

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
UNIVERSIDADE DO RIO GRANDE DO NORTE**

Severino Bezerra Cabral

CONTROLE BOVINO

**NATAL
2012**

SEVERINO BEZERRA CABRAL

CONTROLE BOVINO

Trabalho de Relatório Técnico de Curso apresentado à diretoria de Pós-Graduação da Universidade do Rio Grande do Norte como requisito para conclusão do Curso de Especialização em Desenvolvimento de Sistemas Corporativos.

Orientador:
Prof. Alexandre Luiz G. Damasceno

Natal
2012

SEVERINO BEZERRA CABRAL

CONTRLE BOVINO

Trabalho de Conclusão de Curso apresentado diretoria de Pós-Graduação da
Universidade do Rio Grande do Norte, aprovada em ____ de ____ de
_____, pela banca examinadora composta pelos seguintes membros:

Prof. Alexandre Luiz G. Damasceno

Orientador

SOFTWARE CONTROLE BOVINO

Severino Bezerra Cabral
Alexandre Luiz G. Damasceno

RESUMO

Este relatório tem como premissa abordar as principais características técnicas referentes ao software Controle Bovino desenvolvido em plataforma web e tecnologia Java. Sendo assim, o mesmo pode ser utilizado por alunos, professores e funcionários da UNI-RN que desejam obter maiores informações sobre o assunto. Neste documento estão descritas as funcionalidades atendidas pelo software e ferramentas de frameworks utilizadas durante o seu desenvolvimento.

SOFTWARE CONTROLE BOVINO("Cattle Control")

ABSTRACT

This report aims to comment on the main technical characteristics of the Controle Bovino ("Cattle Control") software application, developed for the Web platform using Java technology. In this manner, it can be used as a reference by students, faculty and employees of UNI-RN in the event they wish to obtain greater information on these subjects. This document describes the functionality and features offered by the software, as well as the tools and frameworks employed during its development.

Introdução

O objetivo desse trabalho é apresentar a construção de um sistema web, desenvolvido em Java, denominado “Controle Bovino”. Este sistema se propõe a auxiliar o gerenciamento de informações de animais de um proprietário de terra, no tocante a compra, nascimento, descarte, cobertura, peso e definição de grupo sanguíneo em relação à raça do animal. Dessa forma, este trabalho pretende apresentar o sistema, suas funcionalidades e as tecnologias usadas para desenvolvê-lo.

1.1 Objetivos específicos:

- Objetivos do estudo:
 - Estudar e aplicar as tecnologias estudadas durante o curso (JSF, Spring, JPA e PostgreSQL).
 - Aplicar conhecimentos de engenharia de requisitos e UML, com o uso de especificação de casos de uso e modelo de domínio.
 - Desenvolver um sistema web, com base nas necessidades de um cliente real.
- Objetivos do sistema:
 - Manter os dados dos animais do rebanho integro e confiáveis.
 - Manter os dados dos animais descartados integro e confiáveis.
 - Fornecer informações de animais do rebanho de forma segura e ágil.
 - Fornecer informações de animais descartados de forma segura e ágil.
 - Fornecer informações de futuros nascimento de animais.
 - Controlar cobertura de animais.
 - Acompanhar evolução de pesos dos animais do rebanho.
 - Manter dados de raças de animais.
 - Calcular grau de sangue de animais tanto no caso de nascimento do rebanho como no caso de compra de animais.

2 DELIMITAÇÃO DO ESCOPO DO SISTEMA

Como ponto de partida deste relatório, para melhor entender o problema a qual o sistema pretende resolver, faz necessário a apresentação do escopo a ser trabalho e quais requisitos o sistema pretende resolver.

O escopo das funcionalidades do sistema Controle Bovino objetiva auxiliar o proprietário a realizar um controle eficiente de seu rebanho de forma a acompanhar desde o nascimento, passando pela compra e descarte dos animais. Em detalhes, este escopo consiste nas seguintes funcionalidades:

- Cadastrar Raças de animais – permite ao usuário realizar manutenção do cadastrado das raças, para posterior identificação do grau de sangue/raça de cada animal.
- Informar Compra de animais – permite ao usuário realizar manutenção de diversas informações dos animais adquiridos.
- Informar cria de animais (Nascimento) – permite ao usuário entrar com informações dos animais nascidos a partir do próprio rebanho.
- Excluir animais – permite ao usuário realizar exclusão de informações dos animais, eliminando o animal da lista do sistema.
- Alterar animais – permite ao usuário alterar os dados dos animais cadastrados no sistema.
- Informar grau de Sangue de animais comprados – permite ao usuário informar dados de sangue/raça de animais adquiridos.
- Informar Descarte de animais – permite ao usuário realizar a operação de descarte de animais do rebanho.
- Informar Peso de animais – permite ao usuário realizar o cadastrado do peso dos animais do rebanho para que seja possível realizar o acompanhamento da evolução do peso dos animais.
- Cadastrar Coberturas de animais – permite ao usuário cadastrar as coberturas dos animais do rebanho.
- Listar animais do rebanho – concede ao usuário um relatório com a lista das informações dos animais do rebanho.
- Listar animais descartados - concede ao usuário um relatório com a lista das informações dos animais descartados, isto é, animais que não

pertencem mais ao rebanho.

3 Caso de Usos

Considerando a lista de requisitos de negócio apresentadas, o primeiro passo no entendimento dos requisitos do sistema foi realizar o levantamento dos casos de uso a serem desenvolvidos neste trabalho. Após elaboradas reuniões com o cliente, um conjunto de casos de uso iniciais foram especificados.

O caso de uso apresentado na Tabela 1 objetiva apresentar a funcionalidade de cadastro (inclusão, exclusão e alteração) dos dados animais a serem gerenciados pela aplicação.

Tabela 1 - UC1- Manter Animais - Inserir compra de animais

Caso de Uso	Inserir compra de animais.
Caso de Uso Geral	Manter Animais
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados da compra do animal 2. O ator informa os dados da compra do animal e aciona a opção para salvar. 3. O sistema informa que a operação foi bem sucedida. 4. O caso de uso termina.
Fluxos Alternativos	1. Brinco já cadastrado 1. Se no passo 2 do fluxo básico, o ator informar dados de um animal já cadastrado, então: 2. O sistema informa que o animal já encontra-se cadastrado 3. O fluxo retorna para o passo 1 do fluxo básico. 2. Data Nascimento não poder ser futura 1. Se no passo 2 do fluxo básico, o ator informar uma data de nascimento futura, então: 2. O sistema informa que a data de nascimento não pode ser futura. 3. O fluxo retorna para o passo 1 do fluxo básico.

Pós-condição	Dados de compra de animal cadastrada com sucesso.
--------------	---

Tabela 2 - UC2 - Manter Animais - Inserir Cria

Caso de Uso	Inserir Cria
Caso de Uso Geral	Manter Animais
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados de cria do animal 2. O ator informa os dados de cria do animal e aciona a opção para salvar. 3. O sistema informa que a operação foi bem sucedida. 4. O caso de uso termina.
Fluxos Alternativos	1. Brinco já cadastrado. 1. Se no passo 2 do fluxo básico, o ator informar dados de um animal já cadastrado, então: 2. O sistema informa que o animal já encontra-se cadastrado 3. O fluxo retorna para o passo 1 do fluxo básico. 2. Data Nascimento não poder ser futura 1. Se no passo 2 do fluxo básico, o ator informar uma data de nascimento futura, então: 2. O sistema informa que a data de nascimento não pode ser futura. 3. O fluxo retorna para o passo 1 do fluxo básico.
Pós-condição	Inserção efetuada com sucesso

Tabela 3 - UC3 - Manter Animais - Excluir Animal

Caso de Uso	Excluir Animal
Caso de Uso Geral	Manter Animais
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de animais. 2. O ator seleciona o animal a ser excluído e aciona a opção para excluir. 3. O sistema pede confirmação. 4. O sistema informa que a operação foi bem sucedida. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Exclusão efetuada com sucesso

Tabela 4 - UC4 - Manter animais – Alterar dados do Animal

Caso de Uso	Alterar dados do animal
Caso de Uso Geral	Manter animais
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de animais. 2. O ator seleciona o animal a ter seus dados alterados e aciona a opção detalhe. 3. O ator informa os dados do animal e aciona a opção para salvar. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Dados do animal alterado com sucesso

O caso de uso apresentado na Tabela 5 objetiva apresentar a funcionalidade de cadastro (inclusão, exclusão e alteração) dos dados de coberturas de animais a serem gerenciados pela aplicação.

Tabela 5 - UC5 - Manter Cobertura

Caso de Uso	Inserir Cobertura
Caso de Uso Geral	Manter Cobertura
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados da cobertura 2. O ator informa os dados da cobertura e aciona a opção para salvar. 3. O sistema informa que a operação foi bem sucedida. 4. O caso de uso termina.
Fluxos Excepcionais	1 Data Cobertura não poder ser futura 1.1 Se no passo 2 do fluxo básico, o ator informar uma data de cobertura futura, então: 1.2 O sistema informa que a data de cobertura não pode ser futura. 1.3 O fluxo retorna para o passo 1 do fluxo básico. 2 Só animais adultos podem estar disponíveis para serem selecionados como pai e mãe.
Fluxos Alternativos	
Pós-condição	Inserção cobertura efetuada com sucesso

Tabela 6 - UC6 - Manter Cobertura

Caso de Uso	Excluir Cobertura
Caso de Uso Geral	Manter Cobertura
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de coberturas de animais. 2. O ator seleciona a cobertura a ser excluída e aciona a opção para excluir. 3. O sistema pede confirmação. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Exclusão efetuada com sucesso

Tabela 7 - UC7 - Manter Cobertura

Caso de Uso	Informar Gestação
Caso de Uso Geral	Manter Cobertura
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de cobertura de animais. 2. O ator seleciona a cobertura a qual quer informar a gestação e aciona a opção informar gestação. 3. O ator informa a gestação para a cobertura e aciona opção para salvar. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Dados da cobertura alterados com sucesso

O caso de uso apresentado na Tabela 8 objetiva apresentar a funcionalidade de cadastro (inclusão, exclusão e alteração) dos dados de grau de sangue/raça dos animais a serem gerenciados pela aplicação.

Tabela 8 - UC8 - Manter Sangue

Caso de Uso	Inserir Sangue
Caso de Uso Geral	Manter Sangue
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados do grau de sangue para animal selecionado 2. O ator informa os dados do grau de sangue e aciona a opção para salvar. 3. O sistema informa que a operação foi bem sucedida. 4. O caso de uso termina.
Fluxos Excepcionais	1. A soma dos sangues/raça informada não pode ser diferente de 1 1. Se no passo 2 do fluxo básico, o ator informar dados de grau de sangue que a soma não seja igual a 1 2. O sistema informa que a soma de grau de sangue tem que ser igual a 1. 3. O fluxo retorna para o passo 1 do fluxo básico.
Fluxos Alternativos	
Pós-condição	Inserção sangue efetuada com sucesso

O caso de uso apresentado na Tabela 9 objetiva apresentar a funcionalidade realizar o descarte de animais a serem gerenciados pela aplicação.

Tabela 9 - UC9 - Descarte de Animais

Caso de Uso	Realizar descarte de Animais
Caso de Uso Geral	Descarte Animais
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados de descarte de animais selecionado 2. O ator informa os dados de descarte e aciona a opção para descarte. 3. O sistema pede confirmação. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Descarte de animais realizado com sucesso

O caso de uso apresentado na Tabela 10 objetiva apresentar a funcionalidade de cadastro (inclusão, exclusão e alteração) dos dados das raças de animais a serem gerenciados pela aplicação.

Tabela 10 - UC10 - Manter Raça

Caso de Uso	Inserir Raça
Caso de Uso Geral	Manter Raça
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados da raça 2. O ator informa os dados da raça e aciona a opção para salvar. 3. O sistema informa que a operação foi bem sucedida. 4. O caso de uso termina
Fluxos Excepcionais	1. Raça já existe 1. Se no passo 2 do fluxo básico, o ator informar dados de uma raça já cadastrada, então: 2. O sistema informa que a raça já encontra-se cadastrado 3. O fluxo retorna para o passo 1 do fluxo básico.
Fluxos Alternativos	
Pós-condição	Inserção raça efetuada com sucesso

Tabela 11 - UC11 - Manter Raça

Caso de Uso	Excluir Raça
Caso de Uso Geral	Manter Raça
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de raças de animais. 2. O ator seleciona a raça a ser excluída e aciona a opção para excluir. 3. O sistema pede confirmação. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	1. Raça já informada para animais 1. Se no passo 2 do fluxo básico, o ator informar dados de um raça já cadastrada, para um animal qualquer, então 2. O sistema informa que a raça já encontra-se cadastrado para animais 3. O fluxo retorna para o passo 1 do fluxo básico.
Fluxos Alternativos	
Pós-condição	Exclusão raça efetuada com sucesso

Tabela 12 - UC12 - Manter Raça

Caso de Uso	Alterar Raça
Caso de Uso Geral	Manter Raça
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de raças. 2. O ator seleciona a raça a ter seus dados alterados e aciona a opção detalhe. 3. O ator informa os dados da raça e aciona a opção para salvar. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Dados da raça alterados com sucesso

O caso de uso apresentado na Tabela 13 objetiva apresentar a funcionalidade de cadastro (inclusão, exclusão e alteração) dos dados dos pesos de animais a serem gerenciados pela aplicação.

Tabela 13 - UC13 - Manter Peso

Caso de Uso	Inserir Peso
Caso de Uso Geral	Manter Peso
Ator Principal	Usuário do sistema
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema requisita os dados do peso 2. O ator informa os dados de peso do animal e aciona a opção para salvar. 3. O sistema informa que a operação foi bem sucedida. 4. O caso de uso termina.
Fluxos Excepcionais	3 1. Data medição não poder ser futura 3.1 Se no passo 2 do fluxo básico, o ator informar uma data de medição futura, então: 3.2 O sistema informa que a data de medição não pode ser futura. 3.3 O fluxo retorna para o passo 1 do fluxo básico.
Fluxos Alternativos	
Pós-condição	Inserção Peso efetuada com sucesso

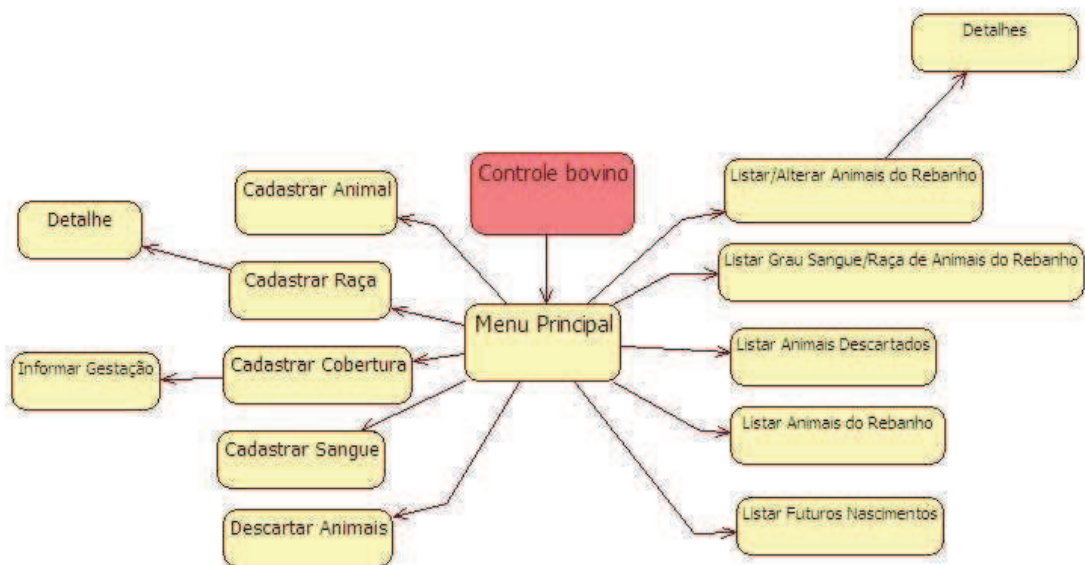
Tabela 14 - UC14 - Manter Peso

Caso de Uso	Excluir Peso
Caso de Uso Geral	Manter Peso
Ator Principal	Usuário
Pré-condição	Acessar o sistema
Fluxo Normal	1. O sistema exibe lista de pesos de animais. 2. O ator seleciona a data do peso a ser excluída e aciona a opção para excluir. 3. O sistema pede confirmação. 4. O sistema informa que a operação foi bem sucedida. 5. O caso de uso termina.
Fluxos Excepcionais	
Fluxos Alternativos	
Pós-condição	Exclusão peso efetuada com sucesso

4 Descrição das principais telas do sistema

Após a elaboração dos casos de uso, foi feita uma prototipagem das telas do sistema para melhor visualizar o funcionamento final do sistema e facilitar o entendimento dos casos de uso. Abaixo seguem as principais telas prototipadas do sistema e seu fluxo.

Figura 1 - Fluxo de telas.



4.1 Apresentação das telas

A tela principal do sistema apresenta as principais funcionalidades do mesmo, permitindo ao usuário selecionar a opção desejada. Segue a lista de funcionalidades: Cadastrar Animal, Cadastrar Raça, Cadastrar Cobertura, Cadastrar Peso, Cadastrar Sangue, Descarte de Animais, Listar/Alterar Animais do Rebanho, Listar Animais Descartados, Listar Animais do Rebanho e Listar Futuros Nascimento.

Figura 2 - Tela principal do sistema.



Na tela de cadastro de animais realiza-se a inserção dos dados dos animais tanto os animais adquiridos, como as crias de animais do rebanho.

Figura 3 - Tela de Cadastro de Animais.

The screenshot shows a web browser window titled 'Controle Bovino - Mozilla Firefox'. The address bar shows a local URL. The page has a header 'Controle Bovino' and a main section 'Cadastro de Animais'. This section contains a form with the following fields and options:

- Aquisição:** Radio buttons for 'Aquisição' (selected) and 'Cria'.
- Nome:** A text input field.
- Brinco:** A text input field with the value '0'.
- Sexo:** Radio buttons for 'Masculino' (selected) and 'Feminino'.
- Data Nascimento:** A date input field.
- Semen?:** A checkbox.
- Mae:** A dropdown menu with the text '-- Selecione --'.
- Pai:** A dropdown menu with the text '-- Selecione --'.

Below the form are three buttons: 'Cadastrar Cria', 'Cadastrar Aquisição', and 'Voltar'. Below the buttons is a table titled 'Lista de Animais' with the following data:

Animal	Brinco	Sexo	Data Nascimento
Pegador	1	M	2005-01-01
Princesa	2	F	2005-01-01
olimpia	4	F	2009-12-25
Filha1	7	F	2010-01-01
criasans	19	F	2010-01-01
criaolimpia	20	F	2012-05-11

Para a inserção de nova aquisição deve-se escolher a opção de “aquisição” neste caso o botão de “cadastrar cria” e as opções de escolher pai e mãe serão desabilitados, então devem ser preenchidos os dados solicitados na tela e clicar no botão “cadastrar aquisição”. Após a operação o animal deve constar na lista de animais do rebanho.

Para inserção de uma nova cria deve-se escolher a opção de “cria”, neste caso o botão de “cadastrar aquisição” será desabilitado, então devem ser preenchidos os dados solicitados na tela e clicar no botão “cadastrar cria”. Após a operação o animal deve constar na lista de animais do rebanho.

Para a inserção de dados de uma nova cobertura dos animais deve-se informar a possível futura mãe, o possível futuro pai e a data da cobertura e clicar no botão “Inserir”. Após a operação a cobertura deve constar na lista de coberturas.

Figura 4 - Tela de Cadastro de Coberturas.

The screenshot shows a web browser window titled 'Controle Bovino - Mozilla Firefox'. The address bar shows 'localhost:8084/WEB-INF/template_admin.jsf'. The page title is 'Controle Bovino'. Below the title is a section titled 'Cadastro de coberturas' with a form containing three fields: 'Mae:' with a dropdown menu showing 'criasans', 'Pai:' with a dropdown menu showing 'Pegador', and 'Data Cobertura:' with a text input field. Below these fields are two buttons: 'Inserir' and 'Voltar'. Below the form is a table titled 'Lista de Coberturas' with the following data:

Data Cobertura	pai	mae	Gestação		
03/01/2012	Pegador	Princesa	false	Excluir	Informar Gestação
04/10/2011	Pegador	criasans	false	Excluir	Informar Gestação
01/04/2009	Pegador	Princesa	false	Excluir	Informar Gestação
01/01/2006	Pegador	Princesa	true	Excluir	Informar Gestação
01/09/2010	teste3semem	Princesa	false	Excluir	Informar Gestação

Nesta tela para o usuário também estão disponíveis as opções de excluir uma cobertura e informar uma gestação gerada a partir de uma cobertura.

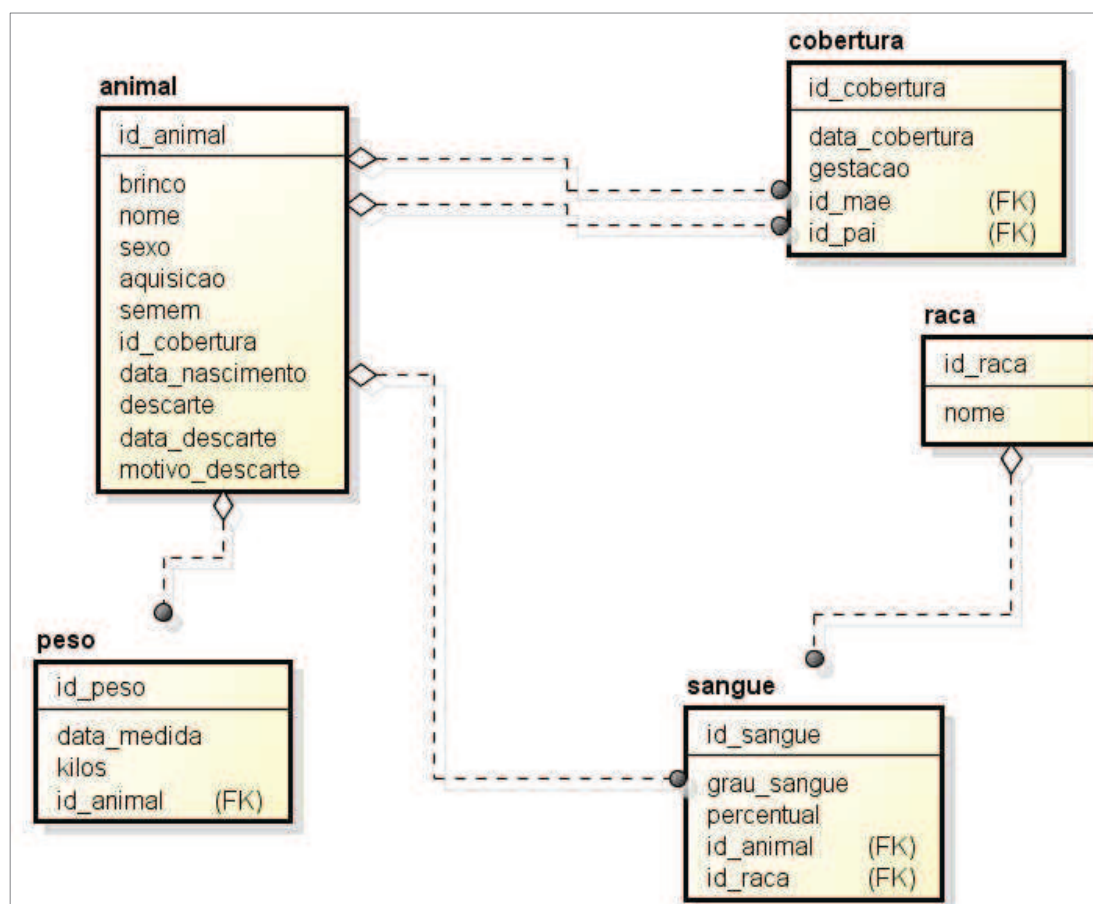
Para excluir uma cobertura que tenha sido realizado de forma indevida, basta escolher a cobertura na lista de coberturas, clicar no botão “Excluir” e confirmar a exclusão.

Para informar uma gestação, basta escolher a cobertura na lista de coberturas, clicar no botão “Informar Gestação” onde abrirá uma nova tela para informação da gestação.

6 Diagrama de Entidade-Relacionamento

Com base no modelo de domínio especificado, foi possível realizar o projeto banco de dados. Abaixo segue o Diagrama Entidade-Relacionamento (DER) do sistema, representando o projeto lógico do banco de dados usado.

Figura 6 - Diagrama de Entidade-Relacionamento.



Com a elaboração do DER, e finalizado o projeto lógico do banco de dados, foi passado para elaboração o modelo físico e, assim, criado o banco de dados no PostgreSQL. Esta foi a última etapa executada para o entendimento e elaboração dos requisitos do sistema, não sendo mais necessárias outras atividades, dada a simplicidade dos requisitos.

7 DESENVOLVIMENTO

Com a finalização do DER, casos de uso e modelo de domínio, a documentação mínima necessária para entender os requisitos do sistema fica pronta, permitindo o início do desenvolvimento do seu código. Mas, antes de detalhar a programação do sistema, faz-se necessário apresentar superficialmente as tecnologias usadas em seu desenvolvimento.

7.1 Tecnologias utilizadas

7.1.1 JavaServer Faces (JSF)

O JavaServer Faces (JSF) é um framework web MVC desenvolvido com uma filosofia baseada em componentes. A sua versão 2.0 foi criada e usada por grupos de especialistas da Apache, Oracle, Redhat, IceSoft entre outras empresas, responsáveis pela especificação e desenvolvimento de sistemas. Ele já é o framework Java mais usado para desenvolvimento de aplicações web. Seu excelente conjunto de funcionalidades faz com que esta tecnologia seja a favorita no cenário de desenvolvimento em que está inserida. Isto se deve a vários motivos: [Magazine78]

- É uma especificação que faz parte da Enterprise Edition desde a versão 5;
- É um framework cuja API também foi pensada para os desenvolvedores de IDEs (Eclipse, NetBeans, JDeveloper, ...), não somente para os desenvolvedores de aplicação;
- Tem soluções para necessidades comuns do desenvolvimento web, propostas e consolidadas a partir da experiência de outros frameworks de mercado moldado em 3 camadas (MVC – Model View Controller), tais como Struts, Tapestry e Spring Web Flow;
- Faz parte da certificação SCWCD 6 junto com Servlets e JSP;
- Possui uma comunidade bastante ativa em fóruns e no IRC;

- Possui bons materiais de estudo no formato de livros, documentos digitais e sites;
- Exige pouco conhecimento inicial para construção de interfaces de usuário tradicionais, como consultas baseadas em lista e cadastros;
- Preparada para ser estendida, possui ótimas bibliotecas de componentes livres ou pagas desenvolvidas por terceiros. Estas bibliotecas oferecem recursos interativos similares aos encontrados na interface de usuário para desktop, evitando assim a necessidade da equipe de desenvolvimento dominar tecnologias cliente como HTML, Javascript e CSS;
- Oferece capacidades interativas integradas com AJAX. Isto com mínima necessidade de escrever Javascript.

No ambiente corporativo atual, os sistemas de informação são ferramentas indispensáveis que auxiliam as empresas a automatizar seus processos de negócio. O uso de aplicações web para desenvolver sistemas de informações é uma prática comum desde pequenas empresas a grandes corporações. As aplicações web objetivam: converter e validar dados, integrar com o banco de dados ou outra forma de persistência, oferecer mecanismos de segurança, internacionalização, localização, acessibilidade, suporte a css, suporte a criação e uso de templates, entre muitas outras funcionalidades.

Frameworks que comportam a criação de sistemas web, mantendo as características supracitadas, tornam as tarefas de desenvolvimento mais simples. O JSF é um framework que oferece todas estas soluções. É uma estrutura de interface de usuário (UI) para aplicativos Java Web projetado para facilitar significativamente a trabalhosa tarefa de escrever e manter os aplicativos que são executados em um servidor de aplicativo Java e a processar as UIs de volta ao cliente de destino. [JSF - NetBeans]

O JSF oferece facilidade de uso das seguintes formas:

- Facilita a construção de uma IU a partir de um conjunto de componentes de IU reutilizáveis
- Simplifica a migração de dados do aplicativo para e a partir da IU
- Ajuda a gerenciar o estado da IU nas solicitações do servidor
- Oferece um modelo simples de conectar os eventos gerados pelo cliente ao código do aplicativo do lado do servidor

- Permite personalizar os componentes de UI para que sejam facilmente construídos e reutilizados

Desvantagens de usar JSF

- JSF é um framework baseado em componentes. Este fato traz consigo restrições inerentes que têm a ver com obediência à componente do modelo.
- As mensagens de erro que são exibidos por padrão não são boas. Existe a necessidade de personalizar a mensagem de erro padrão.
- Não é escalável. Objeto de sessão é usado para armazenar o estado do componente.
- Curva de aprendizado, não só de JSF, mas das bibliotecas de componentes necessárias em para uso é bastante longa.

7.1.1.1 Managed Beans

Além da tecnologia JSF, esse trabalho fez uso de Managed Beans. A principal responsabilidade de um Managed Bean é intermediar a comunicação entre as páginas (componentes do JSF) e nosso modelo. [Rafael Ponte]

Escutar eventos, processá-los e delegar para a camada de negócios são apenas algumas de suas características.

Suas principais responsabilidades são: **[Apostilas K19]**

- Fornecer dados que serão exibidos nas telas.
- Receber os dados enviados nas requisições.
- Executar tarefas de acordo com as ações dos usuários.

7.1.2 Spring

O Framework escolhido para a construção da camada de negócio do sistema foi o framework Spring. Ele provê diversos benefícios para o desenvolvimento de projetos de Sistemas de Informação, aumentando a produtividade de desenvolvimento e a performance das aplicações, fornecendo ao mesmo tempo uma cobertura para testes dentre outras melhorias para melhorar a qualidade da aplicação. [Diego Pacheco]

O Spring provê uma solução de Framework MVC completa de forma transparente e elegante, direcionando o desenvolvimento da arquitetura do sistema de forma a torná-la extremamente modular e desacoplada.

O Spring objetiva os seguintes valores: [Diego Pacheco]

- É melhor programar para interfaces do que para classes, reduz o custo e a complexidade de construir interfaces.

- JavaBeans oferece uma maneira excelente para configurar uma aplicação.

– Inversion of control: Com a grande demanda de desenvolvimento para o Java corporativo (JEE), surgiu um problema comum, como ligar a camada de acesso os dados com a camada de regra de negócio e por ventura a camada de apresentação. O Spring framework possui um container de loc leve e implementa o pattern de setter injection.

A Idéia principal da injeção de dependências é separar os Objetos e principalmente o objeto que usa outro objeto não instanciar diretamente esse objeto. Sendo assim, a principal vantagem da Dependency Injection é separar uma interface de sua implementação concreta.

Vantagens de usar Spring

- A injeção de dependências permite a eliminação/redução do acoplamento, porque não é necessário mais instanciar a classe diretamente de outra classe, e sim injetá-la no local desejado.
- Simplificação no código da aplicação. Com a injeção de dependência, os objetos da aplicação são liberados da tarefa de trazer suas próprias dependências (redução de código para inicialização e configuração de dependências); eles ficam livres para executar apenas suas regras negociais, pois, sabem que as suas dependências estarão lá quando necessárias.
- Testabilidade. Se as dependências podem ser injetadas em um componente, torna-se possível (e talvez até mais fácil) injetar implementações para essas dependências.
- Baixo acoplamento entre os objetos. O objeto conhece suas dependências apenas por sua interface pública (não por sua implementação, nem por como foram instanciadas). Assim, a dependência pode ser trocada por uma implementação diferente, sem que o objeto dependente conheça a diferença.
- Flexibilidade no gerenciamento do ciclo de vida dos objetos. Objetos podem ser cacheados, serem um singleton ou ter uma vida curta – tudo controlado por meio de configurações e/ou pelo container.
- Facilidade de integração com outros frameworks existentes (ex:Hibernate).

Desvantagens de usar Spring

- Dificuldade no acompanhamento do código. A existência de um método no código torna possível saber que ele está sendo chamado para injetar um valor, mas não se tem certeza de quando isso irá acontecer ou qual o valor que será injetado. Isso pode dificultar, inclusive, a rastreabilidade.
- Quando se utiliza a DI baseada em propriedades (injeção por métodos), faz-se necessário marcar suas dependências como públicas. Para que se utilize uma injeção via `setXxx()`, por exemplo, esse método tem de ser tornado público. Isso apenas por causa da injeção de dependência. Isso faz com que quebre um dos bons princípios de OO: a imutabilidade dos objetos.
- A cadeia de objetos pode se tornar muito grande. Mesmo que eu precise fazer apenas uma operação, todas as dependências são criadas e/ou precisam ser resolvidas.

7.1.3 JPA e Hibernate

Nas linguagens de programação modernas como Java, o conceito de orientação a objetos é inerente à tecnologia. Os dados são manipulados no formato de objetos, porém na maioria das vezes são persistidos em bancos de dados relacionais, pois os bancos de dados orientados a objetos não são tão robustos, eficientes, desenvolvidos ou possuem grande suporte como os relacionais. Como uma alternativa para esse problema, foram desenvolvidas ferramentas e frameworks de mapeamento objeto/relacional (MOR), e dentre essas ferramentas se destaca o JPA (Java Persistence API) e o Hibernate.

A JPA foi criada com o objetivo de padronizar as ferramentas ORM para aplicações Java e consequentemente diminuir a complexidade do desenvolvimento, ela especifica um conjunto de classes e métodos que as ferramentas ORM devem implementar. Ela é apenas uma especificação, ou seja, ela não implementa nenhum código. Por isso foi necessário usar uma das implementações da JPA, o Hibernate. [Apostilas K19]

Para a camada de persistência, a aplicação desenvolvida neste trabalho fez uso do framework Hibernate, baseada na JPA.

7.1.3.1 Hibernate

O Hibernate é um framework para mapeamento objeto/relacional em Java, que abstrai o código SQL da aplicação, permitindo, dentre outras coisas, modificar a base de dados para outro SGBD (Sistema Gerenciador de Banco de Dados) sem modificar uma linha de código Java. Ele transforma objetos definidos pelo desenvolvedor em dados tabulares de uma base de dados, portanto com ele o programador se livra de escrever uma grande quantidade de código de acesso ao banco de dados e de SQL. [André Roberto Ortoncelli, Carlos Alberto Jóia Lazarin].

Vantagens de usar Hibernate

- JPA/Hibernate abstrai a lógica de banco de dados, tornando o acesso ao sistema mais próximo ao paradigma Orientado a Objetos.
- Abstrai o banco de dados: o desenvolvedor não necessitará ficar tão atrelado com as tarefas de criação e manutenção das tabelas, uma vez que elas serão conseguidas a partir do mapeamento realizado pelo Hibernate.
- Código limpo: necessidade de escrever menos código repetitivo de persistência, sem perder flexibilidade e o acesso a recursos nativos/específicos dos bancos de dados com menos preocupação na escrita de código de persistência, os arquitetos podem se preocupar mais com o modelo de domínio e com as regras de negócio.
- Tratamento de exceções: melhor tratamento de exceções e gerenciamento de recursos do banco de dados (ex: tratamento de erros relativos à integridade referencial, gerenciamento de transações, pool de conexões etc..).

Desvantagens de usar Hibernate

- Pode existir algum overhead na construção do sistema e o framework possui uma forma diferente da habitual (para desenvolvedores muito acostumados com SQL) – o que exige alguma curva de aprendizagem.
- Nem sempre é possível realizar consultas e operações complexas de forma otimizada, sendo necessário realizar operações relacionais diretas usando *strings* SQL.

7.1.3.2 Mapeamento

Como os bancos e dados não entendem dados orientados a objetos, a solução utilizada pelo Hibernate é utilizar um identificador não natural, assim o banco de dados é capaz de compreender os objetos e montar seus relacionamentos. Dessa forma, os mapeamentos consistem em associar cada tabela da base de dados e seus campos a uma classe, seus atributos e relações, e também montar os identificadores não naturais. [André Roberto Ortoncelli, Carlos Alberto Jóia Lazarin]

As interfaces do Hibernate podem ser classificadas como:

- Interfaces chamadas pela aplicação para executar operações básicas do CRUD (create, retrieve, update, delete). Essas são as principais interfaces de dependência entre a lógica de negócios/controle da aplicação e o Hibernate. Estão incluídas Session, Transaction e Query.
- Interfaces chamadas pela infra-estrutura da aplicação para configurar o Hibernate, mais especificamente Configuration;
- Interfaces responsáveis por realizar a interação entre os eventos do Hibernate e a aplicação: Interceptor, Lifecycle e Validatable.
- Interfaces que permitem a extensão das funcionalidades de mapeamento do Hibernate: UserType, CompositeUserType, IdentifierGenerator.

No Hibernate, há duas maneiras de fazer o mapeamento, via XML, ou utilizando anotações.

7.1.3.2.1 Mapeamento via XML

No mapeamento via XML são criados arquivos XML que devem ter a extensão .hbm.xml, e também devem ser referenciados no arquivo de configuração.

A desvantagem desse tipo de mapeamento é que para cada tabela da base de dados deve ser criado um arquivo de mapeamento e uma classe POJO.

Esse processo é mais trabalhoso que o mapeamento utilizando anotações, porém há ferramentas, como o XDoclet, que podem ser utilizadas para gerar os mapeamentos.

7.1.3.2.2 Mapeamento via Anotações

Com o mapeamento via anotações, não é necessário criar nenhum arquivo XML (ou em qualquer outro formato) para fazer o mapeamento, basta colocar as anotações na classe POJO relacionada à tabela.

7.1.4 PostgreSQL

Como gerenciador de banco de dados para o ambiente de desenvolvimento foi utilizado o banco de dados POSTGRESQL, pelo fato do mesmo ser livre e robusto.

O PostgreSQL é um SGBD (Sistema Gerenciador de Banco de Dados) objetorelacional de código aberto, com mais de 15 anos de desenvolvimento. Baseado no POSTGRES, Versão 4.2, desenvolvido na Universidade da Califórnia no Departamento de Ciências da Computação em Berkeley, É robusto e confiável, além de ser extremamente flexível e rico em recursos. Um de seus atrativos é possuir recursos comuns a banco de dados de grande porte (SQL Server), o que o deixa apto a trabalhar, inclusive, com operações de missão crítica. Além disso, trata-se de um banco de dados versátil, seguro e gratuito disponível sob uma licença BSD.[Documentacao oficial do PostgreSQL]

O PostgreSQL é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados com conformidade e padrões. Ele roda em todos os grandes sistemas operacionais, incluindo GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e MS Windows.

Além de tudo é totalmente compatível com ACID, têm suporte completo à chaves estrangeiras, junções (JOIN), visões, gatilhos e procedimentos armazenados em múltiplas linguagens. Inclui a maior parte dos tipos de dados do ISO SQL: 1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e uma documentação excepcional.

O PostgreSQL teve patrocínio de diversas empresas, entre as quais se destacam: Fujitsu, Hub.Org, NTT Group, Red Hat, Skype, SRA e Sun Microsystems. O software tem adquirido prestígio na comunidade Linux, tendo recebido diversas vezes o prêmio Linux Journal Editor's Choice de melhor Sistema de Gerenciamento de Banco de Dados.

8 Trabalho desenvolvido

A implementação foi realizada através de paginas JSF com uso do frameworks Primefaces como camada de visualização começando com a pagina do menu do

sistema. Veja abaixo trecho de código da pagina do menu principal

Figura 7 - Trecho código menu principal.

```
<h:form >
  <p:panel id="panel" header="Menu Principal">
    <span style="font-family: Verdana; font-size: 20px; color: aqua">
      <li><h:commandLink value="Cadastrar Animal" action="App/animal/formulario"/></li>
      <li><h:commandLink value="Cadastrar Raça" action="App/raça/formulario"/></li>
      <li><h:commandLink value="Cadastrar Cobertura" action="App/cobertura/formulario"/></li>
      <li><h:commandLink value="Cadastrar Peso" action="App/peso/formulario"/></li>
      <li><h:commandLink value="Cadastrar Sangue" action="#{sangueBean.carregaListaSangue}"/></li>
      <li><h:commandLink value="Descartar Animais" action="App/animal/formularioDescarte"/></li>
      <br/><br/>
      <li><h:commandLink value="Listar/Alterar Animais do Rebanho" action="App/animal/listagem"/></li>
      <li><h:commandLink value="Listar Grau Sangue/Raça de Animais do Rebanho" action="App/sangue/listagem"/></li>
      <li><h:commandLink value="Listar Animais Descartados" action="App/animal/listagemDescarte"/></li>
      <li><h:commandLink value="Listar Animais do Rebanho" action="App/animal/listagemfiltro"/></li>
      <li><h:commandLink value="Listar Futuros Nascimento" action="#{coberturaBean.nascimentoPrevistos}"/></li>
    </span>
  </p:panel>
</h:form>
```

e a cada função de menu criado foi gerados outras paginas JSF com uso do frameworks Primefaces. Veja trecho do código da pagina do cadastro de cobertura abaixo

Figura 8 - Trecho código Cadastro de Cobertura.

```
<h:form>
  <p:panel id="panel" header="Cadastro de coberturas">
    <h:panelGrid columns="2">
      <h:outputText value="Mãe:" />
      <h:selectOneMenu value="#{coberturaBean.entidade.mae.id}" required="true">
        <f:selectItems value="#{coberturaBean.animaisFemeas}"
          var="c" itemLabel="#{c.nome}" itemValue="#{c.id}"/>
      </h:selectOneMenu>
      <h:outputText value="Pai:" />
      <h:selectOneMenu value="#{coberturaBean.entidade.pai.id}" required="true">
        <f:selectItems value="#{coberturaBean.animaisMachos}"
          var="p" itemLabel="#{p.nome}" itemValue="#{p.id}"/>
      </h:selectOneMenu>
      <h:outputText value="Data Cobertura:" />
      <p:calendar value="#{coberturaBean.entidade.dataCobertura}" id="popupCal" />
      <p:commandButton value="Inserir" action="#{coberturaBean.cadastrarCobertura}"
        ajax="false"/>
      <p:commandButton value="Voltar" action="#{coberturaBean.voltarMenu}" immediate="true"
        ajax="false"/>
    </h:panelGrid>
  </p:panel>
</h:form>
```

para funções específicas do sistema e a para cada pagina criada foi vinculada a um ou mais ManagerBean.

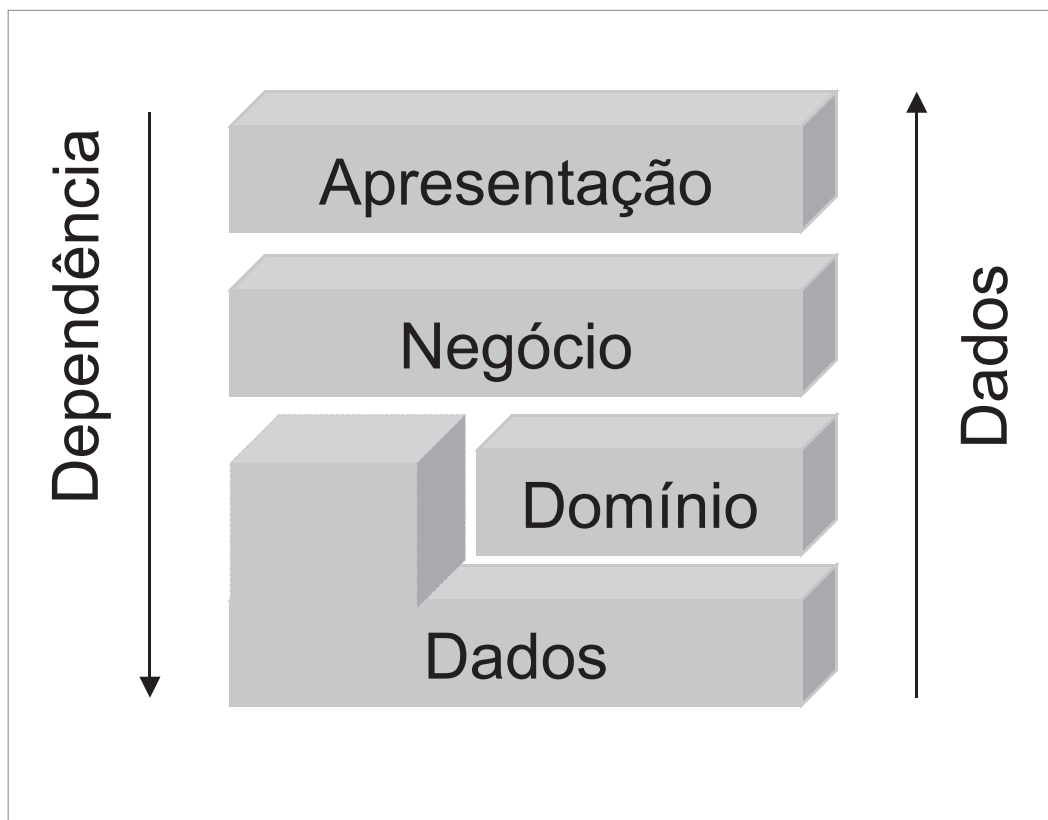
8.1 Estruturação dos pacotes

Na implementação procurou-se organizar o trabalho em camadas e a

organização de componentes por responsabilidades em comum. A explicação do por que da estrutura do projeto estar montada assim foi seguir a idéia de boas praticas.

Neste modelo a lógica de apresentação esta separada em sua própria camada lógica .A separação em camadas lógicas torna os sistemas mais flexíveis permitindo que as partes possam ser alteradas de forma independente. As funcionalidades da camada de negócio podem ser divididas em classes e essas classes podem ser agrupadas em pacotes ou componentes reduzindo as dependências entre as classes e pacotes.

Figura 9 - Apresentação de Camadas.



8.1.1 Pacote 'br.farn.camadas.apresentacao'

Este pacote contém os chamados ManagerBeans que são classes que fazem interface das páginas JSF com a camada de negocio utilizando para isso as classes de domínio, os chamados Beans, são as classes onde estarão os acessos as outras partes do sistema.

Nas classes desse pacote foi usada a injeção de dependência com o uso frameworks Spring, veja trecho do código abaixo

Figura 10 - Trecho código ManegerBeanSangue.

```

40     private List<Animal> listaValidos;
41     @Inject
42     private GerenciadorAnimal gerenciador;
43     @Inject
44     private GerenciadorSangueImpl gerenciadorSangue;
45     @Inject
46     private GerenciadorCobertura gerenciadorCobertura;
47
48     public AnimalBean() {
49         super();
50     }
51
52
53     @PostConstruct
54     public void inicializaGerenciadorAnimal() {
55         atualizaListaAnimal();
56     }
57
58     public void atualizaListaAnimal() {
59         animaisSemCadSangue = gerenciador.buscarAnimaisSemCadSangue();
60         listaValidos = gerenciador.buscarTodosdoRebanho();
61         animaisFemeas = gerenciador.buscarAnimaisFemeas();
62         animaisMachos = gerenciador.buscarAnimaisMachos();
63         descartados = gerenciador.buscarAnimaisDescartados();
64     }
65
66
67     public String listaDescarte() {
68         listaValidos = gerenciador.buscarTodosdoRebanho();
69         return getPastEntidade() + "formularioDescarte";
70     }

```

8.1.2 Pacote 'br.farn.camadas.dominio'

Este pacote possui as classes que contém os domínios dos objetos. Por exemplo, caso queira incluir um objeto no banco (um animal), o formulário irá capturar as informações dos campos do formulário, será instanciado um objeto da classe 'animal', e os valores provenientes do formulário serão atribuídos a esse novo objeto de valor, que por sua vez será persistido no banco de dados.

Nas classes desse pacote foram usadas as anotações do Hibernate para persistência dos dados. Veja no trecho do código abaixo

Figura 11 - Trecho código domínio Animal.

```

34 @Table(name = "animal")
35 public class Animal implements Serializable {
36
37     @Id
38     @GenericGenerator(name = "seq_animal", strategy = "sequence")
39     @GeneratedValue(generator = "seq_animal")
40     @Column(name = "id_animal")
41     private long id;
42     @OneToMany(mappedBy = "animal")
43     @Cascade(CascadeType.DELETE)
44     private List<Peso> pesos = new ArrayList<Peso>();
45     @OneToMany(mappedBy = "mae")
46     private List<Cobertura> coberturasMae = new ArrayList<Cobertura>();
47     @OneToMany(mappedBy = "pai")
48     private List<Cobertura> coberturasPai = new ArrayList<Cobertura>();
49     @ManyToOne
50     @JoinColumn(name = "id_cobertura")
51     private Cobertura cobertura;
52     @Cascade(CascadeType.DELETE)
53     @OneToMany(mappedBy = "animal")
54     private List<Sangue> sangues = new ArrayList<Sangue>();
55
56     @Min(value = 1, message = "Brinco: não pode ser abaixo de 1 unidade")
57     private long brinco;
58     private String nome;
59     private String sexo;
60     private String aquisicao;
61     private boolean semem = false;
62     @NotNull(message = "Data Nascimento não pode ser nula")
63     @Past(message = "Data Nascimento não pode ser futura")

```

8.1.3 Pacote 'br.farn.camadas.negocio'

Neste pacote se encontram as classes que irão fazer a interface entre a camada de apresentação e a camada de persistência de dados trabalhando a parte lógica e de negócios da aplicação.

Nas classes estão contidas as lógicas de negócios. Veja no trecho do código abaixo

Figura 12 - Trecho código GerenciadorSangue.

```

39 public HashMap<Long, Float> calculaSangueAnimal(long id_mae, long id_pai, long id_filho) {
40     HashMap<Long, Float> tsanguesFilho = new HashMap<Long, Float>();
41     List<Sangue> listaSanguePai = new ArrayList<Sangue>();
42     List<Sangue> listaSangueMae = new ArrayList<Sangue>();
43     listaSangueMae = buscarSanguePorIdAnimal(id_mae);
44     listaSanguePai = buscarSanguePorIdAnimal(id_pai);
45     for (Sangue sangue : listaSangueMae) {
46         tsanguesFilho.put(sangue.getRaca().getId(), sangue.getPercentual() / 2);
47     }
48
49     for (Sangue sangue : listaSanguePai) {
50         if (tsanguesFilho.get(sangue.getRaca().getId()) == null) {
51             tsanguesFilho.put(sangue.getRaca().getId(), sangue.getPercentual() / 2);
52         } else {
53             tsanguesFilho.put(sangue.getRaca().getId(),
54                             (tsanguesFilho.get(sangue.getRaca().getId()).floatValue()
55                              + (sangue.getPercentual() / 2)));
56         }
57     }
58     cadastraVarios(tsanguesFilho, id_filho);
59     return tsanguesFilho;
60 }
61
62
63
64

```

8.1.4 Pacote 'br.farn.camadas.dados'

Neste pacote se encontram as classes com a responsabilidade de tratar a parte de dados, de persistência no banco de dados.

Nas classes desse pacote foram usadas os comandos SQL de persistência e recuperação de dados do Hibernate. Veja no trecho do código abaixo

Figura 13 - Trecho código AnimalDao.

```

14 import org.springframework.stereotype.Component;
15
16 /**
17  *
18  * @author severino
19  */
20 @Component("AnimalDao")
21 public class AnimalDaoImpl extends AbstractDao<Animal> implements AnimalDao {
22
23     @Inject
24     public List<Animal> buscarAnimaisFemeas() {
25         Date dataModificada = Util.adicionaMeses(new Date(), -15);
26         return ht.find("from Animal a where a.descarte = 'FALSE' and a.semm = 'FALSE' and "
27                       + "a.sexo = 'F' and a.dataNascimento < ? order by a.nome", dataModificada);
28     }
29
30     public List<Animal> buscarAnimaisMachos() {
31         Date dataModificada = Util.adicionaMeses(new Date(), -15);
32         return ht.find("from Animal a where a.descarte = 'FALSE' and a.sexo = 'M' and "
33                       + "a.dataNascimento < ? order by a.nome", dataModificada);
34     }
35
36     public List<Animal> buscarTodosdoRebanho() {
37
38         return ht.find(" from Animal a where a.semm = 'FALSE' and a.descarte = 'FALSE' order by a.brinco ");
39     }
40
41     }
42
43     public void updateDescarte(long id, String motivo, Date dataDescarte) {
44         boolean descarte = true;
45         ht.bulkUpdate(" update Animal a set a.descarte = ?, a.motivoDescarte = ?, a.dataDescarte = ? "
46                     + "where a.id = ?", descarte, motivo, dataDescarte, id);
47     }
48

```

8.2 Algumas dificuldades encontradas na implementação

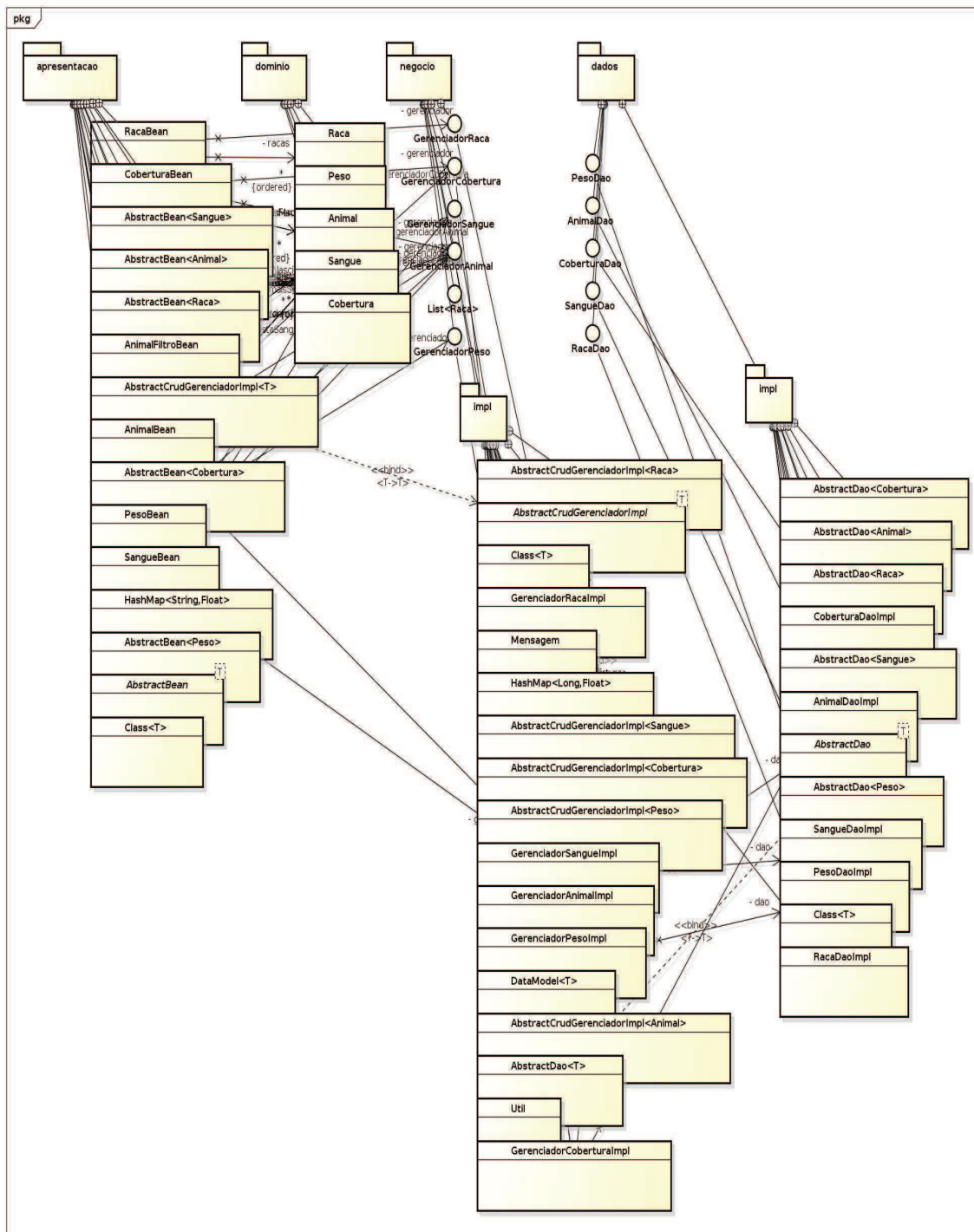
Dificuldade ou imaturidade ou desconhecimento do uso do Primefaces como por exemplo, tentar usar para descarte de vários animais o componente DataTable RadioCheckBox não se sabe porque, não se conseguiu pegar os dados da tela no Maganerbean para usar nas regras de negocio e assim atualizar dados de animais descartados. Então a solução que foi o uso de DataTable – Multi Row Selection que ai sim conseguiu a partir da seleção de varias linhas fazer o descarte de vários animais de uma só vez.

Outra dificuldade foi usar a classe abstract como classe genérica e tentar a partir da classe genérica, usar todas as funcionalidades dos objetos, sendo que por exemplo, o objeto animal foi modelado com um atributo boolean para identificar os animais descartados, e sendo assim, a consulta de recuperação de todos os animais no hibernate não atende pois neste caso trazia todos os animais inclusive os descartados e para trazer só animais do rebanho é preciso atender a condição de não trazer os animais descartados. E nessa mesma linha teve-se problemas para ordenar as consultas por determinado atributo, ou seja para muitos casos o genérico não dar para ser usado sendo necessário umas consultas mais especificas, e como foi gasto muito tempo na tentativa do uso de classes genéricas ficou a duvida se é valido o uso dessas classes.

8.3 Arquitetura final

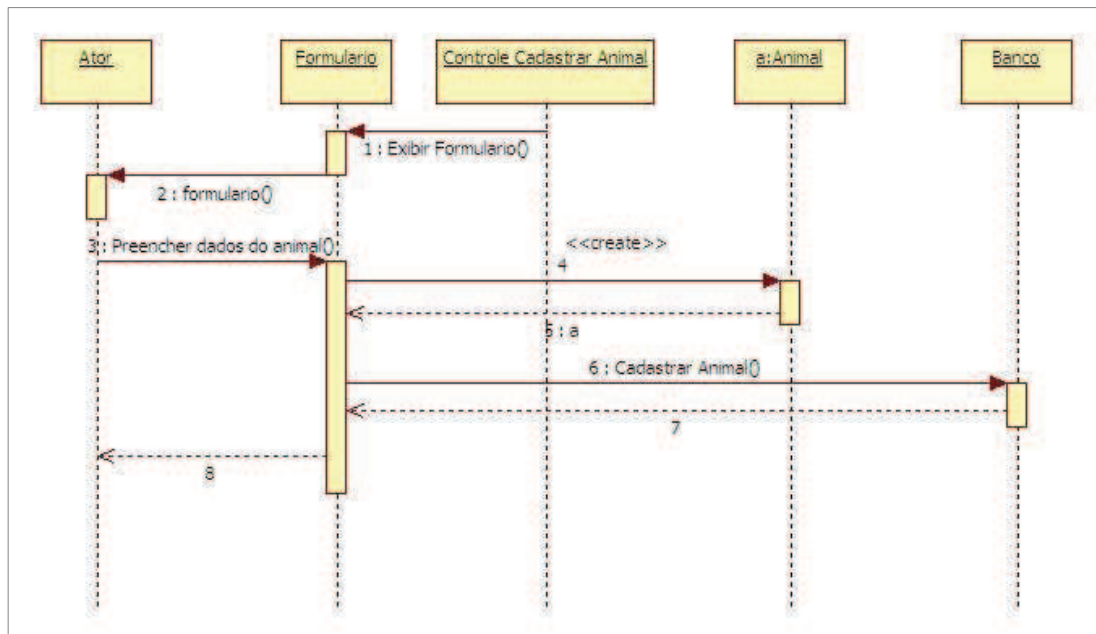
Para construir o sistema, alguns artefatos foram necessários para melhor agrupar as responsabilidades de cada classe e montar de forma eficiente uma arquitetura de sistema WEB. Abaixo, este trabalho apresenta o diagrama de pacotes organizado para apresentar a arquitetura usada e alguns exemplos de diagramas de seqüência usado na criação dos principais métodos controladores.

Figura 14 - Diagrama Arquitetura Final.



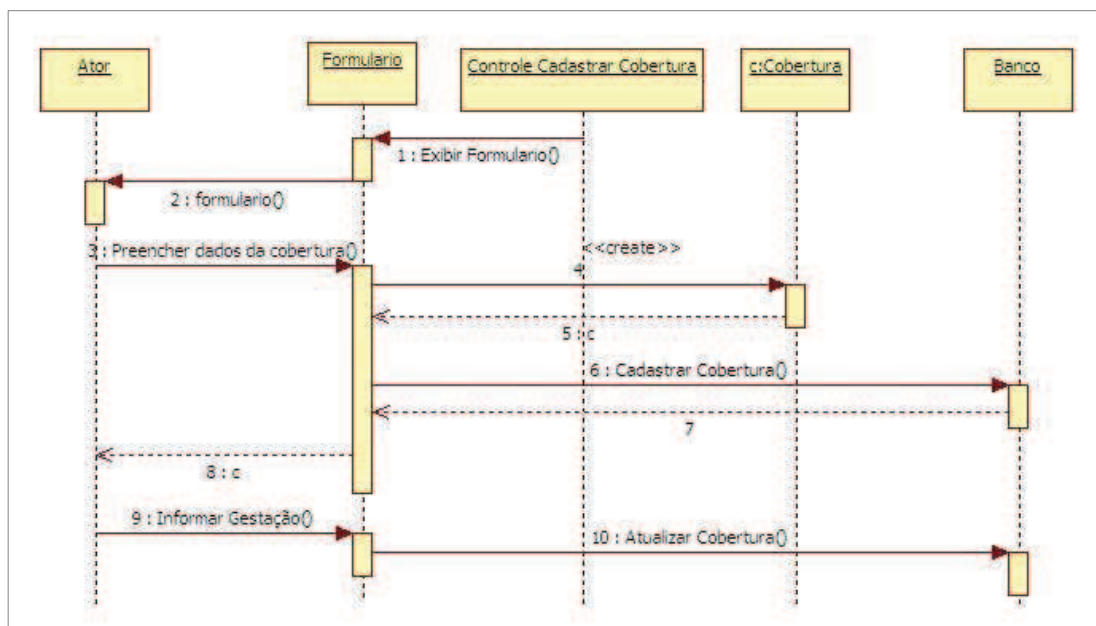
8.4 Diagrama de sequencia - Cadastrar Animal

Figura 15 - Diagrama de sequencia Cadastrar Animal.



8.5 Diagrama de sequencia - Cadastrar Cobertura

Figura 16 - Diagrama de sequencia Cadastrar Cobertura.



9 Conclusão

Este trabalho foi desenvolvido com o objetivo de aplicar o conhecimento adquirido durante o curso, tendo seu foco principal o estudo e desenvolvimento de aplicações WEB. Dentre os principais benefícios adquiridos neste trabalho, é possível citar o conhecimento adquirido no uso dos frameworks JSF, Primefaces, SPRING e Hibernate, aliados ao uso das melhores práticas do mercado de programação Java.

A solução desenvolvida usando este conhecimento foi um software criado para auxiliar o manejo de rebanhos, mantendo o controle de animais, coberturas, raças e grupo de sangue, além de informações a respeito de descartes destes animais.

Durante o desenrolar das atividades realizadas, dado que se tratava do primeiro projeto Java implementado pelo autor, foi inevitável passar por muitas dificuldades no uso da linguagem Java e os frameworks acima citados. Entretanto, estes problemas puderam ser contornados com estudos e pesquisas apontadas nas seções deste texto.

Os resultados alcançados, apesar de satisfatórios, indicam a necessidade de continuação do trabalho, tanto através de melhorias nos artefatos desenvolvidos quanto na expansão do escopo do projeto construído. Seguem algumas sugestões para trabalhos futuros:

- Especificar e desenvolver o módulo responsável pelo controle de custo e de receita do rebanho.
- Especificar e desenvolver o módulo responsável pelo controle de vacinas do rebanho.
- Especificar e desenvolver o módulo responsável por formular as dietas totais de custo mínimo (volumoso + concentrado) de acordo com as características de peso e produtividade dos rebanhos, de modo que a ração formulada atenda às exigências nutricionais do gado leiteiro e/ou de corte.

REFERENCIAS

[André Roberto Ortoncelli Carlos Alberto Jóia Lazarin] Apostila Persistência de dados com Java e Hibernate Annotations Bandeirantes, 2009 Universidade Estadual do Norte do Paraná
Campus Luiz Meneghel
pt.scribd.com/doc/65849905/Apostila-Hibernate

[Apostilas K19] K19 treinamentos Apostila JAVA Desenvolvimento Web com JSF2 e JPA2 www.k19.com.br

[Diego Pacheco] Spring Framework (2.0)Framework paraDesenvolvimento de Aplicaçõesem Java. blog <<http://diego-pacheco.blogspot.com/>>
pt.scribd.com/doc/77704931/Spring-Framework-20-Diego-Pacheco

[Documentacao oficial do PostgreSQL] <http://www.postgresql.org/>

[JSF - NetBeans] Introdução ao JavaServer Faces2.0
[Netbeans.org/kb/docs/web/jsf20-intro_pt_BR.html](http://netbeans.org/kb/docs/web/jsf20-intro_pt_BR.html)

[Magazine 78]Artigo Java Magazine 78 - JSF 2.0
Aprendendo JSF 2.0 com ScrumToys
As novidades da nova versão da especificação padrão para desenvolvimento de interfaces de usuário web com Java

[Rafael Ponte] Managed Beans. Não complique, simplifique.
<http://www.rponte.com.br/2009/08/27/managed-beans-nao-complique-simplifique/>

