Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Hugo Filipe de Pina Jorge
novembro | 2012

Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Aplicação de Gestão dos Animais da Exploração Agrícola Quinta das Marietas

Hugo Filipe de Pina Jorge
nº1009071

Projeto de Informática em contexto de estágio do curso Engenharia Informática

15 de Novembro de 2012

Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Aplicação de Gestão dos Animais da Exploração Agrícola Quinta das Marietas

Hugo Filipe de Pina Jorge
nº1009071

Projeto de Informática em contexto de estágio do curso Engenharia Informática

Supervisor: João Pedro Fernandes Ribeiro, Sócio-Gerente da Casa Agricola das Marietas, Unipessoal, LDA.
Orientador: Mestre José Alberto Quitério Figueiredo, Professor Adjunto da Unidade Técnico-Científica de Informática da ESTG.

15 de Novembro de 2012

Agradecimentos

Primeiro que tudo gostaríamos de agradecer ao Sr. João Pedro Ribeiro por nos propor e ter dado a oportunidade de poder fazer parte do desenvolvimento deste Projeto.

Gostaríamos também de agradecer ao Professor José Quitério por ter aceite o desafio de ser nosso orientador neste projeto, foi sem dúvida uma mais-valia para este projeto pois o seu conhecimento ajudou-nos muito para tornar este projeto uma realidade.

Gostaríamos de agradecer ao Professor José Fonseca pelo precioso apoio e disponibilidade na construção da Base de Dados, foi muito importante pois conseguimos ligar todos os conceitos de uma forma coerente o que facilitou em muito a construção física da nossa aplicação.

Também gostaríamos de agradecer ao professor Paulo Nunes pela disponibilidade demonstrada e fornecida, bem como pela ajuda prestada na planificação e elaboração do relatório em latex.

Por fim mas não menos importante gostaríamos também de agradecer a Professora Doutora Maria Clara Silveira pela grande disponibilidade em nos apoiar com a metodologia, planificação, organização, e análise de todo o projeto foi muito importante para nós. Mais uma vez um muito obrigado a todos.

Num contexto mais pessoal quero ainda agradecer aos meus amigos e à minha família por todo o apoio prestado nesta fase de extrema importância.

Quero ainda agradecer de um modo muito especial ao meu colega e amigo André Gonçalves uma vez que sem ele seria extremamente difícil fazer um trabalho tão complexo e interessante.

Resumo

A evolução da tecnologia permitiu que a informática fosse introduzida na agricultura de modo a ajudar e a facilitar a vida dos agricultores e dos gestores das explorações agrícolas. Com um único Software é possível gerir uma exploração inteira, podendo os gestores incidir a sua gestão e/ou consulta a uma área especifica. Este relatório descreve o trabalho que foi realizado no âmbito da unidade curricular Projeto de Informática na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão da Guarda e consiste na criação de uma aplicação desktop para a gestão de uma exploração agrícola.

O trabalho que nos foi pedido pelo sócio/gerente da exploração Quinta das Marietas consiste no estudo do funcionamento e do necessário para a gestão de uma exploração agrícola, mais propriamente uma exploração de criação de gado bovino. Após o estudo pretende-se proceder ao desenvolvimento de uma aplicação desktop onde seja possível gerir os animais bovinos, manadas e estados produtivos. Esta aplicação irá permitir ao gestor da exploração conter toda a informação sobre a gestão dos animais bovinos, manadas e estados produtivos mais relevante na mesma aplicação, poupando assim tempo e erros. Esta aplicação irá ser criada em Java na plataforma NetBeans IDE 7.2 e com base de dados embutida criada em Java DB, esta escolha foi feita devido ao facto de o gestor da exploração pretender uma aplicação desktop sem ter necessidade de colocar a base de dados num servidor, ou de instalar outro programa para poder aceder à base de dados.

Palavras Chave

Aplicação Desktop, Java, Base de Dados, Exploração Agrícola, Gestão Agrícola.

Abstract

The evolution of technology has enabled computing of being introduced in agriculture in order to assist and facilitate the lives of farmers and farm managers. With a single software it is possible to manage an entire farm, and managers can focus their management and / or query to a specific area. This report describes the work done within the course in Computer Project Degree in Computer Science from the School of Technology and Management of Guarda and consists of creating a desktop application for managing a farm.

The work that we have been asked by the member/manager of farm 'Quinta das Marietas' is the study of the functioning and necessities for the operation of a farm, more specifically an exploration of raising cattle. After the study we pretend to develop a desktop application where you can manage your farm animals, herd and productive state. This application will enable the manager of exploration contain all relevant information in the same application, thus saving time and errors. This application will be created in java in NetBeans IDE 7.2 Platform with embedded database created in java DB. This choice was made due to the fact that the manager want a desktop application without having to put the database on a server, or install another program to access the database.

Key words

Application Desktop, Java, Database, Farm, Agricultural Management.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Glossário

Java — Linguagem de programação.

JavaDB — Compilador com base em apache Derby que vem com o Netbeans IDE 7.2 - serve para construir uma base de dados.

MADg — Matéria Azotada Digestível por grama.

NetBeans IDE 7.2 — Plataforma de desenvolvimento de programação com compilador.

PDIEg — Proteína Digestível no Intestino Permitida pela Energia do Alimento por grama.

PDINg — Proteína Digestível no Intestino Permitida pelo Azoto do Alimento por grama.

UEB — Capacidade de Ingestão.

UFL — Unidades Forrageiras Leiteiras.

UFV — Ou mais propriamente UFC - Unidades Forrageiras de Carne (em françês Carne é Viande).

Capítulo 1

Introdução

No presente a tecnologia faz parte do quotidiano dos mais diversos setores não sendo a Agricultura uma exceção a regra. Na busca de maior produtividade e qualidade, os agricultores procuram cada vez mais os equipamentos tecnológicos para usarem nas suas explorações. Os que se recusam a entrar nesta realidade perdem espaço no mercado e competitividade. Qualquer empresa precisa de ter habilidade para competir com a concorrência, sendo os meios mais eficazes oferecer agilidade e qualidade, mas quando se trata de tecnologia não nos podemos esquecer que ela está em constante evolução e que o facto de adotar uma solução tecnológica hoje não significa que seja uma boa solução tecnológica amanhã, ou seja temos de acompanhar a mudança e sempre a procura de uma solução melhor [2].

Nem só os vários equipamentos que as empresas implementam nas suas explorações são importantes, os programas usados também o são cada vez mais de modo a facilitar a gestão das mesmas, tornando assim os seus registos mais completos e mais facilmente acessíveis.

Numa exploração agrícola os registos dos dados dos animais tais como o número do brinco, a raça, o peso, a categoria corporal em que se encontra o animal (se são muito magros têm uma categoria corporal menor, se são mais bem constituídos têm uma categoria corporal mais elevada, sendo que esta categoria é enumerada de 1 a 5), o estado produtivo, entre outros dados são de extrema importância uma vez que vai ser isso que vai distinguir uns animais dos outros além de influenciar no tipo de alimentação que cada animal necessita e que é distribuído de acordo com o estado produtivo em que o animal se encontra (aleitamento, manutenção, gestação ou engorda).

Outra parte de extrema importância numa exploração agrícola é a escolha dos vários alimentos relativamente aos seus constituintes nutritivos e o seu tipo (Forragem ou Concentrados (grãos, frutos, raízes, etc.)), bem como as necessidades energéticas diárias dos animais de acordo com a disponibilidade na exploração. Esta parte é fulcral uma vez que vai ser uma boa alimentação o principal factor de produtividade da exploração pois facilita em obter animais bem constituídos e saudáveis, sendo ao mesmo tempo uma parte em que se tem que ter muito cuidado, pois se a aplicação de uma alimentação aos animais não for a mais correta pode ser extremamente prejudicial aos animais.

Outro conceito importante numa exploração agrícola, principalmente quando esta se foca na criação e venda de animais, é ter uma aplicação dedicada à sanidade dos animais de modo a se poder registar as doenças de cada animal assim como as intervenções veterinárias e os medicamentos prescritos num caso específico, tendo em conta o tempo de duração da doença.

As aplicações informáticas hoje em dia são de extrema importância uma vez que conseguem executar tudo o que foi descrito anteriormente melhorando substancialmente o trabalho, ajudando a reduzir despesas e a tornar as explorações agrícolas muito mais eficientes e competitivas.

1.1 Motivação

A principal motivação para o desenvolvimento deste projeto é a possibilidade em contribuir para o desenvolvimento de uma aplicação para uma exploração agrícola de forma a ajudar a sua evolução bem como a sua integração com as ferramentas de gestão.
Além desta motivação o que nos cativou e motivou a desenvolver esta aplicação foi o reconhecermos a importância da agricultura no passado, presente e futuro.
O facto de esta área de trabalho, apesar de não se tratar do mesmo, ter algum relacionamento com uma outra aplicação já desenvolvida no âmbito da unidade curricular Engenharia de Software II em que o objetivo era criar uma solução para o problema da fome em algumas regiões do mundo, fez com que ficássemos ligados com a proposta e cativou-nos a aprender e a estudar para nos ser possível fazer esta aplicação de uma forma empenhada e concreta.
Também o facto de nos obrigar a pesquisar e a abrir os nossos horizontes de forma a por em prática muitos dos conhecimentos adquiridos ao longo do curso e acima de tudo criar um Software que vai ser implementado e facilitar a vida de alguém dá uma enorme satisfação e vontade de fazer o melhor possível. Estes foram alguns dos factores que nos influenciaram na escolha e desenvolvimento deste projeto.

1.2 Solução

A solução encontrada para a proposta que nos foi feita e de acordo com os requisitos pedidos e pretendidos foi a criação de uma aplicação desktop de gestão, solução desenvolvida em Java na plataforma Netbeans IDE 7.2. Esta aplicação pretende ser uma ferramenta de apoio na gestão de uma exploração agrícola de modo a poder facilitar o modo de gerir a exploração, nas vertentes dos animais bovinos, manadas e estados produtivos.

1.3 Contribuição

A contribuição principal deste trabalho é o desenvolvimento, implementação e testes de uma aplicação destktop de gestão de gado bovino, mais propriamente gestão de

gado bovino, manadas e estados produtivos, ajudando assim ao desenvolvimento de uma exploração agrícola de modo a facilitar e a ajudar a inovar no modo de gerir a mesma tornando tudo mais acessível.

1.4 Estrutura do documento

O documento compreende cinco capítulos, para além da presente introdução.

No segundo capítulo é apresentado o estado da arte, onde fazemos referencia a algumas das aplicações já existentes no mercado, e apresentamos a nossa opinião em relação a nossa aplicação.

No terceiro capítulo é descrita a metodologia a seguir e descrição das tarefas.

No quarto capítulo é descrita a análise pormenorizada dos requisitos necessários à nossa aplicação.

No quinto capítulo descreve-se a implementação da solução proposta com algumas imagens de janelas da nossa aplicação acompanhado de algum código.

Finalmente, no capítulo seis, são apresentadas as conclusões mais relevantes do trabalho, e as perspectivas de desenvolvimento que se pretendem efetuar no futuro.

1.5 Definição do problema

Desenvolver uma aplicação desktop para a exploração agrícola Quinta das Marietas, de modo a ter todas as funcionalidades necessárias: Gestão dos animais, manadas, categorias de estado produtivo, brincos, peso, e o utilizador poder ver o histórico de um animal. Gerir e definir alimentação de acordo com cada manada, em que categoria de estado produtivo se encontram os animais da manada e o número de animais em cada categoria. Gerir a sanidade dos animais de modo a poder registar as doenças de cada animal assim como as intervenções veterinárias e os medicamentos prescritos num caso específico, tendo em conta o tempo de duração da doença. A aplicação deve possuir uma base de dados embutida de modo a não ser necessário a instalação de qualquer outro programa ou ter a base de dados num servidor independente. Para a realização do projeto a que nos propomos é necessário ultrapassar vários obstáculos de modo a não haver falhas de troca de informação dentro da aplicação.

Os problemas iniciais que foram necessários resolver para a criação da aplicação a que no propusemos são os seguintes:

- Como obter a informação dinamicamente:
- Perceber o dinamismo e a rotatividade dos animais nos diferentes estados produtivos.
- Criar um modelo entidade relacionamento, pois com o passar do tempo e a medida que melhor compreendíamos o problema, o modelo entidade relacionamento estava em constante mudança impedindo assim o início da componente física da aplicação.

- Em que plataforma criar uma base de dados eficiente e sem falhas mas de modo a ficar embutida no programa evitando assim a necessidade de instalar outro programa para aceder à mesma ou a necessidade de a colocar num servidor.
- Como criar o registo de animais, manadas, categoria de estados produtivos e tudo o que evolve a gestão destes tópicos de modo a poder fazer as seguintes associações e criações:
  - Como inserir números para os brincos;
  - Como atribuir a cada animal um brinco individual e único, brinco este que se vai tornar a identificação do animal;
  - Como poder criar manadas para se poder separar os animais de acordo com as necessidades existentes;
  - Como associar os animais às respetivas manadas;
  - Como poder criar categorias de estado produtivo de acordo com os estados produtivos existentes;
  - Como associar as manadas às respetivas categorias de estado produtivo;
  - Como poder alterar as manadas em que os animais se encontram tendo em atenção os animais que lá se encontram;
  - Como poder alterar as categorias de estado produtivo em que as manadas se encontram;
- Como efetuar o cálculo da alimentação.
  - Integrar a informação — Como relacionar a informação dos animais, manadas e categorias de estado produtivo com a alimentação.
  - Qual o cálculo a utilizar consoante o estado produtivo.
  - Qual a relação nutrientes do alimento como as necessidades energéticas do animal.
  - Como resolver um sistema de equações em Java.
  - Como fazer a verificação do cálculo da alimentação.
- Como efetuar a gestão da sanidade animal.

1.6 Objetivos previstos

Os objetivos que pretendemos atingir consistem:

- Criar, editar, pesquisar informação dos diversos animais.
- Criar, editar, eliminar, pesquisar manadas.
- Criar, editar, eliminar, pesquisar categorias de estados produtivos.
- Permitir a inserção de brincos.
- Atribuir um brinco a cada animal.
- Associar manadas aos animais.
- Associar categorias de estado produtivo às manadas.
- Criar, editar, eliminar, pesquisar os vários tipos de alimentos e alimentos.
- Calcular e definir a alimentação para cada manada de acordo com as categorias de estado produtivo de cada animal que se encontra na respetiva manada.
- Criar, editar, eliminar, pesquisar casos de sanidade de cada animal.
- Criar, editar, eliminar, pesquisar intervenções veterinárias de um determinado caso de sanidade de um animal bem como os medicamentos prescritos por intervenção.

Para atingir os objetivos previstos vamos utilizar para consulta essencialmente o livro sobre a alimentação de bovinos [3], e que vai ser usado sistematicamente ao longo de todos os outros capítulos.

Capítulo 2

Estado da arte

2.1 Introdução

As aplicações existentes são consideradas aplicações objetivas e focam-se essencialmente na gestão financeira da exploração, tal como produtividade das colheitas, receitas e custos, e pelo que nos pesquisamos são apenas aplicações genéricas não se destinam a um tipo de exploração especifico (ex. criação de gado, produção de cereais, etc.), enquanto a nossa foca-se essencialmente na criação de gado bovino permitindo ao utilizador uma gestão ampliada sobre esse assunto.

2.2 Exemplos de Aplicações existentes

Como exemplos de aplicações existentes vamos falar de duas aplicações, SoftAgro: S.A - Produtor descrita no sub-capitulo 2.2.1 e AGROGESTÃO: AgroGestão + ZooGestão descrita no sub-capitulo 2.2.2, que são ambas aplicações de apoio a gestão de explorações agrícolas.

2.2.1 SoftAgro: S.A - Produtor [5]

- Proporciona segurança com acesso somente a utilizadores registados e permissões de operação personalizada;

- Controlo de todos os custos envolvidos na produção agrícola, tais como: atividade de preparação; atividades de cultivo; máquinas, veículos e implementação; depreciação e amortização do imobilizado; juros do capital; mão de obra; atividades de colheita e comercialização;

- Controlo financeiro (contas a pagar/pagas, contas a receber/recebidas, caixa e banco) totalmente integrado com todas atividades envolvidas no processo de produção e com os mais diversos tipos de relatórios para obtenção de resultado, podendo inclusive extrair relatórios pelos mais variados tipos de indexadores existente, criando assim um cenário real para cada necessidade;

- Estrutura de análise de custos e produtividade por cultura, talhão, exploração ou produtor;
- Possibilita PROJEÇÃO dos CUSTOS e das RECEITAS através de ORÇAMENTOS DE PRODUÇÃO a partir de uma estrutura de centro de custos e tipos de operação;
- Controlo de stock de produção;
- Entrada para levantamento de ervas por talhão com controlo de histórico de dados relacionados à infestação e ao controlo;
- Entrada para informações e controlo de veículos, máquinas e implementação da propriedade;
- Análise de Fluxo de Caixa projetado e realizado;
- Completo conjunto de formulários para o levantamento e anotação dos dados no campo com o objetivo de facilitar a inclusão no sistema;
- Consultas, Relatórios e Gráficos Analíticos dos Custos e da Produtividade;
- Entrada para diversos tipos de moeda ou indexadores, como Dólar Comercial de Venda e Cotação de saca de Soja;
- Consultas, Relatórios e Gráficos por indexadores. Exemplo: custo de Herbicidas, comparação nas últimas colheitas, em sacas de soja;
- Registo do Produtor com entrada para: Dados Gerais, Endereço Comercial, Endereço de Cobrança, Sócios, Património e Referências;
- Registo de diversas Propriedades (fazendas) por Produtor com entrada para: Dados Gerais, Endereço, dados do Proprietário em caso de arrendamento, inclusive controlo de valor devido pelo arrendamento;
- Toda movimentação realizada de Funcionários, Clientes e Fornecedores classificados por categoria;

2.2.2 AGROGESTÃO: AgroGestão + ZooGestão [1]

AgroGestão:

- Rendimentos globais, por núcleo e sector;
- Determinação de proveitos e custos por cultura, parcela, talhão, folha ou qualquer outro tipo de unidade de análise - actividade;
- Controlo técnico por operação produtiva e respectiva determinação de custos;
- Análise da utilização do aparelho de produção - máquinas, trabalhadores, terra, construções, etc. - e respectiva determinação de custos;

- Controlo de stocks e respectiva e evolução histórica (por armazém e lote).

ZooGestão:

Maneio Administrativo

- Registo dos movimentos administrativos de cada animal (entrada, saída, perdas de brincos, etc).
- Registo de Existências e Deslocações de Bovinos (RED Bovinos) - Homologado pela Direcção Geral de Veterinária.
- Registo de Existências e Deslocações de Ovinos e Caprinos (RED OC).
- Livro de Medicamentos.
- Impressão de Guias SNIRA.
- Possibilidade de gerir múltiplas marcas de exploração, e múltiplas empresas.
- Múltiplos parâmetros de identificação do animal (ex: n.º SIA, n.º Casa, Chip,?).
- Validação do n.º SIA pelo dígito de controlo.
- Controlo de prémios e períodos de retenção dos animais.
- Valorização dos animais por critério para apoio contabilístico.

Maneio Técnico

- Registo completo de cada animal e/ou rebanho.
- Controlo de toda a informação de carácter técnico, como ocorrências e produtividades.
- Controlo de parições, cobrições e diagnósticos de gestação.
- Validação de periodicidade de parições.
- Cálculo de indicadores produtivos, reprodutivos, genealogia e consanguinidade.
- Avisos com base em previsões parametrizáveis.
- Agrupamento de animais por rebanho, lote e classe.
- Possibilidade de associar fotografias, resenhos e esquemas a cada animal.

2.3 Análise Critica

Como foi apresentado no tópico anterior é visível que já existe algum trabalho nesta área. No entanto a nossa aplicação não se limita unicamente ao registo e consulta do que se passa na exploração, mas também permite calcular a alimentação para os animais da exploração. Deste modo, é possível controlar melhor a produtividade da exploração que é o objetivo fundamental de qualquer gestor/empresário de uma exploração.

Capítulo 3

Definição do problema e Objetivos previstos

3.1 Definição do problema

Desenvolver uma aplicação desktop para a exploração agrícola Quinta das Marietas, de modo a ter todas as funcionalidades necessárias: Gestão dos animais, manadas, categorias de estado produtivo, brincos, peso, e o utilizador poder ver o histórico de um animal. Gerir e definir alimentação de acordo com cada manada, em que categoria de estado produtivo se encontram os animais da manada e o número de animais em cada categoria. Gerir a sanidade dos animais de modo a poder registar as doenças de cada animal assim como as intervenções veterinárias e os medicamentos prescritos num caso específico, tendo em conta o tempo de duração da doença. A aplicação deve possuir uma base de dados embutida de modo a não ser necessário a instalação de qualquer outro programa ou ter a base de dados num servidor independente. Para a realização do projeto a que nos propomos é necessário ultrapassar vários obstáculos de modo a não haver falhas de troca de informação dentro da aplicação.

Os problemas iniciais que foram necessários resolver para a criação da aplicação a que no propusemos são os seguintes:

- Como obter a informação dinamicamente:
- Perceber o dinamismo e a rotatividade dos animais nos diferentes estados produtivos.
- Criar um modelo entidade relacionamento, pois com o passar do tempo e a medida que melhor compreendíamos o problema, o modelo entidade relacionamento estava em constante mudança impedindo assim o início da componente física da aplicação.
- Em que plataforma criar uma base de dados eficiente e sem falhas mas de modo a ficar embutida no programa evitando assim a necessidade de instalar outro programa para aceder à mesma ou a necessidade de a colocar num servidor.

- Como criar o registo de animais, manadas, categoria de estados produtivos e tudo o que evolve a gestão destes tópicos de modo a poder fazer as seguintes associações e criações:

  - Como inserir números para os brincos;
  - Como atribuir a cada animal um brinco individual e único, brinco este que se vai tornar a identificação do animal;
  - Como poder criar manadas para se poder separar os animais de acordo com as necessidades existentes;
  - Como associar os animais às respetivas manadas;
  - Como poder criar categorias de estado produtivo de acordo com os estados produtivos existentes;
  - Como associar as manadas às respetivas categorias de estado produtivo;
  - Como poder alterar as manadas em que os animais se encontram tendo em atenção os animais que lá se encontram;
  - Como poder alterar as categorias de estado produtivo em que as manadas se encontram;

- Como efetuar o cálculo da alimentação.

  - Integrar a informação — Como relacionar a informação dos animais, manadas e categorias de estado produtivo com a alimentação.
  - Qual o cálculo a utilizar consoante o estado produtivo.
  - Qual a relação nutrientes do alimento como as necessidades energéticas do animal.
  - Como resolver um sistema de equações em Java.
  - Como fazer a verificação do cálculo da alimentação.

- Como efetuar a gestão da sanidade animal.

3.2 Objetivos previstos

Os objetivos que pretendemos atingir consistem:

- Criar, editar, pesquisar informação dos diversos animais.
- Criar, editar, eliminar, pesquisar manadas.
- Criar, editar, eliminar, pesquisar categorias de estados produtivos.
- Permitir a inserção de brincos.
- Atribuir um brinco a cada animal.
- Associar manadas aos animais.
- Associar categorias de estado produtivo às manadas.
- Criar, editar, eliminar, pesquisar os vários tipos de alimentos e alimentos.
- Calcular e definir a alimentação para cada manada de acordo com as categorias de estado produtivo de cada animal que se encontra na respetiva manada.
- Criar, editar, eliminar, pesquisar casos de sanidade de cada animal.
- Criar, editar, eliminar, pesquisar intervenções veterinárias de um determinado caso de sanidade de um animal bem como os medicamentos prescritos por intervenção.

Para atingir os objetivos previstos vamos utilizar para consulta essencialmente o livro sobre a alimentação de bovinos [3], e que via ser usado sistematicamente ao longo de todos os outros capítulos.

Capítulo 4

Metodologia e resultados esperados

4.1 Metodologia

A metodologia escolhida e utilizada para desenvolver, implementar e testar a aplicação desktop foi o Desenvolvimento Ágil, mais especificamente uma adaptação do Desenvolvimento Ágil XP. Esta abordagem interativa faz com que o cliente avalie o incremento do Software com alguma periodicidade, sendo que receber um feedback constante torna-se bom para a equipa de trabalho pois facilita as adaptações ao processo de desenvolvimento [4]. De facto foi o que fizemos neste projeto o nosso cliente foi envolvido em todas as versões do nosso projeto.

Os princípios do processo de Desenvolvimento Ágil são:

1. Indivíduos e interações em vez de processos e ferramentas - Existiu sempre uma cooperação constante entre nós e o cliente em vez de mantermos a análise inicial de requisitos.

2. Software a funcionar em vez de documentação abrangente - Ao longo do período de desenvolvimento da aplicação fomos tendo em conta, sempre que possível, uma aplicação funcional para mostrar ao cliente, com o objetivo de o cliente nos dizer se era o pretendido ou o que faltava.

3. Colaboração do cliente em vez de negociação de contratos - O cliente esteve sempre presente no desenvolvimento do projeto, assim garantíamos que estávamos a avançar sempre no mesmo sentido.

4. Resposta a modificações em vez de seguir um plano - Foram feitas enumeras alterações nos requisitos do projeto ao longo do seu desenvolvimento, e nós tentámos sempre responder com eficácia e rapidez.

O desenvolvimento ágil não descarta os métodos tradicionais tais como documentações, ferramentas e processos, planeamentos e negociações, mas procura dar a esses itens uma cotação secundária perante indivíduos e interações, o bom funcionamento de Software, colaboração do cliente e respostas eficazes às mudanças. Uma interação constante da parte do cliente é uma mais valia para qualquer projeto, por esses motivos deve ser um método a utilizar.

4.2 Descrição das tarefas

As principais tarefas de desenvolvimento da aplicação são:

- Tarefa 1 — Análise dos requisitos.
- Tarefa 2 — Separação do projeto em duas partes.
- Tarefa 3 — Obtenção de documentação sobre os animais.
  - Estudo sobre os animais, o necessário para o seu registo e brincagem.
  - Estudo sobre o estado produtivo dos animais e diversas manadas.
- Tarefa 4 — Obtenção de documentação sobre como fazer as associações dos animais ao estado produtivo e às manadas.
  - Estudo sobre como efetuar uma forma eficaz de registo e consulta dos animais na exploração, bem como associação de estado produtivo, manadas e brincos.
- Tarefa 5 — Implementação da solução proposta.
- Tarefa 6 — Juntar partes da aplicação.
- Tarefa 7 — Testes da aplicação.
  1. Inserir informação na Base de dados.
  2. Testar pesquisas.
  3. Testar editar.
  4. Testar eliminar.
- Tarefa 8 — Elaboração do relatório.

O agendamento previsto das tarefas é apresentado na figura 4.1.

Identificação	Nome da tarefa	Início	Término	Duração Total	Média de Horas de Trabalho por Semana
1	Tarefa 1	25-04-2012	04-05-2012	2sem	5h
2	Tarefa 2	11-05-2012	11-05-2012	,5sem	6h
3	Tarefa 3	16-05-2012	23-05-2012	1,5sem	5h
4	Tarefa 4	23-05-2012	30-05-2012	1,5sem	5h
5	Tarefa 5	01-06-2012	27-07-2012	8,5sem	30h
6	Tarefa5 + Tarefa 6	08-08-2012	22-08-2012	2,5sem	30h
7	Tarefa 7	22-08-2012	31-08-2012	2sem	30h
8	Tarefa 8	15-08-2012	31-08-2012	3sem	3h

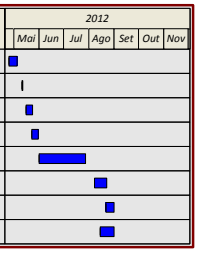

Figura 4.1: Mapa de Gantt Previsto.

O Mapa de Gantt, da figura 4.2 mostra como decorreu o desenvolvimento do projeto.

Identificação	Nome da tarefa	Início	Término	Duração Total	Média de Horas de Trabalho por Semana
1	Tarefa 1	25-04-2012	11-05-2012	3sem	5h
2	Tarefa 2	16-05-2012	16-05-2012	,5sem	3h
3	Tarefa 3	23-05-2012	01-06-2012	2sem	5h
4	Tarefa 4	06-06-2012	15-06-2012	2sem	5h
5	Tarefa 5	03-08-2012	19-09-2012	7sem	25h
6	Tarefa5 + Tarefa 6	05-09-2012	28-09-2012	4sem	20h
7	Tarefa 7	26-09-2012	28-09-2012	1sem	30h
8	Tarefa 8	22-08-2012	10-10-2012	7,5sem	3h

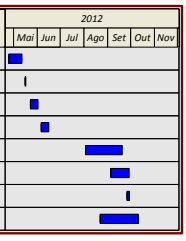

Figura 4.2: Mapa de Gantt Real.

Capítulo 5

Análise dos Requisitos e Conceção da Aplicação

5.1 Diagrama de Contexto

O diagrama de contexto, como o da figura (5.1) apresenta o fluxo de informação entre o sistema e os elementos externos e o modo como eles interagem. Serve para representar o objeto de estudo bem como a sua relação entre o ambiente. Descreve a ideia geral do sistema através de um recurso visual facilitando assim a sua compreensão.

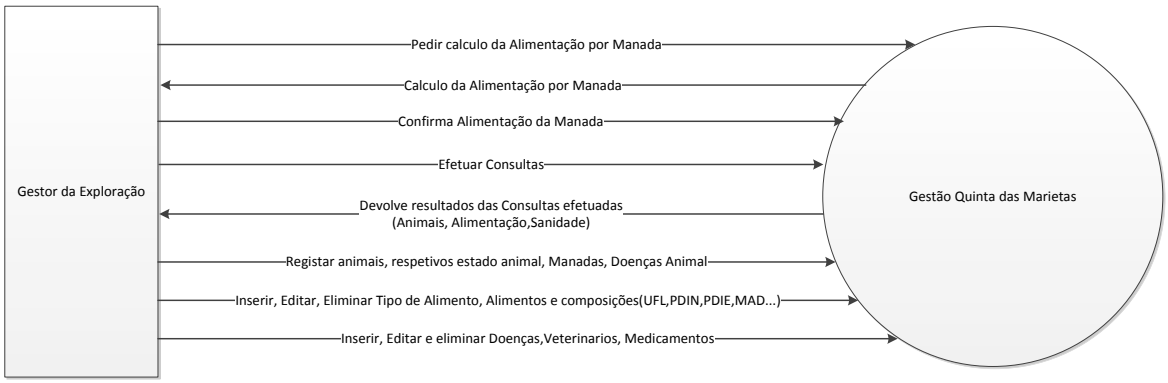

Figura 5.1: Diagrama de Contexto.

5.2 Atores e Respetivos Casos de Uso

Um ator é algo externo ao sistema que interage com ele mas que não tem controlo sobre o mesmo, são os atores que iniciam os casos de uso. Tipicamente o ator é um ser humano, podendo ainda ser outro processo, dispositivo hardware, ou outros.

Na tabela 5.1 encontram-se os atores, casos de uso e respetivos obejtivos.

Tabela 5.1: Ator e respetivos casos de uso

Ator	**Caso de Uso**	**Objetivos**
Gestor da Exploração	Gerir Tipo de Alimento	O objetivo é o Utilizador poder Criar, Editar e Eliminar informação do Tipo de Alimento
	Pesquisar Tipo de Alimento	O objetivo é o Utilizador poder pesquisar informação do Tipo de Alimento
	Criar Novo Alimento	O objetivo é o Utilizador poder Criar um novo alimento.
	Editar Alimento	O objetivo é o Utilizador poder Editar um Alimento.
	Eliminar Alimento	O objetivo é o Utilizador poder Eliminar um Alimento
	Pesquisar Alimento	O objetivo é o Utilizador poder pesquisar informação dos Alimentos
	Gerar Cálculo de Nova Alimentação da Respetiva Manada	O objetivo é o Utilizador pedir a aplicação para gerar o cálculo de uma nova alimentação para manada.
	Pesquisar Alimentações da Manada	O objetivo é o Utilizador poder pesquisar alimentações a que a respetiva manada esteve sujeita.
	Criar Manada	O objetivo é o Utilizador poder Criar uma Nova Manada.
	Editar Manada	O objetivo é o Utilizador poder Editar uma Manada
	Eliminar Manada	O objetivo é o Utilizador poder Eliminar uma Manada
	Associar manada	O objetivo é o Utilizador poder associar um animal a uma Manda existente.
	Gerir Animal*	O objetivo é o Utilizador poder Criar, Editar e Eliminar informação dos Animais
	Associar Categoria de Estado Produtivo ao Animal	O objetivo é o Utilizador poder Atribuir uma nova Categoria de Estado Produtivo ao Animal
	Gerir Categorias de Estado Produtivo*	O objetivo é o Utilizador poder Criar, Editar e Eliminar as Categorias de Estado Produtivo
	Gerar Nova Pesagem Animal	O objetivo é o Utilizador poder Atribuir um novo peso ao Animal
	Criar Brincos	O objetivo é o Utilizador poder introduzir uma sequência de Brincos no sistema.
	Atribuir Brinco ao Animal	O objetivo é o Utilizador poder atribuir um ao Animal
	Pesquisar Animal	O objetivo é o Utilizador poder pesquisar um Animal por nº de brinco, sexo e/ou manada e/ou categoria corporal e/ou se existe na exploração
	Gerir Medicamentos Usados	O objetivo é o Utilizador poder Criar, Editar e Eliminar informação dos Medicamentos
	Pesquisar Medicamentos Usados	O objetivo é o Utilizador poder pesquisar informação dos Medicamentos
	Gerir Veterinário	O objetivo é o Utilizador poder Criar, Editar e Eliminar informação dos Veterinários
	Pesquisar Veterinário	O objetivo é o Utilizador poder pesquisar informação dos Veterinários
	Gerir Doenças	O objetivo é o Utilizador poder Criar, Editar e Eliminar informação das Doenças
	Pesquisar Doença	O objetivo é o Utilizador poder pesquisar informação das Doenças
	Novo caso de Sanidade Animal	O objetivo é o Utilizador poder Criar um novo caso de sanidade animal para um Animal
	Pesquisar caso de Sanidade Animal	O objetivo é o utilizador poder efetuar uma pesquisa de casos de sanidade através do numero de identificação do animal ou quais os animais doentes ou não.
	Nova Intervenção do Veterinário	O objetivo é o utilizador inserir e descrever uma intervenção do veterinário num determinado caso de sanidade de um animal
	Nova Linha de Tratamento	O objetivo é o Utilizador inserir medicamentos prescritos pelo veterinário numa determinada Intervenção

* Todos os caos de uso começados por Gerir têm como base o Criar, Editar, Eliminar e Associar a Manada.

5.3 Diagrama de Casos de Uso

O diagrama de casos de uso, representado pela figura 5.2, é de extrema importância para a análise do sistema, permite definir o ator bem como a interação que este tem com o sistema. Em relação ao diagrama de casos de uso do nosso projeto podemos ver a fronteira que delimita o sistema "Gestão Quinta das Marietas", onde estão inseridos os casos de uso e o ator ("Gestor da Exploração") que está associado aos casos de uso.

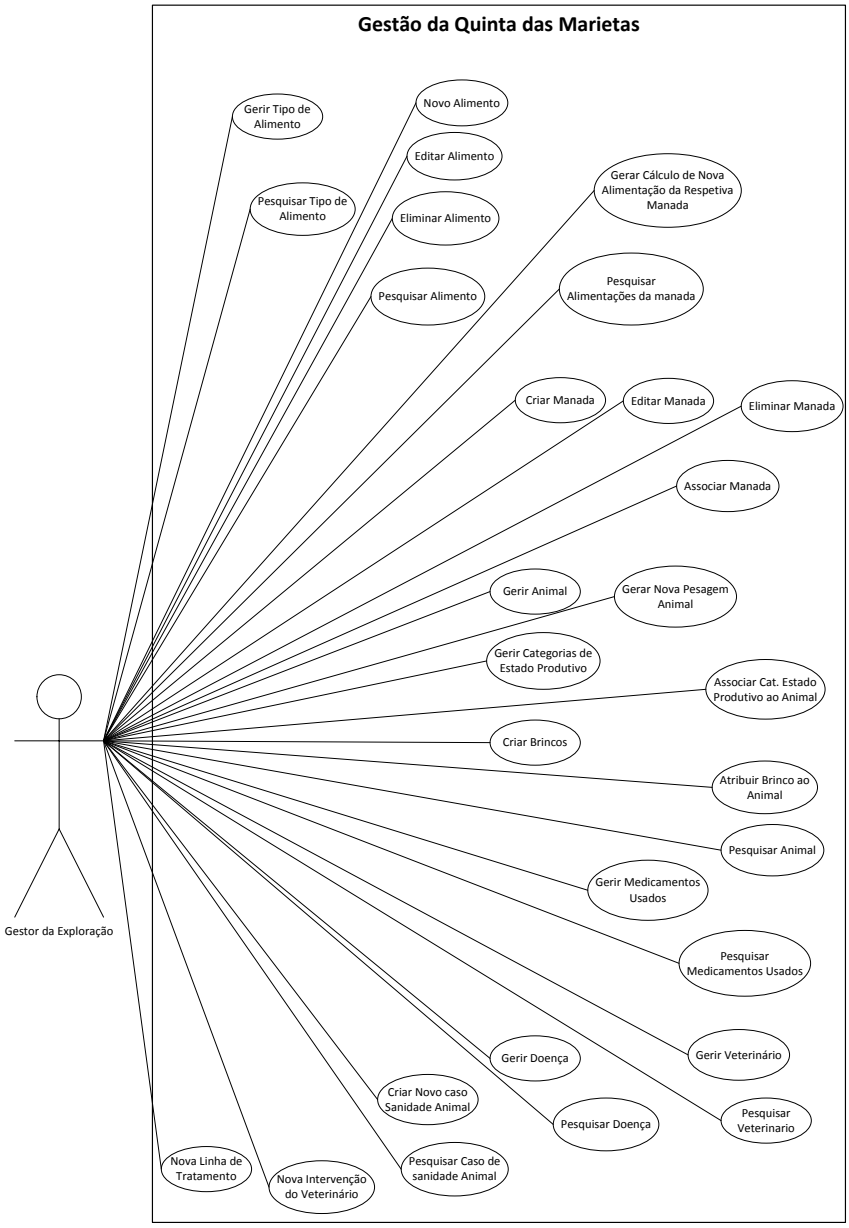

Figura 5.2: Diagrama de Casos de Uso.

5.4 Descrição de Casos de Uso

As tabelas de descrição dos casos de uso são compostas por nome, descrição, atores envolvidos, pré condições, fluxo principal, fluxos alternativos, suplementos e pós condições. O nome indica a designação do caso de uso que estamos a definir. A descrição mostra para que serve o caso de uso que estamos a desenvolver. Os atores envolvidos são todos os que interagem com o caso de uso. Pré condições são as condições necessárias para se poder executar o caso de uso. O fluxo principal é o que o sistema e do ator fazem neste caso de uso. Fluxos alternativos são os fluxos que fazem parte do caso de estudo mas só são executados em certas circunstancias. Suplementos são as validações que são necessárias serem feitas para se poder executar o caso de uso pretendido. Pós condições são as verificações finais que são feitas após a execução do que representa o caso de estudo.

Nas tabelas em baixo está descrito em detalhe os casos de uso de maior relevância para a parte dos animais.

Na tabela 5.2 está descrito em detalhe o caso de uso de criar uma nova manada.

Tabela 5.2: Criar Nova Manada

Nome:	Criar Nova Manada
Descrição:	O objetivo é o Utilizador poder criar uma nova Manada.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Não tem
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Gerir Manada". 2. O sistema disponibiliza o formulário de Gestão de Manadas. 3. O utilizador seleciona a opção "Criar Manada". 4. O sistema disponibiliza o formulário Criar Manada. 5. O utilizador introduz um nome que identifica a nova Manada. 6. O sistema pede para confirmar. 7. O utilizador confirma. 8. O sistema Guarda.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 5a. O sistema cancela se o utilizador não introduzir todos os dados. 7a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Verifica se já existe uma Manada com o mesmo nome.
Pós-Condições:	Não tem

Na tabela 5.3 está descrito em detalhe o caso de uso de editar uma nova manada.

Tabela 5.3: Editar Manada

Nome:	Editar Manada
Descrição:	O objetivo é o Utilizador poder Editar uma Manada.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Não tem
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Gerir Manada". 2. O sistema disponibiliza o formulário de Gestão de Manadas. 3. O utilizador seleciona a Manada que deseja Editar. 4. O sistema disponibiliza o formulário Editar Manada. 5. O utilizador Edita a Identificação da manada. 6. O sistema pede para confirmar. 7. O utilizador confirma. 8. O sistema Atualiza.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 5a. O sistema cancela se o utilizador não introduzir todos os dados. 7a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Verifica se já existe uma Manada com o mesmo nome.
Pós-Condições:	Não tem

Na tabela 5.4 está descrito em detalhe o caso de uso para eliminar uma manada.

Tabela 5.4: Eliminar Manada

Nome:	Eliminar Manada
Descrição:	O objetivo é o Utilizador poder Eliminar uma Manada.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Não tem
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Gerir Manada". 2. O sistema disponibiliza o formulário de Gestão de Manadas. 3. O utilizador seleciona a Manada que deseja Eliminar. 4. O sistema pede para confirmar. 5. O utilizador confirma. 6. O sistema Elimina.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Voltar". 3a. O sistema não permite eliminar se a manada contêm pelo menos um animal ou está sujeita a uma alimentação (o registo tiver dependências). 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se o sistema permite eliminar uma manada que contenha animais ou se está sujeita a uma alimentação.
Pós-Condições:	Não tem

Na tabela 5.5 está descrito em detalhe o caso de uso para associar uma manada a um ou vários animais.

Tabela 5.5: Associar Manada

Nome:	Associar Manada
Descrição:	O objetivo é o Utilizador poder associar um animal a uma Manada.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Não tem
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Associar Manada". 2. O sistema disponibiliza o formulário para associar a Manada. 3. O utilizador vai associar o animal à manada pretendida. 4. O sistema pede para confirmar. 5. O utilizador confirma. 6. O sistema Guarda.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 3a. O sistema cancela se o animal já está na manada a que se quer associar. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Verifica se já existe o animal já se encontra numa manada. Verifica se a Manda existe.
Pós-Condições:	Não tem

Na tabela 5.6 está descrito em detalhe o caso de uso para atribuir um brinco a um animal.

Tabela 5.6: Atribuir Brinco ao Animal

Nome:	Atribuir Brinco ao Animal
Descrição:	O objetivo é o Utilizador poder atribuir a animal um Brinco.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	O animal já foi criado.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Atribuir Brincos". 2. O sistema vai buscar à base de dados o brinco imediatamente a seguir ao último inserido. 3. O sistema atribui esse brinco ao animal. 4. O sistema Guarda.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 3a. O sistema cancela se o utilizador introduzir um brinco num animal que já tem brinco. 4a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Verifica se já o animal já possui um brinco.
Pós-Condições:	Não tem

Na tabela 5.7 está descrito em detalhe o caso de uso para associar uma categoria de estado produtivo a um animal.

Tabela 5.7: Associar Categoria de Estado Produtivo Animal

Nome:	Associar Categoria de Estado Produtivo Animal
Descrição:	O objetivo é o Utilizador poder Associar Cat. Estado Produtivo a um Animal.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Animal selecionado.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Atribuir Categoria de Estado Animal". 2. O Sistema disponibiliza formulário para Atribuir a Categoria de Estado Produtivo. 3. O utilizador seleciona o estado produtivo, a categoria do estado produtivo e a data atual. 4. O sistema pede para confirmar. 5. O utilizador confirma. 6. O sistema Guarda.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 3a. Data atual ou data pretendida. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Não permitir que se coloque um animal numa categoria que não existe Verifica se a Categoria de Estado Produtivo existe.
Pós-Condições:	Não tem

Na tabela 5.8 está descrito em detalhe o caso de uso para pesquisar um animal.

Tabela 5.8: Pesquisar Animal

Nome:	Pesquisar Animal.
Descrição:	O objetivo é o Utilizador poder pesquisar um Animal por nº do brinco, sexo e/ou manada e/ou categoria corporal e/ou se existe na exploração.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Ter Animal selecionado.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Gerir Animal". 2. O Sistema disponibiliza o "formulário"de pesquisa. 3. O utilizador vai pesquisar um Animal por n.º do brinco, sexo e/ou manada e/ou categoria corporal e/ou se existe na exploração. 4. O sistema pede para mostra.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar".
Suplementos:	Não tem.
Pós-Condições:	Não tem

Na tabela 5.9 está descrito em detalhe o caso de uso para pesar um animal.

Tabela 5.9: Pesagem do Animal

Nome:	Pesagem do Animal.
Descrição:	O objetivo é o Utilizador poder pesar um Animal existente na exploração.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Ter Animal selecionado.
Fluxo Principal:	1. 1. O caso de uso começa quando o Utilizador seleciona a opção "Pesar Animal". 2. O Sistema disponibiliza o "formulário"de pesagem. 3. O utilizador vai introduzir o peso do Animal. 4. O sistema pede para confirmar. 5. O utilizador confirma. 6. O sistema Guarda.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar".
Suplementos:	Não tem.
Pós-Condições:	Não tem

5.5 Diagramas de Sequência

O diagrama de sequência mostra a troca de mensagens entre os vários objetos numa sequência temporal. É utilizado para descrever o fluxo de execução dos casos de uso e facilita a visualização do sistema pois mostra que classes são chamadas e em que situações. No diagrama de sequência está representado o ator ("Gestor de Exploração") bem como os objetos e métodos ordenados pelo tempo.

Este diagrama de sequência 5.3 representa as ações do ator(gestor) ao criar uma nova manada.

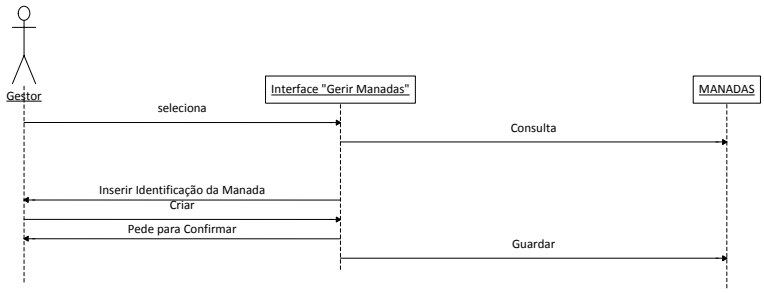

Figura 5.3: Diagrama Sequência: Criar Manada.

Este diagrama de sequência 5.4 representa as ações do ator(gestor) para editar uma manada.

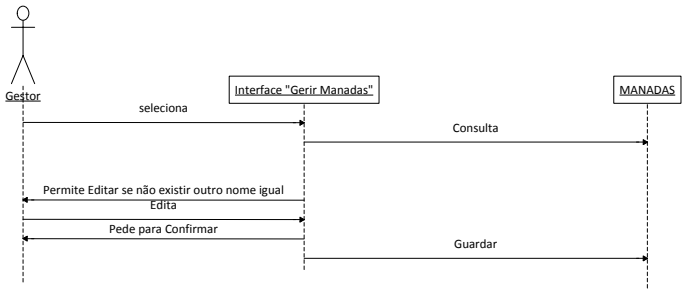

Figura 5.4: Diagrama Sequência: Editar Manada.

Este diagrama de sequência 5.5 representa as ações do ator(gestor) e verificações do sistema para eliminar uma manada.

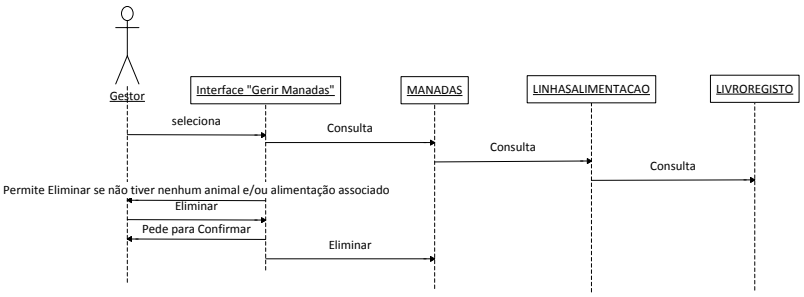

Figura 5.5: Diagrama Sequência: Eliminar Manada.

Este diagrama de sequência 5.6 representa as ações do ator(gestor) e verificações do sistema para associar uma manada a um ou vários animais.

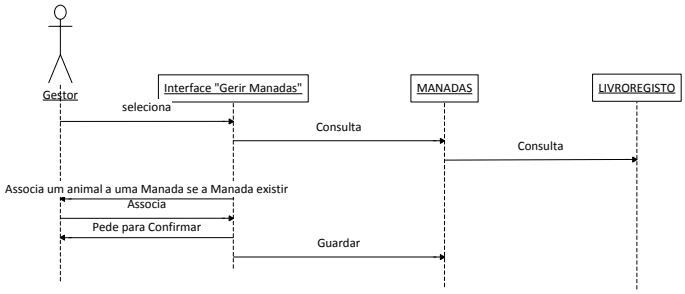

Figura 5.6: Diagrama Sequência: Associar Manada.

Este diagrama de sequência 5.7 representa as ações do ator(gestor) e verificações do sistema para criar um novo brinco animal.

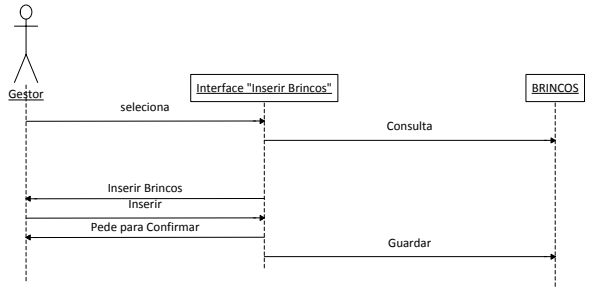

Figura 5.7: Diagrama Sequência: Criar Brinco.

Este diagrama de sequência 5.8 representa as ações do ator(gestor) e verificações do sistema para atribuir um brinco que já tenho sido criado anteriormente.

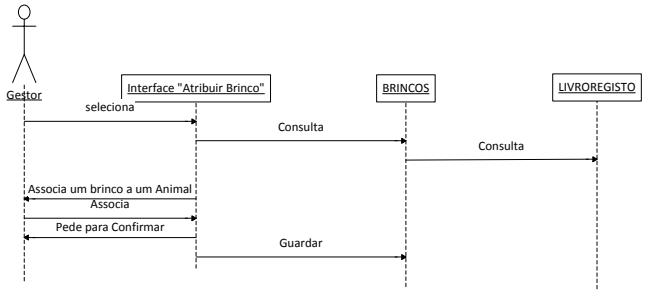

Figura 5.8: Diagrama Sequência: Atribuir Brinco.

Este diagrama de sequência 5.9 representa as ações do ator(gestor) e verificações do sistema para associar um animal a uma categoria de estado produtivo.

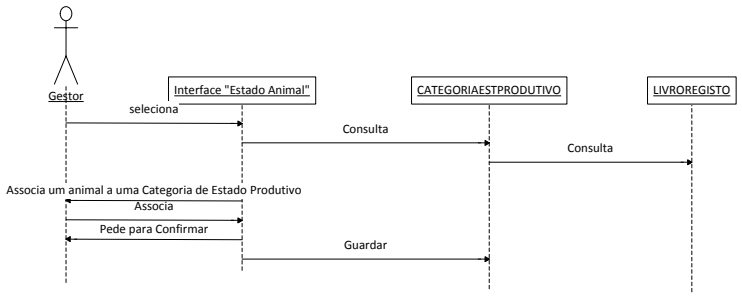

Figura 5.9: Diagrama Sequência: Associar Categoria de Estado Produtivo.

Este diagrama de sequência 5.10 representa as ações do ator(gestor) e verificações do sistema para pesquisar um animal.

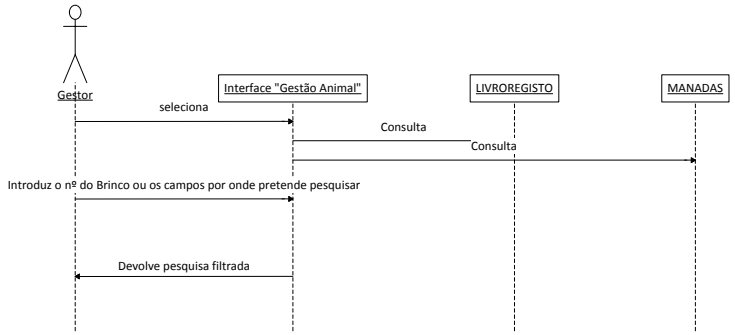

Figura 5.10: Diagrama Sequência: Pesquisar Animal.

5.6 Diagrama de Classes

O diagrama de classes é de extrema importância uma vez que define a estrutura a desenvolver e mostra a relação entre as várias classes implementadas. Este diagrama é consequência de uma breve análise de requisitos previamente efetuada.
Cada classe é composta pelo nome, seus atributos e as operações que representa o papel dos atores no sistema.

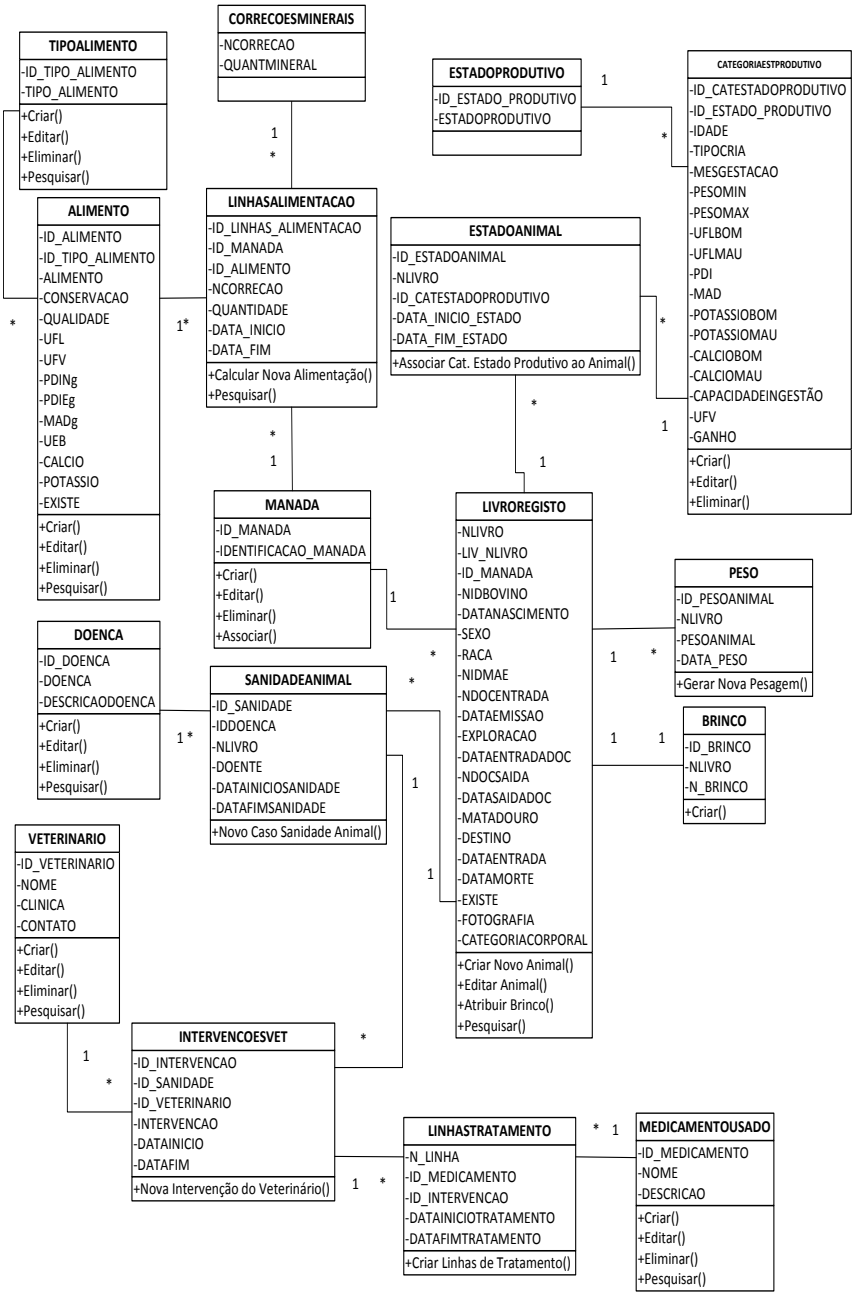

Figura 5.11: Diagrama de Classes.

5.7 Semântica de Classes

A semântica de classes é composta por todos os atributos, tipos de dados, descrição, valores válidos, formato e restrições de cada classe do diagrama. Nos atributos encontra-se o nome da "variável"como foi definida no diagrama de classes. Na descrição é exposto a representação dos dados. Os valores válidos que se podem inserir no campo são todos os valores que sejam numéricos, texto ou data. O formato indica o limite máximo de dados que podem ser inseridos e/ou o modo como estes devem ser inseridos, por exemplo a data é inserida yyyy-MM-dd (ano-mês-dia). Por fim as restrições são as diretrizes que devem ser cumpridas relativamente à inserção de dados bem como a sua obrigatoriedade, e ainda, se podem ser alterados. Em baixo encontram-se as semânticas de classes referentes aos animais bem como os algoritmos mais importantes correspondentes às classes a que estão associados.

Tabela 5.10: Semântica Classes PESO

Semântica: PESO - Têm como objetivo registar o peso de um determinado animal					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_PESOANIMAL (PK)	Inteiro	Número sequencial que identifica univocamente cada pesagem	Maior que 0	Até 4 dígitos	Obrigatório e não alterável
NLIVRO	Inteiro	Número que identifica cada Registo Animal	Números de 0 a 9	Até 50 dígitos	Obrigatório e não alterável
PESOANIMAL	Inteiro	Peso do Animal	Números de 0 a 9	Até aos dígitos máximos primitivos	Obrigatório e alterável
DATA_PESO	Data	Data do registo Peso do Animal	Dígitos separados por (-)	yyyy-MM-dd	Obrigatório e alterável

Algoritmo 1 Semântica da Classes +Pesar()

1: **procedure** +PESAR() ▷ Operação que permite Pesar uma Animal.
2: O Sistema gera o ID_PESOANIMAL (incrementa uma unidade ao último ID_PESOANIMAL).
3: Introduzir PESOANIMAL.
4: O sistema introduz a DATA_PESO.
5: Nova Pesagem feita.
6: **end procedure**

Tabela 5.11: Semântica Classes LIVROREGISTO

Semântica: LIVROREGISTO - Têm como objetivo registar um animal					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
NLIVRO (PK)	Inteiro	Número sequencial que identifica univocamente o número do registo de Animal no Livro de Registo	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
LIV_NLIVRO (FK1)	Inteiro	Número que identifica univocamente cada Animal que é Mãe de um animal da Manada	Maior que 0	Até 10 dígitos	Obrigatório e não alterável
ID_MANADA (FK2)	Inteiro	Número que identifica univocamente cada Manada	Maior que 0	Até 10 dígitos	Obrigatório e não alterável
NIDBOVINO	Inteiro	Número que identifica univocamente cada Animal	Números de 0 a 9 e/ou Caracteres de A a Z	Até 11 dígitos XX-XX-XX-XXXXX	Obrigatório e não alterável
DATANASCIMENTO	Data	Data de Nascimento do Animal	Dígitos separados por (-)	yyyy-MM-dd	Obrigatório e alterável
SEXO	Texto	Sexo do Animal	Caracteres de A a Z	Até 50 dígitos	Obrigatório e alterável
RACA	Texto	Raça do Animal	Caracteres de A a Z	Até 50 dígitos	Obrigatório e alterável
NIDMAE	Inteiro	Número de Identificação da Mãe do Animal	Números de 0 a 9 e/ou Caracteres de A a Z	Até 11 dígitos XX-XX-XX-XXXXX	Obrigatório e não alterável
NDOCENTRADA	Inteiro	Número do documento de entrada do animal na exploração	Números de 0 a 9 e/ou Caracteres de A a Z	Até aos dígitos máximos primitivos	Obrigatório e alterável
DATAEMISSAO	Data	Data de Emissão do Documento	Dígitos separados por (-)	yyyy-MM-dd	Obrigatório e alterável
EXPLORACAO	Texto	Nome da Exploração onde se encontra o animal	Caracteres de A a Z	Até 100 dígitos	Não Obrigatório e alterável
DATAENTRADADOC	Data	Data de Entrada do Documento	Dígitos separados por (-)	yyyy-MM-dd	Não Obrigatório e alterável
NDOCSAIDA	Inteiro	Número do documento de Saída da Exploração	Números de 0 a 9 e/ou Caracteres de A a Z	Até aos dígitos máximos primitivos	Não Obrigatório e alterável
DATASAIDADOC	Data	Data de saída do documento da exploração	Dígitos separados por (-)	yyyy-MM-dd	Não Obrigatório e alterável
MATADOURO	Texto	Nome do Matadouro para onde foi o Animal	Caracteres de A a Z	Até 100 dígitos	Não Obrigatório e alterável
DESTINO	Texto	Destino (Nome da exploração) para onde foi o Animal	Caracteres de A a Z	Até 100 dígitos	Não Obrigatório e alterável
DATAENTRADA	Data	Data de Entrada do Animal na exploração	Dígitos separados por (-)	yyyy-MM-dd	Não Obrigatório e alterável
DATAMORTE	Data	Data de Morte do Animal	Dígitos separados por (-)	yyyy-MM-dd	Não Obrigatório e alterável
EXISTE	Booleano	Ver se o Animal existe na exploração ou se morreu ou foi vendido	Selecionado caso exista	Selecionado ou não Selecionado	Obrigatório e alterável
FOTOGRAFIA	Texto	Fotografia do Animal para registo	Icone	Até 1000 dígitos	Não Obrigatório e alterável
CATEGORIACORPORAL	Inteiro	Categoria Corporal do Animal	Números de 1 a 5	Até 1 dígito	Obrigatório e alterável

O Criar é semelhante ao algoritmo 4;

O Editar é semelhante ao algoritmo 5;

Algoritmo 2 Semântica da Classe +Pesquisar().

```
procedure +PESQUISAR()                                   ▷ Operação que permite Pesquisar.
    if Introduz o número do brinco do animal then
        if NIDBOVINO = número introduzido then
            Cancelar operação.
        else
            Devolve "Animal não encontrado"
        end if
    else
        if Seleciona sexo e/ou manada e/ou categoria corporal e/ou se existe na
    exploração then
            Devolve a informação filtrada
        else
            Devolve "Animal não encontrado"
        end if
    end if
end procedure
```

Algoritmo 3 Semântica da Classe +Atribuir Brinco().

```
procedure +ATRIBUIR BRINCO()          ▷ Operação permite atribuir um Brinco a um Animal.
    if Animal têm brinco then
        Sistema cancela.
    else
        if BRINCO.NLIVRO = null then
            LIVROREGISTO.NIDBOVINO=BRINCO.N_BRINCO
        else
            Sistema cancela.
        end if
    end if
end procedure
```

Tabela 5.12: Semântica Classes MANADA

Semântica: MANADA - Têm como objetivo registar uma manada					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_MANADA (PK)	Inteiro	Número que identifica univocamente cada Manada	Maior que 0	Até 10 dígitos	Obrigatório e não alterável
IDENTIFICACAO_MANADA	Texto	Nome que identifica cada Manada	Números de 0 a 9 e/ou Caracteres de A a Z	Até 50 dígitos	Obrigatório e alterável

Algoritmo 4 Semântica da Classe +Criar().

```
1: procedure +CRIAR()                ▷ Operação que permite criar o registo de uma nova Manda
2:     O Sistema gera o ID_MANADA (incrementa uma unidade ao último
       ID_MANADA).
3:     Introduzir IDENTIFICACAO_MANADA
4:     Criar nova Manada.
5: end procedure
```

Algoritmo 5 Semântica da Classe +Editar().

```
1: procedure +EDITAR()               ▷ Operação que permite alterar uma Manada.
2:     Selecionar a Manada através do ID_MANADA.
3:     Introduzir nova IDENTIFICACAO_MANADA.
4:     Alterar dados da Manada.
5: end procedure
```

Algoritmo 6 Semântica da Classe +Eliminar().

```
1: procedure +ELIMINAR()             ▷ Operação que permite eliminar uma Manada.
2:     Selecionar a Manada através do ID_MANADA.
3:     if ID_MANADA está a ser utilizado then
4:         Cancelar operação.
5:     else
6:         Executar operação apagar Manada
7:     end if
8: end procedure
```

Algoritmo 7 Semântica da Classe +Associar().

1: **procedure** +ASSOCIAR() ▷ Operação que permite Associar um Animal a uma Manada.
2: Selecionar um Animal através do NIDBOVINO.
3: Selecionar a Manada através do ID_MANADA
4: **if** NIDBOVINO está na Manada Pretendida **then**
5: Cancelar operação.
6: **else**
7: Executar operação Associar Manada (NIDBOVINO associa IDENTIFICACAO_MANADA).
8: **end if**
9: **end procedure**

Tabela 5.13: Semântica Classes ESTADOANIMAL

Semântica: ESTADOANIMAL - Têm como objetivo registar um estado animal					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_ESTADOANIMAL (PK)	Inteiro	Número sequencial que identifica cada estado animal	Maior que 0	Até 10 dígitos máximos	Obrigatório e não alterável
NLIVRO	Inteiro	Número que identifica cada registo Animal	Números de 0 a 9	Até a10 dígitos máximos	Obrigatório e não alterável
ID_CATESTADOPRODUTIVO	Inteiro	Número de Animais que se encontra em cada manada	Números de 0 a 9	Até 10 dígitos máximos	Obrigatório e não alterável
DATA_INICIO_ESTADO	Data	Data de Inicio do Estado Animal	Dígitos separados por (-)	yyyy-MM-dd	Obrigatório e alterável
DATA_FIM_ESTADO	Data	Data de Fim do Estado Animal	Dígitos separados por (-)	yyyy-MM-dd	Obrigatório e alterável

Algoritmo 8 Semântica da Classe +Associar Categoria de Estado().

1: **procedure** +ASSOCIAR CATEGORIA DE ESTADO() ▷ Operação que permite Associar um Animal a um Estado.
2: Selecionar um Animal através do NLIVRO.
3: Selecionar a Categoria de Estado Produtivo através do ID_CATESTADOPRODUTIVO.
4: **if** NLIVRO está na Categoria de Estado Produtivo Pretendida **then**
5: Cancelar operação.
6: **else**
7: Executar operação Associar ESTADOANIMAL (NLIVRO associa ID_CATESTADOPRODUTIVO).
8: **end if**
9: **end procedure**

Tabela 5.14: Semântica Classes Categoria Estado Produtivo

Semântica: CATEGORIAESTPRODUTIVO - Têm como objetivo registar a categoria de um estado produtivo					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_CATESTADOPRODUTIVO (PK)	Inteiro	Número de Animais que se encontra em cada manada	Números de 0 a 9	Até aos dígitos máximos primitivos	Obrigatório e não alterável
ID_ESTADO_PRODUTIVO (FK)	Inteiro	Número sequencial que identifica univocamente cada estado produtivo	Maior que 0	Até aos dígitos máximos primitivos	Obrigatório e não alterável
IDADE	Inteiro	Nome que identifica a idade do animal	Números de 0 a 9	Até 50 dígitos	Obrigatório e alterável
TIPOCRIA	Texto	Classifica a cria como precoce ou tardia	Caracteres de A a Z	Até 50 dígitos	Obrigatório e alterável
MESGESTACAO	Inteiro	Guarda o mês em que a cria se encontra	Valor numérico separado por ponto	Até 2 dígitos	Obrigatório e alterável
PESOMIN	Texto	Peso mínimo do animal para ver a categoria em que se enquadra	Valor numérico separado por ponto	###.##	Obrigatório e alterável
PESOMAX	Texto	Peso máximo do animal para ver a categoria em que se enquadra	Valor numérico separado por ponto	###.##	Obrigatório e alterável
UFLBOM	Texto	Componente Unidade Forrageira Leiteira UFL (Máximo)	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
UFLMAU	Texto	Componente Unidade Forrageira Leiteira UFL (Mínimo)	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
PDI	Texto	Componente Proteína Digestível no Intestino	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
MAD	Texto	Componente Matéria Azotada Digestível MAD	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
POTASSIOBOM	Texto	Componente Potássio (Máximo)	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
POTASSIOMAU	Texto	Componente Potássio (Mínimo)	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
CALCIOBOM	Texto	Componente Cálcio (Máximo)	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
CALCIOMAU	Texto	Componente Cálcio (Mínimo)	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
CAPACIDADEINGESTAO	Texto	Capacidade Máxima de ingestão do Animal	Valor numérico separado por ponto	###.##	Obrigatório e não alterável
UFV	Texto	Componente Unidade Forrageira Carne (Viande) UFV	Valor numérico separado por ponto	##.##	Obrigatório e não alterável
Ganho	Texto	Ganho diário do animal	Valor numérico separado por ponto	###.##	Obrigatório e alterável

Tabela 5.15: Semântica Classes ESTADOPRODUTIVO

Semântica: ESTADOPRODUTIVO - Têm como objetivo registar um estado produtivo					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_ESTADO_PRODUTIVO (PK)	Inteiro	Número sequencial que identifica cada estado produtivo	Maior que 0	Até 10 dígitos máximos	Obrigatório e não alterável
ESTADOPRODUTIVO	Texto	Nome que identifica o estado produtivo	Caracteres de A a Z	Até 50 dígitos	Obrigatório e alterável

Tabela 5.16: Semântica Classes BRINCO

Semântica: BRINCO Têm como objetivo registar um brinco					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_BRINCO (PK)	Inteiro	Número sequencial que identifica cada brinco	Maior que 0	Até 100 dígitos máximos	Obrigatório e não alterável
NLIVRO	Inteiro	Número que identifica cada Animal	Números de 0 a 9	Até 50 dígitos	Obrigatório e não alterável
N_BRINCO	Texto	Número de Animais que se encontra em cada manada	Números de 0 a 9 e/ou Caracteres de A a Z	Até 12 dígitos máximos	Obrigatório e alterável

- O Criar é semelhante ao algoritmo 4;

5.8 Diagrama de Atividades

Um diagrama de atividades é basicamente um gráfico de fluxo que controla uma atividade para outra e é usado para fazer a modelagem dos aspetos dinâmicos do sistema. Este diagrama mostra de uma forma organizada as operações que constituem a aplicação.

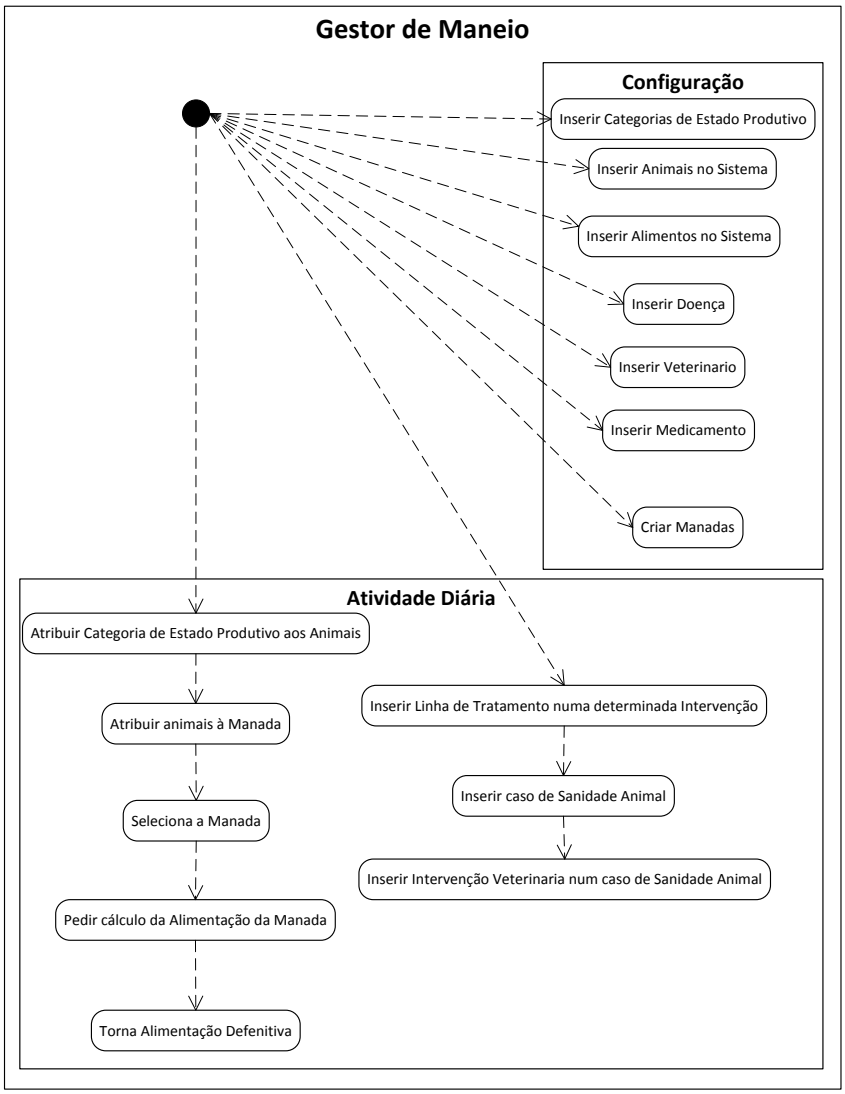

Figura 5.12: Diagrama de Atividades.

5.9 Diagrama de Estados

Um diagrama de estados em UML ilustra os eventos e os estados interessantes de um objeto e o comportamento de um objeto em resposta a um evento, ou seja mostra o ciclo de vida de um objeto, os eventos por onde passa, as suas transições e os e os estados em que ele está entre estes eventos.

Este diagrama mostra a transição do estado da existência do animal na exploração, onde inicialmente do estado da existência do animal é existe, ou seja encontra-se na exploração, e com o decorrer do processo que a data da morte ou o destino for inserido ele passa a não existe.

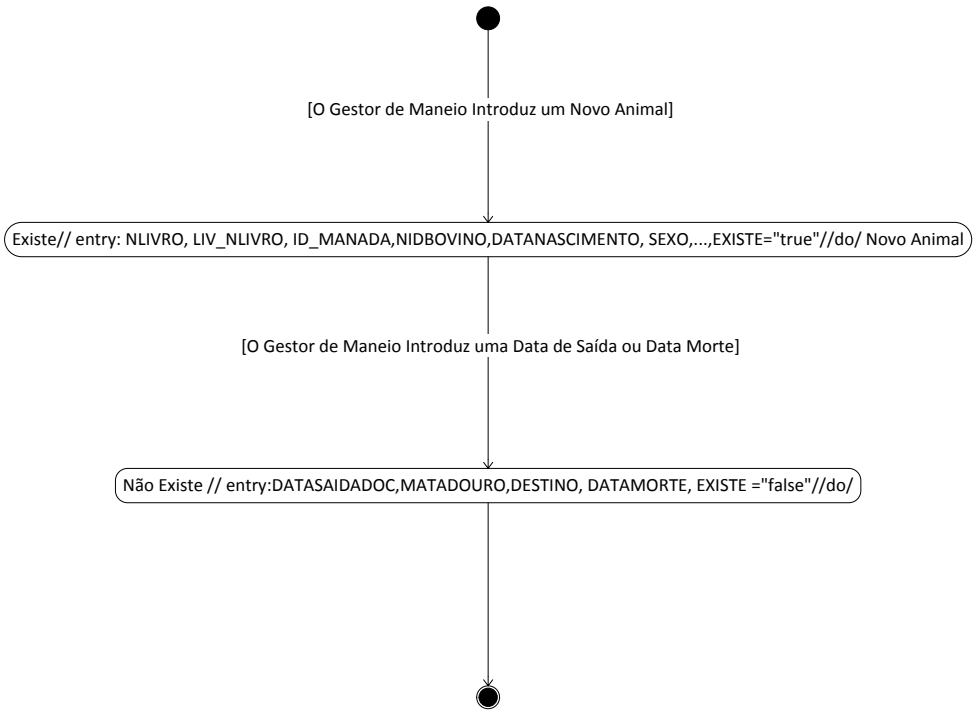

Figura 5.13: Diagrama de Estados.

5.10 Diagrama de Componentes

O diagrama de componentes mostra a organização e as ligações entre a nossa aplicação e a base de dados. Este diagrama mostra que a nossa aplicação é dependente da base de dados e vice-versa.

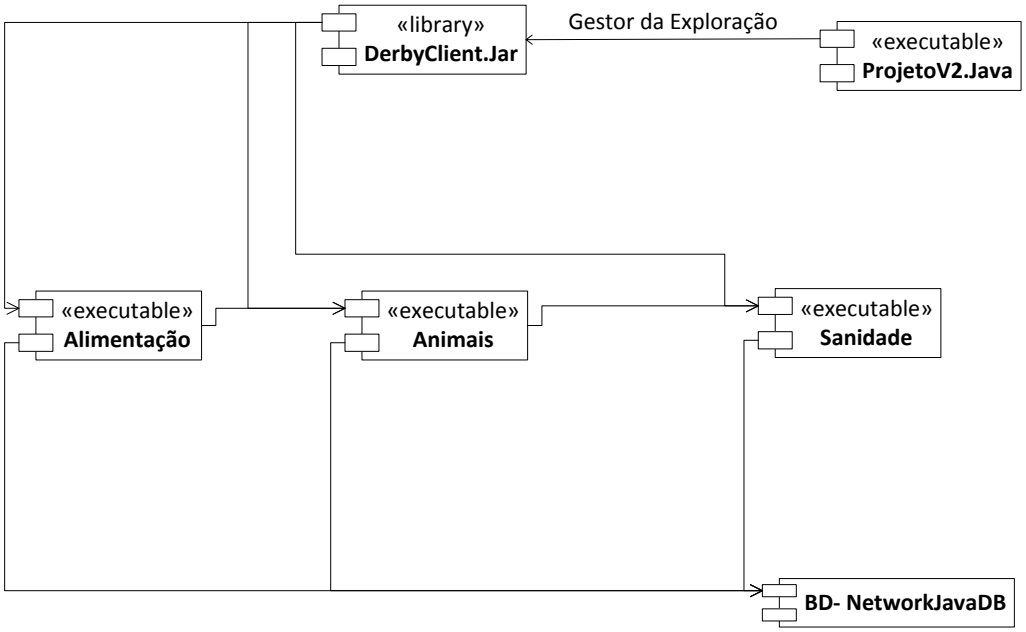

Figura 5.14: Diagrama de Componentes.

Tabela 5.17: Relação das componentes com as classes

Componente:	**Alimentação**	**Animais**	**Sanidade**
Classes utilizadas:	TIPOALIMENTO ALIMENTO MANADA LIVROREGISTO ESTADOANIMAL CATEGORIAESTPRODUTIVO ESTADOPRODUTIVO	LIVROREGISTO PESO BRINCO MANADA ESTADOANIMAL CATEGORIAESTPRODUTIVO ESTADOPRODUTIVO	SANIDADEANIMAL DOENCA INTERVENCOESVET VETERINARIO LINHASTRAMENTO MEDICAMENTOSUSADOS LIVROREGISTO

5.11 Diagrama de Instalação

O diagrama de Instalação mostra as ligações entre o código fonte, bibliotecas, base de dados,etc. E serve para visualizar como tudo se interliga de modo a se tornar a aplicação final.

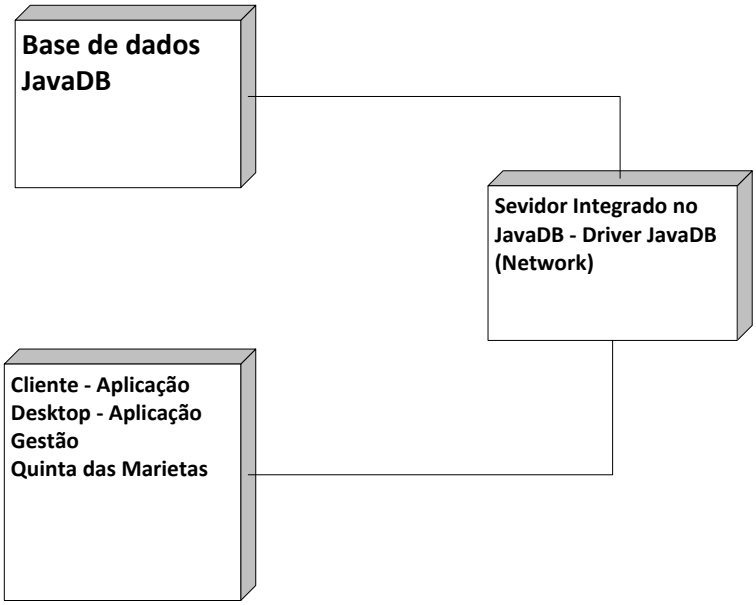

Figura 5.15: Diagrama de Componentes.

Capítulo 6

Implementação da solução

6.1 Introdução

Uma vez que foi feito um estudo aprofundado na análise de requisitos e tínhamos uma solução bem estruturada facilitou a implementação de modo a que se tornou mais fácil do que inicialmente prevíamos. Na implementação tentámos fazer sempre o pretendido o mais fácil e eficazmente possível de modo a fazer um programa com qualidade, rapidez e eficácia, sem descurar a simplicidade e o pretendido.

6.2 Código Utilizado na Aplicação

De seguida vou demonstrar algumas das interfaces da aplicação importantes da parte dos Animais acompanhados de excertos de código que acho relevantes. Vou demonstrar pelo menos um inserir, editar, pesquisar e associar visto que o código é relativamente parecido nos vários aspetos de gestão da nossa aplicação.

A figura 6.1 mostra o menu "Gerir Animal"bem como a pesquisa que pode ser efectuada para encontrar um animal pretendido. Neste menu além de se poder pesquisar um animal específico ou um conjunto de animais, temos acesso a toda a parte de gestão dos animais podendo criar ou editar um animal, alterar a manda onde este se encontra, atribuir um estado animal, pesar um animal ou mesmo ter acesso à sanidade do mesmo.

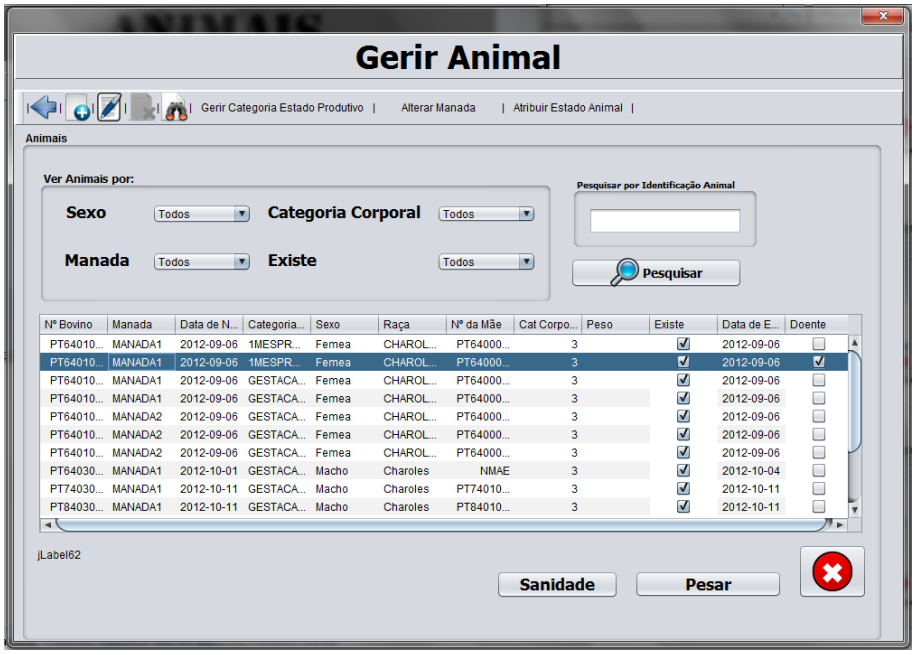

Figura 6.1: Gerir/Pesquisar Animal.

O algoritmo 10 mostra o código da classe que preenche a tabela do menu "Gerir Animal".
O algoritmo12 mostra o código necessário para efectuar a pesquisa.

Algoritmo 9 Classe Preencher Gerir Animal.

```
1: procedure PREENCHER GERIR ANIMAL        ▷ Operação que Preencher a Tabela de Gerir Animal.
2:     PreencheGerirAnimais(JTable jTableGerirAnimal, JLabel jLabelErroGerirAnimais, JComboBox jComboBoxGerirAnimalCatCorporal, JComboBox jComboBoxGerirAnimalExiste, JComboBox jComboBoxGerirAnimalManada, JComboBox jComboBoxGerirAnimalSexo)
3:         preencheTabela(jTableGerirAnimal, jLabelErroGerirAnimais);
4:     private void preencheTabela(JTable jTableGerirAnimal, JLabel jLabelErroGerirAnimais)
5:         DefaultTableModel ModeloTabelaAnimais = (DefaultTableModel) jTableGerirAnimal.getModel();
6:         int tamanho = ModeloTabelaAnimais.getRowCount();
7:         for (int i = 0; i < tamanho; i++)
8:             ModeloTabelaAnimais.removeRow(0);
9:         try
10:            Class.forName("org.apache.derby.jdbc.ClientDriver");
11:            Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
12:            String sql = "select * from LIVROREGISTO";
13:            PreparedStatement st = (PreparedStatement) con.prepareStatement(sql);
14:            ResultSet rs = st.executeQuery();
15:            String vIDENTIFICACAO_MANADA2 = ;
16:            String vCATEGORIAESTPRODUTIVO2 = ;
17:            boolean vDOENTE = true;
18:            while (rs.next())
19:                String vPESO2 = ;
20:                String vNLIVRO = rs.getString("NLIVRO");
21:                String vNID_BOVINO = rs.getString("NIDBOVINO");
22:                String vIDENTIFICACAO_MANADA = rs.getString("ID_MANADA");
23:                String vDATA_NASCIMENTO = rs.getString("DATANASCIMENTO");
24:                String vSEXO = rs.getString("SEXO");
25:                String vRACA = rs.getString("RACA");
26:                String vNIDMAE = rs.getString("NIDMAE");
27:                String vCATEGORIA_CORPORAL = rs.getString("CATEGORIACORPORAL");
28:                boolean vEXISTE = rs.getBoolean("EXISTE");
29:                String vDATA_ENTRADA = rs.getString("DATAENTRADA");
30:                DefaultTableModel modelo = (DefaultTableModel) jTableGerirAnimal.getModel();
                                                                        ▷ Ir buscar a Manada
31:                String sqlMANADA = "select IDENTIFICACAOMANADA from MANADAS where ID_MANADA = "+ vIDENTIFICACAO_MANADA;
32:                PreparedStatement NANi = (PreparedStatement) con.prepareStatement(sqlMANADA);
33:                ResultSet rs2 = NANi.executeQuery();
34:                while (rs2.next())
35:                    vIDENTIFICACAO_MANADA2 = rs2.getString("IDENTIFICACAOMANADA");
36: end procedure
```

Algoritmo 10 Classe Preencher Gerir Animal - Continuação.

```
1: procedure PREENCHER GERIR ANIMAL - CONTINUAÇÃO ▷ Operação que Preencher a Tabela de Gerir Animal - Continuação.
                                                                ▷ Ir buscar o Peso
2:                 String sqlPESO = "select PESOANIMAL from PESO where NLIVRO = "+ vNLIVRO;
3:                 PreparedStatement stPeso = (PreparedStatement) con.prepareStatement(sqlPESO);
4:                 ResultSet rs3 = stPeso.executeQuery();
5:                 while (rs3.next())
6:                     vPESO2 = rs3.getString("PESOANIMAL");
                                                                ▷ Ir buscar a categoria de estado animal
7:                 String sqlCatEstAni = "select ID_CATESTADOPRODUTIVO from ESTADOANIMAL where NLIVRO = "+ vNLIVRO;
8:                 PreparedStatement stCatEstAni = (PreparedStatement) con.prepareStatement(sqlCatEstAni);
9:                 ResultSet rs4 = stCatEstAni.executeQuery();
10:                while (rs4.next())
                                                                ▷ Ir buscar a categoria de estado produtivo
11:                    String sqlCatEstProd = "select NOME_CATEGORIA from CATEGORIAESTPRODUTIVO where ID_CATESTADOPRODUTIVO = "+ rs4.getString("ID_CATESTADOPRODUTIVO");
12:                    PreparedStatement stCatEstProd = (PreparedStatement) con.prepareStatement(sqlCatEstProd);
13:                    ResultSet rs5 = stCatEstProd.executeQuery();
14:                    while (rs5.next())
15:                        vCATEGORIAESTPRODUTIVO2 = rs5.getString("NOME_CATEGORIA");
                                                                ▷ Ir buscar a sanidade do animal
16:            String sqlSanidade = "select DOENTE from SANIDADEANIMAL where NLIVRO = "+ vNLIVRO;
17:            PreparedStatement stSanidade = (PreparedStatement) con.prepareStatement(sqlSanidade);
18:            ResultSet rs6 = stSanidade.executeQuery();
19:            if (rs6.next())
20:                vDOENTE = rs6.getBoolean("DOENTE");
21:            else
22:            vDOENTE = false;
23:            modelo.addRow(new Object[]vNID_BOVINO, vIDENTIFICACAO_MANADA2, vDATA_NASCIMENTO, vCATEGORIAESTPRODUTIVO2, vSEXO, vRACA, vNIDMAE, vCATEGORIA_CORPORAL, vPESO2, vEXISTE, vDATA_ENTRADA,vDOENTE);
24:        catch (Exception e)
25:            jLabelErroGerirAnimais.setText(e.toString());
26: end procedure
```

Algoritmo 11 Código da Pesquisa do Animal.

```
procedure CÓDIGO DA PESQUISA DO ANIMAL
                                                    ▷ Operação que permite Pesquisar.
    private void jButtonPesquisarGerirAnimalActionPerformed(java.awt.event.ActionEvent evt)
        String vManadaP = ;
        String vSexoP = ;
        String vExisteP1 = "1";
        String vExisteP2 = "0";
        String vCatCorporal = ;                     ▷ Combobox Sexo Retira valor.
            if (jComboBoxGerirAnimalSexo.getSelectedItem().equals("Todos"))
                vSexoP = ;
            else
                vSexoP = (String) jComboBoxGerirAnimalSexo.getSelectedItem();
                                                    ▷ Comobox Manada Retira Valor.
            if (jComboBoxGerirAnimalManada.getSelectedItem().equals("Todos"))
                vManadaP = "is not null";
            else
            vManadaP = (String) jComboBoxGerirAnimalManada.getSelectedItem();
            try
                Class.forName("org.apache.derby.jdbc.ClientDriver");
                Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF","admi
"administrador");
                String sqlId_Manada = "SELECT ID_MANADA FROM MANADAS WHERE IDENTIFICACAOMANADA = '"+ vManadaP + "'";
                PreparedStatement Pst2 = (PreparedStatement) con.prepareStatement(sqlId_Manada);
                ResultSet rs2 = Pst2.executeQuery();
                while (rs2.next())
                    vManadaP = "="+ rs2.getInt("ID_MANADA");
            catch (Exception e)
                jLabelErroGerirAnimais.setText(e.toString());
                                                    ▷ ComboBox CatCorporal retira valor.
        if (jComboBoxGerirAnimalCatCorporal.getSelectedItem().equals("Todos"))
            vCatCorporal = "is not null";
        else
            vCatCorporal = "="+ ((jComboBoxGerirAnimalCatCorporal.getSelectedIndex()));
                                                    ▷ ComboBox Existe retira valor.
        if (jComboBoxGerirAnimalExiste.getSelectedItem().equals("Todos"))
            vExisteP1 = "1";
            vExisteP2 = "0";
        else if (jComboBoxGerirAnimalExiste.getSelectedItem().equals("Existe"))
            vExisteP1 = "1";
            vExisteP2 = "1";
        else
            vExisteP1 = "0";
            vExisteP2 = "0";
        DefaultTableModel ModeloTabelaAnimais = (DefaultTableModel) jTableGerirAnimal.getModel();
        int tamanho = ModeloTabelaAnimais.getRowCount();
        for (int i = 0; i < tamanho; i++)
            ModeloTabelaAnimais.removeRow(0);
end procedure
```

Algoritmo 12 Código da Pesquisa do Animal - Continuação.

```
1: procedure CÓDIGO DA PESQUISA DO ANIMAL - CONTINUAÇÃO
                                        ▷ Operação que permite Pesquisar - Continuação.
2: State    try
3:              Class.forName("org.apache.derby.jdbc.ClientDriver");
4:              Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF",
   "administrador", "administrador");                    ▷ Ir buscar o codigo da Manada.
5:              String sql = "select * from LIVROREGISTO where ID_MANADA "+ vMa-
   nadaP + "and SEXO like '"+ vSexoP + "%' and (EXISTE = '"+ vExisteP1 + "' or EXISTE
   = '"+ vExisteP2 + "') and CATEGORIACORPORAL "+ vCatCorporal + ;
6:              PreparedStatement st = (PreparedStatement) con.prepareStatement(sql);
7:              ResultSet rs = st.executeQuery();
8:              String vIDENTIFICACAO_MANADA2 = ;
9:              while (rs.next())
10:                 String vNID_BOVINO = rs.getString("NIDBOVINO");
11:                 String vIDENTIFICACAO_MANADA =
   rs.getString("ID_MANADA");
12:                 String vDATA_NASCIMENTO =
   rs.getString("DATANASCIMENTO");
13:                 String vSEXO = rs.getString("SEXO");
14:                 String vRACA = rs.getString("RACA");
15:                 String vNIDMAE = rs.getString("NIDMAE");
16:                 String vEXPLORACAO = rs.getString("EXPLORACAO");
17:                 String vDESTINO = rs.getString("DESTINO");
18:                 String vCATEGORIA_CORPORAL =
   rs.getString("CATEGORIACORPORAL");
19:                 boolean vEXISTE = rs.getBoolean("EXISTE");
20:                 String vDATA_ENTRADA = rs.getString("DATAENTRADA");
21:                 String sqlMANADA = "select IDENTIFICACAOMANADA from MA-
   NADAS where ID_MANADA = "+ vIDENTIFICACAO_MANADA;
22:                 PreparedStatement NANi = (PreparedStatement)
   con.prepareStatement(sqlMANADA);
23:                 ResultSet rs2 = NANi.executeQuery();
24:                 while (rs2.next())
25:                     vIDENTIFICACAO_MANADA2 =
   rs2.getString("IDENTIFICACAOMANADA");
26:                 ModeloTabelaAnimais.addRow(new Object[]vNID_BOVINO, vIDENTI-
   FICACAO_MANADA2, vDATA_NASCIMENTO, vSEXO, vRACA, vNIDMAE, vEXPLO-
   RACAO, vDESTINO, vCATEGORIA_CORPORAL, vEXISTE, vDATA_ENTRADA);
27:         catch (Exception e)
28:             jLabelErroGerirAnimais.setText(e.toString());
29: end procedure
```

A figura 6.2 mostra o formulário de criação de um novo animal onde se inserem todos os campos necessário à criação de um novo animal bem como caso se pretenda uma fotografia.

Figura 6.2: Criar Animal.

O algoritmo13 mostra o código de criação de um novo animal.
O algoritmo14 mostra o código da classe de criação do novo animal.

Algoritmo 13 Código Criar Novo Animal

```
1: procedure CRIAR NOVO ANIMAL                    ▷ Operação que permite Criar um Novo Animal.
2:     if(jTextFieldNBovino.getText().equals()
3:         || jTextFieldRaca.getText().equals()
4:         || jTextFieldNMae.getText().equals()
5:         || jTextFieldNDocEntrada.getText().equals()
6:         || jTextFieldExploracao.getText().equals()
7:         || jDateChooserDataNascimento.getDate().equals()
8:         || jDateChooserDataEmissao.getDate().equals()
9:         || jDateChooserDataEntradaDoc.getDate().equals()
10:        || jDateChooserDataEntrada.getDate().equals())
11:        jLabelErroNovoAnimal.setText("Por favor preencha todos os campos correta-
   mente");
12:    else                                                                  ▷ Variáveis.
13:        String vIDENTIFICACAO_MANADA = jComboBoxMa-
   nada.getSelectedItem().toString();
14:        String vNID_BOVINO = jTextFieldNBovino.getText().toUpperCase();
15:        String vDATA_NASCIMENTO = new SimpleDateFormat("yyyy-MM-
   dd").format(jDateChooserDataNascimento.getDate());
16:        String vSEXO = (String) jComboBoxSexo.getSelectedItem();
17:        String vRACA = jTextFieldRaca.getText();
18:        String vNIDMAE = jTextFieldNMae.getText();
19:        String vN_DOC_ENTRADA = jTextFieldNDocEntrada.getText();
20:        String vDATA_EMISSAO = new SimpleDateFormat("yyyy-MM-
   dd").format(jDateChooserDataEmissao.getDate());
21:        String vEXPLORACAO = jTextFieldExploracao.getText();
22:        String vDATA_ENTRADA_DOC = new SimpleDateFormat("yyyy-MM-
   dd").format(jDateChooserDataEntradaDoc.getDate());
23:        String vDATA_ENTRADA = new SimpleDateFormat("yyyy-MM-
   dd").format(jDateChooserDataEntrada.getDate());
24:        String vFOTOGRAFIA = jLabelFoto.getIcon().toString();
25:        int vCATEGORIA_CORPORAL = (jComboBoxNAnimalCatCorpo-
   ral.getSelectedIndex()) + 1;
26:        boolean vEXISTE;
27:        if (jCheckBoxExisteBov.isSelected())
28:            vEXISTE = true;
29:         else
30:            vEXISTE = false;
31:                                                   ▷ Procedimento para Inserir novo Animal.
32:        InserirNovoAnimal inserirNovoAnimal = new InserirNovoAni-
   mal(jLabelErroNovoAnimal,
33:        vIDENTIFICACAO_MANADA,
34:        vNID_BOVINO,
35:        vDATA_NASCIMENTO,
36:        vSEXO,
37:        vRACA,
38:        vNIDMAE,
39:        vN_DOC_ENTRADA,
40:        vDATA_EMISSAO,
41:        vEXPLORACAO,
42:        vDATA_ENTRADA_DOC,
43:        vDATA_ENTRADA,
44:        vFOTOGRAFIA,
45:        vCATEGORIA_CORPORAL,
46:        vEXISTE);                                  ▷ Eliminar os campos preenchidos.
47:        jTextFieldNBovino.setText();
48:        jTextFieldRaca.setText("Charolês");
49:        jTextFieldNMae.setText();
50:        jTextFieldNDocEntrada.setText();
51:        jTextFieldExploracao.setText();
52:
53: end procedure
```

Algoritmo 14 Classe Criar Novo Animal

```
1: procedure CLASSE CRIAR NOVO ANIMAL ▷ Operação que permite Criar um Novo Animal.
2:     public class InserirNovoAnimal
3:         int vManada;
4:         InserirNovoAnimal(JLabel jLabelErroNovoAnimal, String vIDENTIFICA-
   CAO_MANADA, String vNID_BOVINO, String vDATA_NASCIMENTO, String vSEXO,
   String vRACA, String vNIDMAE, String vN_DOC_ENTRADA, String vDATA_EMISSAO,
   String vEXPLORACAO, String vDATA_ENTRADA_DOC, String vDATA_ENTRADA,
   String vFOTOGRAFIA, int vCATEGORIA_CORPORAL, boolean vEXISTE)
5:         try                                          ▷ Criar ligação com a base de dados.
6:         Class.forName("org.apache.derby.jdbc.ClientDriver");
7:         Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF",
   "administrador", "administrador");
8:         Statement st = (Statement) con.createStatement();
9:         String verifica = "select NIDBOVINO from LIVROREGISTO where NIDBOVINO
   = '"+ vNID_BOVINO + "'";
10:        PreparedStatement pst = (PreparedStatement) con.prepareStatement(verifica);
11:        ResultSet rs = pst.executeQuery();
12:        if (rs.next())
13:            jLabelErroNovoAnimal.setText("Ja existe este Animal: "+ vNID_BOVINO);
14:         else
                                                        ▷ Ir buscar o código do Animal.
15:            String sqlAnimal = "SELECT ID_MANADA FROM MANADAS WHERE
   IDENTIFICACAOMANADA = '"+ vIDENTIFICACAO_MANADA + "'";
16:            PreparedStatement Pst2 = (PreparedStatement)
   con.prepareStatement(sqlAnimal);
17:            ResultSet rs2 = Pst2.executeQuery();
18:            while (rs2.next())
19:                vManada = rs2.getInt("ID_MANADA");
20:
                                                        ▷ Inserir novo Animal.
21:            String sql = "INSERT INTO LIVROREGISTO
   (LIV_NLIVRO,ID_MANADA, NIDBOVINO, DATANASCIMENTO, SEXO, RACA,
   NIDMAE, NDOCENTRADA, DATAEMISSAO, EXPLORACAO, DATAENTRADADOC,
   DATAENTRADA, EXISTE, FOTOGRAFIA, CATEGORIACORPORAL) VALUES
   (1,"+ vManada + ",'"+ vNID_BOVINO + "','"+ vDATA_NASCIMENTO + "','"+
   vSEXO + "','"+ vRACA + "','"+ vNIDMAE + "','"+ vN_DOC_ENTRADA + "','"+
   vDATA_EMISSAO + "','"+ vEXPLORACAO + "','"+ vDATA_ENTRADA_DOC
   + "','"+ vDATA_ENTRADA + "','"+ vEXISTE + "','"+ vFOTOGRAFIA + "','"+
   vCATEGORIA_CORPORAL + ")";
22:            st.executeUpdate(sql);
23:         //fecha as ligações
24:        st.close();
25:        con.close();
26:         catch (Exception e)
27:            jLabelErroNovoAnimal.setText(e.toString());
28:
29:
30:
31: end procedure
```

A figura 6.3 mostra o formulário de edição de um animal. Neste formulário pode-se editar todos os dados referentes a um animal específico com exceção do número do seu brinco.

Figura 6.3: Editar Animal.

O algoritmo15 mostra o código que verifica qual o animal selecionado na tabela para ser editado.
O algoritmo16 mostra o código de edição de um animal.
O algoritmo17 mostra o código da classe de edição de um animal.

Algoritmo 15 Código de Selecionar Editar Animal.

```
1: procedure CÓDIGO DE SELECIONAR EDITAR ANIMAL        ▷ Operação que seleciona
   Editar Animal.
2:     private void jButtonGerirEditarActionPerformed(java.awt.event.ActionEvent evt)
3:         int linhaSelecionada = jTableGerirAnimal.getSelectedRow();
4:         if (linhaSelecionada < 0 || linhaSelecionada > jTableGerirAnimal.getRowCount())
5:             jLabelErroGerirAnimal.setText("Nao tem nenhuma linha seleccionada");
6:         else
7:             jDialogEditarAnimal.setVisible(true);
8:
9:
10: end procedure
```

Algoritmo 16 Código Editar Animal.

```
1: procedure CÓDIGO EDITAR ANIMAL                    ▷ Operação que permite Editar Animal.
2:     private void jDialogEditarAnimalWindowActivated(java.awt.event.WindowEvent evt)
3:         preencherComboManada();
4:         DefaultTableModel modelo = (DefaultTableModel) jTableGerirAnimal.getModel();
                                                                          ▷ valor da linha selccionado
5:         int linha = jTableGerirAnimal.getSelectedRow();
                                                                          ▷ valor que esta na linha selecionada
6:         String vNIDBOVINOS = (String) modelo.getValueAt(linha, 0);
7:         CarregarDadosEditarAnimal cdeAni = new CarregarDadosEditarAnimal(jLabelErroGerirAnimal, jComboBoxManadaEdit, vNIDBOVINOS, jTextFieldNBovinoEdit, jFormattedTextFieldDataNascimentoEdit, jComboBoxSexoEdit, jTextFieldRacaEdit, jTextFieldNMaeEdit, jTextFieldNDocEntradaEdit, jFormattedTextFieldDataEmissaoEdit, jTextFieldExploracaoEdit, jFormattedTextFieldDataEntDocEdit, jTextFieldNDocSaidaEdit, jFormattedTextFieldDataSaidaDocEdit, jTextFieldMatadouroEdit, jTextFieldDestinoEdit, jFormattedTextFieldDataEntradaEdit, jFormattedTextFieldDataMorteEdit, jCheckBoxExisteBovEdit, jComboBoxNAnimalCatCorporalEdit);
8:         CarregarComboManadas ccME = new CarregarComboManadas(jLabelErroGerirAnimal, jComboBoxManadaEdit);
9:
10: end procedure
```

Algoritmo 17 Classe Editar Animal.

```
1: procedure CLASSE EDITAR ANIMAL
                                                    ▷ Classe que permite Editar um animal.
2:     public class AtualizarAnimal
3:         AtualizarAnimal(JLabel jLabelErroGerirAnimal, String vNIDBOVINOS, String vIDENTIFICACAO_MANADA_EDITAR, String vNID_BOVINO_EDITAR, String vDATA_NASCIMENTO_EDITAR, String vSEXO_EDITAR, String vRACA_EDITAR, String vNIDMAE_EDITAR, String vN_DOC_ENTRADA_EDITAR, String vDATA_EMISSAO_EDITAR, String vEXPLORACAO_EDITAR, String vDATA_ENTRADA_DOC_EDITAR, String vNDOC_SAIDA_EDITAR, String vDATA_SAIDA_DOC_EDITAR, String vMATADOURO_EDITAR, String vDESTINO_EDITAR, String vDATA_ENTRADA_EDITAR, String vDATA_MORTE_EDITAR, boolean vEXISTE_EDITAR, String vFOTOGRAFIA_EDITAR, int vCATEGORIA_CORPORAL_EDITAR)
4:
5:         try
6:             Class.forName("org.apache.derby.jdbc.ClientDriver");
7:             Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
8:             Statement st = (Statement) con.createStatement();
                                                    ▷ Ir buscar a manada
9:             String vIdentificacaoManada=;
10:            String sqlManada = "select ID_MANADA from MANADAS WHERE IDENTIFICACAOMANADA = '"+ vIDENTIFICACAO_MANADA_EDITAR + "'";
11:            PreparedStatement Pst2 = (PreparedStatement) con.prepareStatement(sqlManada);
12:            ResultSet rs2 = Pst2.executeQuery();
13:            while (rs2.next())
14:                vIdentificacaoManada = rs2.getString("ID_MANADA");
15:
16:            String sql = "UPDATE LIVROREGISTO SET ID_MANADA = "+ vIdentificacaoManada + ", ID_MANADA = "+ vIdentificacaoManada + ", NIDBOVINO='"+ vNID_BOVINO_EDITAR + "', DATANASCIMENTO='"+ vDATA_NASCIMENTO_EDITAR + "',SEXO='"+ vSEXO_EDITAR + "',RACA='"+ vRACA_EDITAR + "',NIDMAE='"+ vNIDMAE_EDITAR + "',NDOCENTRADA='"+ vN_DOC_ENTRADA_EDITAR + "',DATAEMISSAO='"+ vDATA_EMISSAO_EDITAR + "',EXPLORACAO='"+ vEXPLORACAO_EDITAR + "',DATAENTRADADOC='"+ vDATA_ENTRADA_DOC_EDITAR + "',NDOCSAIDA='"+ vNDOC_SAIDA_EDITAR + "',DATASAIDADOC='"+ vDATA_SAIDA_DOC_EDITAR + "',MATADOURO='"+ vMATADOURO_EDITAR + "',DESTINO='"+ vDESTINO_EDITAR + "',DATAENTRADA='"+ vDATA_ENTRADA_EDITAR + "',DATAMORTE='"+ vDATA_MORTE_EDITAR + "', EXISTE='"+ vEXISTE_EDITAR + "', FOTOGRAFIA='"+ vFOTOGRAFIA_EDITAR + "',CATEGORIACORPORAL="+ vCATEGORIA_CORPORAL_EDITAR + "WHERE NIDBOVINO = '"+ vNIDBOVINOS + "'";
17:            st.executeUpdate(sql);
18:            jLabelErroGerirAnimal.setText("Animal Atualizado "+ vNID_BOVINO_EDITAR);
19:            st.close();
20:            con.close();
21:             catch (Exception e)
22:                jLabelErroGerirAnimal.setText(e.toString());
23:
24:
25:
26: end procedure
```

A figura 6.4 mostra o formulário para se eliminar uma manada, o programa vai ser se a manada está associada a algum animal ou estado produtivo, se não estiver "desbloqueia"o botão para eliminar.

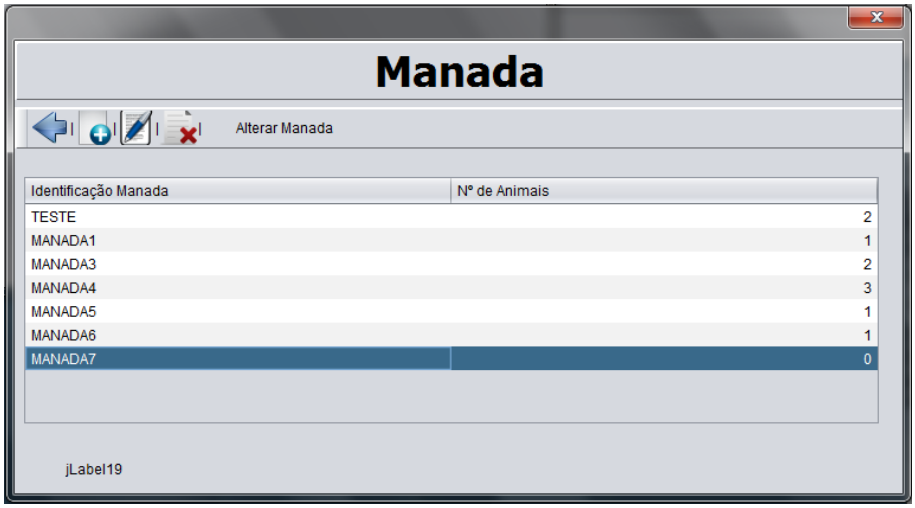

Figura 6.4: Eliminar Manada.

O algoritmo19 mostra o código que verifica há algum animal que está selecionado na tabela e se sim qual.
O algoritmo18 mostra o código da classe que verifica se há algum animal ou estado produtivo associado à manada que se pretende eliminar e se não existir elimina o animal pretendido.

Algoritmo 18 Classe Eliminar Manada.

```
procedure CLASSE ELIMINAR MANADA        ▷ Operação que Preencher a Tabela de Gerir Animal.
    class EliminarManada
        EliminarManada(String vManadaS)
            try
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
            Statement st = (Statement) con.createStatement();
            String sql = "DELETE from MANADAS where IDENTIFICACAOMANADA = '"+ vManadaS + "'";
            st.executeUpdate(sql);
            st.close();
            con.close();
             catch (Exception e)



end procedure
```

Algoritmo 19 Código Eliminar Manada.

```
1: procedure CÓDIGO ELIMINAR MANADA        ▷ Operação qunado se clica no botão eliminar.
2:     DefaultTableModel modelo = (DefaultTableModel) jTableManada.getModel();   ▷ valor da
   linha selccionado
3:     int linha = jTableManada.getSelectedRow();          ▷ valor que esta na linha selecionada
4:     String vManadaS = (String) modelo.getValueAt(linha, 0);
5:     EliminarManada EM = new EliminarManada(vManadaS);
6:     jLabelErroManada.setText("Registo Eliminado");
7:     PreparararGerirManada PGM = new PreparararGerirManada(jTableManada, jLabelErroManada);
8: end procedure
```

A figura 6.5 mostra o formulário para alterar um ou vários animais de uma manada para outra. Onde diz manada atual escolhe-se a manada onde se encontra o animal ou animais que se deseja alterar, onde diz manada seguinte encontra-se a manada para onde se deseja alterar o animal ou animais.

Figura 6.5: Alterar Manada.

O algoritmo20 mostra o código que preenche as tabelas e as combobox do formulário alterar manada.
O algoritmo21 mostra o código da classe que procede à alteração da manada.

Algoritmo 20 Código Alterar Manada.

```
1: procedure CÓDIGO ALTERAR MANADA ▷ Operação que permite alterar a Manda do animal
2:     private void jButtonAlterarManadaActionPerformed(java.awt.event.ActionEvent evt)
3:         DefaultTableModel ModeloManAlterada = (DefaultTableModel) jTableManadaAlterada.getModel();
4:         int linhaAtual = 0;
5:         int tamanho = ModeloManAlterada.getRowCount();
6:         for (int i = 0; i < tamanho; i++)
7:             //valor da linha selccionado
8:             linhaAtual = 0;  ▷ valor que esta na linha selecionada na coluna alimentos
9:             String vID_BovinoS = (String) ModeloManAlterada.getValueAt(linhaAtual, 0);
10:            String vManadaAAlterarS = jComboBoxManadaDestino.getSelectedItem().toString();
11:            AlteraManada alterarManAtualizada = new AlteraManada(jLabelMessageAlterarErro, vID_BovinoS, vManadaAAlterarS);  ▷ remove a linha que passa
12:        ModeloManAlterada.removeRow(linhaAtual);
13:
14:
15: end procedure
```

Algoritmo 21 Classe Alterar Manada.

```
1: procedure CLASSE ALTERAR MANADA ▷ Classe da Operação que permite alterar a Manda do animal
2:     class AlteraManada
3:         public AlteraManada(JLabel jLabelMessageAlterarErro, String vID_BovinoS, String vManadaAAlterarS)
4:             try
5:                 String vID_MANADA2=;
6:                 Class.forName("org.apache.derby.jdbc.ClientDriver");
7:                 Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
8:                 Statement st = (Statement) con.createStatement();
9:                 String sqlEstPro = "select ID_MANADA from MANADAS where IDENTIFICACAOMANADA = '"+ vManadaAAlterarS+"'";
10:                PreparedStatement NANi = (PreparedStatement) con.prepareStatement(sqlEstPro);
11:                ResultSet rs2 = NANi.executeQuery();
12:                while (rs2.next())
13:                    vID_MANADA2 = rs2.getString("ID_MANADA");
14:
15:                String sql = "UPDATE LIVROREGISTO SET ID_MANADA= "+ vID_MANADA2 + "WHERE NIDBOVINO = '"+ vID_BovinoS + "'";
16:                st.executeUpdate(sql);
17:                jLabelMessageAlterarErro.setText("Animais na Manada Atualizada ");
18:                st.close();
19:                con.close();
20:                 catch (Exception e)
21:                    jLabelMessageAlterarErro.setText(e.toString());
22:
23:
24:
25: end procedure
```

A figura 6.6 mostra o formulário para atribuir um animal de uma categoria de estado produtivo. Neste menu podemos escolher o estado pretendido onde se deseja inserir o animal bem como a sua categoria.

Figura 6.6: Atribuir Estado Animal.

O algoritmo22 mostra o código dos dados que vão ser introduzidos na atribuição do estado animal.
O algoritmo23 mostra o código da classe que procede à atribuição do estado animal.

Algoritmo 22 Código Atribuir Estado Animal.

1: **procedure** CÓDIGO ATRIBUIR ESTADO ANIMAL ▷ Operação quando se clica no botão Guardar.
2: String vNID_BOVINO_ESTADO = jTextFieldAtribuirNBovino.getText().toUpperCase();
3: String vCatEstadoProdutivo = jComboBoxAtribuirCatEstAnimal.getSelectedItem().toString();
4: String vDATA_INICIOESTADO = jFormattedTextFieldDataAtribuir.getText();
5: AtribuirEstAnimal atribuirEstAnimal = new AtribuirEstAnimal(jLabelErroGerirAnimal, vNID_BOVINO_ESTADO, vCatEstadoProdutivo, vDATA_INICIOESTADO);
6: jDialogAtribuirEstadoAnimal.setVisible(false);
7: **end procedure**

Algoritmo 23 Classe Atribuir Estado Animal.

```
1: procedure CLASSE ATRIBUIR ESTADO ANIMAL                ▷ Operação para Atribuir Estado Animal.
2:     class AtribuirEstAnimal
3:         AtribuirEstAnimal(JLabel jLabelErroGerirAnimal, String vNID_BOVINO_ESTADO, String vCatEstadoProdutivo, String vDATA_INICIOESTADO)
4:             try
5:                 Class.forName("org.apache.derby.jdbc.ClientDriver");
6:                 Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
7:                 Statement st = (Statement) con.createStatement();
                                                          ▷ Ir buscar a o N.º Livro de Registos
8:                 String vAtribuirEstAnimal = ;
9:                 String sqNLIVRO = "select NLIVRO from LIVROREGISTO WHERE NIDBOVINO = '"+ vNID_BOVINO_ESTADO + "'";
10:                PreparedStatement Pst1 = (PreparedStatement) con.prepareStatement(sqNLIVRO);
11:                ResultSet rs1 = Pst1.executeQuery();
12:                while (rs1.next())
13:                    vAtribuirEstAnimal = rs1.getString("NLIVRO");
14:
                                                          ▷ Ir buscar a o ID_CATEGORIAESTADO
15:                String vCatEstadoProd2 = ;
16:                String sqlCatEst = "select ID_CATESTADOPRODUTIVO from CATEGORIAESTPRODUTIVO WHERE NOME_CATEGORIA = '"+ vCatEstadoProdutivo + "'";
17:                PreparedStatement Pst2 = (PreparedStatement) con.prepareStatement(sqlCatEst);
18:                ResultSet rs2 = Pst2.executeQuery();
19:                while (rs2.next())
20:                    vCatEstadoProd2 = rs2.getString("ID_CATESTADOPRODUTIVO");
21:
22:                String sql = "INSERT INTO ESTADOANIMAL(NLIVRO, ID_CATESTADOPRODUTIVO, DATA_INICIOESTADO, DATA_FIMESTADO)VALUES ("+vAtribuirEstAnimal+ ","+vCatEstadoProd2+ ",'"+vDATA_INICIOESTADO+ "',NULL)";
23:                st.executeUpdate(sql);
24:                jLabelErroGerirAnimal.setText("Estado Animal atribuido");
25:                st.close();
26:                con.close();
27:            catch (Exception e)
28:                jLabelErroGerirAnimal.setText(e.toString());
29:
30:
31:
32: end procedure
```

A figura 6.7 mostra o formulário para alterar um estado produtivo e/ou uma categoria de estado produtivo de um animal. Este formulário é parecido ao alterar manada com a diferença que é preciso escolher o estado animal e a categoria de estado.

Figura 6.7: Alterar Estado Animal.

O algoritmo24 mostra o código dos dados que vão ser alterados na estado animal. O algoritmo25 mostra o código da classe que procede às alterações do estado animal.

Algoritmo 24 Código Alterar Estado Animal.

```
1: procedure CÓDIGO ALTERAR ESTADO ANIMAL    ▷ Operação quando se clica no botão.
2:     DefaultTableModel ModeloEstAlterada = (DefaultTableModel) jTableEstadoAlterado.getModel();
3:     int linhaAtual = 0;
4:     int tamanho = ModeloEstAlterada.getRowCount();
5:     for (int i = 0; i < tamanho; i++)    ▷ valor da linha selccionado
6:         linhaAtual = 0;    ▷ valor que esta na linha selecionada na coluna alimentos
7:         String vID_BovinoS = (String) ModeloEstAlterada.getValueAt(linhaAtual, 0);
8:         String vEstadoAlterarS = jComboBoxEstadoDestino.getSelectedItem().toString();
9:         String vCatEstadoAlterarS = jComboBoxCatEstadoDestino.getSelectedItem().toString();
10:        AlteraEstado alterarEstAtualizada = new AlteraEstado(jLabelMessageErroAlterarEstadoAnimal, vID_BovinoS, vEstadoAlterarS, vCatEstadoAlterarS);    ▷ remove a linha que passa
11:        ModeloEstAlterada.removeRow(linhaAtual);
12:
13: end procedure
```

Algoritmo 25 Classe Alterar Estado Animal.

```
1: procedure CLASSE ALTERAR ESTADO ANIMAL ▷ Operação de Alterar Estado Animal.
2:     class AlteraEstado
3:         AlteraEstado(JLabel jLabelMessageErroAlterarEstadoAnimal, String vID_BovinoS, String vEstadoAlterarS, String vCatEstadoAlterarS)
4:             try
5:                 String vID_ESTADO2=;
6:                 String vID_CatESTADO2 = ;
7:                 Class.forName("org.apache.derby.jdbc.ClientDriver");
8:                 Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
9:                 Statement st = (Statement) con.createStatement();
10:                String sqlEstPro = "select NOME_CATEGORIA from CATEGORIA-ESTPRODUTIVO where NOME_CATEGORIA = '"+ vCatEstadoAlterarS+"'";
11:                PreparedStatement EstProd = (PreparedStatement) con.prepareStatement(sqlEstPro);
12:                ResultSet rs = EstProd.executeQuery();
13:                while (rs.next())
14:                    vID_CatESTADO2 = rs.getString("NOME_CATEGORIA");
15:
16:                String sqlCatEstPro = "select ESTADOPRODUTIVO from ESTADO-PRODUTIVO where ESTADOPRODUTIVO = '"+ vEstadoAlterarS+"'";
17:                PreparedStatement CatEst = (PreparedStatement) con.prepareStatement(sqlCatEstPro);
18:                ResultSet rs2 = CatEst.executeQuery();
19:                while (rs2.next())
20:                    vID_ESTADO2 = rs2.getString("ESTADOPRODUTIVO");
21:
22:                String sql = "UPDATE ESTADOANIMAL SET ID_ESTADOANIMAL= "+ vID_ESTADO2 + "WHERE NLIVRO = '"+ vID_BovinoS + "'";
23:                st.executeUpdate(sql);
24:                jLabelMessageErroAlterarEstadoAnimal.setText("Estado Animal Atualizado");
25:                st.close();
26:                con.close();
27:            catch (Exception e)
28:                jLabelMessageErroAlterarEstadoAnimal.setText(e.toString());
29:
30:
31:
32: end procedure
```

A figura 6.8 mostra o formulário para introduzir o peso de um animal.

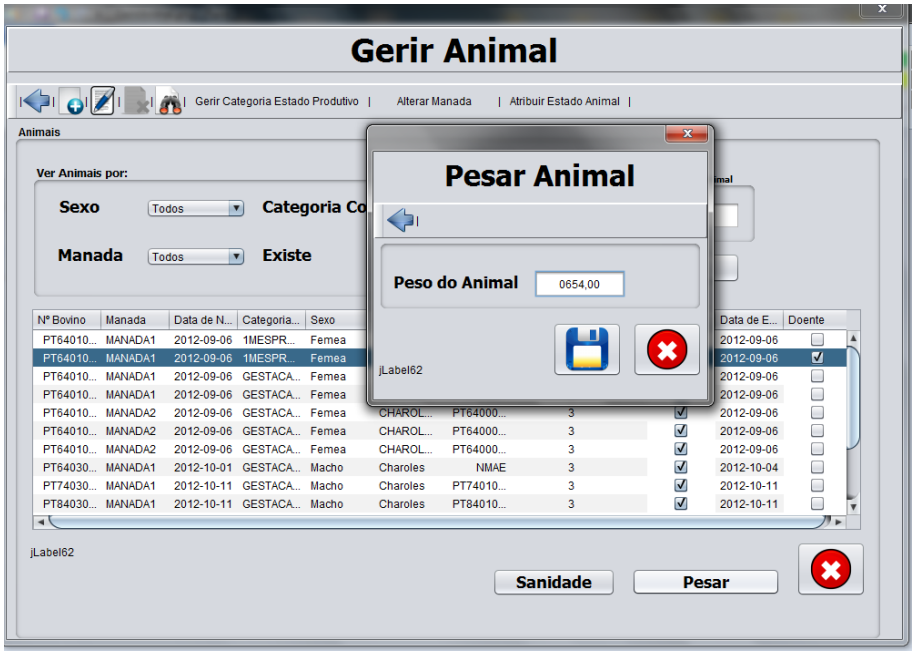

Figura 6.8: Pesar Animal.

O algoritmo26 mostra o código que vai "buscar"o animal para se introduzir o peso do animal.
O algoritmo27 mostra o código da classe que procede à introdução do peso.

Algoritmo 26 Código Pesar Animal

```
procedure CÓDIGO PESAR ANIMAL                ▷ Operação que permite Pesar uma Animal.
    String vPESOANIMAL = jFormattedTextFieldPesoAnimal.getText();
    DefaultTableModel modelo = (DefaultTableModel) jTableGerirAnimal.getModel();   ▷ valor da linha selccionado.
    int linha = jTableGerirAnimal.getSelectedRow();   ▷ valor que esta na linha selecionada.
    String vAnimalS = (String) modelo.getValueAt(linha, 0);   ▷ Procedimento para Inserir novo Peso.
    InserirPesoAnimal inserirPesoAnimal = new InserirPesoAnimal(jLabelErroPeso, vAnimalS, vPESOANIMAL);   ▷ Eliminar os campos preenchidos.
    jFormattedTextFieldPesoAnimal.setText();
end procedure
```

Algoritmo 27 Classe Pesar Animal

```
1: procedure CLASSE PESAR ANIMAL                    ▷ Operação que permite Pesar uma Animal.
2:     public class InserirPesoAnimal
3:         InserirPesoAnimal(JLabel jLabelErroPeso, String vAnimalS, String vPESOANIMAL)
4:             try
                                                                    ▷ Criar ligação com a base de dados
5:                 Class.forName("org.apache.derby.jdbc.ClientDriver");
6:                 Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
7:                 Statement st = (Statement) con.createStatement();
8:                 String vNLIVRO = ;
                                                                    ▷ Ir buscar o NLIVROAnimal
9:                 String sqlManada = "select NLIVRO from LIVROREGISTO WHERE NIDBOVINO = '"+ vAnimalS + "'";
10:                PreparedStatement Pst2 = (PreparedStatement) con.prepareStatement(sqlManada);
11:                ResultSet rs2 = Pst2.executeQuery();
12:                while (rs2.next())
13:                    vNLIVRO = rs2.getString("NLIVRO");
14:
                                                                    ▷ Inserir o novo peso na tabela
15:                String sql = "INSERT INTO PESO (NLIVRO, PESOANIMAL, DATAPESO) VALUES ("+ vNLIVRO + ",'"+ vPESOANIMAL + "',CURRENT_DATE)";
16:                st.executeUpdate(sql);
                                                                    ▷ Fecha as ligações
17:                st.close();
18:                con.close();
19:            catch (Exception e)
20:                jLabelErroPeso.setText(e.toString());
21:
22:
23:
24: end procedure
```

6.3 Base de dados

A figura 6.9 representa o modelo físico da nossa aplicação, modelo este que representa a base de dados utilizada.

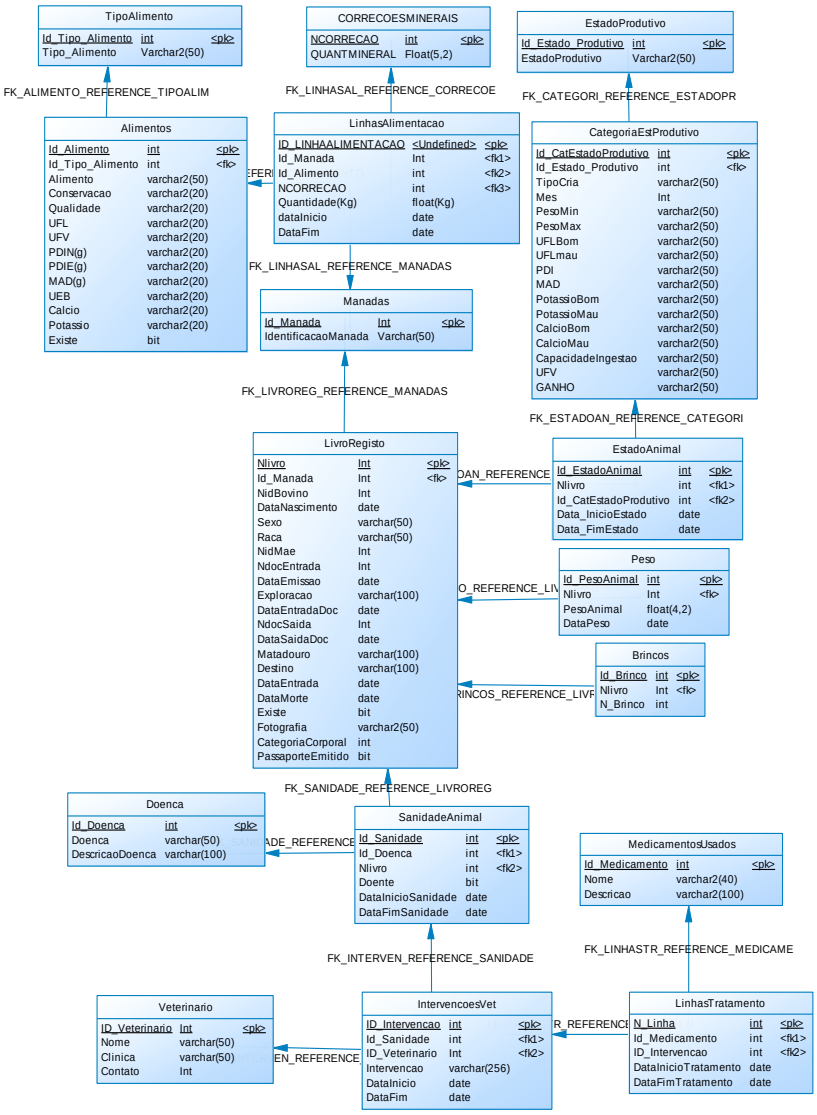

Figura 6.9: Modelo ER.

O modelo físico foi elaborado durante a fase inicial do desenvolvimento da nossa aplicação, e representa a nossa base de dados. Porém ao longo do estudo que fomos realizando e conforme fomos desenvolvendo a aplicação, o modelo físico foi sofrendo alterações conforme o necessário de modo a torná-lo mais eficiente e mais apto para o que pretendíamos. Este modelo que aqui apresentamos é a versão final do modelo físico e aquele que com base no necessário nos permite implementar a nossa aplicação sem problemas.

Capítulo 7

Conclusões e trabalho futuro

7.1 Conclusões

Inicialmente foi difícil entender a dimensão e funcionamento da exploração agrícola. O facto de os nossos conhecimentos em relação à agricultura serem muito limitados e em alguns aspetos mesmo inexistentes dificultou a nossa interpretação sobre o pretendido e o que implementar. Essa dificuldade foi ultrapassada como uma visita à exploração agrícola e ver no terreno como as coisas se processam. A implementação automática dos números dos brincos não foi possível solucionar. Este deveu-se ao facto de não termos conseguido adquirir o dígito de controlo dos brincos impedindo assim de criar um método que fosse fiável. Por fim pessoalmente a maior dificuldade que senti foi o SQL, que apesar de ter desenvolvido os meus conhecimentos e aprendido muito, é uma parte de extrema importância que tenho que desenvolver mais. No entanto o facto de ter aprendido muito e de ter ganho noções diferentes sobre o SQL e ter desenvolvido mais os meus conhecimentos de Java, foi sem dúvida uma mais-valia e muito gratificante.

Para mim como individuo bem como na minha vida profissional foi de extrema importância ter a oportunidade de desenvolver esta aplicação para um "cliente", foi muito produtivo e interessante criar um aplicação completamente pensada e elaborada por nós tendo só como base o que o "cliente"nos pediu deixando ao nosso critério a melhor maneira de o fazer. Independentemente disso o cliente esteve sempre presente no desenvolvimento, e nós estivemos sempre sujeitos à sua aprovação, o que nos motivou a dar o nosso melhor e cumprir o máximo de objetivos a que nos propusemos. Sem dúvida foi muito gratificante fazer uma aplicação para ser implementada e ficar por dentro do funcionamento e conceito de Gerir uma Exploração Agrícola de criação de gado bovino.

7.2 Trabalho Futuro

Como trabalho futuro de modo a tornar a aplicação o mais completa possível a implementação de uma parte que ajude nas culturas da exploração de modo a se plantar os alimentos necessários de forma eficiente(quantidades necessárias, tamanho do terreno necessário, ou outros), para alimentar o gado sem ter que se adquirir os

alimentos que contém as proteínas necessárias fora da exploração, poupando assim dinheiro e aproveitando melhor os terrenos da exploração agrícola.

Para além da parte das culturas um "upgrade"à inserção dos brincos, tornando essa inserção automatizada, bem como a implementação de gráficos de histórico animal por data, sanidade, entre outros, será uma mais valia que tornará a aplicação mais completa e eficiente.

Bibliografia

[1] AGROGESTAO. Agrogestao - solucao integrada de gestao. http://agrogestao.com/pagina.asp?ID=15. Visitado a 01 de Outubro de 2012.

[2] CulturaMix. Tecnologia na agricultura. http://meioambiente.culturamix.com/agricultura/tecnologia-na-agricultura. Visitado a 16 de Setembro de 2012.

[3] Jean-Paul Desgranges Raymond Gadoud Marie-Madeleine Joseph Gerard Joyaux Roland Jussiau Jean Metge Pierre Pelekhine Jean-Louis Tisserand Raoul Rives Gilbert Bonnes, Jeanine Desclaude. Alimentation des bovins. Coordination I.N.R.A.P., 1978.

[4] Junior Goncalves. Metodologia xp, extreme programming, desenvolvimento agil. http://www.hiperbytes.com.br/miscelanea/sem-categoria/metodologia-xp-extreme-programming-%E2%80%93-desenvolvimento-agil/. Visitado a 23 de Setembro de 2012.

[5] SoftAgro Sistemas. S.a - produtor. http://www.softagro.com.br/two.php?flag=prod&tit=1. Visitado a 01 de Outubro de 2012.

Apêndice A

Anexo - Artigo da aplicação

Aplicação de Gestão da Exploração da Quinta das Marietas

Hugo Filipe de Pina Jorge
André Martins Gonçalves
Mestre José Quitério Figueiredo
Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda
Av. Dr. Francisco Sá Carneiro, 50 – 6300 Guarda
hugojorge_35@hotmail.com

***Resumo* — A evolução da tecnologia permitiu que a informática fosse introduzida na agricultura de modo a ajudar e a facilitar a vida dos agricultores e dos gestores das explorações agrícolas. Este artigo descreve o trabalho que foi realizado no âmbito da unidade curricular Projeto de Informática na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão da Guarda e consiste na criação de uma aplicação desktop para a gestão de uma exploração agrícola.**
O trabalho consiste no estudo do funcionamento de uma exploração agrícola, mais propriamente uma exploração de criação de gado bovino, e no desenvolvimento de uma aplicação desktop onde seja possível gerir os animais da exploração, a alimentação e sanidade dos mesmos. A aplicação foi criada em Java e com base de dados criada em Java DB.

1. Introdução

Atualmente a tecnologia está presente em todos os sectores do nosso quotidiano e a agricultura não é exceção. Um bom programa de gestão agrícola que consiga ter todos os dados referentes à exploração agrícola numa só aplicação torna muito mais eficientes os registos efetuados e a consulta dos mesmos, impedindo ainda muitos erros na parte da gestão poupando assim tempo e dinheiro aos responsáveis pela exploração. Numa exploração agrícola os registos dos dados dos animais, o estado produtivo, a manada em que se encontram inseridos entre outros dados são de extrema importância uma vez que vai distinguir os animais uns dos outros além de influenciar no tipo de alimentação de cada um. Outra parte fulcral é a escolha dos alimentos relativamente aos seus constituintes bem como as necessidades energéticas. O registo da sanidade animal ao longo do tempo é outro aspeto a ter em conta quando se desenvolve um Software como o pretendido.

2. Motivação

O desenvolver um Software para uma exploração agrícola, ajudar a sua evolução bem como a noção que temos da sua importância foi uma enorme motivação para a realização deste projeto. Além de que poder contribuir e ajudar assim ao desenvolvimento de uma exploração agrícola de modo a facilitar e a ajudar a inovar no modo de gerir a mesma tornando tudo mais acessível. Foi extremamente motivador e cativante.

3. Objetivos

Desenvolver uma aplicação desktop para a exploração agrícola Quinta das Marietas, de modo a ter todas as funcionalidades necessárias à gestão dos animais, manadas, categorias de estado produtivo, brincos, peso, poder ver o histórico de um animal, gerir e definir alimentação de acordo com cada manada, em que categoria de estado produtivo se encontram os animais da manada e o número de animais em cada categoria, gerir a sanidade dos animais de modo a se poder registar as doenças de cada animal assim como as intervenções veterinárias e os medicamentos prescritos num caso específico, tendo em conta o tempo de duração da doença.

Os objetivos a atingir consistem:

- Criar, Editar, Pesquisar informação dos diversos animais.
- Criar, Editar, Eliminar, Pesquisar manadas.
- Criar, Editar, Eliminar, Pesquisar categorias de estados produtivos.
- Permitir a inserção de brincos.
- Atribuir um brinco a cada animal.
- Associar manadas aos animais.
- Associar categorias de estado produtivo às manadas.
- Criar, Editar, Eliminar, Pesquisar os vários tipos de alimentos e alimentos.
- Calcular e Definir a alimentação para cada manada de acordo com as categorias de estado produtivo de cada animal que se encontra na respetiva manada.
- Criar, Editar, Eliminar, Pesquisar casos de sanidade de cada animal.
- Criar, Editar, Eliminar, Pesquisar intervenções veterinárias de um determinado caso de sanidade de um animal bem como os medicamentos prescritos por intervenção.

4. Metodologia

A metodologia escolhida e utilizada para desenvolver, implementar e testar a aplicação desktop é a Metodologia Ágil XP (EXTREME PROGRAMMING).
Esta metodologia foi escolhida devido ao facto de existir uma cooperação constante entre nós e o cliente. Ao longo do período de desenvolvimento da aplicação fomos tendo sempre que possível uma aplicação funcional para mostrar ao cliente, com o objetivo de o cliente nos dizer se era o pretendido ou o que faltava. Ou seja desenvolvemos a programação em primeiro lugar e delegámos a documentação para o fim tendo sempre em atenção à opinião do cliente e ao pretendido.

5. Desenvolvimento

Na implementação tentámos fazer sempre o pretendido o mais fácil e eficazmente possível de modo a fazer um programa com qualidade, rapidez e eficácia, sem descurar a simplicidade e o pretendido. Devido ao tamanho da aplicação decidimos dividi-la em duas partes tendo eu desenvolvido a parte da gestão animal e o meu colega André Gonçalves a parte da gestão alimentar e da gestão da sanidade animal.

De seguida vamos demonstrar alguns dos vários menus da aplicação, os que achamos mais importantes, referentes à parte dos animais.

A. Menu Inicial da Aplicação

Este menu é o menu inicial da aplicação a partir daqui pode-se escolher o que se deseja fazer, gerir animais, alimentação, sanidade. Também contém os separadores dos avisos onde a aplicação alerta sobre vários aspetos relacionados com as várias gestões que são possíveis serem feitas.

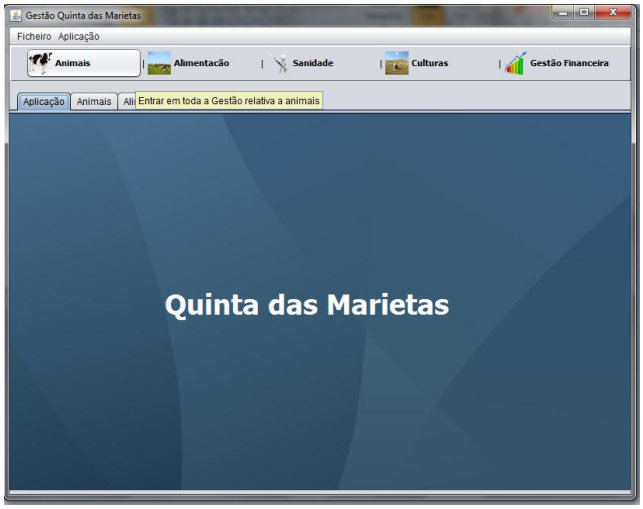

Figura 1 – Menu Inicial da Aplicação

B. Menu Inicial da Gestão Animal

A partir deste menu podemos decidir o que desejamos fazer em relação à gestão dos animais. Temos a opção de inserir brincos, gerir mandada, estado animal ou animais.

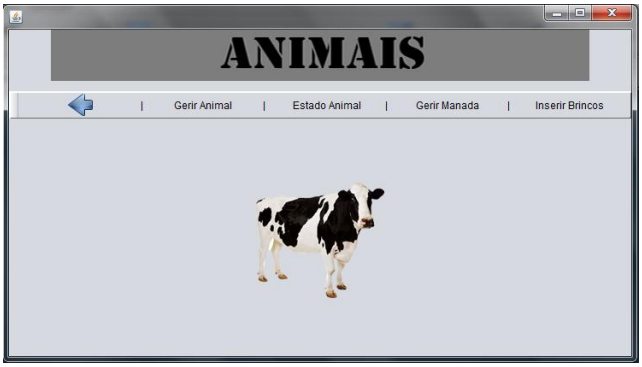

Figura 2 – Menu Inicial da Gestão Animal

C. Menu Gerir Animal

Neste menu além de se poder pesquisar um animal específico ou um conjunto de animais, temos acesso a toda a parte de gestão dos animais podendo criar ou editar um animal, alterar a manda onde este se encontra, atribuir um estado animal, pesar um animal ou mesmo ter acesso à sanidade do mesmo.

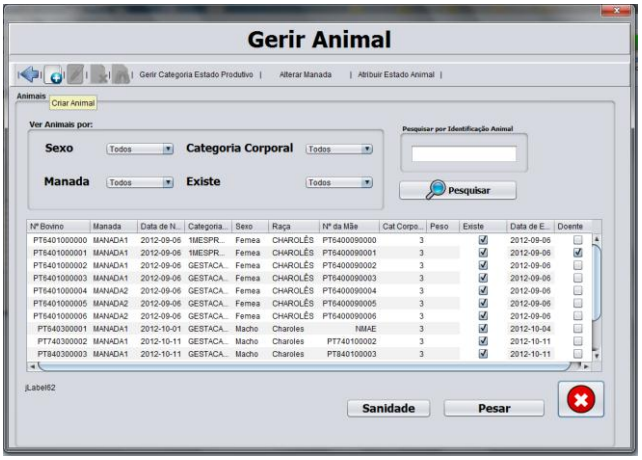

Figura 3 – Menu Gerir Animal

D. *Menu Novo Animal*

Este menu é o de criação de um novo animal onde se inserem todos os campos necessário à criação de um novo animal bem como caso se pretenda uma fotografia.

Figura 4 – Menu Novo Animal

E. *Formulário Editar Animal*

Neste formulário de edição de um animal. Neste formulário pode-se editar todos os dados referentes a um animal específico com exceção do número do seu brinco

Figura 5 – Menu Editar Animal

F. *Menu Atribuir Estado Animal*

Mostra o formulário para atribuir um animal a um estado animal e a uma categoria de estado produtivo. Neste menu podemos escolher o estado pretendido onde se deseja inserir o animal bem como a sua categoria.

Figura 6 – Menu Atribuir Estado Animal

G. *Menu Gerir Categoria Estado Produtivo*

Este menu permite gerir as categorias de estado produtivo existentes, bem como adicionar novas categorias, editar, ou eliminar.

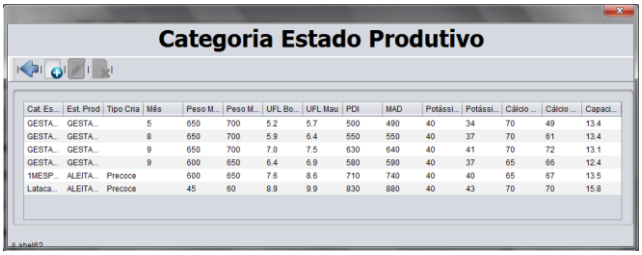

Figura 7 – Menu Gerir Categoria Estado Produtivo

H. Menu Alterar Estado Animal

Este formulário serve para alterar um estado produtivo e/ou uma categoria de estado produtivo de um animal. Escolhe-se o estado do animal e a categoria de onde se pretende alterar e coloca-se no estado e categoria pretendida.

Figura 8 – Menu Alterar Estado Animal

I. Menu Gerir Brincos

Neste formulário podemos inserir um ou vários brincos que depois vão ser utilizados na criação dos novos animais.

Figura 9 – Menu Gerir Brincos

6. Algoritmos

De seguida estão exemplos dos algoritmos que acho mais importantes, como um exemplo de um criar, editar, eliminar, associar e pesquisar.

1. Inicio Criar
2. O Sistema gera o ID_MANADA (incrementa uma unidade ao último ID_MANADA).
3. Introduzir IDENTIFICACAO_MANADA
4. Criar nova Manada
5. Fim

1. Inicio Editar
2. Selecionar a Manada através do ID_MANADA
3. Introduzir nova IDENTIFICACAO_MANADA
4. Alterar dados Manada
5. Fim

1. Inicio Eliminar
2. Selecionar a Manada através do ID_MANADA
3. Se ID_MANADA está a ser utilizado
   3.1. Cancelar operação
4. Senão
   4.1. Executar operação de apagar a manada
5. Fim

1. Inicio Associar
2. Selecionar um Animal através do NIDBOVINO
3. Selecionar a Manada através do ID_MANADA
4. Se NIDBOVINO está na Manada Pretendida
   4.1. Cancelar operação
5. Senão
   5.1. Executar operação Associar Manada (NIDBOVINO associa IDENTIFICACAO_MANADA).
6. Fim

1. Inicio Pesquisar
2. Se Introduz o número do brinco do animal
   2.1. Se NIDBOVINO = número introduzido
      2.1.1. Cancelar operação.
   2.2. Senão
      2.2.1. Devolve "Animal não encontrado"
3. Senão
   3.1. Se Seleciona sexo e/ou manada e/ou categoria corporal e/ou se existe na exploração
      3.1.1. Devolve a informação filtrada
   3.2. Senão
      3.2.1. Devolve "Animal não encontrado"
4. Fim

7. Considerações Finais

Como individuo foi de extrema importância ter a oportunidade de desenvolver esta aplicação completamente pensada e elaborada por nós tendo só como base o que o "cliente" nos pediu deixando ao nosso critério a melhor maneira de o fazer. O facto de o cliente estar sempre presente no desenvolvimento, e nós estarmos constantemente sujeitos à sua aprovação, motivou-nos a dar o nosso melhor e cumprir o máximo de objetivos a que nos propusemos. Sem dúvida foi muito gratificante fazer uma aplicação para ser implementada e ficar por dentro do funcionamento e conceito de Gerir uma Exploração Agrícola de criação de gado bovino.

8. Conclusões

Pelo que pudemos verificar não existe nada relativamente à área específica que nos foi proposta, a criação de gado bovino. A nossa aplicação não se limita unicamente ao registo e consulta do que se passa na exploração, mas também permite calcular a alimentação para os animais da exploração. Deste modo, é possível controlar melhor a produtividade da exploração que é o objetivo fundamental de qualquer gestor/empresário de uma exploração.

9. Referências

1. AGROGESTAO. Agrogestao - solucao integrada de gestao. http://Agrogestao.com/pagina.asp?ID=15. Visitado a 01 de Outubro de 2012.
2. CulturaMix. Tecnologia na agricultura. http://meioambiente.culturamix.com/agricultura/tecnologia-na-agricultura. Visitado a 16 de Setembro de 2012.
3. Junior Goncalves. Metodologia xp, extreme programming, desenvolvimento agil. http://www.hiperbytes.com.br/miscelanea/sem-categoria/metodologia-xp-extreme-programming-%E2%80%93-desenvolvimento-agil/. Visitado a 23 de Setembro de 2012.
4. SoftAgro Sistemas. S.a - produtor. http://www.softagro.com.br/two.php?flag=prod&tit=1. Visitado a 01 de Outubro de 2012.