Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

André Martins Gonçalves
novembro | 2012

Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Aplicação de Gestão da alimentação e sanidade dos animais da Exploração Quinta das Marietas

André Martins Gonçalves
nº1008921

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 da alimentação e sanidade dos animais da Exploração Quinta das Marietas

André Martins Gonçalves
nº1008921

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

Gostaríamos de agradecer:

Ao Sr. João Pedro Ribeiro por nos propor e ter dado a oportunidade de poder fazer parte do desenvolvimento deste projeto.

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.

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.

A Professora Maria Clara Silveira pela grande disponibilidade em nos apoiar com a metodologia, planificação, organização, e analise de todo o projeto foi muito importante para nós.

E por fim mas não menos importante gostaríamos também de agradecer ao Professor Paulo Nunes pela disponibilidade e prontidão em nos ajudar em qualquer assunto, e principalmente pela ajuda fornecida na planificação do relatório e com o latex.

Mais uma vez um muito obrigado a todos.

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 o desenvolvimento de uma aplicação desktop onde seja possível gerir a alimentação e sanidade dos animais da exploração. Esta aplicação irá permitir ao gestor da exploração calcular a alimentação para os animais assim como registar todos os casos de sanidade para cada animal. 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 diet and health of farm animals. This application will enable the manager of exploration to calculate the feed for animals and as recording all cases of health for each animal. 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

Atualmente 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 recusarem a entrar nesta realidade perdem espaço no mercado. 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 adoptar 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 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 têm 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 ser possível 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, assim como o reconhecimento da importância da agricultura no passado, presente e futuro.

Outro factor de extrema importância foi que 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.

Pelos motivos descritos ficamos ligados com a proposta, e cativou-nos a estudá-la 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 os 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 de uma exploração agrícola, 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 de gestão da alimentação e sanidade dos animais.

1.3 Contribuição

A contribuição principal deste trabalho é o desenvolvimento, implementação e teste de uma aplicação destktop, 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 e de um capítulo de conclusões.

No segundo capítulo é apresentado o estado da arte, onde fazemos referência 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 que foram seguidas em todo o processo de desenvolvimento da nossa aplicação.

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

No capítulo cinco 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.

Neste trabalho utilizamos o livro "Alimentation des Bouvins"([3]) e durante todos os capítulos vai ser usado sistematicamente.

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 (exemplo: criação de Gado, produção de cereais, etc...), enquanto a nossa, tendo em conta que é uma aplicação personalizada e elaborada de acordo com as necessidades do cliente, foca-se essencialmente na criação de gado bovino permitindo ao gestor da exploração 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 alfaias; 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 alfaias 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 partições, cobrições e diagnósticos de gestação.
- Validação de periodicidade de partições.
- 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, desenhos e esquemas a cada animal.

2.3 Análise Crítica do Estado de Arte

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

Metodologia e resultados esperados

3.1 Metodologia

A metodologia escolhida e utilizada para desenvolver, implementar e testar a aplicação desktop é o Desenvolvimento Ágil com uma ligeira adaptação a Metodologia XP *(EXTREME PROGRAMMING)* -.

Uma 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, gestor da exploração, 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 ás mudanças. Uma interação constante da parte do cliente é uma mais valia para qualquer projeto, por esses motivos deve ser um método a utilizar.

3.2 Descrição das tarefas

As principais tarefas de toda a organização e desenvolvimento da nossa aplicação são:

- Tarefa 1 — Análise dos requisitos – Definição das funcionalidade da aplicação.
- Tarefa 2 — Separação do projeto em duas partes.
- Tarefa 3 — Obtenção de documentação sobre a alimentação.
  - Estudo sobre os alimentos essencialmente sobre os nutrientes necessários para o calculo.
  - Estudo sobre o calculo da alimentação e relação alimentação por categorias de estado produtivo.
- Tarefa 4 — Obtenção de documentação sobre a sanidade do gado.
  - Estudo sobre como efetuar de forma eficaz de registo e consulta de caso de sanidade animal, tal como intervenções do veterinário num determinado caso de sanidade e medicamentos associados a cada intervenção.
- Tarefa 5 — Implementação da solução proposta.
- Tarefa 6 — Juntar partes da aplicação.
- Tarefa 7 — Teste da aplicação — Para cada uma das funcionalidade da aplicação proceder da seguinte forma:
  1. Inserir informação na Base de dados.
  2. Testar pesquisas.
  3. Testar editar.
  4. Testar eliminar.
- Tarefa 8 — Elaboração do relatório.

O agendamento das tarefas previsto é apresentado na figura 3.1.

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

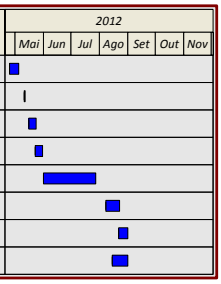

Figura 3.1: Mapa de Gant.

Na figura 3.2 é apresentado o agendamento das tarefas real.

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	06-07-2012	31-08-2012	8,5sem	30h
6	Tarefa5 + Tarefa 6	05-09-2012	19-09-2012	2,5sem	30h
7	Tarefa 7	26-09-2012	28-09-2012	1sem	30h
8	Tarefa 8	22-08-2012	10-10-2012	7,5sem	3h

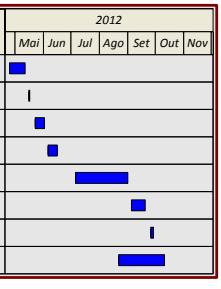

Figura 3.2: Mapa de Gant.

Capítulo 4

Análise dos requisitos e Conceção

4.1 Diagrama de Contexto

Um diagrama de contexto, como mostra a figura 4.1, permite interligar o estudo/projeto "Aplicação de Gestão da Exploração Quinta das Marietas"ao factor externo que neste caso é o gestor da exploração bem como as interações que o gestor da exploração têm com o sistema através de fluxos de dados.

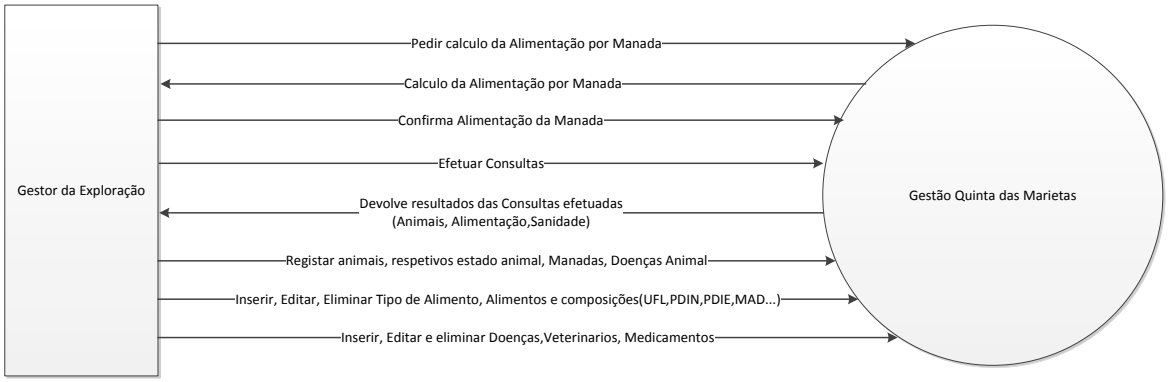

Figura 4.1: Diagrama de Contexto.

4.2 Atores e Respetivos Casos de Uso

A tabela seguinte 4.1 tem como objetivo definir o ator/atores (Gestor da Exploração) bem como os respetivos casos de uso que interferem com o sistema, os casos de uso definem a maioria dos requisitos de um sistema computacional.

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 Cat. 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

Tabela 4.1: Atores e respetivos casos de uso

*Todos os casos de uso que começam por "Gerir"são semelhantes tendo como referencia o Criar, Editar e Eliminar Alimento.

**Todos os casos de uso que começam por "pesquisar"são semelhantes tendo como referencia o Pesquisar Alimento.

4.3 Diagrama de Casos de Uso

O diagrama de casos de uso (4.2) permite mostrar quais são os usos do sistema assim como definir o ator que está relacionado com o uso, neste caso podemos ver a fronteira que delimita o sistema "Gestão da Exploração Quinta das Marietas", onde estão inseridos os casos de uso respetivos e o ator "Gestor da exploração"que esta associado aos caso de uso.

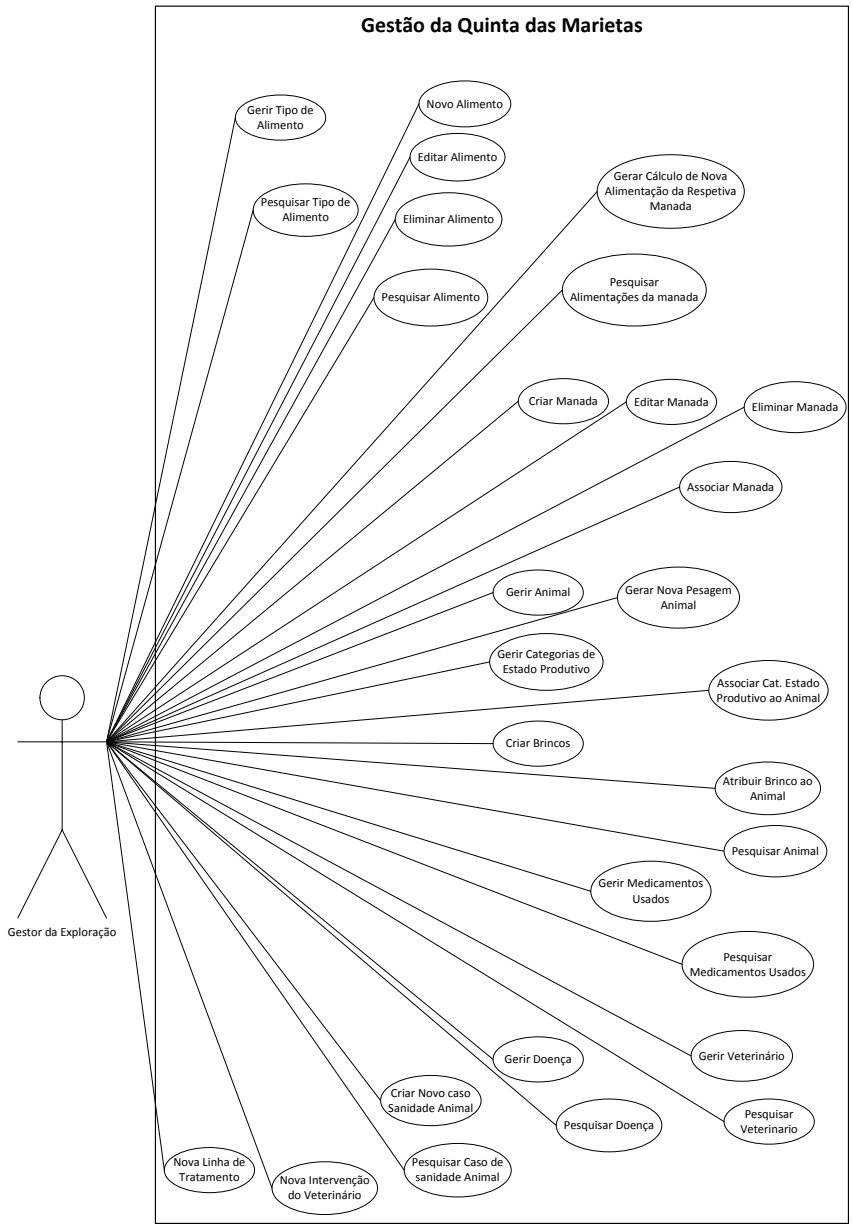

Figura 4.2: Diagrama de Casos de Uso.

4.4 Descrição de Casos de Uso

Aqui vamos descrever com detalhe os casos de uso mais relevantes para a parte da alimentação e para a sanidade animal. Cada tabela é constituída por:

Nome – Indica o nome do caso de uso que se trata.
Descrição – Descreve o objetivo do caso de uso.
Atores Envolvidos – Indica os atores que interagem no caso de uso.
Pré Condições – Indica se existir a pré condição necessária para se puder dar inicio ao caso de uso
Fluxo Principal – Descreve as varias etapas do caso de uso entre o ator e o sistema.
Fluxos Alternativos – Descreve Validações de campos e operações anormais ao fluxo principal.
Suplementos – Indica os casos de teste concretos ao caso de uso.
Pós Condições – Se existirem descrevem alguma operação efetuada após o término do caso de uso.

Criar Novo Alimento:

A tabela (4.2) descreve com detalhe o caso de uso Criar Novo Alimento.

Nome:	Criar Novo Alimento
Descrição:	O objetivo é o Utilizador poder Criar um novo alimento.
Atores Envolvidos:	Gestor da Exploração
Pré Condições:	Não têm
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Novo alimento". 2. O Sistema disponibiliza o formulário Com todos os campos a preencher. 3. O Utilizador introduz: Id Tipo de Alimento, Conservação, Qualidade, UFL, UFV, PDINg, PDIEg, MADg, UEB, Cálcio, potássio e se existe na exploração. 4. O sistema pede para confirmar. 5. O utilizador confirma (Botão Guardar). 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 utilizador não preencher todos os campos com a exceção do MADg e UEB e UFL tem que ser constituído por # . # # , UFV tem que ser constituído por # . # #, PDINg tem que ser constituído por # # # , PDIEg tem que ser constituído por # # # , MADg tem que ser constituído por: # # # , UEB tem que ser constituído por # . # # , Cálcio tem que ser constituído por # # . # # , potássio tem que ser constituído por # # . # # . Onde o # identifica um valor numérico. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se o sistema deixa inserir sem os campos estarem todos preenchidos com a exceção do MADg e UEB.
Pós-Condições:	Não Têm

Tabela 4.2: Descrição Caso de Uso: Criar Novo Alimento.

Editar Alimento:

A tabela (4.3) descreve com detalhe o caso de uso Editar Alimento.

Nome:	Editar Alimento.
Descrição:	O objetivo é o Utilizador poder Editar um Alimento.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Não têm.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Editar Alimento". 2. O Sistema disponibiliza o formulário Com todos os campos do alimento a editar. 3. O Utilizador edita: Id Tipo de Alimento, Conservação, Qualidade, UFL, UFV, PDINg, PDIEg, MADg, UEB, Cálcio, potássio e se existe na exploração. 4. O sistema pede para confirmar. 5. O utilizador confirma (Botão Guardar). 6. O sistema atualiza.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 3a. O sistema cancela se o utilizador não preencher todos os campos com a exceção do MADg e UEB e UFL tem que ser constituído por # . # # , UFV tem que ser constituído por # . # #, PDINg tem que ser constituído por # # # , PDIEg tem que ser constituído por # # # , MADg tem que ser constituído por: # # # , UEB tem que ser constituído por # . # # , Cálcio tem que ser constituído por # # . # # , potássio tem que ser constituído por # # . # # . Onde o # identifica um valor numérico. . 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se o sistema permite criar uma doença já inserida anteriormente. Testar se o sistema deixa inserir sem os campos estarem todos preenchidos.
Pós-Condições:	Não Têm

Tabela 4.3: Descrição Caso de Uso: Editar Alimento.

Eliminar Alimento:
A tabela (4.4) descreve com detalhe o caso de uso Eliminar Alimento.

Nome:	Eliminar Alimento.
Descrição:	O objetivo é o Utilizador poder Eliminar um alimento.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Não têm.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Gerir Alimentos". 2. O Sistema disponibiliza formulário com todos os Alimentos. 3. O Utilizador seleciona o Alimento que pretende 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 pelo menos uma manada estiver a ser alimentada ou foi alimentada por esse alimento. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se o sistema permite eliminar um alimento registada em alguma alimentação da manada.
Pós-Condições:	Não têm.

Tabela 4.4: Descrição Caso de Uso: Eliminar Alimento

Pesquisar Alimento:
A tabela (4.5) descreve com detalhe o caso de uso Eliminar Alimento.

Nome:	Pesquisar Alimento.
Descrição:	O objetivo é o Utilizador poder pesquisar informação dos alimentos.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Não têm.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Gerir Alimentos". 2. O Sistema disponibiliza o interface "Gerir Alimentos"com o formulário de pesquisa e uma tabela com todos os alimentos existentes. 3. O utilizador vai pesquisar um Alimento por Nome, Tipo de Alimento e/ou se Existe ou não na Exploração. 4. O sistema mostra os resultados filtrados.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar".
Suplementos:	Testar se os resultados coincidem na perfeição com o pedido.
Pós-Condições:	Não Têm

Tabela 4.5: Descrição Caso de Uso: Pesquisar Alimento.

Gerar Cálculo de Nova Alimentação da Respetiva Manada:

A tabela (4.6) descreve com detalhe o caso de uso Gerar Cálculo de Nova Alimentação da Respetiva Manada.

Nome:	Gerar Cálculo de Nova Alimentação da Respetiva Manada.
Descrição:	O objetivo é o Utilizador pedir a aplicação para gerar o cálculo de uma nova alimentação para manada.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Manada Selecionada.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "nova alimentação da Manada". 2. O Sistema mostra a Manada com as respetivas Categorias de Estado Produtivo e Numero de Animais de cada Categoria e as características alimentares de cada categoria, e disponibiliza o formulário "Nova Alimentação da Manada". 3. O Utilizador faz Correções a Categorias dos Estados Produtivos: Gestação e Aleitamento e pede para gerar calculo. 4. O Sistema efetua o cálculo e disponibiliza-o ao Utilizador. 5. O utilizador confirma (Botão guardar). 6. O sistema atualiza as alimentações anteriores da manada Data Fim = data atual, e o sistema Guarda a nova alimentação data fim = vazio.
Fluxos Alternativos:	A qualquer momento o sistema cancela se o utilizador pressionar o botão "Cancelar". 5a. Se o utilizador pretender fazer um novo cálculo da alimentação volta ao ponto 3. 5b. O sistema cancela se o utilizador pressionar o botão "Cancelar".
Suplementos:	Testar se o Sistema atualiza a Data Fim = data atual das alimentações anteriores daquela Manada.
Pós-Condições:	Não têm.

Tabela 4.6: Descrição Caso de Uso: Gerar Cálculo de Nova Alimentação da Respetiva Manada.

Novo Caso Sanidade Animal:

A tabela (4.7) descreve com detalhe o caso de uso Novo Caso Sanidade Animal.

Nome:	Novo Caso Sanidade Animal.
Descrição:	O objetivo é o Utilizador poder Criar um novo caso de sanidade animal para um Animal.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Não têm.
Fluxo Principal:	1. O caso de uso começa quando o Utilizador seleciona a opção "Novo Caso de Sanidade". 2. O Sistema disponibiliza o formulário "Nova Caso de Sanidade Animal". 3. O Utilizador Introduz: IdBovino, Doença, Doente = sim , Data de Inicio. 4. O Sistema pede para Confirmar. 5. O utilizador confirma (Botão guardar). 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 utilizador não preencher todos os campos. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se quando um novo caso de sanidade é criado se o animal fica considerado atualmente doente.
Pós-Condições:	Não têm.

Tabela 4.7: Descrição Caso de Uso: Novo Caso Sanidade Animal.

Nova Intervenção do Veterinário:

A tabela (4.8) descreve com detalhe o caso de uso Nova Intervenção do Veterinário.

Nome:	Nova Intervenção do Veterinário.
Descrição:	O objetivo é o utilizador inserir e descrever uma intervenção do veterinário num determinado caso de sanidade de um animal.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Existir um Caso de Sanidade Selecionado para um animal.
Fluxo Principal:	1. Utilizador seleciona a opção "Nova Intervenção Veterinária". 2. O Sistema devolve o formulário "Nova Intervenção". 3. O utilizador Introduz: Veterinário, Intervenção, Data de início, Data de Fim. 4. O Sistema pede para Confirmar. 5. O utilizador confirma (Botão guardar). 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 utilizador não preencher todos os campos. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se a Intervenção fica bem relacionada com o caso de sanidade a que pertence.
Pós-Condições:	Não têm.

Tabela 4.8: Descrição Caso de Uso: Nova Intervenção do Veterinário.

Criar Linhas de Tratamento:

A tabela (4.9) descreve com detalhe o caso de uso Criar Linhas de Tratamento.

Nome:	Criar Linhas de Tratamento.
Descrição:	O objetivo é o Utilizador inserir medicamentos prescritos pelo veterinário numa determinada Intervenção.
Atores Envolvidos:	Gestor da Exploração.
Pré Condições:	Existir uma Intervenção Veterinária selecionada num determinado caso de Sanidade de um animal.
Fluxo Principal:	1. Utilizador seleciona a opção "Nova Medicação". 2. O Sistema devolve o formulário "Nova Medicação". 3. O utilizador Introduz: Medicamento, Data de início tratamento, Data de Fim tratamento. 4. O Sistema pede para Confirmar. 5. O utilizador confirma (Botão guardar). 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 utilizador não preencher todos os campos. 5a. O sistema cancela quando o utilizador não confirma.
Suplementos:	Testar se a Medicação fica bem relacionada com a Intervenção a que pertence.
Pós-Condições:	Não tem.

Tabela 4.9: Descrição Caso de Uso: Criar Linhas de Tratamento.

4.5 Diagramas de Sequência

Os diagramas de Sequência são utilizados para representar casos de uso com o objetivo de modelar o fluxo de mensagens, eventos e ações entre objetos e componentes.

Neste caso vamos mostrar os diagramas de sequência dos casos de uso já descritos em cima.

Novo Alimento

O diagrama de sequência representado pela figura (4.3) descreve os eventos e ações com o sistema quando o gestor da exploração insere um novo alimento.

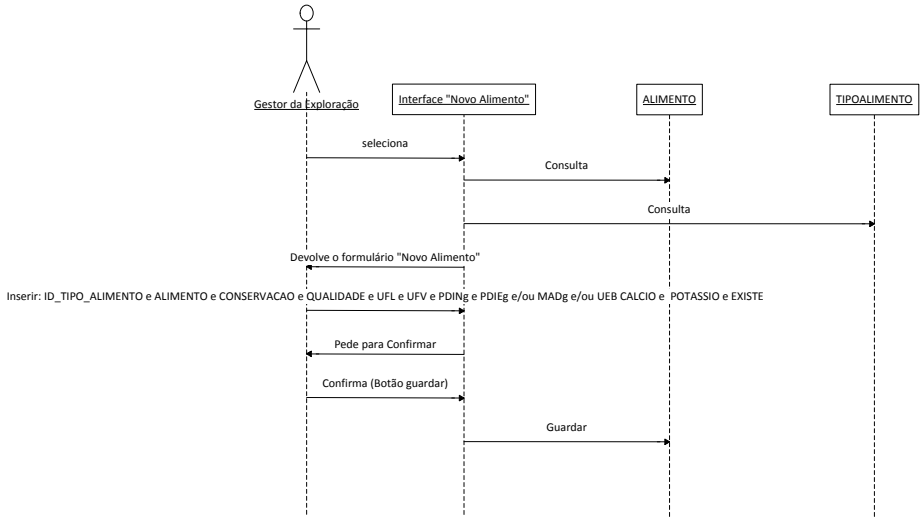

Figura 4.3: Diagrama de Sequência: Novo Alimento.

Editar Alimento

O diagrama de sequência representado pela figura (4.4) descreve os eventos e ações com o sistema quando o gestor da exploração edita um alimento.

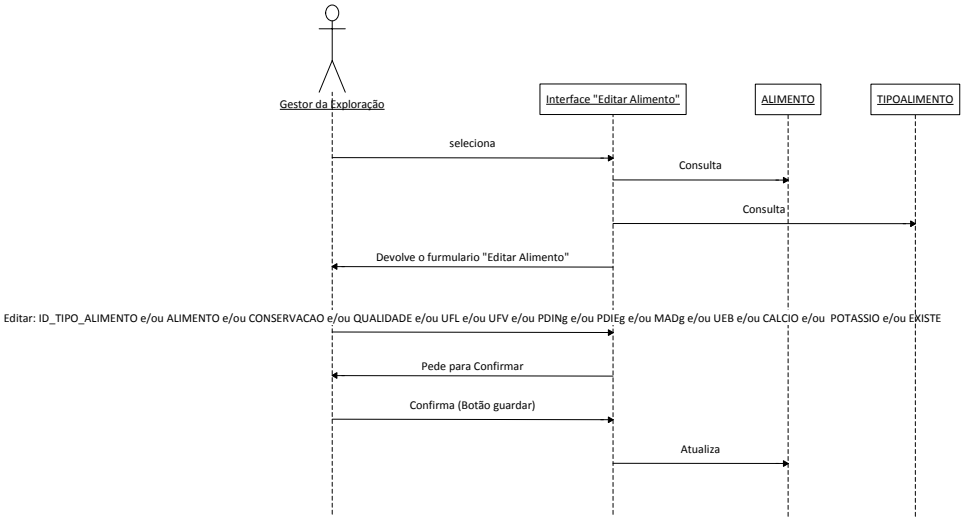

Figura 4.4: Diagrama de Sequência: Editar Alimento.

Eliminar Alimento

O diagrama de sequência representado pela figura (4.5) descreve os eventos e ações com o sistema quando o gestor da exploração elimina um alimento.

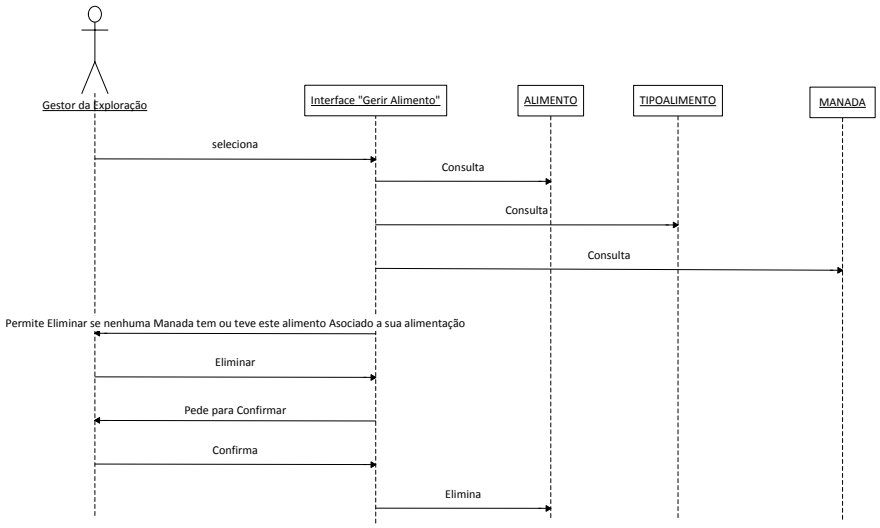

Figura 4.5: Diagrama de Sequência: Eliminar Alimento.

Pesquisar Alimento

O diagrama de sequência representado pela figura (4.6) descreve os eventos e ações com o sistema quando o gestor da exploração pesquisa um alimento.

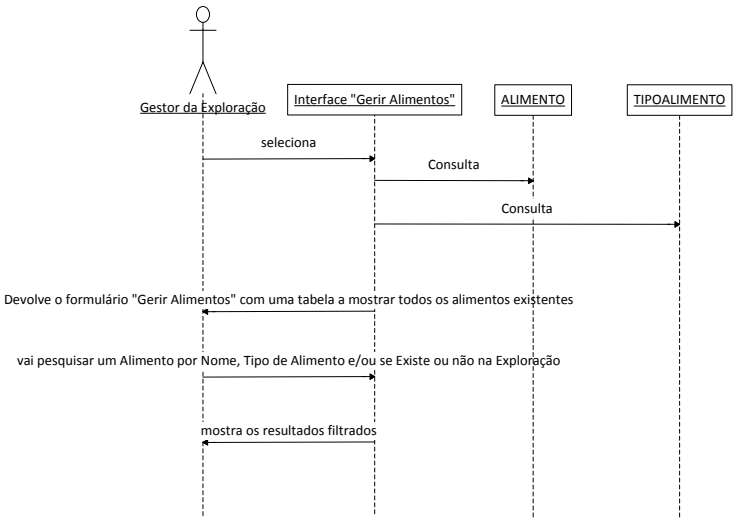

Figura 4.6: Diagrama de Sequência: Pesquisar Alimento.

Gerar Cálculo de Nova Alimentação da Respetiva Manada

O diagrama de sequência (4.7) descreve os eventos e ações com o sistema quando o gestor da exploração pede um cálculo de uma nova alimentação de uma manada.

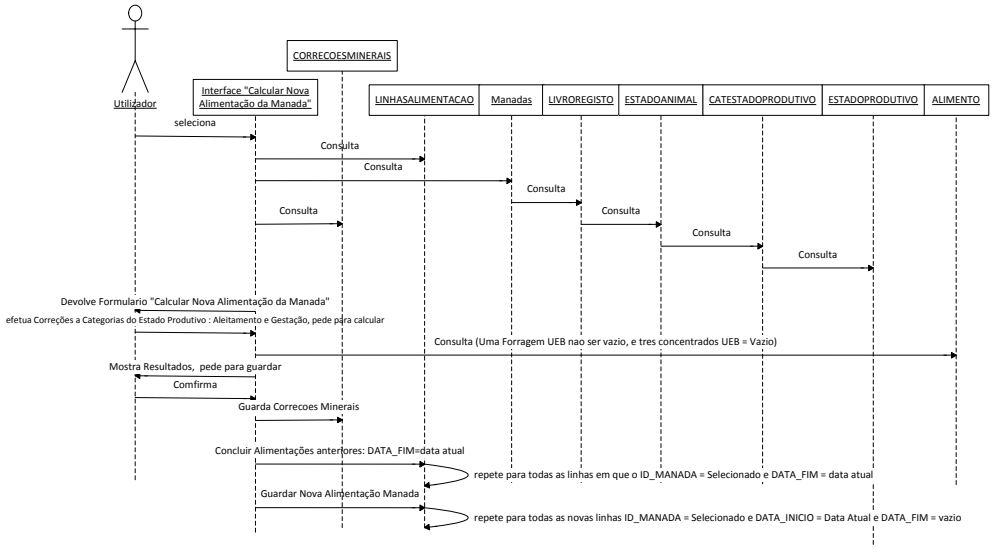

Figura 4.7: Diagrama de Sequência: Gerar Cálculo de Nova Alimentação da Respetiva Manada.

Novo Caso Sanidade Animal

O diagrama de sequência representado pela figura (4.8) descreve os eventos e ações com o sistema quando o gestor da exploração insere um novo caso de sanidade animal.

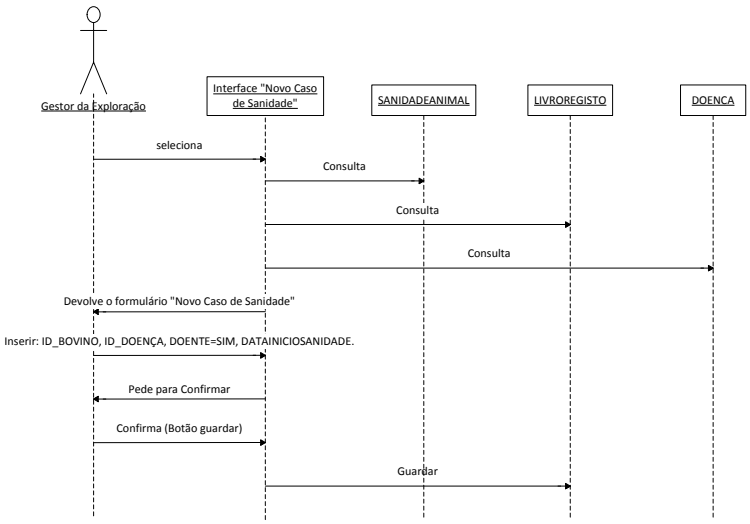

Figura 4.8: Diagrama de Sequência: Novo Caso Sanidade Animal.

Nova Intervenção do Veterinário

O diagrama de sequência representado pela figura (4.9) descreve os eventos e ações com o sistema quando o gestor da exploração insere uma nova intervenção do veterinário num determinado caso de sanidade animal.

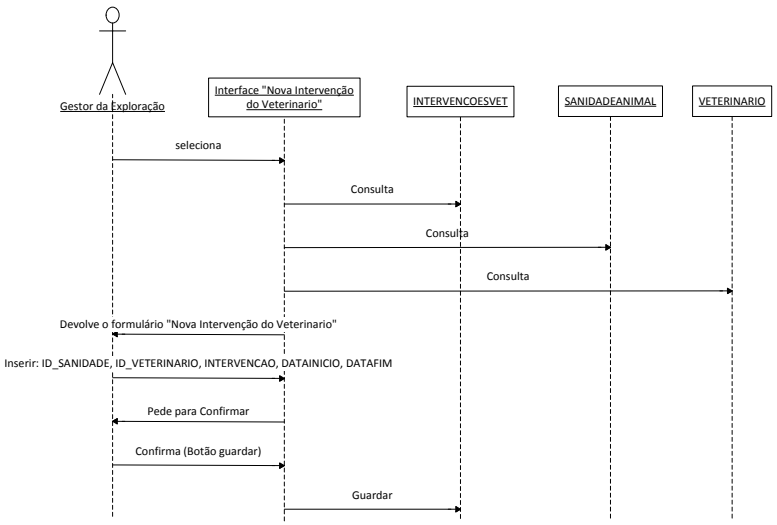

Figura 4.9: Diagrama de Sequência: Nova Intervenção do Veterinário.

Criar Linhas de Tratamento

O diagrama de sequência representado pela figura (4.10) descreve os eventos e ações com o sistema quando o gestor da exploração insere um novo medicamento prescrito pelo veterinário numa determinada intervenção.

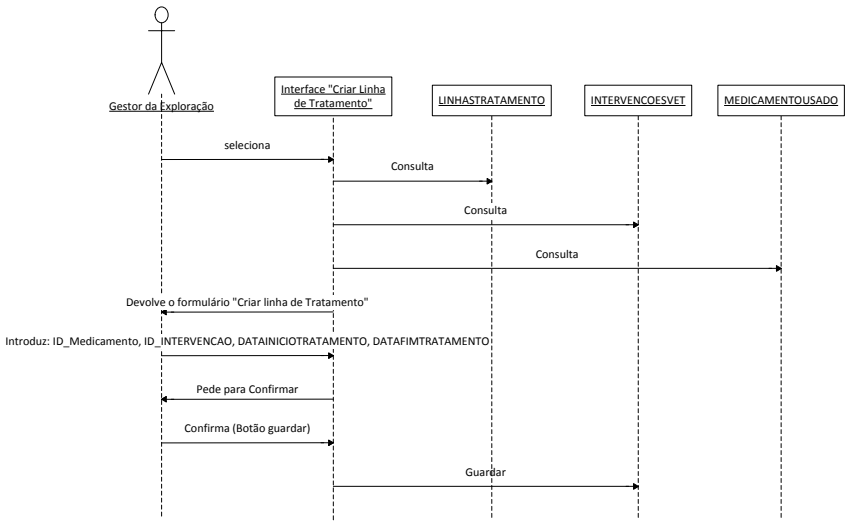

Figura 4.10: Diagrama de Sequência: Criar Linhas de Tratamento.

4.6 Diagrama de Classes

Aqui podemos ver o diagrama de Classes da nossa aplicação, representado pela figura (4.11) este diagrama mostra como as diferentes classes se relacionam entre si, agrupando especialidades e especializa comportamentos, cada classe é constituída pelo nome (o que representa a classe no mundo real), atributos (informação que deve ser analisada e/ou armazenada) e por fim as operações que representa o papel dos atores no sistema.

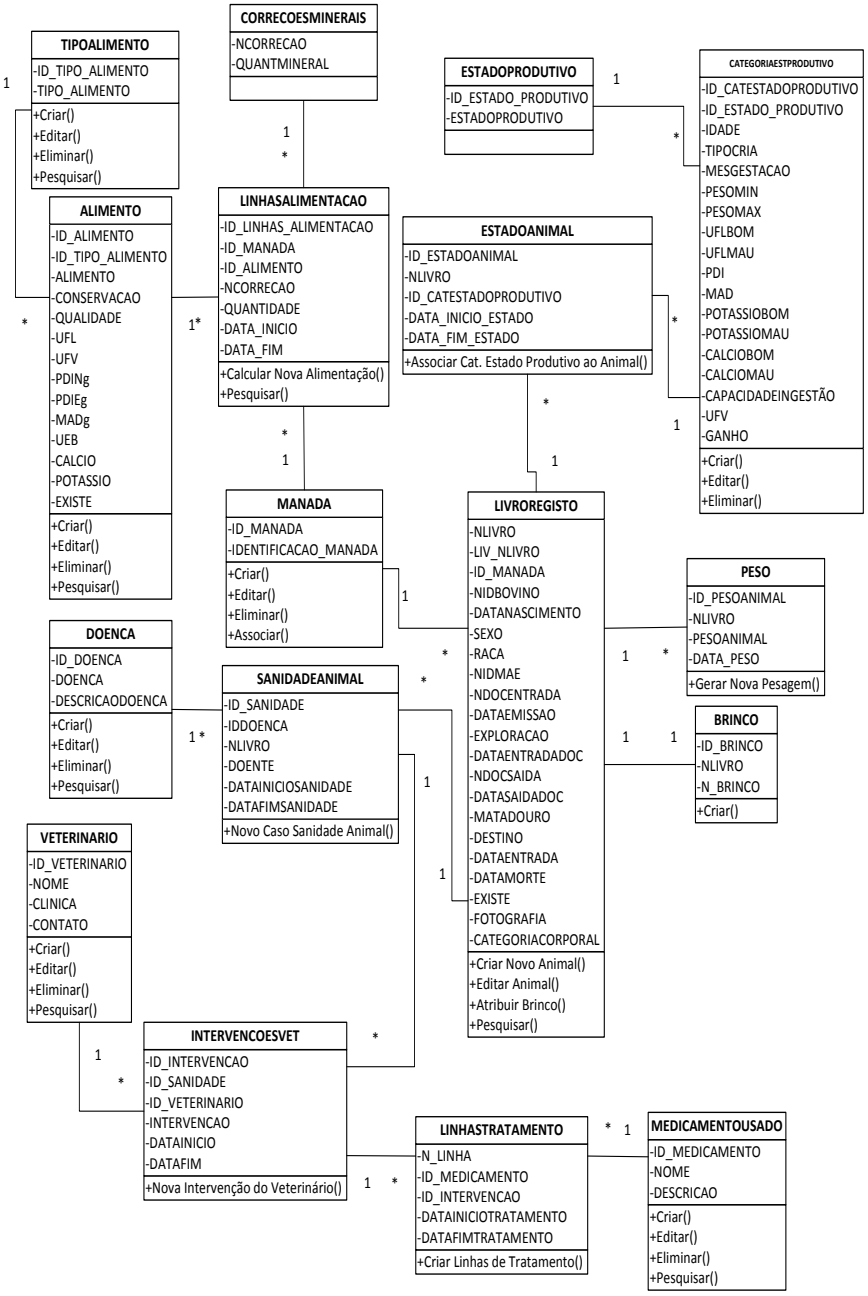

Figura 4.11: Diagrama de Classes.

4.7 Semântica de Classes

Esta secção tem como objetivo descrever as classes utilizadas

atributos - campos da classe;

Tipo de dados - valores que compõem o atributo;

Descrição - o que representa o atributo na classe;

Valores Validos - Apesar do tipo de dados este coluna tem como objetivo referenciar os valores validos no contexto em que os valores vão ser usados;

Formato - Representação do atributo, por exemplo se for data yyyy-MM-dd (Ano - Mês - dia);

Restrição - Como é tratado o atributo, se é tratado pelo sistema ou pelo utilizador;

Classe: CORRECOESMINERAIS - Têm como objetivo registar as correções minerais para cada a alimentação					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
NCORRECAO	Inteiro	Número sequencial que identifica a Correcao dos Minerais	Maior que 0	Até 10 dígitos	Gerado pelo sistema/Não alterável
QUANTMINERAL	float	Quantidade Mineral	Carateres	Valores numéricos separados por ponto	Preenchido pelo sistema

Tabela 4.10: Semântica Classe CORRECOESMINERAIS.

Classe: ALIMENTO - Têm como objetivo registar todos os alimentos bem como os seus constituintes nutritivos					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_ALIMENTO	Inteiro	Número sequencial que identifica o Alimento	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
ID_TIPO_ALIMENTO	Inteiro	Número que identifica o Tipo de Alimento	Maior que 0	Opção	Obrigatório
ALIMENTO	Texto	Nome do Alimento	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
CONSERVACAO	Texto	Tipo de conservação	(Verde, Palha, grão, Seco)	Opção	Obrigatório
QUALIDADE	Texto	Qualidade do Alimento (Bom, Mau, Muito Bom)	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
UFV	Texto	Componente Unidade Forrageira Carne (Viande) UFV	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
UFL	Texto	Componente Unidade Forrageira Leiteira UFL	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
PDIN	Texto	Componente Proteína Digestível no Intestino Permitida pelo Azoto do Alimento PDIN	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
PDIE	Texto	Componente Proteína Digestível no Intestino Permitida pela Energia do Alimento PDIE	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
MAD	Texto	Componente Matéria Azotada Digestível MAD	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
UEB	Texto	Componente Capacidade Ingestão do Alimento	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
CALCIO	Texto	Componente Cálcio	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
POTASSIO	Texto	Componente Potássio	Carateres	150 Carateres	Obrigatório/ Alterável/ Preenchido pelo Utilizador
EXISTE	Booleano	Existe na Exploração	Sim/Não	Sim/Não	Obrigatório/ Alterável/ Preenchido pelo Utilizador

Tabela 4.11: Semântica Classe ALIMENTO.

Algoritmo das operações:

Algoritmo 1 Algoritmo Classe ALIMENTO +Criar().

1: **procedure** +CRIAR()
▷ Operação que permite criar um Novo Alimento
2: O sistema gera o ID_TIPO_ALIMENTO.
3: Introduzir TIPO_ ALIMENTO e CONSERVACAO e QUALIDADE e UFL e UFV e PDING e PDIEG e/ou MADG e/ou UEB e CALCIO e POTASSIO e EXISTE
4: Se Campos todos preenchidos com a exceção do MADG e UEB
5: Criar Novo Alimento
6: Fim Se
7: **end procedure**

Algoritmo 2 Algoritmo Classe ALIMENTO +Editar().

1: **procedure** +EDITAR()
▷ Operação que permite Editar um Alimento
2: O sistema disponibiliza o Alimento onde o ID_ALIMENTO = ID_ALIMENTO Selecionado.
3: Editar TIPO_ALIMENTO e/ou CONSERVACAO e/ou QUALIDADE e/ou UFL e/ou UFV e/ou PDING e/ou PDIEG e/ou MADG e/ou UEB e/ou CALCIO e/ou POTASSIO e/ou EXISTE
4: Se (Campos todos preenchidos com a exceção do MADG e UEB)
5: Atualizar Alimento
6: Fim se
7: **end procedure**

Algoritmo 3 Algoritmo Classe ALIMENTO +Eliminar().

1: **procedure** +ELIMINAR()
▷ Operação que permite Eliminar um Alimento
2: Se (ID_ALIMENTO Selecionado Existir na Classe LINHASALIMENTACAO).
3: Devolver "Não pode Eliminar este Alimento".
4: Senão
5: Eliminar Alimento Selecionado
6: **end procedure**

Algoritmo 4 Algoritmo Classe ALIMENTO +Pesquisar().

1: **procedure** +PESQUISAR()
▷ Operação que permite pesquisar informação dos Alimentos
2: Introduzir meios pelo qual pretende pesquisar um Alimento por Nome, Tipo de Alimento e/ou se Existe ou não na Exploração.
3: Confirmar
4: **end procedure**

Classe: TIPOALIMENTO - Têm como objetivo registar os tipos de alimentos existentes ou necessários para qualificar os alimentos					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_TIPO_ALIMENTO	Inteiro	Número sequencial que identifica o Tipo de Alimento	Maior que 0	Até 10 dígitos	Gerado pelo sistema/Não alterável
TIPO_ALIMENTO	Texto	Nome do Tipo de Alimento	Carateres	150 Carateres	Preenchido pelo utilizador

Tabela 4.12: Semântica Classe TIPOALIMENTO.

Tendo em conta que as operações de gestão são muito semelhantes podem ver os exemplos da Classe ALIMENTO:

+Criar(): Ver Algoritmo 1.

+Editar(): Ver Algoritmo 2.

+Eliminar(): Ver Algoritmo 3.

+Pesquisar(): Ver Algoritmo 4.

Classe: LINHASALIMENTACAO - Têm como objetivo registar as varias linhas da alimentação calculada para uma manada					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_LINHAS_ALIMENTACAO	Inteiro	Número sequencial que identifica a Linha de Alimentação	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
ID_ALIMENTO	Inteiro	Número que identifica o Alimento	Maior que 0	Opção	Gerado pelo sistema / Não alterável
ID_MANADA	Inteiro	Número que identifica a Manada	Maior que 0	Opção	Não alterável
QUANTIDADE	float	Quantidade de Alimento necessária	Números separados por ponto	Valor com Duas casa decimais	Não alterável
DATA_INICIO	Data	Data de Inicio da Alimentação	Dígitos separados por (-)	yyyy-MM-dd	Data do sistema / Não alterável
DATA_FIM	Data	Data de Fim da Alimentação	Dígitos separados por (-)	yyyy-MM-dd	Data do sistema / Não alterável

Tabela 4.13: Semântica Classe LINHASALIMENTACAO.

Algoritmo das operações:

Algoritmo 5 Algoritmo Classe LINHASALIMENTACAO: +Gerar Cálculo de Nova Alimentação da Respetiva Manada().

```
1: procedure +Gerar Cálculo de Nova Alimentação da Respetiva Manada()
                ▷ Operação que permite criar o calculo de uma nova alimentação da Manada
2:     O Utilizador seleciona uma Manada
3:     if o utilizador efetuar correções em alguma categoria pertencente a Aleitamento ou Gestação ([3]) then
4:         Efetuar correção na categoria selecionada
5:     end if
6:     O sistema seleciona 1 forragem (Alimento com Capacidade de Ingestão (UEB)) e 3 Concentrados (Alimento sem Capacidade de Ingestão (UEB))
7:     for O Numero de categorias existentes na manada Fazer do
8:         Calcula a quantidade da forragem ([3])
9:         if Existir défice nas Componentes UFL, PDIN, PDIE then
10:            if défice nas 3 then
11:                Calcula a quantidade de mais 3 Concentrados ([3])
12:            end if
13:            if défice em 2 then
14:                Calcula a quantidade de mais 2 Concentrados ([3])
15:            end if
16:            if défice em 1 then
17:                Calcula a quantidade de mais 1 Concentrados ([3])
18:            end if
19:        end if
20:    end for
21:    if Existe Défice no cálcio e/ou no potássio then
22:        Calcula a necessidade de cálcio e / ou de Potássio
23:    end if
24:    if tornar a alimentação da manada Definitiva then
25:        Guardar Alimentação para a manada
26:    elseRecalcular
27:        volta a linha 3
28:    end if
29: end procedure
```

+Pesquisar(): Ver Algoritmo 4.

Classe: DOENCA - Têm como objetivo registar todas as doenças que já passaram ou existem na exploração					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_DOENCA	Inteiro	Número sequencial que identifica a Doença	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
DOENCA	Texto	Nome da Doença	Carateres	150 Carateres	Preenchido pelo utilizador
DESCRICAODOENCA	Texto	Descrição da Doença	Carateres	150 Carateres	Preenchido pelo utilizador

Tabela 4.14: Semântica Classe DOENCA.

Tendo em conta que as operações de gestão são muito semelhantes podem ver os exemplos da Classe ALIMENTO:

+Criar(): Ver Algoritmo 1.

+Editar(): Ver Algoritmo 2.

+Eliminar(): Ver Algoritmo 3.

+Pesquisar(): Ver Algoritmo 4.

Classe: MEDICAMENTOSUSADOS - Têm como objetivo registar todos os medicamentos usados na exploração					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_MEDICAMENTO	Inteiro	Número sequencial que identifica Medicamento	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
NOME	Texto	Nome do Medicamento	Carateres	150 Carateres	Preenchido pelo utilizador
DESCRICAO	Texto	Descrição do Medicamento	Carateres	150 Carateres	Preenchido pelo utilizador

Tabela 4.15: Semântica Classe MEDICAMENTOSUSADOS.

Tendo em conta que as operações de gestão são muito semelhantes podem ver os exemplos da Classe ALIMENTO:

+Criar(): Ver Algoritmo 1.

+Editar(): Ver Algoritmo 2.

+Eliminar(): Ver Algoritmo 3.

+Pesquisar(): Ver Algoritmo 4.

Classe: VETERINARIO - Têm como objetivo registar todos os Veterinários que já passaram pela exploração					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_VETERINARIO	Inteiro	Número sequencial que identifica o Veterinario	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
NOME	Texto	Nome do Veterinário	Carateres	150 Carateres	Preenchido pelo utilizador
CLINICA	Texto	Clinica onde o Veterinário exerce	Carateres	150 Carateres	Preenchido pelo utilizador
CONTATO	Texto	Contato do Veterinário	Carateres	9 Carateres numéricos	Preenchido pelo utilizador

Tabela 4.16: Semântica Classe VETERINARIO.

Tendo em conta que as operações de gestão são muito semelhantes podem ver os exemplos da Classe ALIMENTO:

+Criar(): Ver Algoritmo 1.

+Editar(): Ver Algoritmo 2.

+Eliminar(): Ver Algoritmo 3.

+Pesquisar(): Ver Algoritmo 4.

Classe: SANIDADEANIMAL - Têm como objetivo registar todos os casos de sanidade dos animais da exploração					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_SANIDADE	Inteiro	Número sequencial que identifica o Caso de Sanidade	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
ID_DOENCA	Inteiro	Numero que identifica a doença	Maior que 0	Até 10 dígitos	Selecionado pelo utilizador
NLIVRO	Inteiro	Numero que identifica o Animal	Maior que 0	Até 10 dígitos	Selecionado/ Introduzido pelo utilizador
DOENTE	Booleano	Marca o Estado da Sanidade de um animal Doente/Não Doente	Sim/Não	Sim/Não	Gerado pelo Sistema
DATAINICIOSANIDADE	Data	Data de Inicio da Sanidade	Dígitos separados por (-)	yyyy-MM-dd	Preenchido pelo utilizador
DATAFIMSANDIADE	Data	Data de Fim da Sanidade	Dígitos separados por (-)	yyyy-MM-dd	Preenchido pelo utilizador

Tabela 4.17: Semântica Classe SANIDADEANIMAL.

Algoritmo 6 Algoritmo Classe SANIDADEANIMAL +Novo caso de Sanidade().

```
1: procedure +Novo caso de Sanidade()
                ▷ Operação que permite Criar uma novo caso de sanidade para um determinado animal
2:     Selecionar/ Introduzir o NIDBOVINO
3:     Selecionar a DOENCA
4:     Sistema inicia o Estado DOENTE = Sim
5:     Introduzir DATAINIOSANIDADE
6:     Confirmar (Guardar)
7:     Confirmar
8: end procedure
```

Classe: INTERVENCOESVET - Têm como objetivo registar todas as intervenções efetuadas por um veterinário num determinado caso de					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
ID_INTERVENCAO	Inteiro	Número sequencial que identifica a Intervenção veterinaria	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
ID_SANIDADE	Inteiro	Numero que identifica o caso de Sanidade	Maior que 0	Até 10 dígitos	Selecionado pelo Sistema
ID_VETERINARIO	Inteiro	Numero que identifica o Veterinário	Maior que 0	Até 10 dígitos	Selecionado pelo utilizador
INTERVENCAO	Texto	descreve a informação que foi passada pelo veterinário	Carateres	250 Carateres	Preenchido pelo utilizador
DATAINICIO	Data	Data de Inicio Intervenção	Dígitos separados por (-)	yyyy-MM-dd	Preenchido pelo utilizador
DATAFIM	Data	Data de Fim da Intervenção	Dígitos separados por (-)	yyyy-MM-dd	Preenchido pelo utilizador

Tabela 4.18: Semântica Classe INTERVENCOESVET.

Algoritmo 7 Algoritmo Classe INTERVENCOESVET +Nova Intervenção Veterinaria().

1: **procedure** +Nova Intervenção Veterinaria()
▷ Operação que permite Criar uma Intervenção Veterinária num determinado caso de Sanidade Animal
2: Selecionar o Caso de Sanidade ao qual se refere a Intervenção
3: Selecionar o VETERINARIO
4: Inserir DATAINICIO
5: Inserir DATAFIM
6: Confirmar (Guardar)
7: **end procedure**

Classe: LINHASTRATAMENTO - Têm como objetivo registar todos os tratamentos efetuados numa determinada intervenção					
Atributo	**Tipo de dados**	**Descrição**	**Valores Válidos**	**Formato**	**Restrição**
N_LINHA	Inteiro	Número sequencial que identifica a linha de tratamento	Maior que 0	Até 10 dígitos	Gerado pelo sistema / Não alterável
ID_MEDICAMENTO	Inteiro	Numero que identifica o Medicamento	Maior que 0	Até 10 dígitos	Selecionado pelo utilizador
ID_INTERVENCAO	Inteiro	Numero que identifica a Intervenção do Veterinário	Maior que 0	Até 10 dígitos	Selecionado pelo utilizador
DATAINICIOTRATAMENTO	Data	Data de Inicio de Tratamento	Dígitos separados por (-)	yyyy-MM-dd	Preenchido pelo utilizador
DATAFIMTRATAMENTO	Data	Data de Fim de Tratamento	Dígitos separados por (-)	yyyy-MM-dd	Preenchido pelo utilizador

Tabela 4.19: Semântica Classe LINHASTRATAMENTO.

Algoritmo 8 Algoritmo Classe LINHASTRATAMENTO +Criar Linha de Tratamento().

1: **procedure** +Criar Linha de Tratamento()
▷ Operação que permite criar uma nova linha de tratamento
2: Selecionar a Intervenção do Veterinário a qual a medicação está atribuída
3: Selecionar o MEDICAMENTO
4: Introduz DATAINICIOTRATAMENTO
5: Introduz DATAFIMTRATAMENTO
6: **end procedure**

4.8 Diagrama de Atividades

Este diagrama 4.12 representa os fluxos operacionais do sistema descrevendo assim de uma forma genérica e organizada as operações que constituem a aplicação.

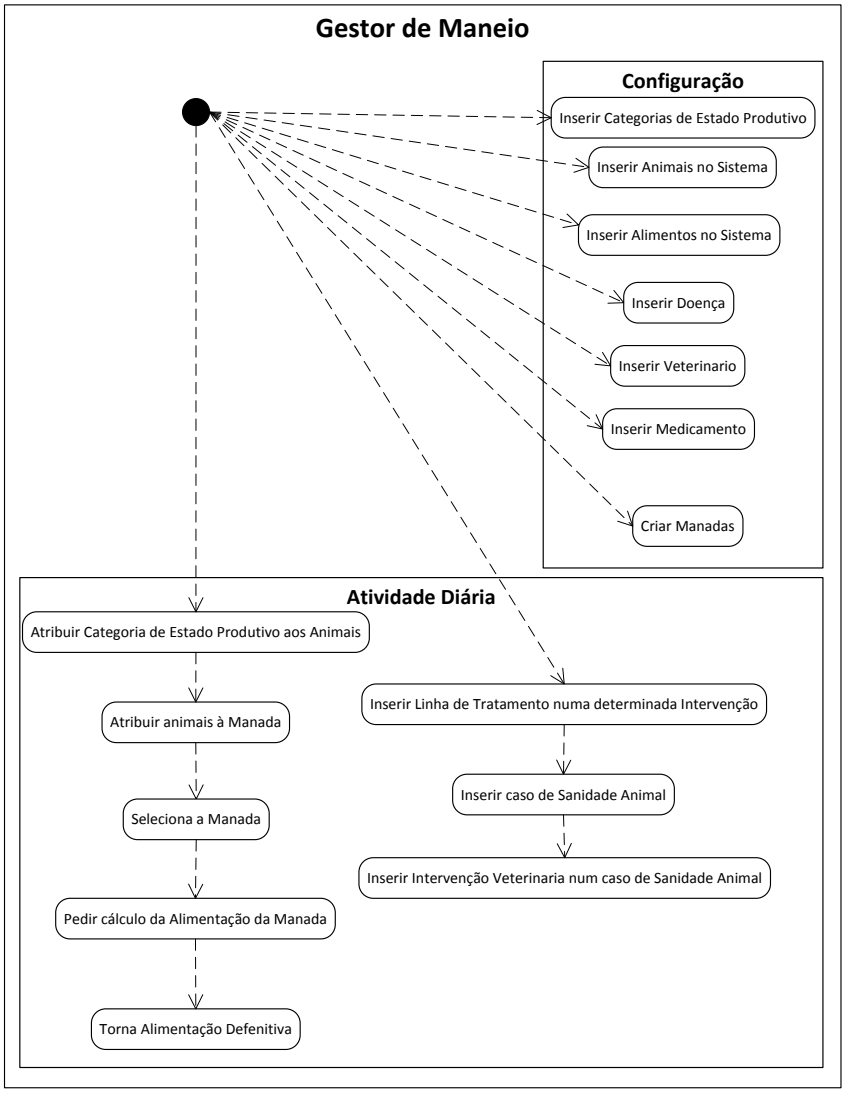

Figura 4.12: Diagrama de Atividades.

4.9 Diagrama de Estados

Este diagrama 4.13 mostra a transição de estado da sanidade do animal, em que inicialmente o estado de sanidade do animal é "doente", e quando a data final de doença for inserida obtém um estado final que passa a ser "não doente".

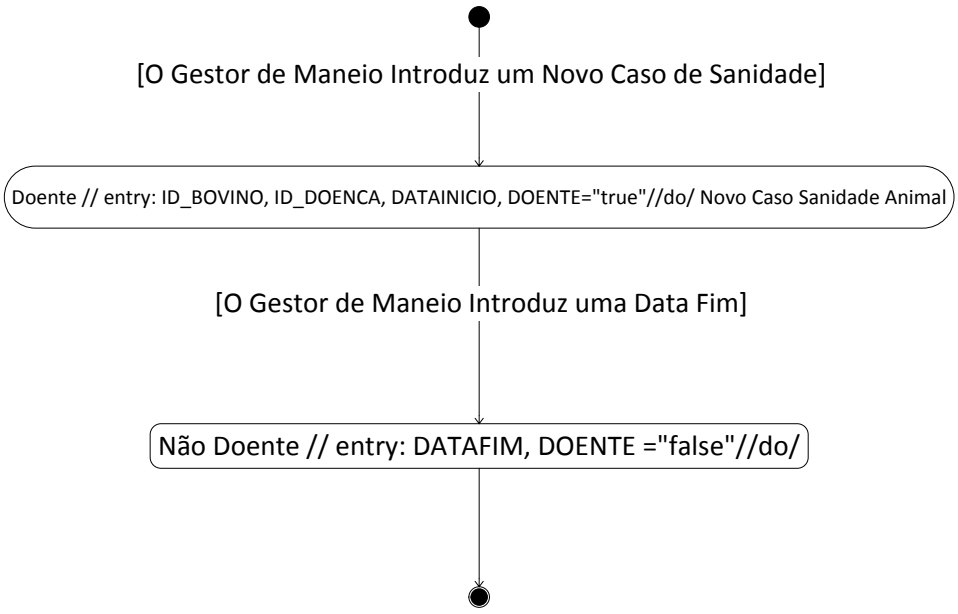

Figura 4.13: Diagrama de Estados.

4.10 Diagrama de Componentes

Este diagrama 4.14 descreve as componentes da aplicação, a seguir ao diagrama na tabela 4.20 podem ver a relação das componentes com as respetivas classes.

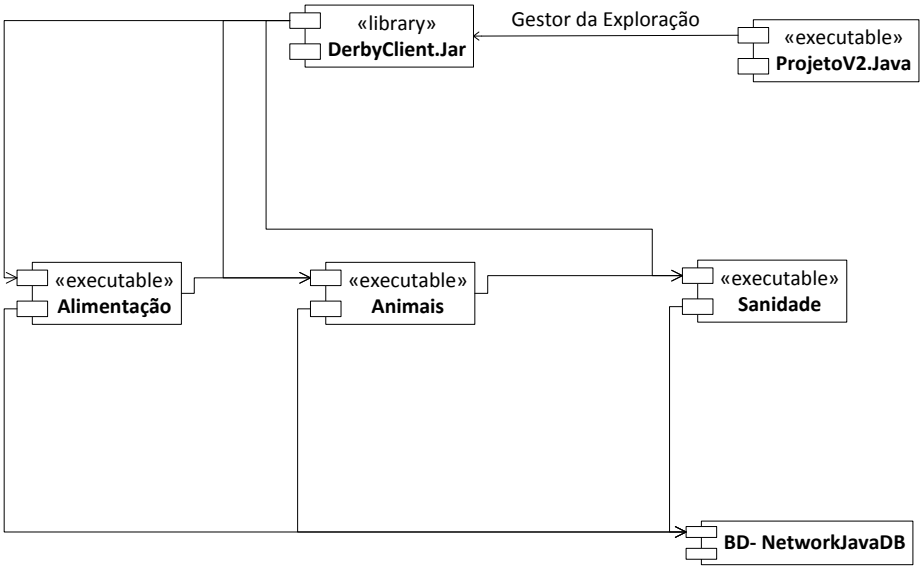

Figura 4.14: Diagrama de Componentes.

ProjetoV2.java – Esta é a componente de arranque da aplicação.

DerbyClient.jar – Biblioteca dedicada a base de dados derby, tem como objetivo permitir a ligação das componentes a base de dados.

BD - NetworkJavaDB – Representa a Base de dados desenvolvida em java DB.

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

Tabela 4.20: Relação das componentes com as classes

4.11 Diagrama de Instalação

Este diagrama 4.15 descreve as vertentes de software e de hardware do sistema.

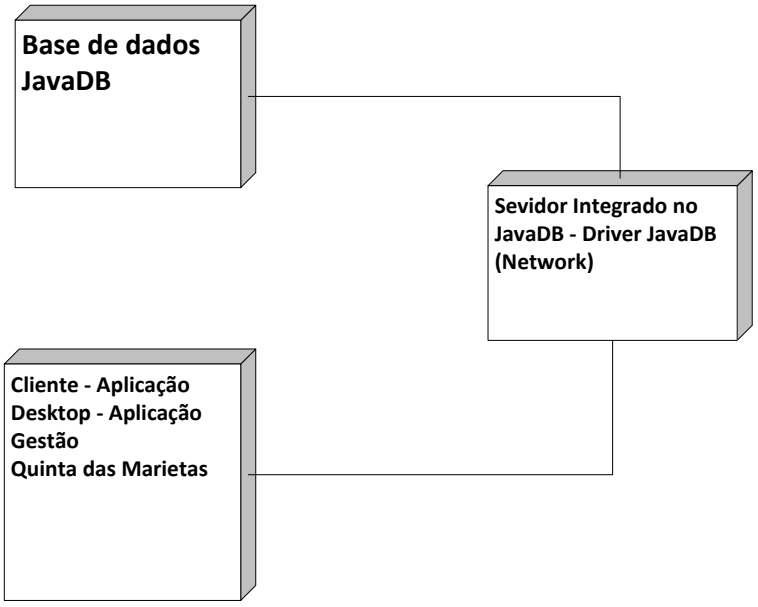

Figura 4.15: Diagrama de Instalação.

Capítulo 5

Implementação da solução

5.1 Introdução

Devido ao estudo que foi feito na análise de requisitos verificamos que tínhamos uma solução bem definida o que iria tornar a implementação mais fácil, só faltava mesmo tornar o estudo numa realidade e passar a implementação. Na implementação tentámos sempre procurar as formas mais eficazes de realizar o pretendido, e essa é a missão de um programador tornar o produto final o mais simples e fácil de utilizar e sem esquecer a rapidez de execução da aplicação.

5.2 Código Utilizado na aplicação

Neste capitulo vamos debruçar-nos essencialmente sobre o código mais relevante da aplicação da parte da Alimentação e da parte da Sanidade acompanhado dos Interfaces apropriados para a situação descrita.

Vamos mostrar pelo menos um exemplo de inserir, editar e eliminar e Pesquisar visto que o código é bastante semelhante em todos os aspetos de gestão da nossa aplicação.

Este figura 5.1 mostra o Menu "Gerir Alimentos":

Figura 5.1: Menu "Gerir Alimentos".

Quando se clica no "Novo Alimento"como podemos ver na figura 5.1 o sistema disponibiliza-nos o formulário representado pela figura 5.2

Esta figura 5.2 mostra o formulario "Novo Alimento":

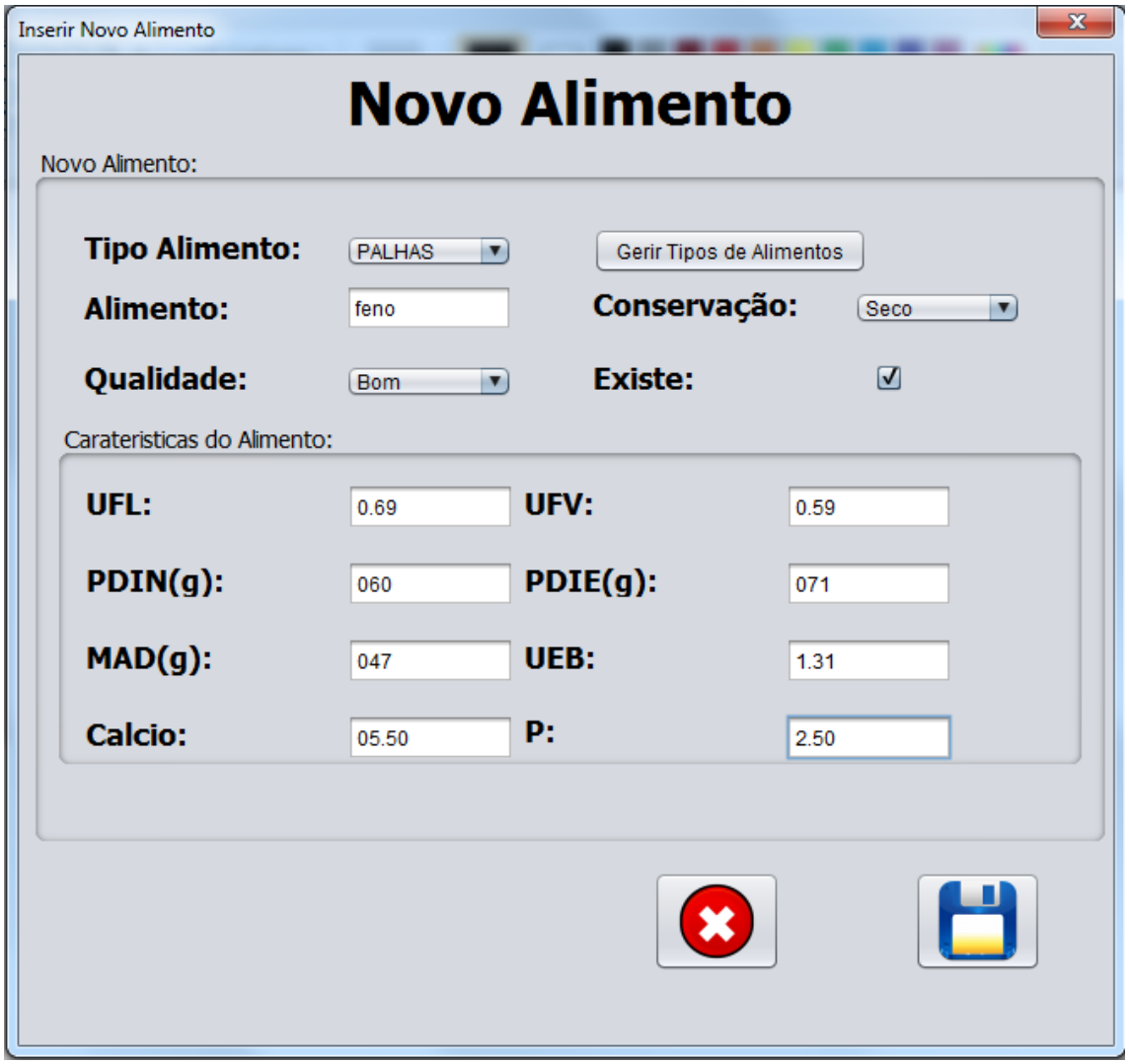

Figura 5.2: Formulário "Novo Alimento".

Neste algoritmo 9 pretendemos mostrar como é feita a inserção de um novo alimento na base de dados, ação reservada ao botão "Guardar"da figura 5.2 e servir de exemplo para todos os "Criar Novos existentes na Aplicação".

Algoritmo 9 Código de criação de Novo Alimento.

```
1:  procedure NovoAlimento
                                                    ▷ Verificar se todos os campos necessários estão preenchidos
2:      if (jTextFieldAlimento.getText().equals()
          || jFormattedTextFieldUFL.getText().equals(". ")
          || jFormattedTextFieldUFV.getText().equals(". ")
          || jFormattedTextFieldPDIN.getText().equals()
          || jFormattedTextFieldPDIE.getText().equals()
          || jFormattedTextFieldCB.getText().equals(". ")
          || jFormattedTextFieldP.getText().equals(". "))
                                                    ▷ Se verificar a condição acima retorna a seguinte mensagem
3:              jLabelMessageA.setText("Por favor preencha todos os campos corretamente");
4:      else
                                                    ▷ ir buscar todos os valores fornecidos pelo utilizador
5:              String vALIMENTO = jTextFieldAlimento.getText().toUpperCase();
6:              String vCONSERVACAO = jComboBoxConservacaoNovo.getSelectedItem().toString();
7:              String vQUALIDADE = jComboBoxQualidade.getSelectedItem().toString();
8:              String vUFL = jFormattedTextFieldUFL.getText();
9:              String vUFV = jFormattedTextFieldUFV.getText();
10:             String vPDIN_G_ = jFormattedTextFieldPDIN.getText();
11:             String vPDIE_G_ = jFormattedTextFieldPDIE.getText();
12:             String vMAD_G_ = jFormattedTextFieldMAD.getText();
13:             String vUEB = jFormattedTextFieldUEBAli.getText();
14:             String vCB = jFormattedTextFieldCB.getText();
15:             String vP = jFormattedTextFieldP.getText();
16:             boolean vEXISTE;
17:             if (jCheckBoxExiste.isSelected())
18:                     vEXISTE = true;
19:             else
20:                     vEXISTE = false;
21:
22:             InserirAlimento(); Algoritmo 10
                                                    ▷ Eliminar os campos preenchidos (repor valores para nova insereção)
23:             jTextFieldAlimento.setText();
24:             jFormattedTextFieldUFL.setText(". ");
25:             jFormattedTextFieldUFV.setText(". ");
26:             jFormattedTextFieldPDIN.setText(". ");
27:             jFormattedTextFieldPDIE.setText(". ");
28:             jFormattedTextFieldMAD.setText(". ");
29:             jFormattedTextFieldCB.setText(". ");
30:             jFormattedTextFieldP.setText(". ");
31:
32: end procedure
```

Algoritmo 10 Código de Inserir de Novo Alimento.

```
1: procedure InserirAlimento
2:     try
3:         //Criar ligação com a base de dados
4:         Class.forName("org.apache.derby.jdbc.ClientDriver");
5:         Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527 /BDpF", "administrador", "administrador");
6:         Statement st = (Statement) con.createStatement();
7:         //Ir buscar o codigo do Tipo de alimento
8:         String sqlTipoAlimento = "SELECT ID_TIPO_ALIMENTO FROM TIPOALIMENTO WHERE TIPO_ALIMENTO = '"+ jComboBoxTipoAlimento.getSelectedItem().toString() + "'";
9:         PreparedStatement Pst2 = (PreparedStatement) con.prepareStatement(sqlTipoAlimento);
10:        ResultSet rs2 = Pst2.executeQuery();
11:        while (rs2.next())
12:            vTipo_Alimento = rs2.getInt("ID_TIPO_ALIMENTO");
13:
14:        //Inserir novo alimento na tabela alimentos
15:        String sql = "INSERT INTO ALIMENTOS (ID_TIPO_ALIMENTO, ALIMENTO, CONSERVACAO, QUALIDADE, UFL, UFV, PDIN_G_, PDIE_G_, MAD_G_ , UEB, Calcio, Potassio, EXISTE) VALUES ("+ vTipo_Alimento + ",'"+ vALIMENTO + "','"+ vCONSERVACAO + "','"+ vQUALIDADE + "','"+ vUFL + "','"+ vUFV + "','"+ vPDIN_G_ + "','"+ vPDIE_G_ + "','"+ vMAD_G_ + "','"+ vUEB + "','"+ vCB + "','"+ vP + "','"+ vEXISTE + "')";
16:        st.executeUpdate(sql);
17:        jLabelMessageA.setText("Novo Alimento Inserido: "+ vALIMENTO);
18:        //fecha as ligações
19:        st.close();
20:        con.close();
21:    catch (Exception e)
22:        jLabelMessageA.setText(e.toString());
23:
24: end procedure
```

Esta figura 5.3 mostra o formulário "Editar Alimento":

Figura 5.3: Formulário "Editar Alimento".

Neste algoritmo 12 pretendemos mostrar como é feita a atualização de um determinado alimento da base de dados, e servir de exemplo para todos os "Editar existentes na Aplicação".

Este processo é constituído por duas vertentes que são elas:

Quando o Utilizador seleciona um alimento no "Gerir Alimentos"o sistema prepara o formulário com os valores do alimento selecionado:

Algoritmo 11 Código para Preparar Formulario "Editar Alimento".

```
1: procedure PrepararEditarAlimento
2:     DefaultTableModel modelo = (DefaultTableModel) jTableGerirAlimentos.getModel();
3:     //valor da linha selccionado
4:     int linha = jTableGerirAlimentos.getSelectedRow();
5:     //valor que esta na linha selecionada na coluna alimentos
6:     String vAlimentoS = (String) modelo.getValueAt(linha, 0);
7:     try
8:         //Criar ligação com a base de dados
9:         Class.forName("org.apache.derby.jdbc.ClientDriver");
10:        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/BDpF", "administrador", "administrador");
11:        String verifica = "select * from ALIMENTOS where ALIMENTO = '"+ vAlimentoS + "'";
12:        PreparedStatement pst = (PreparedStatement) con.prepareStatement(verifica);
13:        ResultSet rs = pst.executeQuery();
14:        while (rs.next())
15:            jTextFieldAlimentoEdit.setText(rs.getString("ALIMENTO"));
16:            //Tipo Alimento
17:            String sqlTA = "select TIPO_ALIMENTO from TIPOALIMENTO where ID_TIPO_ALIMENTO = "+ rs.getString("ID_TIPO_ALIMENTO");
18:            PreparedStatement pstTA = (PreparedStatement) con.prepareStatement(sqlTA);
19:            ResultSet rsTA = pstTA.executeQuery();
20:            String TipoAl=;
21:            while (rsTA.next())
22:                TipoAl = rsTA.getString("TIPO_ALIMENTO");
23:
24:            //fim
25:            jComboBoxTipoAlimentoEditar.setSelectedItem(TipoAl);
26:            jComboBoxQualidadeEdit.setSelectedItem(rs.getString("QUALIDADE"));
27:            jComboBoxConservacaoEdit.setSelectedItem(rs.getString("CONSERVACAO"));
28:            jFormattedTextFieldUFLEdit.setText(rs.getString("UFL"));
29:            jFormattedTextFieldUFVEdit.setText(rs.getString("UFV"));
30:            jFormattedTextFieldPDINEdit.setText(rs.getString("PDIN_G_"));
31:            jFormattedTextFieldPDIEEdit.setText(rs.getString("PDIE_G_"));
32:            jFormattedTextFieldMADEdit.setText(rs.getString("MAD_G_"));
33:            jFormattedTextFieldUEBAliEdit.setText(rs.getString("UEB"));
34:            jFormattedTextFieldCBEdit.setText(rs.getString("CALCIO"));
35:            jFormattedTextFieldPEdit.setText(rs.getString("POTASSIO"));
36:            jCheckBoxExisteEdit.setSelected(rs.getBoolean("EXISTE"));
37:
38:    catch (Exception e)
39:        jLabelMessageA1.setText(e.toString());
40:
41: end procedure
```

Após o utilizador editar o Alimento e confirmar (Botão Guardar):

Algoritmo 12 Código para Editar um Alimento.

```
1: procedure EditarAlimento
2:     if (jTextFieldAlimento.getText().equals()
         || jFormattedTextFieldUFL.getText().equals(". ")
         || jFormattedTextFieldUFV.getText().equals(". ")
         || jFormattedTextFieldPDIN.getText().equals()
         || jFormattedTextFieldPDIE.getText().equals()
         || jFormattedTextFieldCB.getText().equals(". ")
         || jFormattedTextFieldP.getText().equals(". "))
                                                ▷ Se verificar a condição acima retorna a seguinte mensagem
3:         jLabelMessageA.setText("Por favor preencha todos os campos corretamente");
4:     else
                                                ▷ ir buscar todos os valores fornecidos pelo utilizador
5:         String vALIMENTO = jTextFieldAlimento.getText().toUpperCase();
6:         String vCONSERVACAO = jComboBoxConservacaoNovo.getSelectedItem().toString();
7:         String vQUALIDADE = jComboBoxQualidade.getSelectedItem().toString();
8:         String vUFL = jFormattedTextFieldUFL.getText();
9:         String vUFV = jFormattedTextFieldUFV.getText();
10:        String vPDIN_G_ = jFormattedTextFieldPDIN.getText();
11:        String vPDIE_G_ = jFormattedTextFieldPDIE.getText();
12:        String vMAD_G_ = jFormattedTextFieldMAD.getText();
13:        String vUEB = jFormattedTextFieldUEBAli.getText();
14:        String vCB = jFormattedTextFieldCB.getText();
15:        String vP = jFormattedTextFieldP.getText();
16:        boolean vEXISTE;
17:        if (jCheckBoxExiste.isSelected())
18:            vEXISTE = true;
19:        else
20:            vEXISTE = false;
21:        AtualizarAlimento AA = new AtualizarAlimento(); //Algoritmo 13
22:        jLabelMessageA1.setText();
23:        jDialogEditarAlimento.setVisible(false);
24:
25: end procedure
```

Algoritmo 13 Código para Atualizar o Alimento na BD.

```
1: procedure EditarAlimento
2:     try
3:         Class.forName("org.apache.derby.jdbc.ClientDriver");
4:         Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/ BDpF", "administrador", "administrador");
5:         Statement st = (Statement) con.createStatement();
6:         //Tipo Alimento
7:         String sqlTA = "select ID_TIPO_ALIMENTO from TIPOALIMENTO where TIPO_ALIMENTO = '"+vTipo_Alimento+ "'";
8:         PreparedStatement pstTA = (PreparedStatement) con.prepareStatement(sqlTA);
9:         ResultSet rsTA = pstTA.executeQuery();
10:        String TipoAl = ;
11:        while (rsTA.next())
12:            TipoAl = rsTA.getString("ID_TIPO_ALIMENTO");
13:
14:    //fim
15:        String sql = "UPDATE ALIMENTOS SET ID_TIPO_ALIMENTO = "+TipoAl+", ALIMENTO = '"+ vALIMENTO + "', CONSERVACAO='"+ vCONSERVACAO + "', QUALIDADE='"+ vQUALIDADE + "',UFL='"+ vUFL + "',UFV='"+ vUFV + "',PDIN_G_='"+ vPDIN_G_ + "',PDIE_G_='"+ vPDIE_G_ + "',MAD_G_='"+ vMAD_G_ + "',UEB='"+ vUEB + "',CALCIO='"+ vCB + "', POTASSIO='"+ vP + "', EXISTE='"+ vEXISTE + "' WHERE ALIMENTO = '"+ vAlimentoS + "'";
16:        st.executeUpdate(sql);
17:        jLabelgerirAerro.setText("Alimento Atualizado "+ vALIMENTO);
18:        st.close();
19:        con.close();
20:    catch (Exception e)
21:        jLabelMessageA1.setText(e.toString());
22:
23: end procedure
```

Em seguida mostraremos como funciona o processo de eliminar um alimento (exemplo para todos os eliminar), o eliminar têm dois processos muito importantes, um deles é o facto de só disponibiliza o botão "Eliminar Alimento"se este não estiver ou teve inserido em nenhuma alimentação de nenhuma manada como mostra o algoritmo 14, e se se verificar esta condição só assim permite eliminar o alimento como mostra o algoritmo 15.

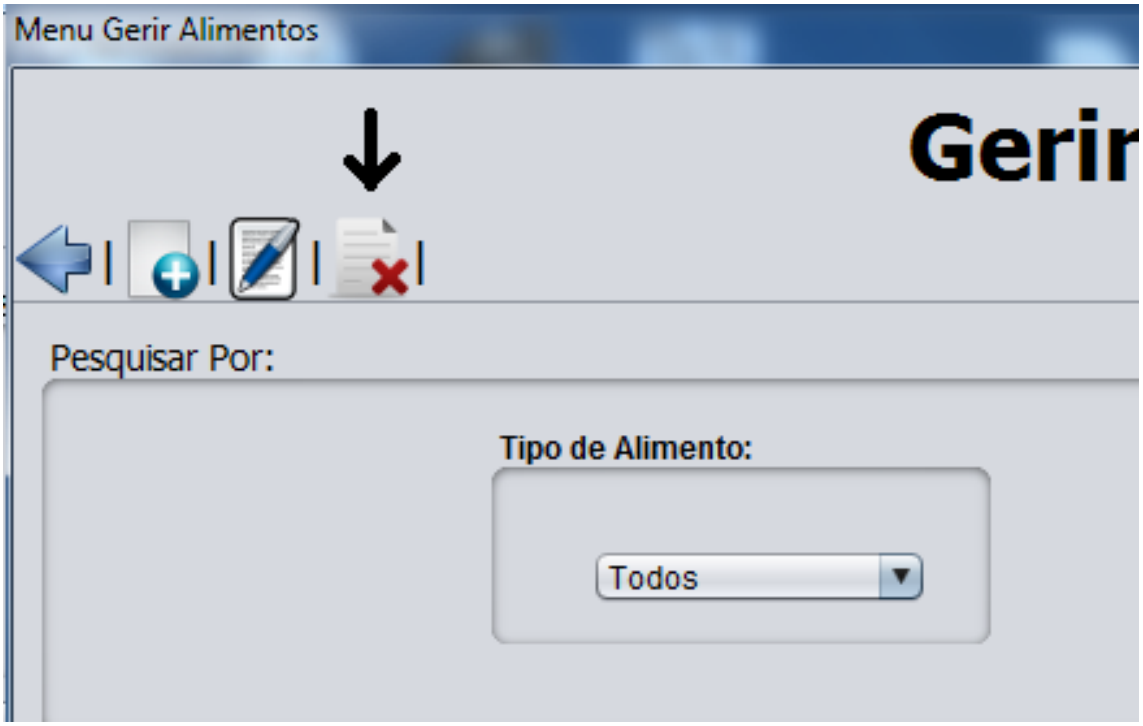

Figura 5.4: Formulário "Eliminar Alimento".

Algoritmo 14 Código Disponibilizar botão "Eliminar um Alimento".

```
1: procedure ELIMINARALIMENTO
2:     private void jTableGerirAlimentosMouseReleased(java.awt.event.MouseEvent evt)
3:         jButtonEditarEditarAlimento.setEnabled(true);
4:         DefaultTableModel modelo = (DefaultTableModel) jTableGerirAlimentos.getModel();
5:         int linha = jTableGerirAlimentos.getSelectedRow();
6:         //valor que esta na coluna da linha selecionada
7:         String AlimentoS = (String) modelo.getValueAt(linha, 0);
8:         //Verifica se é possivel eliminar ou não o registo
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 ID_ALIMENTO from ALIMENTOS WHERE ALIMENTO = '"+ AlimentoS + "'";
13:            PreparedStatement st = (PreparedStatement) con.prepareStatement(sql);
14:            ResultSet rs = st.executeQuery();
15:            if (rs.next())
16:                String sql1 = "select * from LINHASALIMENTACAO WHERE ID_ALIMENTO = "+ rs.getString("ID_ALIMENTO");
17:                PreparedStatement st1 = (PreparedStatement) con.prepareStatement(sql1);
18:                ResultSet rs1 = st1.executeQuery();
19:                if (rs1.next())
20:                    jButtonGerirEliminarAliment.setEnabled(false);
21:                else
22:                    jButtonGerirEliminarAliment.setEnabled(true);
23:
24:
25:        catch (Exception e)
26:            System.out.println( + e.toString());
27:
28:
29: end procedure
```

Algoritmo 15 Código para Eliminar um Alimento.

```
1: procedure ELIMINARALIMENTO
2:     DefaultTableModel modelo = (DefaultTableModel) jTableGerirAlimentos.getModel();
3:     int linha = jTableGerirAlimentos.getSelectedRow();
4:     //valor que esta na coluna da linha selecionada
5:     String AlimentoS = (String) modelo.getValueAt(linha, 0);
6:     try
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 sql = "DELETE from ALIMENTOS where ALIMENTO = '"+AlimentoS+"'";
11:        st.executeUpdate(sql);
12:        st.close();
13:        con.close();
14:    catch (Exception e)
15:
16: end procedure
```

A figura 5.5 têm como objetivo mostrar o menu "Gestão da Alimentação", e destaca essencialmente o factor que só quando uma manada está selecionada como podemos ver, é que disponibiliza o Botão "Nova Alimentação":

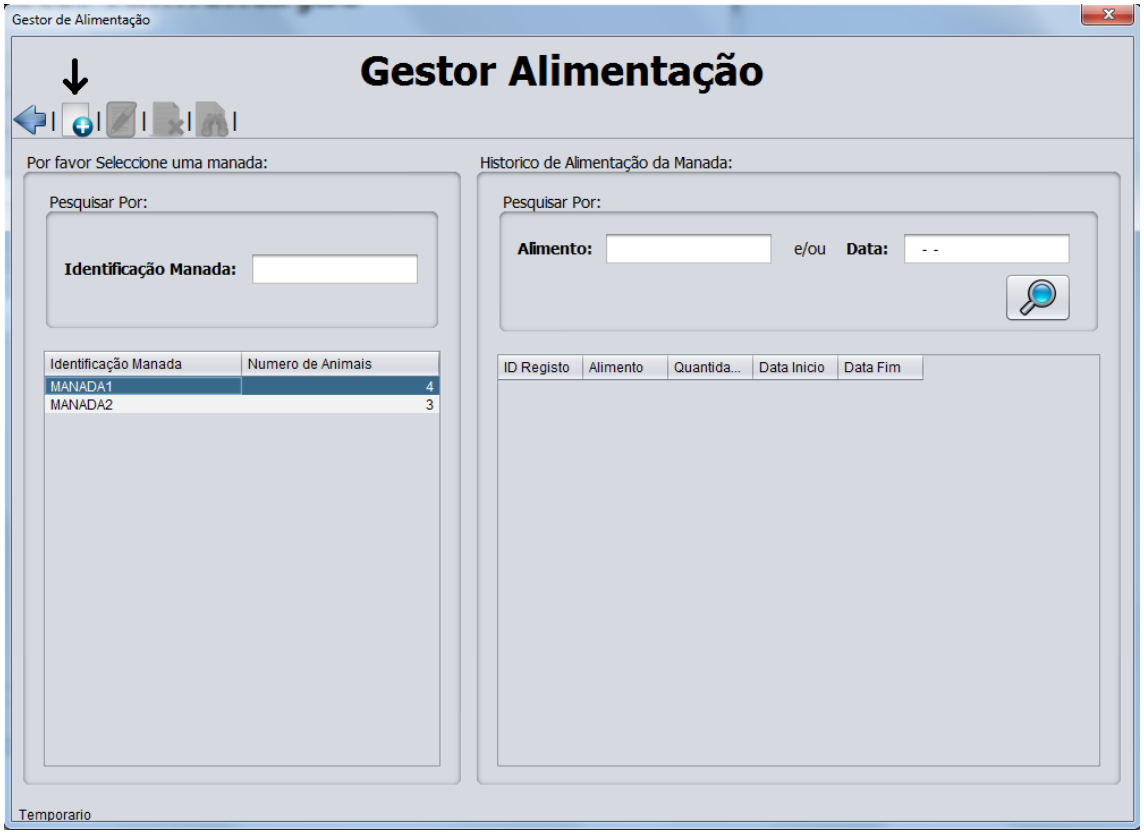

Figura 5.5: Menu "Gerir Alimentação".

Após clicar na botão "Nova Alimentação"da figura 5.5 é disponibilizado o formulário "Nova Alimentação", representado pela figura 5.6.

Figura 5.6: Formulário "Nova Alimentação".

Este algoritmo 16, tem como objetivo mostrar como o sistema seleciona todas as categorias de estado produtivo existentes na manada selecionada pelo utilizador bem como o numero de animais respetivos a cada uma, e preenche a jTableCategoriasManada (Primeira tabela do menu da Figura 5.6) .

Algoritmo 16 Código para Preencher jTabel Categorias da Manada.

```
1: procedure PreencherTabelaCategoriasManada
2:     Classe para Preparar formulário "Nova Alimentação"
3:     DefaultTableModel ModeloTabelaDadosEstProd = (DefaultTableModel) jTableDadosEst-
   Prod.getModel();
4:     int NRegisto = 1;
5:     int NAnimaisCat = 0;
6:     String vNLIVRO2 = ;
7:     //String NAnimais2 = 1;
8:     // numero do livro do animal que esta na manada
9:     String vNLIVRO = ;
10:    //Verificar as categorias dos animais que estao nas manadas
11:    String vCategoriAnimal = ;
12:    //Para cada categoria diferente vai buscar as necessidades que os animais precisão
13:    String vNomeCat = ;
14:    String vPesoMin = ;
15:    String vPesoMax = ;
16:    String vUFLouUFV = ;
17:    String vPDI = ;
18:    String vMAD = ;
19:    String vPotassio = ;
20:    String vCalcio = ;
21:    String vCapacidadeIngestão = ;
22:    try
23:        Class.forName("org.apache.derby.jdbc.ClientDriver");
24:        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527 /BDpF", "adminis-
   trador", "administrador");
25:        String sql = "select ID_MANADA from MANADAS where IDENTIFICACAOMANADA= '"+
   ManadaS + "'";
26:        PreparedStatement st = (PreparedStatement) con.prepareStatement(sql);
27:        ResultSet rs = st.executeQuery();
28:        while (rs.next())
29:            int id_Manada = rs.getInt("ID_MANADA");
30:            //Ir buscar o numero de animais
31:            String sqlANimais = "select NLIVRO from LIVROREGISTO where ID_MANADA = "+
   id_Manada;
32:            PreparedStatement pstANimais = (PreparedStatement) con.prepareStatement(sqlANimais);
33:            ResultSet rs2 = pstANimais.executeQuery();
34:            while (rs2.next())
35:                vNLIVRO = rs2.getString("NLIVRO");
36:                //Ir buscar a categoria de est prod
37:                String sqlCat = "select ID_CATESTADOPRODUTIVO from ESTADOANIMAL
   where NLIVRO = "+ vNLIVRO + "and DATA_FIMESTADO is null";
38:                PreparedStatement pstCat = (PreparedStatement) con.prepareStatement(sqlCat);
39:                ResultSet rs3 = pstCat.executeQuery();
40:                while (rs3.next())
41:                    vCategoriAnimal = rs3.getString("ID_CATESTADOPRODUTIVO");
42:                    //Ir buscar o numero de animais
43:                    String NAnimais = "select NLIVRO from ESTADOANIMAL where
   ID_CATESTADOPRODUTIVO = "+ vCategoriAnimal;
44:                    PreparedStatement NANi = (PreparedStatement)
   con.prepareStatement(NAnimais);
45:                    ResultSet rs5 = NANi.executeQuery();
46:                    while (rs5.next())
47:                        vNLIVRO2 = rs5.getString("NLIVRO");
48:                        //Ir buscar o numero de animais
49:                        String NAnimais2 = "select * from LIVROREGISTO where NLIVRO =
   "+vNLIVRO2+"and ID_MANADA = "+ id_Manada;
50:                        PreparedStatement NANi2 = (PreparedStatement)
   con.prepareStatement(NAnimais2);
51:                        ResultSet rs6 = NANi2.executeQuery();
52:                        while (rs6.next())
53:                            NAnimaisCat++;
54:
55:
56:    //Continua do Algoritmo 17
57: end procedure
```

Algoritmo 17 Código para Preencher jTabel Categorias da Manada.

```
1: procedure PreencherTabelaCategoriasManadaContinuacao
2:     //Continuação do Algoritmo// 16
3:                         //Ir buscar as necessidades da categoria de est prod
4:                         String sqlNecesCat = "select * from CATEGORIAESTPRODUTIVO where
   ID_CATESTADOPRODUTIVO = "+ vCategoriAnimal;
5:                         PreparedStatement pstNecesCat = (PreparedStatement)
   con.prepareStatement(sqlNecesCat);
6:                         ResultSet rs4 = pstNecesCat.executeQuery();
7:                         while (rs4.next())
8:                             vNomeCat = rs4.getString("NOME_CATEGORIA");
9:                             vPesoMin = rs4.getString("PESOMIN");
10:                            vPesoMax = rs4.getString("PESOMAX");
11:                            vUFLouUFV = rs4.getString("UFLBOM");
12:                            vPDI = rs4.getString("PDI");
13:                            vMAD = rs4.getString("MAD");
14:                            vPotassio = rs4.getString("POTASSIOBOM");
15:                            vCalcio = rs4.getString("CALCIOBOM");
16:                            vCapacidadeIngestão = rs4.getString("CAPACIDADEINGESTAO");
17:                            tamanho = ModeloTabelaDadosEstProd.getRowCount();
18:                            for (int i = 0; i < tamanho; i++)
19:                                int linha = jTableDadosEstProd.getSelectedRow();
20:                                //valor que esta na linha selecionada na coluna Doenca
21:                                String vNomecatS = (String) ModeloTabelaDadosEst-
   Prod.getValueAt(i, 2);
22:                                if (vNomecatS.equals(vNomeCat))
23:                                    ModeloTabelaDadosEstProd.removeRow(i);
24:                                    tamanho = ModeloTabelaDadosEstProd.getRowCount();
25:
26:
27:                            ModeloTabelaDadosEstProd.addRow(new Object[]"Registo "+ NRegisto,
   NAnimaisCat, vNomeCat, vUFLouUFV, vPDI, vMAD, vPotassio, vCalcio, vCapacidadeIngestão, false);
28:                            NRegisto++;
29:                            NAnimaisCat=0;
30:
31:
32:
33:
34:     catch (Exception e)
35:         System.out.println( + (e.toString()));
36:
37: end procedure
```

Relativamente ao método de calculo utilizado, a pedido do Sr João Pedro Ribeiro, foi nos pedido algum cuidado em não relatar pormenorizadamente todos os passos utilizados, por isso disponibiliza-mos o algoritmo 5, apresentado na semântica das classes, disponibilizamos também um fluxograma do calculo da nova alimentação da manada, que tem como objetivo explicar melhor como funciona, e qual o método usado para o cálculo de arraçoamento da alimentação para manada.

Fluxograma de calculo:

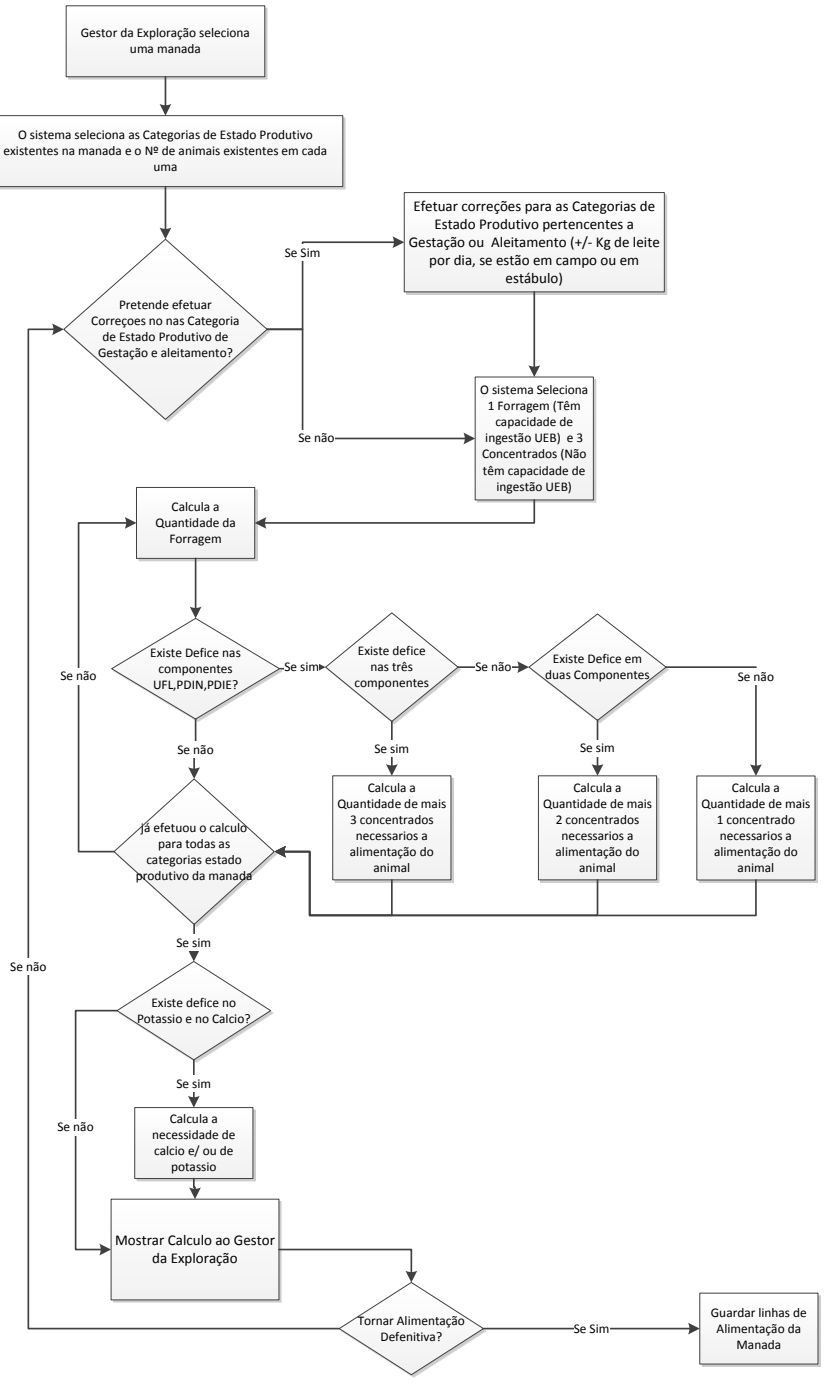

Figura 5.7: Fluxograma de Calculo.

Todos os "cálculos"referidos no fluxograma, assim como as "correções efetuadas"as categorias de estado produtivo, têm como referencia o livro "Alimentation des Bouvins"([3]).

5.3 Base de dados

A figura 5.8 representa o modelo físico da nossa aplicação.

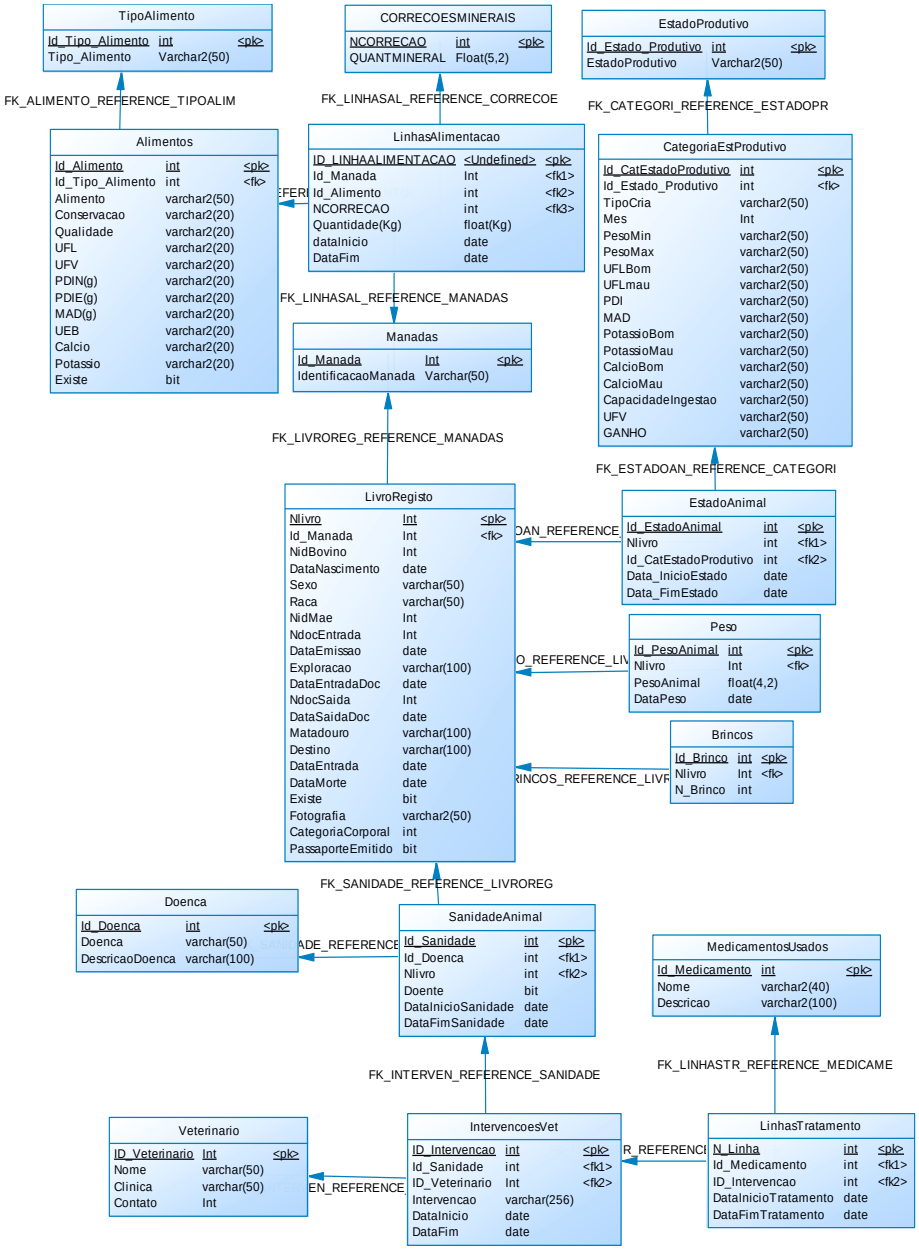

Figura 5.8: Modelo físico da base de dados.

Foi através do modelo físico representado pela figura (5.8) que implementamos a base de dados da nossa aplicação, a descrição dos atributos das tabelas pode ser observado com mais pormenor na analise de requisitos na secção semântica de classes (4.7).

5.4 Testes

A fase de Testes da nossa aplicação foi essencialmente um teste fiel descrito em cada tabela na linha suplementos no capitulo Analise de Requisitos (4) , no sub capitulo Descrição dos casos de uso (4.4), onde fazemos referencia aos casos de teste essenciais para á nossa aplicação.

Um dos casos de teste que também foi feito á nossa aplicação foi, criar um caso de utilização do quotidiano do gestor da exploração, com a sua presença, e fazer uma reprodução fiel á utilização da aplicação.

Sem esquecer também os testes efetuados a integridade dos campos, se respeitava ou não a sua formatação.

Capítulo 6

Conclusões e trabalho futuro

6.1 Conclusões

Inicialmente encontramos algumas dificuldades em perceber a dinâmica de uma exploração agrícola. No inicio parecia simples e fácil conseguir interligar tudo, mas com o avançar do estudo verificamos que tinha conceitos muito complexos. Tudo se resolveu com uma visita à exploração e ver no terreno do que se tratava afinal, e a partir dai começamos a efetuar um estudo sobre o problema mais preciso e concreto.

Outro dos problemas com que nos deparamos foi o método de como se efetua o calculo da Alimentação para a Manada. Pois é um método bastante complexo.

Este projeto foi de extrema importância para mim como pessoa assim como na minha vida profissional, o facto de estar a desenvolver uma aplicação para um "cliente", foi ótimo pois é uma situação totalmente diferente de desenvolver uma aplicação totalmente pensada e elaborada por nós, o "cliente"deu-nos os critérios para efetuar aplicação estivemos sempre sujeitos a aprovação do cliente, o que nos motivou em tentar dar sempre o nosso melhor e obter os resultados esperados, para mim como pessoa foi importante pois nunca é demais aprender e além dos inúmeros conhecimentos aprofundados sobre a área da programação mais especificamente em java e SQL, foi sem duvida muito interessante ficar por dentro do conceito de Gerir uma Exploração Agrícola de criação de gado bovino.

6.2 Trabalho Futuro

Como trabalho futuro vamos elaborar primeiro que tudo uma secção de "Gestão Financeira", uma vez que o objetivo primário de qualquer negócio é o seu desenvolvimento e o lucro que se obtém através dele de modo a ser competitiva uma exploração agrícola não pode descurar os gastos subjacentes à mesma tais como os gastos com a maquinaria utilizada, a compra de alimentos que não se encontram disponíveis na exploração, os ordenados dos trabalhadores, etc. assim sendo um programa onde seja possível registar todas as despesas para saber os gastos que se tem com a mesma e poder analisar onde se está a gastar muito e onde se pode poupar é uma mais-valia para qualquer gestor e empresário.

Na parte da Sanidade Animal estão a ser configuradas algumas mudanças. Está em estudo criar um "upgrade"da aplicação de modo que ao final de algum tempo de utilização da aplicação seja possível de algum modo automatizar um sistema de profilaxia de doenças encontradas nos animais através dos sintomas encontrados, o gestor da exploração poder inserir todos os sintomas visíveis no animal, e quando a aplicação filtrar as doenças através dos sintomas, se o resultado for apenas um, poder mostrar qual a medicação prescrita por um veterinário em outro caso que já tenha decorrido.

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 Quinta das Marietas

André Martins Gonçalves
Hugo Filipe de Pina Jorge
Mestre José Quitério Figueiredo
Escola Superior De Tecnologia e Gestão
Instituto Politécncico da Guarda
Av. Dr. Francisco Sá Carneiro, 50 – 6300 Guarda
andre.mg@live.com.pt

***Resumo* — Este projeto tem como objetivo o desenvolvimento de uma aplicação de gestão de uma exploração agrícola de criação de gado bovino, foca-se essencialmente nos tópicos: animais, alimentação dos animais e sanidade animal, e a aplicação é desenvolvida na linguagem de programação Java e a base de dados desenvolvida em java DB.**

1. Introdução

Este Artigo tem como objetivo descrever o funcionamento e a estrutura da aplicação, nas vertentes alimentação dos animais e sanidade animal, serão descritos alguns passos a tomar para a utilização da aplicação assim como serão também exibidos algumas das janelas mais importantes da aplicação.

2. Estrutura do Aplicação

O objetivo geral das vertentes que vamos descrever neste artigo é:

- Para a alimentação, facilitar ao utilizador todo o processo de cálculo da alimentação da manada, pois se o utilizador fizer uma gestão cautelosa dos alimentos e dos seus nutrientes também sem esquecer que os animais bem inseridos no seu estado produtivo e na manada certa, para o utilizador tudo se processa apenas com um clique.

- Para a sanidade Animal, esta vertente permite ao utilizador criar um histórico da situação de sanidade dos seus animais, através deste tópico o utilizador pode inserir todos os casos de sanidade animal existentes na sua exploração, assim como sempre que queira poder fazer uma pesquisa sobre qualquer situação de sanidade que tenha sido inserida na aplicação.

Funcionamento da aplicação e Janelas associadas

O funcionamento da aplicação é simples e intuitivo pois todos os botões e áreas de interesse contém legendas que dizem exatamente o que fazem e para onde vão, assim como todos os formulários se o utilizador colocar o rato em cima de qualquer zona de preenchimento pelo utilizador, aparece uma legenda que diz o que é para ser inserido nela assim como se existir alguma validação especial é dado um exemplo de como se pretende que seja preenchido.

A. Janela Inicial da Aplicação

Inicialmente é mostrado ao utilizador a janela exibida na **Ilustração 1**, que permite ao utilizador selecionar a área de gestão específica que pretende

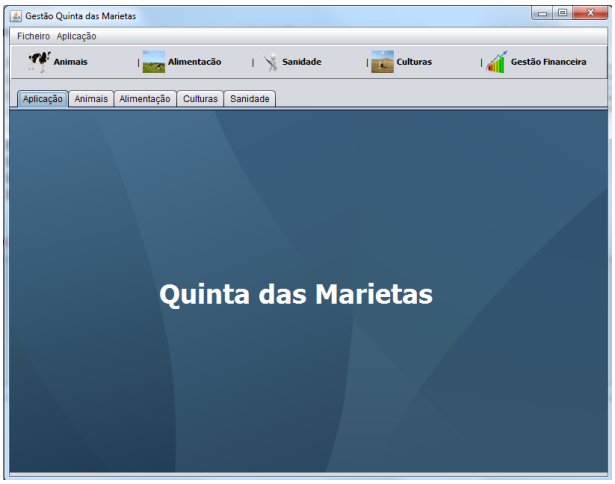

Ilustração 1 – Menu Inicial

B. Menu Alimentação

A **Ilustração 2** representa o menu relativo a gestão da alimentação.

Ilustração 1 – Menu Alimentação

C. Menu gerir tipo de alimento

Quando o utilizador escolhe "Gerir tipo de alimento", é mostrado o menu exibido na **Ilustração 2**.

Ilustração 3 - Gerir Tipo de Alimento

D. Novo Tipo de Alimento

Quando é pretendido pelo utilizador inserir um novo tipo de alimento, é exibido o formulário representado pela **Ilustração 4.**

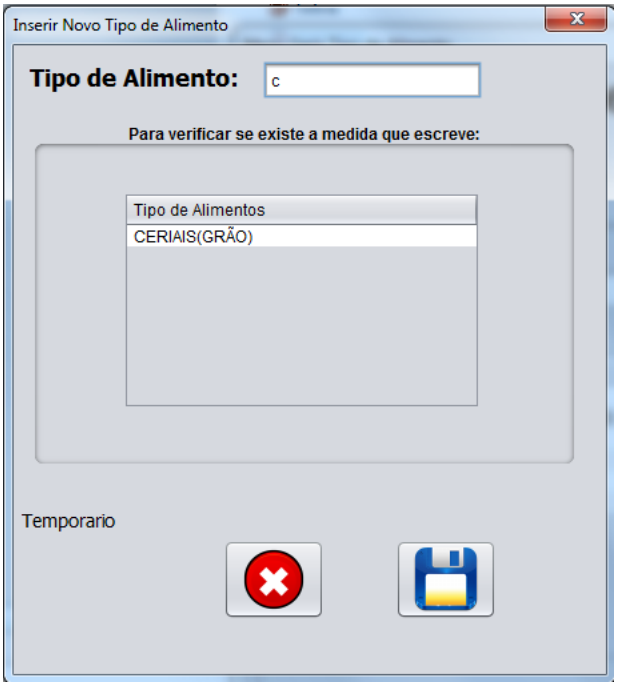

Ilustração 4 – Novo Tipo de Alimento

E. Editar Tipo de Alimento

Quando é pretendido pelo utilizador editar um tipo de alimento, é exibido o formulário representado pela **Ilustração 5,** só e possível editar um tipo de alimento se estiver algum selecionado no menu "gerir tipo de alimentos".

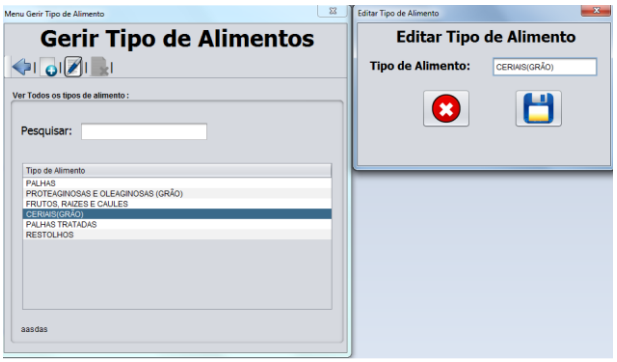

Ilustração 5 - Editar Tipo de Alimento

F. Eliminar Tipo de Alimento

Só é possível eliminar um tipo de alimento se estiver algum selecionado no menu "gerir tipo de alimentos" e se este não tiver nenhum alimento relacionado como mostra a **Ilustração 6**.

Ilustração 6 – Eliminar Tipo de Alimento

G. Gerir Alimentos

Quando o utilizador escolhe "Gerir Alimentos", é mostrado o menu exibido na **Ilustração 7**.

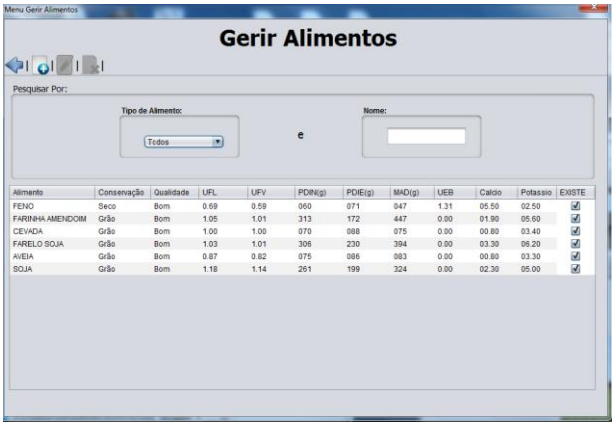

Ilustração 7 – Eliminar Tipo de Alimento

H. Novo Alimento

Quando é pretendido pelo utilizador inserir um novo alimento, é exibido o formulário representado pela **Ilustração 8.**

Ilustração 8 – Novo Alimento

I. Editar Alimento

Quando é pretendido pelo utilizador editar um alimento, é exibido o formulário representado pela **Ilustração 9,** só e possível editar um alimento se estiver algum selecionado no menu "gerir alimentos" na **Ilustração 7**.

Ilustração 9 – Editar Alimento

J. Eliminar Alimento

Só é possível eliminar um alimento se estiver algum selecionado no menu "gerir alimentos" e se este não tiver nenhuma manada tiver esse alimento inserido na alimentação como mostra a **Ilustração 10**.

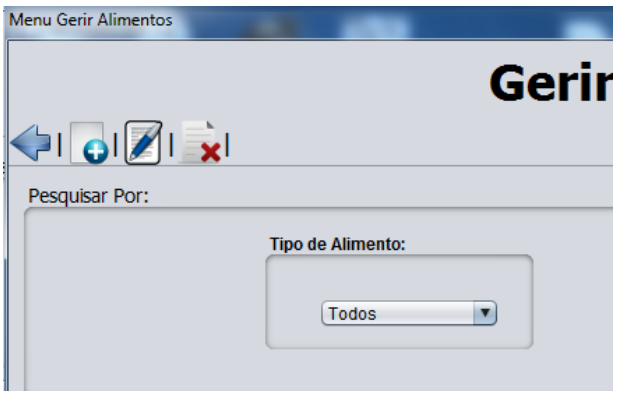

Ilustração 10 – Eliminar Alimento

K. Gerir Alimentação

Quando o utilizador escolhe "Gerir Alimentação", é mostrado o menu exibido na **Ilustração 11**.

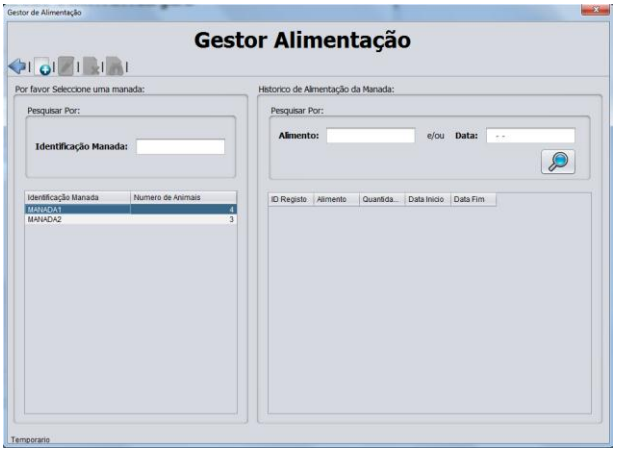

Ilustração 11 – Gerir Alimentação

L. Nova Alimentação

Quando é pretendido pelo utilizador criar uma nova alimentação para uma manada selecionada no menu "Gerir Alimentação" **Ilustração 11**, o sistema seleciona todas as categorias de estado produtivo e número de animais que pertence a cada categoria, da manada, é permitido ao utilizador efetuar correções nas categorias de estado produtivo que pertençam ao Aleitamento ou Gestação, e o utilizador pede o calculo da alimentação, apos lhe ser mostrado o resultado ele dispõem das operações guardar, cancelar, recalcular e para o caso de as correções não serem as pretendidas pode fazer "reset" a tabela que contem as categorias de estado produtivo.

Ilustração 12 – Nova Alimentação

M. Menu Sanidade

A **Ilustração 13** representa o menu relativo a gestão da sanidade animal.

Ilustração 13 – Menu Sanidade

N. Gerir Sanidade

Quando o utilizador escolhe "Gerir Sanidade", é mostrado o menu exibido na **Ilustração 14**.

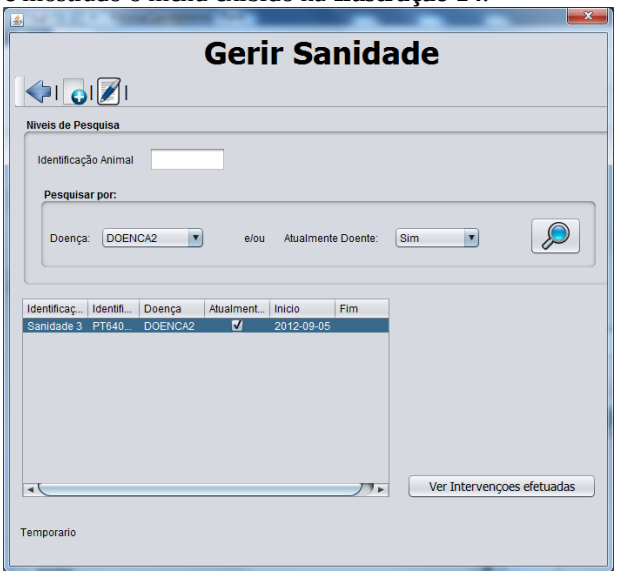

Ilustração 14 – Gerir Sanidade

O. Novo Caso de Sanidade

Quando é pretendido pelo utilizador inserir um novo caso de sanidade animal, é exibido o formulário representado pela **Ilustração 15.**

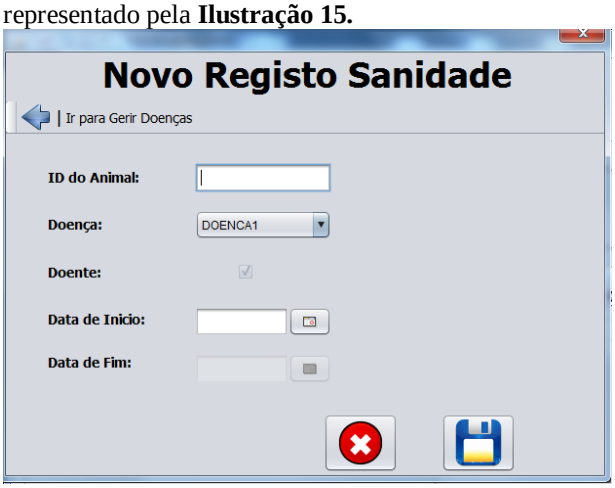

Ilustração 15 – Novo Caso de Sanidade

P. Editar Caso de Sanidade

Quando é pretendido pelo utilizador editar um caso de sanidade animal, é exibido o formulário representado pela **Ilustração 16,** só e possível editar um caso de sanidade se estiver algum selecionado no menu "Gerir Sanidade" na **Ilustração 14**.

Ilustração 16 – Editar Caso de Sanidade

Q. Eliminar Sanidade

O botão eliminar caso sanidade só é disponibilizado se o registo selecionado no menu "Gerir Sanidade" na **Ilustração 14** não tiver associado qualquer Intervenção veterinária.

R. Gerir Intervenção Veterinária

Após o utilizador se selecionar um caso de sanidade no menu "Gerir Sanidade" na Ilustração **14** e após isso clicar no botão "Ver Intervenções Veterinárias ", é exibido o menu "Gerir Intervenções" representado pela **Ilustração 17.**

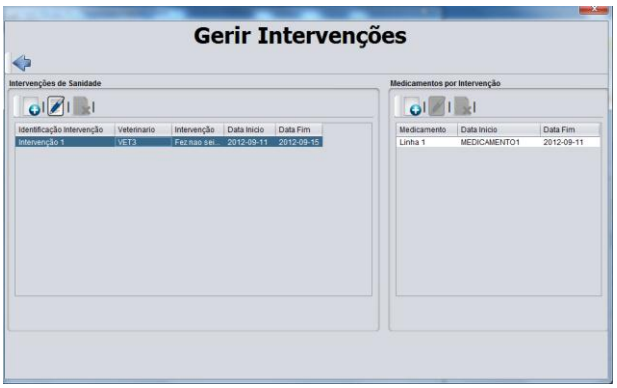

Ilustração 17 – Nova Intervenção Veterinária

S. Nova Intervenção Veterinária

Quando é pretendido pelo utilizador inserir uma nova intervenção do veterinário, é exibido o formulário representado pela **Ilustração 18.**

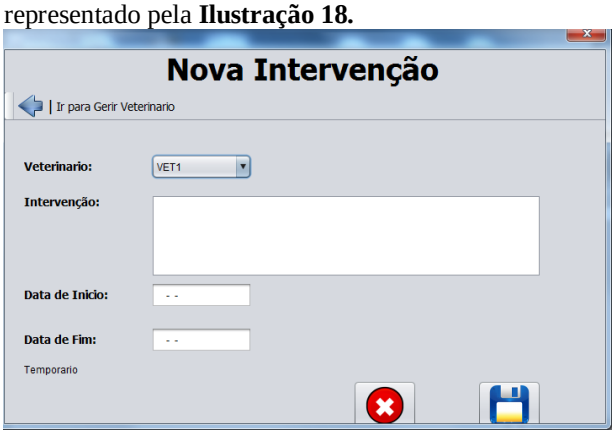

Ilustração 18 – Nova Intervenção Veterinária

T. Editar Intervenção Veterinária

Quando é pretendido pelo utilizador editar uma intervenção, é exibido o formulário semelhante ao representado pela **Ilustração 18,** só e possível editar uma intervenção se estiver algum selecionado no menu "Gerir Intervenção Veterinária" na **Ilustração 17**.

U. Eliminar Intervenção Veterinária

O botão eliminar intervenção só é disponibilizado se o registo selecionado no menu "Gerir Intervenção Veterinária" na **Ilustração 17** não tiver associado qualquer medicamento prescrito.

V. Nova Medicação para a Intervenção Veterinária

Quando é pretendido pelo utilizador inserir um novo medicamento prescrito por um veterinário numa determinada intervenção, é exibido o formulário representado pela **Ilustração 15.**

Ilustração 19 – Novo Alimento

W. Editar Medicação para a Intervenção Veterinária

Quando é pretendido pelo utilizador editar um medicamento prescrito por um veterinário numa determinada intervenção, é exibido o formulário semelhante ao representado pela **Ilustração 19,** só é possível editar um medicamento prescrito por um veterinário numa determinada intervenção se estiver algum selecionado no menu "Gerir Intervenção Veterinária" na **Ilustração 17** na secção de medicamentos por intervenção.

X. Eliminar medicação para a intervenção veterinária

O botão eliminar intervenção só é disponibilizado se existir um registo selecionado no menu "Gerir Intervenção Veterinária" na **Ilustração 17**.

Y. Gerir Doenças

Quando o utilizador escolhe "Gerir Doenças", é mostrado o menu exibido na **Ilustração 20**.

No caso de "Gerir Medicamentos" e "Gerir Veterinários" o processo e idêntico ao apresentado.

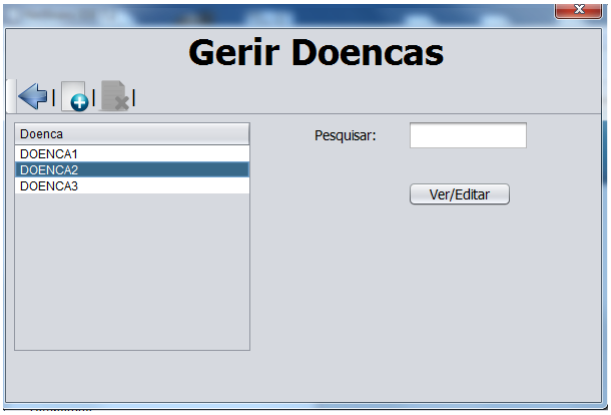

Ilustração 20 – Gerir Doenças

Z. Nova Doença

Quando é pretendido pelo utilizador inserir uma nova doença, é exibido o formulário representado pela **Ilustração 21.**
No caso de "Gerir Medicamentos" e "Gerir Veterinários" o processo e idêntico ao apresentado.

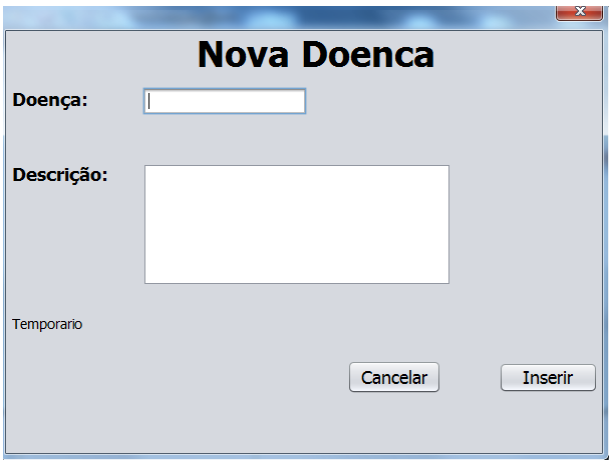

Ilustração 21 – Nova Doença

AA. Ver/Editar Doença

Quando é pretendido pelo utilizador ver em detalhe uma doença, é exibido o formulário representado pela **Ilustração 22.**

Se pretender editar a doença clicar no botão editar na barra de tarefas e é disponibilizado o botão atualizar tal como os campos a editar.
No caso de "Gerir Medicamentos" e "Gerir Veterinários" o processo e idêntico ao apresentado.

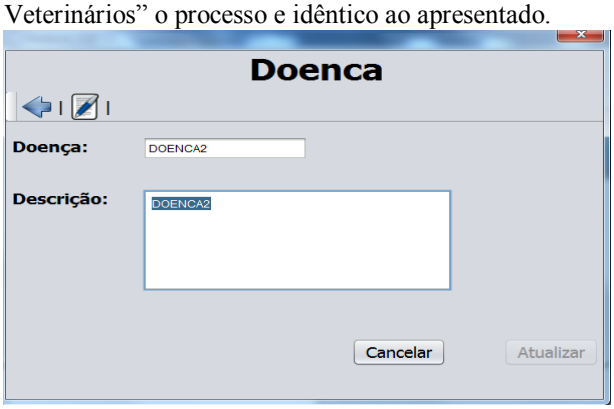

Ilustração 22 – Ver/Editar Doença

3. Conclusões

Tendo em conta alguma pesquisa verificamos que já existe algum trabalho nesta área. No entanto, 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.