"><h:inputText
style="width: 277px" value="#{cadastrarAnimal.animal.pelagem}"
maxlength="30"></h:inputText></td>

                                <td class="linhaFormulario"><h:outputLabel
value="Comportamento:"></h:outputLabel></td>
                                <td class="linhaFormulario"><h:inputText
style="width: 277px"
value="#{cadastrarAnimal.animal.comportamento}"/></td>
                        </tr>

                        <tr>
                                <td class="linhaFormulario"><h:outputLabel
value="Porte:"></h:outputLabel></td>
                                <td class="linhaFormulario"><h:inputText
style="width: 277px" value="#{cadastrarAnimal.animal.porte}"
maxlength="15"></h:inputText></td>
                                <td class="linhaFormulario"></td>
                                <td class="linhaFormulario"></td>
                        </tr>
                </table>
                </fieldset>
                <br/>
                <div align="center">
                        <h:commandButton id="botaoSalvar"
action="#{cadastrarAnimal.salvar}" value="Salvar" onclick="return
cadastrar()" styleClass="botaoSalvar"/> 
```

```
                    <h:commandButton id="botaoCancelar"
action="consultaAnimal" value="Cancelar" immediate="true"
styleClass="botaoCancelar"></h:commandButton>
                </div>
        </h:form>
</f:view>
</body>
</html>
```

```
package br.com.veterinaria.controle;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.event.ActionEvent;
import javax.faces.model.SelectItem;

import br.com.veterinaria.beans.Animal;
import br.com.veterinaria.beans.Especie;
import br.com.veterinaria.beans.Proprietario;
import br.com.veterinaria.beans.Raca;
import br.com.veterinaria.dao.AnimalDao;
import br.com.veterinaria.dao.EspecieDao;
import br.com.veterinaria.dao.RacaDao;
import br.com.veterinaria.util.Constantes;
import br.com.veterinaria.util.FacesUtil;

public class CadastrarAnimal implements Serializable {

        private static final long serialVersionUID = 1L;

        private Animal animal;
        private List<Animal> animais = new ArrayList<Animal>();

        private List<SelectItem> especies = new ArrayList<SelectItem>();
        private Integer especieSelecionada;

        private List<SelectItem> racas = new ArrayList<SelectItem>();
        private Integer racaSelecionada;

        private List<SelectItem> sexo = new ArrayList<SelectItem>();
        private String sexoSelecionado;

        private List<SelectItem> comboPesquisa = new
ArrayList<SelectItem>();
        private String pesquisaSelecionada;
        private String textPesquisa;

        private boolean cadastroAnimal = false;
        private boolean inclusao;

        public String prepararConsulta() {
                try {
                        animais = new AnimalDao().lista();
                        listaComboPesquisa();
```

```
                } catch (Exception e) {
                        FacesUtil.mensagem(FacesMessage.SEVERITY_ERROR,
e.getMessage());
                        e.printStackTrace();
                }
                return "consultaAnimal";
        }

        public String prepararCadastro() {
                animal = new Animal();
                animal.setProprietario(new Proprietario());

                inicializaCombos();

                especieSelecionada = 0;
                racaSelecionada = 0;
                sexoSelecionado = "";

                cadastroAnimal = true;
                inclusao = true;
                return "cadastroAnimal";
        }

        public String editar(){
                inicializaCombos();

                animal = (Animal) FacesUtil.getLinha("animal");
                especieSelecionada =
animal.getRaca().getEspecie().getCodigo();
                racaSelecionada = animal.getRaca().getCodigo();
                sexoSelecionado = animal.getSexo();

                cadastroAnimal = true;
                inclusao = false;
                return "cadastroAnimal";
        }

        public void excluir(ActionEvent evt){
                animal = (Animal) FacesUtil.getLinha("animal");
                        try {
                                new AnimalDao().deletar(animal);
                                animais.remove(animal);
                        } catch (Exception e) {

        FacesUtil.mensagem(FacesMessage.SEVERITY_ERROR, e.getMessage());
                                e.printStackTrace();
                        }
        }

        public String salvar(){
                try {
                        animal.setRaca(new
RacaDao().recuperar(racaSelecionada));
                        animal.setSexo(sexoSelecionado);

                        if (inclusao) {
                                new AnimalDao().salvar(animal);
                                animais.add(animal);
                        } else {
```

```
                                new AnimalDao().atualizar(animal);
                        }
                        return "consultaAnimal";
                } catch (Exception e) {
                        FacesUtil.mensagem(FacesMessage.SEVERITY_ERROR,
e.getMessage());
                        e.printStackTrace();
                }
                return "";
        }

        public String selecionarProprietario() {
                animal.setProprietario((Proprietario)
FacesUtil.getLinha("proprietario"));
                return "cadastroAnimal";
        }

        public void pesquisar(ActionEvent evt){
                try {
                        if (Constantes.ANIMAL.equals(pesquisaSelecionada)) {
                                animais = new
AnimalDao().animaisPorNome(textPesquisa);
                        } else {
                                animais = new
AnimalDao().animaisPorProprietario(textPesquisa);
                        }
                } catch (Exception e) {
                        FacesUtil.mensagem(FacesMessage.SEVERITY_WARN,
e.getMessage());
                        e.printStackTrace();
                }
        }

        private List<SelectItem> listaEspecies() {
                List<SelectItem> lista = null;
                try {
                        lista = new ArrayList<SelectItem>();
                        lista.add(new SelectItem(""));
                        for (Especie especie : new EspecieDao().lista()) {
                                lista.add(new SelectItem(especie.getCodigo(),
especie.getDescricao()));
                        }
                } catch (Exception e) {
                        FacesUtil.mensagem(FacesMessage.SEVERITY_ERROR,
e.getMessage());
                        e.printStackTrace();
                }
                return lista;
        }

        private List<SelectItem> listaRacas() {
                List<SelectItem> lista = null;
                try {
                        lista = new ArrayList<SelectItem>();
                        lista.add(new SelectItem(""));
                        for (Raca raca : new RacaDao().lista()) {
                                lista.add(new SelectItem(raca.getCodigo(),
raca.getDescricao()));
                        }
```

```
                } catch (Exception e) {
                        FacesUtil.mensagem(FacesMessage.SEVERITY_ERROR,
e.getMessage());
                        e.printStackTrace();
                }
                return lista;
        }

        public void filtraRaca(ActionEvent evt) {
                try {
                        racas = new ArrayList<SelectItem>();
                        for (Raca raca : new RacaDao().racasPorEspecie(new
EspecieDao().recuperar(especieSelecionada))) {
                                racas.add(new SelectItem(raca.getCodigo(),
raca.getDescricao()));
                        }
                } catch (Exception e) {
                        FacesUtil.mensagem(FacesMessage.SEVERITY_ERROR,
e.getMessage());
                        e.printStackTrace();
                }
        }

        private void listaSexo() {
                sexo = new ArrayList<SelectItem>();
                sexo.add(new SelectItem(""));
                sexo.add(new SelectItem(Constantes.MACHO));
                sexo.add(new SelectItem(Constantes.FEMEA));
        }

        private void inicializaCombos(){
                especies = listaEspecies();
                racas = listaRacas();
                listaSexo();
        }

        private void listaComboPesquisa(){
                comboPesquisa.clear();
                comboPesquisa.add(new SelectItem(Constantes.ANIMAL));
                comboPesquisa.add(new SelectItem(Constantes.PROPRIETARIO));
        }

        public Animal getAnimal() {
                return animal;
        }

        public void setAnimal(Animal animal) {
                this.animal = animal;
        }

        public List<Animal> getAnimais() {
                return animais;
        }

        public void setAnimais(List<Animal> animais) {
                this.animais = animais;
        }

        public List<SelectItem> getSexo() {
```

```
		return sexo;
	}

	public void setSexo(List<SelectItem> sexo) {
		this.sexo = sexo;
	}

	public String getSexoSelecionado() {
		return sexoSelecionado;
	}

	public void setSexoSelecionado(String sexoSelecionado) {
		this.sexoSelecionado = sexoSelecionado;
	}

	public List<SelectItem> getEspecies() {
		return especies;
	}

	public void setEspecies(List<SelectItem> especies) {
		this.especies = especies;
	}

	public Integer getEspecieSelecionada() {
		return especieSelecionada;
	}

	public void setEspecieSelecionada(Integer especieSelecionada) {
		this.especieSelecionada = especieSelecionada;
	}

	public List<SelectItem> getRacas() {
		return racas;
	}

	public void setRacas(List<SelectItem> racas) {
		this.racas = racas;
	}

	public Integer getRacaSelecionada() {
		return racaSelecionada;
	}

	public void setRacaSelecionada(Integer racaSelecionada) {
		this.racaSelecionada = racaSelecionada;
	}

	public boolean isCadastroAnimal() {
		return cadastroAnimal;
	}

	public void setCadastroAnimal(boolean cadastroAnimal) {
		this.cadastroAnimal = cadastroAnimal;
	}

	public List<SelectItem> getComboPesquisa() {
		return comboPesquisa;
	}
```

```
        public void setComboPesquisa(List<SelectItem> comboPesquisa) {
                this.comboPesquisa = comboPesquisa;
        }

        public String getPesquisaSelecionada() {
                return pesquisaSelecionada;
        }

        public void setPesquisaSelecionada(String pesquisaSelecionada) {
                this.pesquisaSelecionada = pesquisaSelecionada;
        }

        public String getTextPesquisa() {
                return textPesquisa;
        }

        public void setTextPesquisa(String textPesquisa) {
                this.textPesquisa = textPesquisa;
        }

}
```

7.2.3 Página de Agendamento

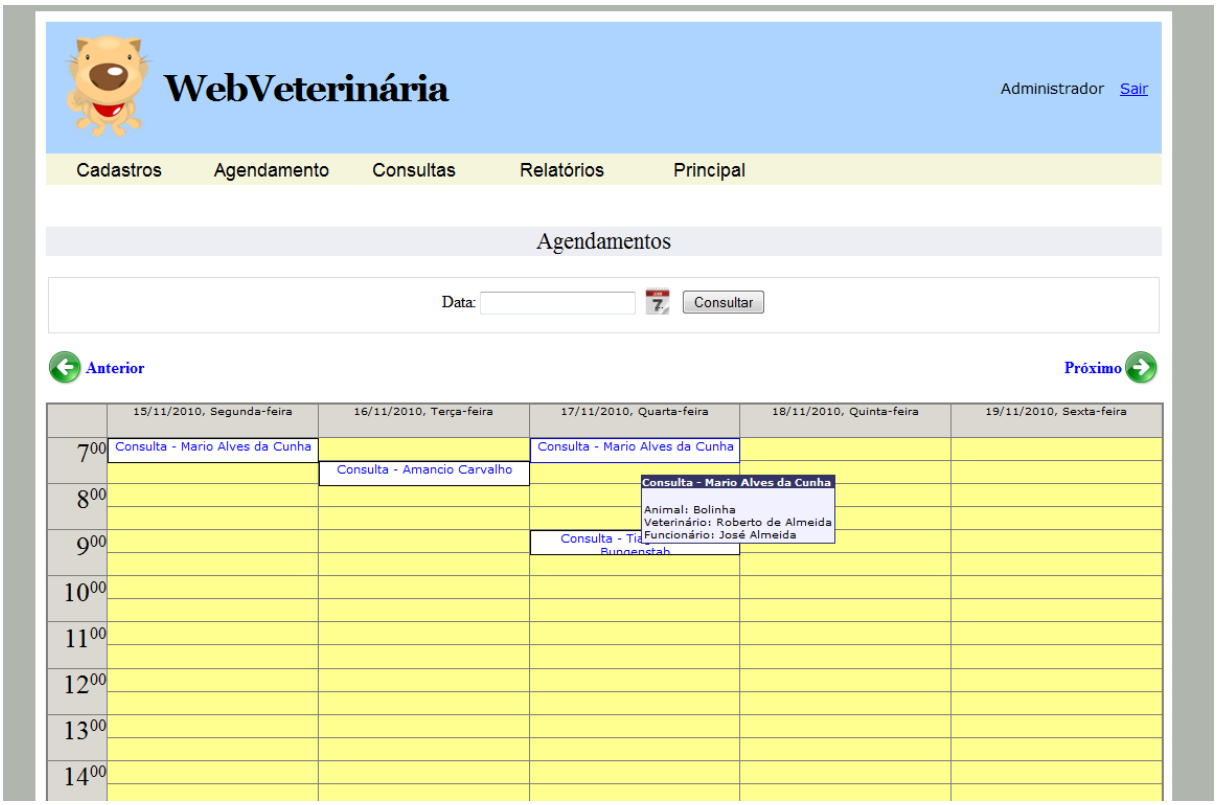

Figura 10. Página de Agendamento

7.2.4 Página de Consulta Clínica

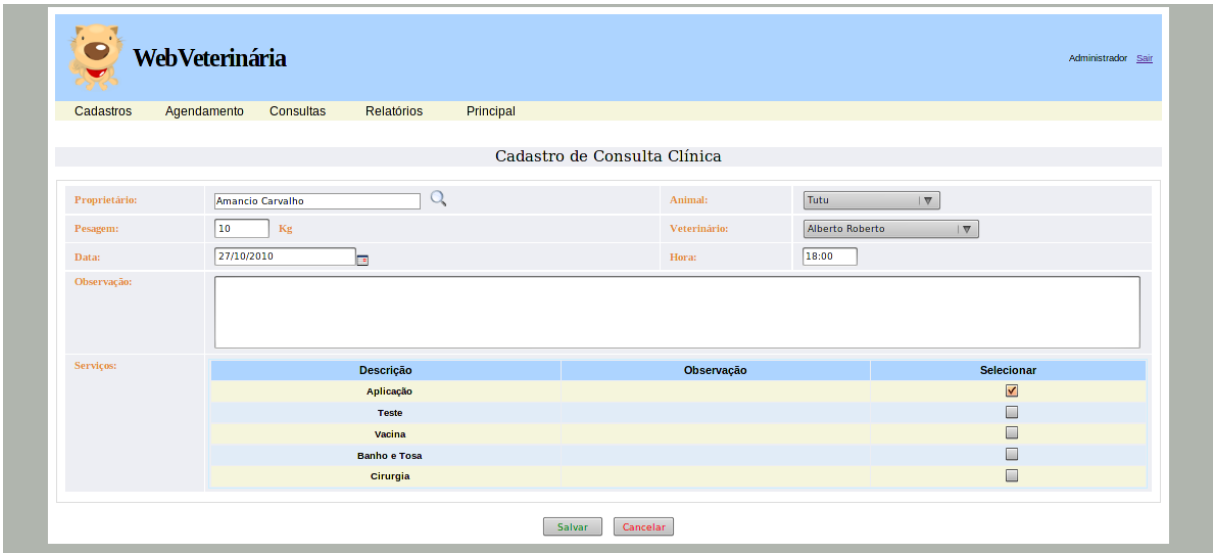

Figura 11. Página de Consulta Clínica

7.3 PÁGINAS DO CLIENTE WEB

7.3.1 Página Meu Animal

Figura 12. Página Meu Animal

7.3.2 Página do Histórico do Animal

Figura 13. Página Histórico do Animal

8 CONCLUSÃO

A implementação do software na empresa Agromotta Veterinária será de grande utilidade para a clínica veterinária, já que facilitará em todos os aspectos as tarefas que serão realizadas no dia-a-dia. Com a implementação do software tanto o usuário quanto o cliente da clínica terão um ganho de tempo. A mudança gerada com a informatização tornará a clínica mais apresentável dificultando a migração de clientes para possíveis concorrentes que possuam sistemas também informatizados.

CRONOGRAMA

Atividades	2010									
	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Levantamento de Requisitos										
Análise do Sistema										
Implementação										
Design do Sistema										
Testes										
Conclusão do TCC										
Defesa do TCC										
Implantação										

Figura 14. Cronograma

REFERÊNCIAS

BLOCH, Joshua**, Java Efetivo**. 2. ed. Tradução de Aldir José Coelho. Rio de Janeiro: Editora Alta Books, 2008.

DEITEL, Harvey M.; DEITEL, Paul J. **Java: Como Programar**. 6. ed. Tradução de Edson Furmankiewicz. São Paulo: Editora Pearson Pratice Hall, 2008.

Caelum. Apostila: FJ-21 **Java para Desenvolvimento Web**. Disponível em www.caelum.com.br/apostilas.

Silva, Elana C.; e Mattos, Karla M.. **Sistema Web para Clínica Veterinária - WEBVET**. 2009. 98p. Trabalho de Conclusão de Curso. Universidade Estadual de Ponta Grossa – UEPG – Setor de Ciências Agrárias de Tecnologia. 2009

GUJ – **Grupo de Usuários Java**. Gerenciado e desenvolvido por Caelum Cursos Java. Acesso em: www.guj.com.br.

KING, Gavin; BAUER, Christian; ANDERSEN, Max R.; BERNARD, Emmanuel e EBERSOLE, Steve. **HIBERNATE – Persistência Relacional para Java Idiomático**. 344p.

SIMPSOM, Blaine e TOUSSI, Fred. **Hsqldb User Guide.** HSQLDB Development Group, 2007.