UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ERICLES ANDREI BELLEI

SISTEMA BASEADO EM CLOUD COMPUTING PARA GESTÃO DE CONFINAMENTO DE GADO DE CORTE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

ERICLES ANDREI BELLEI

SISTEMA BASEADO EM CLOUD COMPUTING PARA GESTÃO DE CONFINAMENTO DE GADO DE CORTE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática – DAINF – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora:
Profa. Dra. Beatriz Terezinha Borsoi

PATO BRANCO
2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e
Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA BASEADO EM CLOUD COMPUTING PARA GESTÃO DE CONFINAMENTOS DE GADO DE CORTE

por

ERICLES ANDREI BELLEI

Este trabalho de conclusão de curso foi apresentado no dia 22 de novembro de 2016, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Banca examinadora:

Profª. Drª. Beatriz Terezinha Borsoi
Orientador

Profª. Me. Andreia Scariot Beulke

Prof. Me. Vinicius Pegorini

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª. Me. Soelaine Rodrigues Ascari
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

"É preciso ter um caos dentro de si para dar à luz uma estrela cintilante."

Friedrich Nietzsche

RESUMO

BELLEI, Ericles Andrei. Sistema baseado em *cloud computing* para gestão de confinamento de gado de corte. 2016. 62 f. Monografia (Trabalho de Conclusão de Curso). Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Pato Branco, 2016.

No contexto da pecuária de corte, confinamento é uma técnica empregada na engorda de gado, em que os animais, agrupados em lotes, ficam dispostos em uma área restrita recebendo alimentação e assistência. As negociações e a manutenção do rebanho de um estabelecimento que confina gado de corte, o consumo e os custos de insumos como alimentos e medicamentos, além do histórico de atividades e manejos efetuados, resultam em um volume de dados considerável e difícil de organizar. Verificou-se, assim, que um sistema computacional que permitisse o registro e a interpretação de todos esses dados possibilitaria a abstração de informações relevantes, que podem facilitar a organização, o planejamento e a administração do estabelecimento. Este trabalho de conclusão de curso apresenta a modelagem elaborada e a implementação de tal sistema computacional: uma aplicação web desenvolvida em linguagem Java, hospedada na nuvem, com banco de dados relacional utilizado como um serviço. Ao longo do texto é exemplificado o uso das tecnologias escolhidas para a construção do trabalho, reproduções de tela do sistema e códigos-fonte gerados.

Palavras-chave: Confinamento de gado de corte. Linguagem Java. Serviços de Computação em Nuvem. Banco de Dados como Serviço.

ABSTRACT

BELLEI, Ericles Andrei. A cloud computing based system for beef cattle feedlot management. 2016. 62 s. Monograph (Final Paper). Technologist Degree on System Analysis and Development. Federal University of Technology - Paraná, 2016.

In beef cattle industry context, feedlot is a technics employed in cattle fattening, where animals, grouped in batches, are arranged in a restricted area receiving food and assistance. Negotiations and maintaining the herd of an establishment whose confines beef cattle, consumption and costs of raw material such as food and medicines, in addition to the historic of performed activities and managements of animals, result in a amount of data which is hard to organize. Thus, it was observed that a computer system that allows the recording and interpretation of all these data would enable the abstraction of relevant informations that can facilitate the organization, planning and management of the property. This final paper presents the elaborate modeling and implementation of such computer system: a cloud hosted web application, developed in Java with relational database employed as a service. In the text the use of technologies chosen for the construction of the work, system screen reproductions and source code generated are exemplified.

Keywords: Beef cattle feedlot. Java programming language. Cloud Computing services. Data Base as a Service.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO FONTE

LISTA DE SIGLAS E ACRÔNIMOS

API	*Application Programming Interface*
AJAX	*Asynchronous Javascript and XML*
CDI	Injeção de Dependência e Contextos
DBaaS	*Data Base as a Service*
DER	Diagrama Entidade-Relacionamento
IaaS	*Infrastructure as a Service*
ISO/IEC	*International Organization for Standardization / International Electrotechnical Commission*
JSF	*JavaServer Faces*
NIST	*National Institut of Standards and Technology*
ORM	*Object-relational mapping*
PaaS	*Plataform as a Service*
PDF	*Portable Document Format*
RF	Requisito Funcional
RNF	Requisito Não Funcional
SaaS	*Software as a Service*
SDK	*Software Development Kit*
SQL	*Structured Query Language*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentadas as considerações iniciais, os objetivos, a justificativa da realização deste trabalho e a organização do texto por meio de uma breve descrição dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

A cadeia da bovinocultura de corte no Brasil atingiu em 2015 um faturamento de R$ 5,9 bilhões, quando o país era o maior exportador e o 2º maior produtor mundial de carne bovina, e também, o detentor do maior rebanho do mundo, com cerca de 212,3 milhões de animais (ASSOCIAÇÃO BRASILEIRA DAS INDÚSTRIAS EXPORTADORAS DE CARNES, 2015, PORTAL BRASIL, 2015). Estimativas de 2015 também apontavam que o Brasil confinou cerca de 5,19 milhões de bovinos, com perspectivas de crescimento contínuo, visto que desde 2014 o aumento foi de mais de 0,5 milhão de cabeças de gado (RALLY DA PECUÁRIA, 2015).

A indústria brasileira de carne bovina mostra-se expressiva mundialmente. Assim, espreita-se a viabilidade de soluções tecnológicas como ferramentas auxiliares para sustentar a estruturação e a otimização dos processos dessa indústria. Um aplicativo computacional que facilite a tarefa de gerenciamento de um confinamento pode ser considerado uma dessas soluções.

No contexto da pecuária de corte, confinamento é um preceito de alocação de bovinos, que ficam dispostos em lotes dentro de uma área restrita para receber alimentação e assistência (CARDOSO, 2000). Tal prática é empregada na engorda do gado, fase em que os animais desenvolvem carne de maneira intensiva, em um menor período de tempo, quando comparada à técnica de pastagem (THIAGO, 1996). O principal propósito de um confinamento é finalizar animais, ou seja, aumentar a sua massa corporal para o abate, promovendo o máximo aproveitamento de seu potencial genético em gerar excedente.

Em geral, estabelecimentos que confinam gado têm funcionamento similar. Inicialmente, bovinos que ainda não estão em condições de abate, são comprados por quilogramas de sua massa corporal, categorizados e dispostos em lotes. Durante a fase de confinamento os animais recebem alimentação, diagnósticos veterinários e

têm suas massas corporais aferidas com determinada frequência, até que seja constatado que cada um atingiu a completude total da finalização — comumente reportado como animal ‘gordo’ — ou quando o animal perfaz a situação que indica o seu melhor custo benefício, proporcional ao seu potencial de ganho de massa corporal *versus* tempo de permanência. Finalmente, o animal é removido do confinamento e vendido, quase sempre, para abatedouros, que pagam por quilograma da carcaça (carne e ossos restantes após o abate), ou para atravessadores, que pagam por quilograma do animal vivo.

No confinamento onde foi efetuada a coleta dos requisitos do sistema computacional desenvolvido, os dados das negociações e dos manejos dos animais são registrados em anotações, devido à falta de uma ferramenta auxiliar específica. Esse procedimento dificulta a obtenção de informações relevantes e consequentemente prejudica a gestão financeira.

Nesse cenário, entendeu-se que um sistema computacional que permitisse o registro e o cruzamento de todos os dados das atividades diárias, teria como resultado a abstração de informações que facilitariam a organização, o planejamento e a administração do estabelecimento como um todo.

Visando suprir essas necessidades, este trabalho apresenta a modelagem elaborada e a implementação de um sistema para um confinamento de gado de corte, exemplificados pelo uso das tecnologias e metodologias escolhidas para seu projeto.

1.2 OBJETIVOS

O objetivo geral refere-se ao resultado principal obtido com este trabalho. Os objetivos específicos o complementam, explicitando as expectativas com as funcionalidades da solução proposta e o viés acadêmico deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema que permita registrar os dados decorrentes das atividades concretizadas no contexto de um confinamento de gado de corte e obter informações relevantes pela análise, gerenciamento e cruzamento desses dados.

1.2.2 Objetivos Específicos

- Facilitar a administração do confinamento, com a gestão de compras e vendas, fornecedores e clientes e os dados dos animais envolvidos nessas negociações.
- Facilitar a constatação de animais gordos, comparando as aferições de massa corporal efetuadas durante a fase de confinamento com uma estimativa de rendimento ponderada nas características zootécnicas individuais no momento da compra.
- Auxiliar nas tarefas de acompanhamento da situação dos animais e na administração da rotina, interpretando os dados dos manejos realizados e resultados de rendimento *versus* variáveis.
- Proporcionar relatórios e cálculos aproximados de investimentos, lucros e eficiência de suprimentos e de animais, analisando os dados de compras e vendas vinculados ao histórico de evolução dos bovinos e ao consumo de insumos do confinamento.
- Empenhar tecnologias de *Platform as a Service*, como *Cloud hosting* e *Database as a Service,* no desenvolvimento de um sistema *web* que auxilie no gerenciamento dos dados obtidos da realização das atividades relacionadas ao confinamento de gado de corte.

1.3 JUSTIFICATIVA

A carência de uma ferramenta auxiliar específica que seja eficaz no contexto de um confinamento de gado de corte, e ainda, os poucos registros e anotações dos dados do cotidiano desse cenário, ocasionam dificuldades e restrições para acompanhamento das atividades, tornando menos eficiente a tarefa de gerenciamento do estabelecimento e mais difícil a monitoração do rebanho e conhecimento de resultados financeiros.

Tantas informações desse cotidiano — que surgem desde a compra de cada animal, com preço, quantidade, características, fornecedor, passando pela fase de confinamento, com alimentações e diagnósticos realizados, aferições de massa

corporal, compra e consumo de suprimentos utilizados e seus valores, até o momento da saída do animal na venda aos clientes — somam um volume de dados grande e difícil de organizar, seja em papel, ou mesmo com o auxílio de planilhas eletrônicas. Ao mesmo tempo, os processos de estimativa de evolução e análise do gado não são específicos e valem-se, frequentemente, de métodos visuais para calcular a completude de finalização. Dificilmente consegue-se acompanhar a taxa de engorda dos bovinos, tampouco a taxa de produtividade de todos ou de animais com raças e procedências diferentes, nem a eficácia de determinado alimento ou medicamento, além de outras variáveis indicadoras de rendimentos.

Essas dificuldades justificam a necessidade da aplicabilidade de um sistema computacional para a intervenção nesse contexto, a fim de facilitar a administração e a rotina de atividades de um confinamento de gado de corte, ao permitir registrar os dados e a partir da gerência desses, abstrair informações relevantes.

Na escolha das tecnologias, o emprego de *Data Base as a Service* (DBaaS) e da hospedagem do servidor de aplicação na nuvem justifica-se pelas vantagens apresentadas, como confiabilidade, integridade e facilidade de acesso aos dados possibilitada pela Internet. Já a aplicação *web* foi indicada para o desenvolvimento por estar pautada dentro do contexto de serviços de computação em nuvem e, com alta compatibilidade, ser independente às restrições de plataformas e de sistemas operacionais.

1.4 ESTRUTURA DO TRABALHO

O texto está organizado em cinco capítulos. Este é o primeiro e relata as considerações iniciais, os objetivos e a justificativa do trabalho. O Capítulo 2 contextualiza o referencial teórico, baseado nos conceitos de serviços de computação em nuvem. No Capítulo 3 está a sequência que compõe o método e os materiais utilizados no desenvolvimento deste trabalho. No Capítulo 4 estão os resultados, a modelagem e a descrição da implementação do sistema, apresentados por meio de diagramas, tabelas de requisitos, explicações textuais, reproduções de telas e partes do código-fonte. Por fim, o Capítulo 5 contém a conclusão do autor acerca das tecnologias aplicadas e dos objetivos alcançados.

2 SERVIÇOS DE COMPUTAÇÃO EM NUVEM

Este capítulo contextualiza o referencial teórico, baseado em uma das ferramentas e tecnologias aplicadas durante a implementação do sistema para gestão de confinamento de gado de corte. Essa ferramenta é o OpenShift Online, descrito pela Red Hat (2016, p.1) como “[...] uma solução de desenvolvimento, criação, implantação e hospedagem de aplicativos na nuvem”. Assim, o referencial teórico apresenta os principais conceitos, divisões e características dos serviços de computação em nuvem, com ênfase em DBaaS, entretanto, sem aprofundar em suas várias especificidades.

2.1 CONTEXTO E DEFINIÇÕES CONCEITUAIS

A crescente popularização da Internet e de recursos computacionais cativa a expansão do número de usuários conectados e de serviços oferecidos aos mesmos. Por conseguinte, a dependência de tais serviços cresce da mesma maneira com que a tecnologia evolui, permitindo que hoje a Internet e serviços disponibilizados por meio dela sejam considerados, muitas vezes, essenciais.

Nesse cenário, Souza et al. (2010) veem a computação em nuvem como uma tendência recorrente, cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda, com pagamento baseado no uso, e que tem pretensões de escalas globais por prover infraestrutura e serviços para as massas, que vão desde o usuário final que hospeda seus arquivos na Internet até grandes corporações que terceirizam toda a divisão de TI.

Mollah et al. (2012) afirmam que a computação em nuvem constitui um grupo virtual de recursos, como redes, armazenamento, unidade central de processamento e memória, para transportar as requisições do usuário e oferecer, sem obstáculos, *hardware* e *software* sob demanda. De tal modo, emprega-se automaticamente vários servidores como se fossem um único computador acessado pela Internet para prover recursos conforme a demanda.

O paradigma da computação em nuvem vislumbra a evolução e reponta o potencial de tendência no mercado de TI. Logo, faz-se necessária a definição clara de seus aspectos e conceitos fundamentais. Porém, não é uma tarefa simples especificar

tecnologias, que constantemente modificam-se, nem estabelecer padrões e barreiras que restringem tal paradigma de conceitos utilizados há algum tempo.

Algumas iniciativas de consórcios e organismos internacionais seguem tratando da consolidação para tal paradigma. Comitês da norma ISO/IEC 20000-7 (2011) trabalham na definição de conceitos da arquitetura de serviços, gestão da segurança, qualidade, operabilidade, portabilidade e outros aspectos da computação em nuvem. Mas, atualmente, as definições conceituais propostas pelo *National Institute of Standards and Technology* (NIST) são, tanto na literatura, quanto comercialmente, as mais abrangentes e aceitas (BEREA, 2012).

2.2 O MODELO NIST

Para conceituar computação em nuvem, o NIST (2010) define cinco atributos e características essenciais, quatro modelos de implantação e três principais modelos de serviço.

Na avaliação de Souza et al. (2010), os atributos e as características básicas citadas pelo NIST são vantagens que as soluções de computação em nuvem oferecem e as definem exclusivamente, distinguindo-a de outros paradigmas:

- *Self-service sob demanda*: O usuário pode demandar maior ou menor quantidade de recursos computacionais, como tempo de processamento no servidor ou armazenamento e transferências na rede. Esses recursos devem ser disponibilizados sem interação humana com seus provedores.
- *Amplo acesso:* Os recursos são disponibilizados por meio da Internet e acessados independentemente de localização geográfica por mecanismos computacionais que possibilitam o uso em plataformas heterogêneas, como computadores pessoais e *smartphones*.
- *Pooling de recursos:* Os recursos computacionais do provedor são organizados para servir diversos usuários em um modelo multiclientes. Os variados recursos físicos e virtuais são dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários, que não precisam conhecer a localização física dos recursos computacionais, mas podem especificar localização com prioridade em um país ou região.

- *Elasticidade rápida*: Com o autogerenciamento do serviço, é a capacidade de escalar, disponibilizar e remover recursos rápida e automaticamente, conforme o aumento ou retração de demanda. Os recursos devem parecer ilimitados aos usuários, que podem adquiri-los ou devolvê-los em qualquer quantidade e a qualquer momento.
- *Serviço medido*: Para possibilitar transparência entre o provedor e o usuário, sistemas em nuvem automaticamente devem monitorar, controlar e otimizar o uso de recursos por meio de uma capacidade de medição que é realizada com base nas funcionalidades, desempenho contratado e no nível de abstração apropriado para o tipo de serviço.

Os modelos de implantação do NIST para ambientes de computação em nuvem descritos por Borges et al. (2011) referem-se às restrições de abertura de acesso e disponibilidade, conforme tipos de informação e processos de negócios. Esses modelos são:

- *Nuvem Privada:* A infraestrutura da nuvem pertence ou é alugada por uma única organização e é exclusivamente empenhada e operada pela mesma.
- *Nuvem Pública*: O provedor, dono de toda a infraestrutura, assume as responsabilidades de implantação, gerenciamento, disponibilização e manutenção para oferecer aos clientes serviços livres de complexidade, com configurações específicas que tentam acomodar os casos de uso mais comuns. O termo ‘público’ não diz respeito à acesso gratuito, mas ao público de usuários em geral que pode contratar serviços.
- *Nuvem Comunidade:* A infraestrutura é compartilhada por organizações que têm interesses em comum, como requisitos de segurança, políticas de acesso, restrições legais, aspectos de compatibilidade etc.
- *Nuvem Híbrida:* A infraestrutura dessa nuvem é composta por pelo menos dois modelos de implantação diferentes que preservam suas características originais, mas, que estão interligados por uma tecnologia proprietária ou padronizada para proporcionar interação e portabilidade dos dados e aplicações entre os mesmos.

Os modelos de serviços de computação em nuvem da literatura do NIST definem um padrão arquitetural de três camadas, que são: *Infrastructure as a Service* (IaaS), *Plataform as a Service* (PaaS) e *Software as a Service* (SaaS).

2.3 MODELOS DE SERVIÇOS DE COMPUTAÇÃO EM NUVEM

Prover serviços sob demanda e com pagamento baseado no uso é a principal finalidade da computação em nuvem. Tais serviços são implementados com finalidades diferentes e específicas, por meio de diversos recursos organizados de maneira interdependente.

O padrão arquitetural proposto pelo NIST (2010) dispõe esses recursos em três camadas: IaaS é a base para as demais, seus serviços compreendem toda a infraestrutura de *hardware* e sistemas operacionais mediadores sobre os quais serão disponibilizados os serviços do grau acima, PaaS, que resume-se em ferramentas de programação que viabilizam aos desenvolvedores criar os serviços da camada mais externa, SaaS, onde estão as aplicações e *softwares* para o usuário final. Assim, valer-se de qualquer recurso de uma camada superior, automaticamente empenha-se recursos das camadas subjacentes. A Figura 1 esquematiza esse padrão com os tipos de recursos computacionais e exemplos comerciais situados em cada nível.

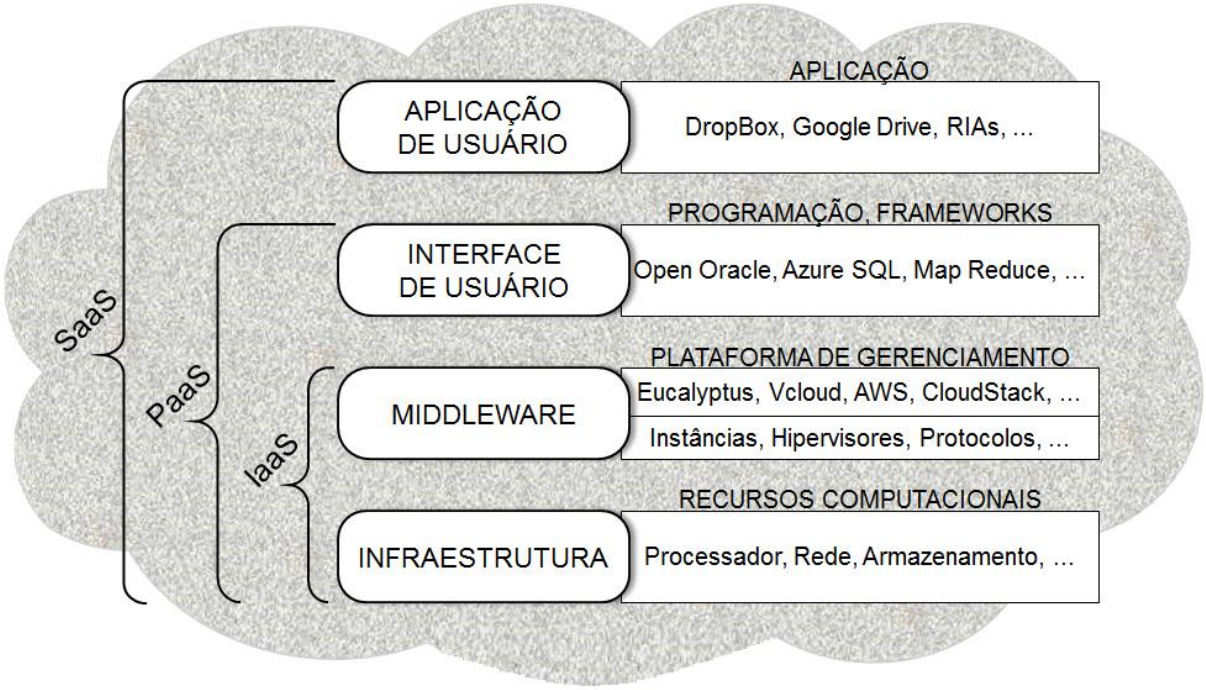

Figura 1 - Padrão arquitetural do NIST para serviços de computação em nuvem
Fonte: Adaptado de Carissimi (2015).

Os serviços da camada de IaaS provêm os recursos computacionais de *hardware* e de *middleware* essencial, como processadores, memórias, armazenamento, redes, sistemas operacionais, protocolos de comunicação, instâncias de máquinas virtuais e hipervisores que dão suporte e assistem a existência

de todos os outros serviços das camadas superiores. Borges et al. (2011) explicam que a IaaS é baseada na virtualização dos recursos computacionais, programada para os escalar dinamicamente de acordo com a demanda das aplicações, característica que proporciona uma série de vantagens aos clientes, como a redução de investimentos e flexibilidade de *hardware*, eliminação de custos com segurança e manutenção, além da otimização de desempenho.

O modelo de PaaS, também segundo Borges et al. (2011), fornece serviços específicos, ambientes compostos por recursos de desenvolvimento de *software*, como ferramentas de codificação, persistência de dados, hospedagem, integração, testes etc. Esses ambientes facilitam a implantação de aplicações, pois os desenvolvedores tornam-se mais livres das complexidades intrínsecas ao gerenciamento da infraestrutura subjacente necessária ao funcionamento de um sistema computacional completo.

SaaS é camada mais externa do padrão NIST, seu modelo visa prover *softwares* prontos para uso, com foco no usuário final. Esses serviços são totalmente executados por recursos de nuvem e estão disponíveis em plataformas heterogêneas, via interfaces *thin client*, como navegadores *web*, situadas em qualquer lugar do mundo conectado à Internet. Carissimi (2015) destaca que nesse modelo o usuário enxerga apenas o *software* que está utilizando e não tem conhecimento da real localização dos recursos empregados, nem das linguagens de programação empenhadas no desenvolvimento do serviço, nem da plataforma e *hardware* sobre o qual a aplicação executa.

Há ainda muitos outros serviços que são nomeados e ofertados comercialmente como algum tipo de serviço de computação em nuvem — *Database as a Service, Testing as a Service, Communication as a Service, Development as a Service, Storage as a Service* etc. — mas, efetivamente são apenas especializações dos três modelos principais propostos pelo NIST.

2.4 DATABASE AS A SERVICE

Os bancos de dados compõem, na maioria dos contextos computacionais, o alicerce central das aplicações de *software*. Atualmente, a tendência dos serviços oferecidos por computação em nuvem, aliada às novas necessidades, faz com que as

tradicionais formas de persistência de dados sofram transformações para adaptar-se aos novos paradigmas de processamento e armazenamento de informações.

Souza et al. (2010) enfatizam que sistemas de banco de dados em nuvem cativam popularidade e potencial de atrair clientes de diversos setores do mercado, desde pequenas empresas que buscam reduzir custos com o uso de infraestrutura e sistemas de terceiros, até grandes corporações que visam soluções para gerenciar milhares de máquinas e permitir o atendimento de um aumento inesperado de tráfego. Tais sistemas adotam o conceito de *Database as a Service* (DBaaS), considerado uma especialização de PaaS.

ScaleDB (2014) define DBaaS como um modelo de serviço de computação em nuvem que fornece aos usuários acesso a um banco de dados sem a necessidade de configurar o *hardware*, instalar *software* e organizar a infraestrutura necessária, pois todas as tarefas administrativas e de manutenção são atendidas pelo provedor.

O cliente pode, ainda conforme ScaleDB (2014), ter mais controle sobre o banco de dados, determinando frequência e agendamento de *backups*, funções de balanceamento de carga, limite de conexões, nível de utilização, além de gerenciar e configurar todas as instâncias subjacentes por meio de uma *Application Programming Interface* (API), geralmente acessível em um aplicativo *web*.

Comparados aos modelos mais tradicionais, os benefícios de banco de dados na nuvem citados por Zhang (2011), incluem a ausência de preocupação do usuário com a manutenção e administração da infraestrutura do servidor de dados — o provedor é responsável pela gerência e disponibilização dinâmica de recursos, tratamento de falhas, manutenção e atualização do *hardware,* estratégias de *backup* e integridade dos dados —, previsibilidade e custos mais baixos devido ao uso de um serviço que tem pagamento baseado na quantidade de utilização, complexidade de implantação reduzida, balanceamento de carga, elasticidade e escalabilidade dos recursos conforme a demanda de utilização, otimização de desempenho etc. Além disso, ainda há a vantagem da informação estar distribuída, já que a Internet permite acesso homogêneo aos dados a partir de qualquer lugar, característica que facilita o compartilhamento desses dados e o trabalho corporativo.

Exemplos comerciais de DBaaS são Amazon RDS e DinamoDB, Apache Cassandra e CouchDB, Google BigTable, IBM Informix, Microsoft SQL Azure, 10gen MongoDB etc. (SOUZA et al., 2010).

3 MATERIAIS E MÉTODO

Este capítulo explicita os materiais, tecnologias, ferramentas e a metodologia empenhada na sequência do desenvolvimento deste trabalho.

3.1 MATERIAIS

O Quadro 1 lista as ferramentas e tecnologias aplicadas durante os processos de modelagem e implementação do sistema proposto.

Material	Versão	Finalidade	Referência
Apache Log4j	2.6.2	Depuração e criação de *log* de dados.	http://logging.apache.org/log4j/2.x/
Apache Maven	2.1 for Eclipse	Compilação e controle de dependências.	https://maven.apache.org/
Apache Tomcat	8.5.4	*Servlet Container* como Servidor de aplicação.	http://tomcat.apache.org/
Eclipse	Neon	Ambiente integrado de desenvolvimento.	https://eclipse.org/
Hibernate ORM	5.1	Mapeamento objeto-relacional e persistência dos dados. ORM de *Object relational mapping.*	http://hibernate.org/orm/
Hibernate Validator	5.2.4	Validação de restrições de atributos e objetos.	http://hibernate.org/validator/
iText PDF	5.5.9	Geração de relatórios no formato *Portable Document Format* (PDF)	http://itextpdf.com/
Linguagem Java	Enterprise Edition 8 SDK	Conjunto de recursos para o desenvolvimento na linguagem Java. SDK de *Software Development Kit.*	http://www.oracle.com/br/java/overview/index.html/
Lombok	1.16.10	Automatização de criação de métodos básicos de classes Java.	https://projectlombok.org/
Mojarra	2.2	Implementação do *JavaServer Faces* (JSF).	https://javaserverfaces.java.net/
OmniFaces	2.4	Biblioteca de utilitários para JSF.	http://showcase.omnifaces.org/
PgAdmin	III	Auxiliar do banco de dados PostgreSQL. SQL de *Structured Query Language.*	https://msdn.microsoft.com/pt-br/library/mt238290.aspx
PostgreSQL	9.4	Servidor de banco de dados.	https://www.postgresql.org/

PrimeFaces	6.0.1 Elite Ultima Theme	Componentes baseados em JSF para a elaboração da interface gráfica de usuário.	http://primefaces.org/layouts/ultima.html
Red Hat OpenShift	Free	Servidor de aplicação e servidor de banco de dados no modelo de *cloud computing*.	https://www.openshift.com/
Spring Security	4.1.3	Controle de acesso de usuários e permissões.	http://projects.spring.io/spring-security/
Visual Paradigm	13.1 Community Edition	Elaboração de diagramas de *Unified Modeling Language* (UML) para modelagem e de Diagrama Entidade-Relacionamento (DER).	https://www.visual-paradigm.com/download/community.jsp
Weld	2.3.5	Injeção de dependências e contextos para Java.	http://weld.cdi-spec.org/

Quadro 1 - Ferramentas e tecnologias utilizadas

3.2 MÉTODO

Durante a concepção da ideia de um sistema computacional como ferramenta auxiliar em um confinamento de gado de corte, foi efetuada uma pesquisa simples, a fim de encontrar referenciais de mercado, como meio de nivelar a solução a ser desenvolvida e verificar, comercialmente, a existência de *softwares* análogos.

Na pesquisa foram analisados materiais de divulgação disponíveis na Internet. A partir desses, foram encontrados poucas soluções com propostas afins, algumas até abrangentes ao versar diversos aspectos de propriedades rurais, como a pecuária de leite e a agricultura. Entretanto, nenhum dos resultados demonstrou ser suficientemente acessível e específico ao abordar o contexto de um confinamento de gado de corte.

Entre os sistemas com finalidades similares encontrados, destacam-se:

- Brabov: um aplicativo gratuito, com versões para iOS e Android, que permite ter um controle financeiro da fazenda, das receitas e despesas classificadas por categorias, além de cadastrar os animais individualmente ou em lotes e controlar suas principais atividades de manejo, como lactação, vacinas, castrações, desmames, reproduções etc. (BRABOV, 2015).

- Agrisoft Módulo Rebanho: produto da Agrisoft TI-Agro, empresa especializada em *softwares* agropecuários para administração rural. Possui funções que permitem acompanhamento de cada animal, controle sanitário e reprodutivo, ocupação dos pastos, acompanhamento da tipagem sanguínea dos bovinos vinculada aos reprodutores, controle de patrimônio, receitas, despesas, custos e investimentos da propriedade (AGRISOFT TI-AGRO, 2015).

Anterior à esta monografia, como o trabalho de estágio supervisionado do autor, foi elaborada a modelagem simplificada e desenvolvido um protótipo do sistema, o que Sommerville (2004, p. 145) conceitua como “[...] uma versão inicial de um sistema de *software,* que é utilizada para mostrar conceitos, experimentar opções de projeto e, em geral, para conhecer mais sobre os problemas e suas possíveis soluções”. A construção desse protótipo objetivou a validação da modelagem e o ensaio de tecnologias, para serem empregadas na implementação do sistema final.

As etapas do trabalho, na sequência em que foram concretizadas são descritas a seguir:

a) Levantamento de requisitos

Inicialmente, os requisitos foram identificados, classificados e registrados com base no conhecimento que o autor tem da rotina, procedimentos e necessidades de um confinamento.

b) Análise inicial e projeto do protótipo do sistema

Com os requisitos devidamente coletados, foram definidos os primeiros casos de uso. Nessa fase foram identificados os atores e algumas funcionalidades para a solução e confeccionados o diagrama de classes e o diagrama de entidade-relacionamento para o sistema.

c) Implementação e testes do protótipo

O protótipo foi desenvolvido para a plataforma *desktop*, com Linguagem Java e banco de dados PostgreSQL. As funcionalidades implementadas estavam centradas em alguns cadastros básicos. Os testes foram simples e de usuário, realizados pelo autor em tempo de codificação, para verificar a adequação das funcionalidades, identificar erros de código e de conexão com o banco de dados.

d) Avaliação de resultados

Com o protótipo finalizado e em funcionamento, foi avaliado o desempenho e a satisfação com as tecnologias empregadas e o sistema até então implementado.

Por conseguinte, foram elencadas algumas possibilidades de alterações e aprimoramento observadas pelo autor e também sugeridas pela banca avaliadora do trabalho de estágio.

e) Refatoramento do projeto do sistema

Incorporadas as ideias e observações do resultado da fase anterior, os requisitos e os diagramas estruturais e comportamentais do projeto foram reformulados e passaram por ajustes para conferir maior adequação à proposta e otimização da solução pretendida.

f) Implementação e testes do sistema final

Na implementação, o pgAdmin III foi utilizado para a estruturação das entidades e suas rotinas associadas no banco de dados PostgreSQL, disponibilizado na plataforma da Red Hat OpenShift. No Eclipse IDE foi elaborada a codificação do *software*, com linguagem Java e outras tecnologias e frameworks listados no Capítulo 3.1. Finalizado, o sistema foi colocado em produção em uma instância do Apache Tomcat também hospedado na plataforma da Red Hat OpenShift.

Os testes foram informais e gerais, realizados pelo autor sem um plano definido. Durante o desenvolvimento, o código fonte foi constantemente testado para identificar erros de programação e analisar a comunicação com a base de dados. A interface gráfica de usuário e as funcionalidades implementadas também foram testadas para avaliar sua conformidade aos requisitos do sistema.

4 RESULTADOS

Este capítulo apresenta a modelagem, códigos e reproduções de tela que exemplificam a implementação e a interação dos usuários com um sistema para facilitar o gerenciamento de um confinamento de gado de corte.

4.1 ESCOPO PARA O SISTEMA

O sistema deve permitir a manutenção e o controle dos dados decorrentes dos acontecimentos de negócio relacionados a um estabelecimento que confina gado de corte e o histórico de seu rebanho. O processo a ser gerenciado envolve a compra de animais, a evolução em termos de ganho de massa corporal e interação medicamentosa, o acompanhamento de consumo de insumos e, finalmente, a venda.

Na compra, o momento da entrada de um animal, com base nos dados zootécnicos de massa corporal, raça, sexo e idade aproximada, o sistema irá calcular e propor um valor que sirva como meta para a engorda do bovino em seu melhor aproveitamento. Esse valor é estimativo e será informado apenas como referência, visto que o desempenho de engorda de um animal depende também de outros fatores.

Durante a fase de confinamento, dados sobre os animais serão atualizados com frequência, como: consumo de alimentos, diagnósticos veterinários efetuados e as aferições de massa corporal individual (principal indicador de produtividade). Comparados com a meta e outras variáveis, esses dados permitirão ao sistema atualizar o *status* de engorda e gerar relatórios e estatísticas da situação geral e evolução dos bovinos, utilização de suprimentos, eficiência de raças ou procedências e o histórico de todos os manejos realizados.

O sistema deve também ter o registro de preços de alimentos, medicamentos e custos veterinários para contabilizar quanto cada animal consumiu de cada insumo quando esteve confinado. Esse custo, somado ao valor pago na compra, corresponde o valor do investimento feito em cada animal.

O registro dos dados da negociação e de cada animal na venda também será feito, para que após, possa ser estimado o lucro obtido.

Os dados de fornecedores, clientes, compras e vendas de animais, alimentos, medicamentos e tudo que envolver negociações e preços são cadastrados e

visualizados apenas pelo administrador ou supervisor. Dados de aferições de massa corporal, alimentação, medicação e acompanhamento serão inseridos pelos funcionários que interagem e cuidam dos animais.

4.2 MODELAGEM

A apresentação da modelagem é arranjada como forma de facilitar o entendimento da proposta de intervenção no contexto abordado com a solução pretendida.

4.2.1 Requisitos Funcionais

O Quadro 2 lista os Requisitos Funcionais (RF) identificados para o sistema.

Identificação	Nome	Descrição
RF01	Cadastrar fornecedores e clientes	Permitir ao usuário administrador fazer e editar o cadastro de fornecedores e clientes do confinamento.
RF02	Cadastrar raças bovinas	Realizar o cadastro das raças de animais que estarão confinados, para ser utilizado durante o cálculo da meta de engorda e no filtro do relatório de rendimentos por raça.
RF03	Cadastrar lotes de animais	Permitir ao usuário administrador cadastrar os lotes, que devem estar associados aos animais que estão alocados em cada um desses.
RF04	Cadastrar compra de animal	Permitir ao usuário administrador cadastrar as compras, individual ao animal, contendo os dados da negociação.
RF05	Cadastrar processo de confinamento	Uma extensão do cadastro de compra, o cadastro de confinamento individual contém os dados do animal, meta e *status* de engorda e os registros de aferições de massa, alimentações e diagnósticos realizados.
RF06	Propor meta de engorda	Baseado nos dados zootécnicos registrados durante a compra, o sistema deve calcular e propor uma meta de engorda para o animal, que é a massa corporal ideal para proporcionar a maior margem de lucro e aproveitar ao máximo seu potencial genético.
RF07	Calcular *status* de engorda	Considerando os registros de aferições de massa corporal de cada animal em relação à sua meta, o sistema deve atribuir automaticamente um *status* de percentagem para a evolução de engorda.

RF08	Registrar aferições de massa corporal	Registrar no respectivo cadastro de processo de confinamento as aferições de massa corporal de cada animal.
RF09	Cadastrar os alimentos utilizados	Cadastrar os alimentos utilizados no confinamento para estimativas de custos. Não envolve o gerenciamento de estoque.
RF10	Registrar alimentações realizadas	Registrar as alimentações que os animais recebem, individualmente ou por lote, contendo a quantidade e o tipo de alimento fornecido.
RF11	Cadastrar os medicamentos utilizados	Cadastrar os medicamentos utilizados no confinamento para estimativas de custos. Não envolve o gerenciamento de estoque.
RF12	Registrar diagnósticos realizados	Registrar os diagnósticos realizados (doença, manejo sanitário, troca de lote, visita de veterinário ou outro evento), individualmente ou por lote, associando os medicamentos que foram utilizados e se houve outros custos, como honorários de médico veterinário.
RF13	Calcular investimento	Calcular o investimento feito em cada animal, somando seu valor de compra com os custos de diagnósticos e alimentações que recebeu.
RF14	Cadastrar venda de animal	Permitir ao usuário administrador cadastrar as vendas, individual por animal, contendo suas características e os dados da negociação.
RF15	Proporcionar relatórios e análise de dados registrados	Proporcionar relatórios para o administrador sobre as condições do confinamento e as atividades concretizadas: - Situação geral e evolução dos bovinos; - Estoque disponível e suas caraterísticas; - Histórico de manejos e consumo de suprimentos; - Investimentos, lucros, custos e correlatos; - Rendimentos e desempenhos de animais vendidos. O sistema deve mostrar também cada animal que está em quarentena, fato que acontece após receber diagnóstico com medicamento que requisita prazo de carência.
RF16	Cadastrar usuários	Permitir ao usuário administrador cadastrar outros usuários e gerenciar suas permissões de acesso às funções e dados no sistema, como os supervisores e funcionários que registrarão os dados dos manejos.

Quadro 2 - Requisitos funcionais

4.2.2 Requisitos Não Funcionais

O Quadro 3 contextualiza os Requisitos Não Funcionais (RNF), aplicáveis ao sistema como um todo, também denominados requisitos suplementares, que aqui explicitam restrições de acesso e regras de negócio.

Identificação	Nome	Descrição
RNF01	Controle de acesso	O acesso ao sistema ocorrerá por meio de credenciais de usuário e senha.
RNF02	Identificação de animal	Cada animal no sistema pode ser identificado pelo código que está impresso em seu brinco.

Quadro 3 - Requisitos não funcionais

4.2.3 Casos de Uso

O diagrama de casos de uso apresentado na Figura 2 ilustra as funcionalidades essenciais do sistema, utilizadas pelos atores administrador e funcionário. O Administrador cadastra compras, vendas, fornecedores, clientes, suprimentos, lotes e raças, além de visualizar relatórios e controlar os usuários e suas permissões. O funcionário registra os dados dos manejos realizados com os bovinos: aferições de massa corporal, alimentações e diagnósticos.

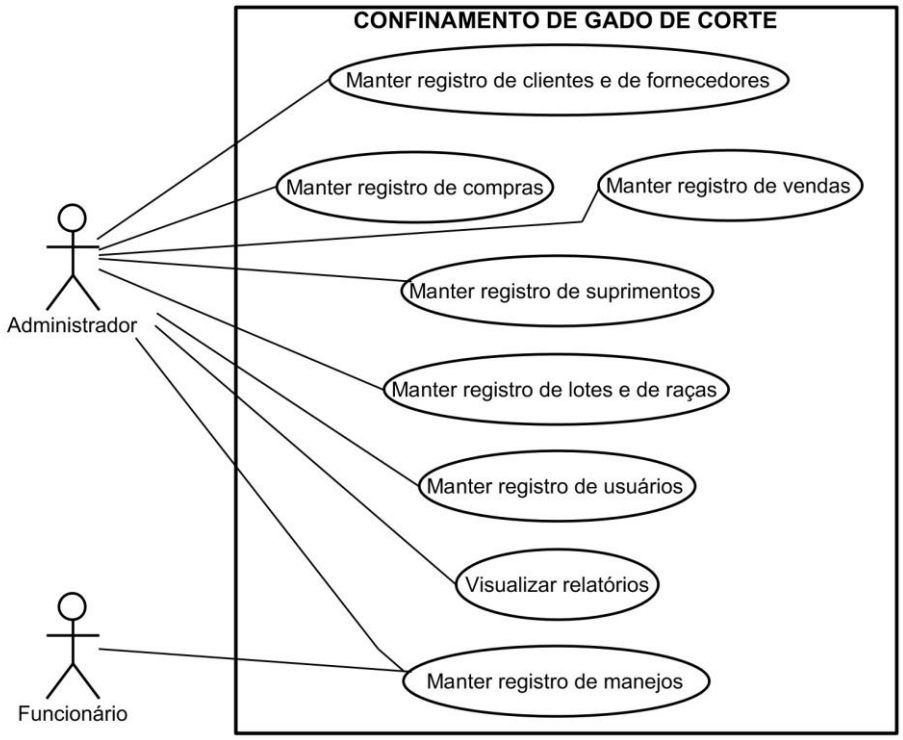

Figura 2 - Diagrama de casos de uso
Fonte: Autoria própria.

Os casos de uso da Figura 2 estão descritos a seguir. No Quadro 4 está a descrição do caso de uso ‘Manter registro de clientes e de fornecedores’.

Caso de uso:
Manter registro de clientes e de fornecedores.
Descrição:
Antes de cadastrar compras ou vendas, é necessário possuir o cadastro dos respectivos fornecedores ou clientes.
Evento Iniciador:
Negociação de bovinos.
Ator:
Administrador.
Pré-condição:
Dados pessoais coletados.
Sequência de Eventos:
1. O ator acessa a tela de cadastro de colaboradores;
2. O ator escolhe fornecedor ou cliente;
3. O ator escolhe cadastrar um novo e insere os dados;
4. O sistema pede a confirmação ao ator e grava os dados.

Pós-Condição:
Cadastro realizado.

Quadro 4 - Caso de uso Manter registro de clientes e de fornecedores

O caso de uso ‘Manter registro de compras’ é denotado no Quadro 5.

Caso de uso:
Manter registro de compras.
Descrição:
Cadastrar as compras e, consequentemente, cada animal e seu processo de confinamento que foi inserido no estabelecimento.
Evento Iniciador:
Compra de animal.
Ator:
Administrador.
Pré-condição:
Dados da compra e características do animal individual coletados.
Cadastro de fornecedor efetuado.
Cadastro do lote do animal efetuado, se for colocado em um.
Cadastro de raças efetuado.
Sequência de Eventos:
1. O ator acessa a tela de compras e a opção cadastrar;
2. O ator seleciona o fornecedor na lista disponível;
3. O ator insere os dados da compra;
4. O ator insere as características do animal;
5. O ator seleciona a(s) raça(s) do animal nas listas disponíveis;
6. O sistema calcula e propõe a estimativa de meta de engorda;
7. O ator verifica a meta e altera-a se julgar necessário;
8. O ator seleciona o lote em que o animal foi inserido na lista disponível;
9. O sistema pede a confirmação ao ator e grava os dados.

Pós-Condição:
Cadastros de compra e de animal realizados.

Quadro 5 - Caso de uso Manter registro de compras

A função de propor meta de engorda (requisito funcional RF06), empregada durante o caso de uso ‘Manter registro de compras’, objetiva maximizar lucros por meio do aproveitamento do potencial do animal, calculando sua meta ponderada nas características zootécnicas de massa corporal, idade, sexo e raça, para sugerir ao administrador, que pode acatar ou não o resultado. Essa função não desempenhará papel fundamental no sistema, sendo apenas a sugestão de uma estimativa, assim, não pode receber grande responsabilidade porque computa genericamente um valor que é relativo de estimar, já que existem outros fatores que condicionam o desempenho de um animal. Ferraz e Eler (2010) particularizam esses fatores como numerosos, alguns até instáveis e complexos, tais como os efeitos climatológicos e estações do ano, sanidade animal, criação, avaliação genética, metabolismo, adaptação ao ambiente e variada composição química dos alimentos da dieta.

O caso de uso ‘Manter registro de vendas’ é explicado no Quadro 6.

Caso de uso: Manter registro de vendas. **Descrição:** Cadastrar as vendas de animais realizadas. **Evento Iniciador:** Venda de animal. **Ator:** Administrador. **Pré-condição:** Dados da venda coletados. Cadastro de cliente efetuado. Cadastro de animal efetuado. **Sequência de Eventos:** 1. O ator acessa a tela de vendas e a opção cadastrar; 2. O ator seleciona o cliente na lista disponível; 3. O ator seleciona o animal que foi vendido na lista disponível; 4. O ator insere as massa e preço do animal na venda; 5. O sistema pede a confirmação ao ator e grava os dados. **Pós-Condição:** Cadastro de venda realizado.

Quadro 6 - Caso de uso Manter registro de vendas

O Quadro 7 contextualiza o caso de uso ‘Manter registro de suprimentos’.

Caso de uso: Manter registro de suprimentos. **Descrição:** Cadastrar novos suprimentos ou alterar o valor de suprimentos já cadastrados. **Evento Iniciador:** Novo suprimento utilizado ou alteração de custo de suprimento. **Ator:** Administrador.

Pré-condição:
Dados do suprimento coletados.
Sequência de Eventos:
1. O ator acessa a tela de suprimentos;
2. O ator escolhe medicamentos ou alimento;
3. O ator escolhe novo ou editar;
4. O ator insere (ou seleciona o suprimento e altera) os dados;
5. O sistema pede a confirmação ao ator e grava os dados.

Pós-Condição:
Cadastro de suprimento realizado ou alterado.

Quadro 7 - Caso de uso Manter registro de suprimentos

O Quadro 8 apresenta a descrição do caso de uso ‘Manter registro de lotes e de raças’.

Caso de uso:
Manter registro de lotes e de raças.
Descrição:
Cadastrar os lotes que conterão os animais confinados ou as raças bovinas presentes no confinamento.
Evento Iniciador:
Criação de novo lote ou utilização de nova raça bovina.
Ator:
Administrador.
Pré-condição:
Dados do lote ou da raça coletados.
Sequência de Eventos:
1. O ator acessa a tela de configurações;
2. O ator escolhe a tela de lotes ou a tela de raças;
3. O ator escolhe a opção cadastrar;
4. O ator insere os dados;
5. O sistema pede a confirmação ao ator e grava os dados.

Pós-Condição:
Cadastro de lote ou raça realizado.

Quadro 8 - Caso de uso Manter registro de lotes e de raças

O caso de uso ‘Manter registro de usuários’ é retratado no Quadro 9.

Caso de uso:
Manter registro de usuários.
Descrição:
O administrador cadastra os funcionários como usuários do sistema e concede as permissões para utilizar determinadas funcionalidades.
Evento Iniciador:
Novo usuário precisa acessar o sistema.
Ator:
Administrador.
Pré-condição:
Dados do usuário coletados.
Sequência de Eventos:
1. O ator acessa a tela de configurações;
2. O ator acessa a tela de usuários e a opção cadastrar;

3. O ator insere os dados; 4. O ator define um *login* e uma senha para o usuário; 5. O ator define as permissões para conceder ao usuário; 6. O sistema pede a confirmação ao ator e grava os dados. **Pós-Condição:** Usuário cadastrado.

Quadro 9 - Caso de uso Manter registro de usuários

O Quadro 10 explicita o caso de uso 'Visualizar relatórios'.

Caso de uso: Visualizar relatórios. **Descrição:** Visualizar os relatórios do confinamento, individual ou por lote, que podem ser de evolução, consumo, investimentos e lucros, eficiência e históricos de manejos. **Evento Iniciador:** Necessidade de informações. **Ator:** Administrador. **Pré-condição:** Relatório gerado/visualizado. **Sequência de Eventos:** 1. O ator acessa a tela de relatórios; 2. O ator escolhe os parâmetros de filtragem para o relatório; 3. O ator visualiza e escolhe se deseja salvar e/ou imprimir; 4. O sistema gera o arquivo e/ou a página de impressão. **Pós-Condição:** Relatório gerado e visualizado.

Quadro 10 - Caso de uso Visualizar relatórios

O caso de uso 'Manter registro de manejos' é apresentado no Quadro 11.

Caso de uso: Manter registro de manejos. **Descrição:** Registrar os manejos realizados nos animais, individualmente ou por lote. **Evento Iniciador:** Manejo realizado. **Ator:** Funcionário, Administrador. **Pré-condição:** Dados do manejo coletados. **Sequência de Eventos:** 1. O ator acessa a tela de manejos de lote ou de manejos de animal; 2. O ator escolhe alimentação, diagnóstico, aferição de massa corporal ou troca de lote; 3. O ator insere os dados; 4. O sistema pede a confirmação ao ator e grava os dados. **Pós-Condição:** Manejo de animal registrado.

Quadro 11 - Caso de uso Manter registro de manejos

4.2.5 Diagrama de Classes

A Figura 3 ilustra o diagrama de classes conceitual com supressão de métodos, proposto para análise na implementação do sistema.

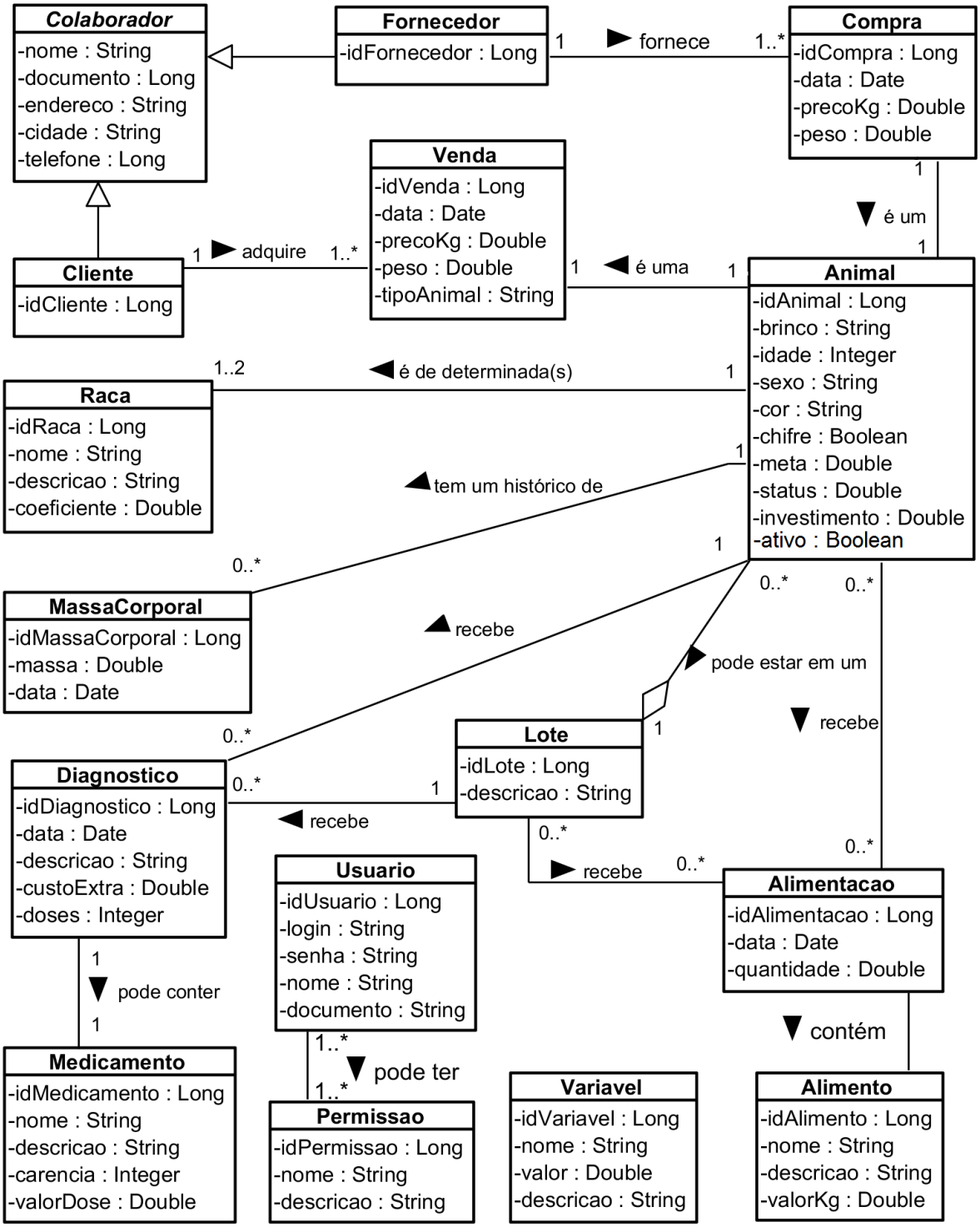

Figura 3 - Diagrama de classes conceitual
Fonte: Autoria própria.

As classes do diagrama da figura anterior estão descritas a seguir, nos Quadros 12 a 25.

Identificação:	Colaborador
Descrição:	Classe abstrata com duas especializações: ‘Fornecedor’ (classe que se relaciona com Compra) e ‘Cliente’ (classe que se relaciona com Venda). Em cada subclasse há um atributo de identificação que diferencia as mesmas.
Atributos:	-nome (String): nome da pessoa ou empresa; -documento (String): CPF ou CNPJ; -endereco (String): endereço; -cidade (String): cidade; -telefone (String): para contato.
Métodos:	adicionar(), atualizar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF01.

Quadro 12 - Descrição da classe Colaborador

Identificação:	Compra
Descrição:	Registro individual dos dados referentes à compra de cada animal.
Atributos:	-idCompra (Long): número de identificação da compra; -data (Date): data da negociação e inserção do animal no confinamento; -valorTotal (Double): valor total pelo animal.
Métodos:	adicionar(), atualizar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF04, RF05, RF15, RNF02.

Quadro 13 - Descrição da classe Compra

Identificação:	Venda
Descrição:	Registro individual dos dados referentes à venda de cada animal.
Atributos:	-idVenda (Long): número de identificação da venda; -data (Date): da venda e retirada do animal no confinamento; - valorTotal (Double): valor total pelo animal; -massa (Double): valor da massa do animal vivo ou da sua carcaça em quilogramas; -tipoAnimal (String): situação em que o animal foi vendido (saiu do confinamento): abatido, vivo, descartado etc.
Métodos:	adicionar(), atualizar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF14, RF15, RNF02.

Quadro 14 - Descrição da classe Venda

Identificação:	Animal
Descrição:	Registro individual dos dados referentes ao processo de confinamento de cada animal.
Atributos:	-idAnimal (Long): número de identificação do animal; -brinco(String): o código que está impresso no brinco do animal; -idade (Integer): idade do animal em meses;

	-sexo (String): boi, touro, novilho, novilha, vaca, matriz (vaca leiteira descartada de sua finalidade original) etc.; -pelagem (String): cores predominantes; -chifre (Boolean): se o animal possuir chifres / cornos; -meta (Double): massa do animal quando estiver gordo; -status (Double): percentagem da evolução de massa corporal; -investimento (Double): soma de todos os valores de custos associados ao animal e estimáveis dentro do sistema; -ativo (Boolean): se o animal está confinado ou já foi vendido.
Métodos:	adicionar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF04, RF05, RF06, RF07, RF13, RF15, RNF02.

Quadro 15 - Descrição da classe Animal

Identificação:	Raca
Descrição:	Registro das raças bovinas (nesse caso, espécies descendentes das subespécies de *Bos taurus*) presentes no confinamento, feito para ser associado ao registro individual de cada animal e possibilitar o cálculo da meta de engorda.
Atributos:	-idRaca (Long): número identificador da raça; -nome (String): nome usual da raça; -descricao (String): características da raça; -coeficiente (Double): variável utilizada no cálculo da meta.
Métodos:	adicionar(), remover(), atualizar(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF02, RF05, RF06.

Quadro 16 - Descrição da classe Raca

Identificação:	MassaCorporal
Descrição:	Registro das aferições de massa corporal de cada bovino.
Atributos:	-idMassaCorporal (Long): número identificador da aferição; -massa (Double): valor da massa corporal em quilogramas; -data (Date): dia em que foi realizada a aferição.
Métodos:	adicionar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF05, RF07, RF08.

Quadro 17 - Descrição da classe MassaCorporal

Identificação:	Diagnostico
Descrição:	Registro de diagnóstico (vacinação, pulverização etc.) efetuado individualmente ou em um lote.
Atributos:	-idDiagnostico (Long): número identificador do diagnóstico; -data (Date): dia em que foi efetuado; -descricao (String): descrição e motivos da realização; -custoExtra (Double): eventual custo com profissionais veterinários e/ou medicamentos que não fazem parte do estoque; -doses (Integer): eventual quantidade de doses do medicamento utilizado.

Métodos:	adicionar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF05, RF12.

Quadro 18 - Descrição da classe Diagnostico

Identificação:	Alimentacao
Descrição:	Registro das alimentações servidas aos lotes e cada bovino isolado que está retido em uma baia individual.
Atributos:	-idAlimentacao (Long): número identificador da alimentação; -quantidade (Double): do alimento utilizado, em quilogramas; -data (Date): dia em que foi registrada a alimentação.
Métodos:	adicionar(), remover(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF05, RF09, RF10.

Quadro 19 - Descrição da classe Alimentacao

Identificação:	Lote
Descrição:	Registro dos lotes alocados no confinamento.
Atributos:	-idLote (Long): número identificador do lote; -descricao (String): características e informações sobre o lote.
Métodos:	adicionar(), atualizar(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF03, RF05.

Quadro 20 - Descrição da classe Lote

Identificação:	Medicamento
Descrição:	Registro dos medicamentos utilizados no confinamento.
Atributos:	-idMedicamento (Long): número identificador do medicamento; -nome (String): nome comercial do medicamento; -descricao (String): informações, posologia; -carencia (Integer): prazo, em dias, em que o animal que recebe o medicamento deve ficar em quarentena; -valorDose (Double): valor da mililitro ou unidade de medida do medicamento.
Métodos:	adicionar(), remover(), atualizar(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF11, RF12.

Quadro 21 - Descrição da classe Medicamento

Identificação:	Alimento
Descrição:	Registro dos alimentos utilizados no confinamento.
Atributos:	-idAlimento (Long): número identificador do alimento; -nome (String): nome utilizado; -descricao (String): informações, características; -valorKg (Double): valor do quilograma do alimento.
Métodos:	adicionar(), remover(), atualizar(), listarTodos(), pesquisar() etc.
Requisitos envolvidos:	RF09, RF10.

Quadro 22 - Descrição da classe Alimento

Identificação:	Usuario
Descrição:	Usuarios que terão acesso ao sistema.
Atributos:	-idUsuario (Long): número identificador do usuario; -login (String): termo para acesso ao sistema; -senha (String): passe do acesso; -nome (String): nome do usuário; -documento (Long): um documento do usuário.
Métodos:	adicionar(), atualizar(), remover(), listarTodos() etc.
Requisitos envolvidos:	RF16, RNF01.

Quadro 23 - Descrição da classe Usuario

Identificação:	Permissao
Descrição:	Permissões que os usuários terão para acessar as funcionalidades do sistema.
Atributos:	-idPermissao (Long): número identificador do usuário; -nome (String): nome da permissão; -descricao (String): características da permissão.
Métodos:	adicionar(), atualizar(), remover() etc.
Requisitos envolvidos:	RF16, RNF01.

Quadro 24 - Descrição da classe Permissao

Identificação:	Variavel
Descrição:	Variáveis, escolhas de usuários para configurações necessárias ao funcionamento do sistema.
Atributos:	-idVariavel (Long): número identificador da variável; -nome (String): nome da variável; -valor (Double): valor da variável; -descricao (String): características da variável.
Métodos:	listarTodos(), atualizar() etc.
Requisitos envolvidos:	RF06, RF15.

Quadro 25 - Descrição da classe Variavel

4.2.6 Diagrama Entidade-Relacionamento

A Figura 4 ilustra o diagrama de entidade-relacionamento que representa o banco de dados identificado para a manutenção das informações do sistema proposto.

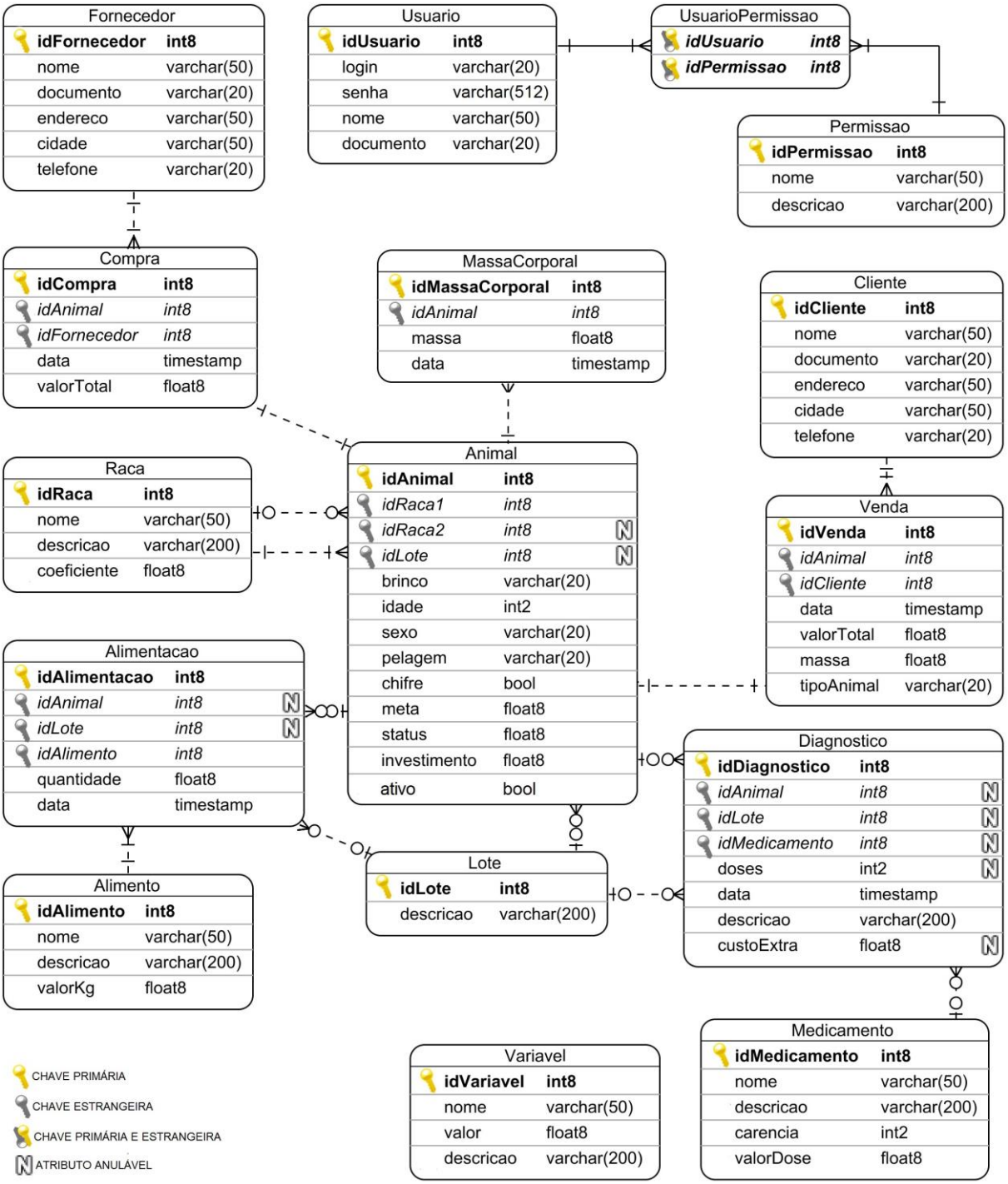

Figura 4 - Diagrama Entidade-Relacionamento
Fonte: Autoria própria.

A entidade ‘Fornecedor’ armazena dados pessoais e de contato das fazendas e indivíduos responsáveis pelo fornecimento dos animais que irão passar pelo processo de engorda. Já a entidade ‘Cliente’ armazena dados de mesmo teor, mas, pertinentes aos abatedouros e atravessadores que compram os animais confinados.

Cada registro na entidade ‘Compra’ contém os dados da negociação de cada animal e a referência ao seu respectivo fornecedor. Já na entidade central ‘Animal’

estão todas as características de cada animal e informações sobre sua evolução durante a estadia no confinamento. A coluna ‘idRaca2’ admite valores nulos porque para cada animal pode-se identificar uma única raça ou até duas, no caso de um bovino mestiço. Na entidade ‘Raca’ são gravadas informações de cada raça bovina — descendentes das subespécies de *Bos taurus* — que pode estar relacionada a algum animal.

Cada agrupamento de animais em uma área em comum, chamado de lote, é registrado na entidade ‘Lote’. Se um animal estiver em uma baia individual, logo, ele não está alocado em nenhum lote, por isso, a chave estrangeira que referencia o lote na entidade ‘Animal’ admite valor nulo.

Os suprimentos consumidos no confinamento, que são os alimentos e os medicamentos, estão registrados, respectivamente, nas entidades ‘Alimento’ e ‘Medicamento’.

As aferições de massa corporal dos animais são registradas na entidade ‘MassaCorporal’. O histórico das alimentações é armazenado na entidade ‘Alimentacao’. As alimentações podem acontecer a um lote ou a um animal isolado em uma baia individual, assim, em um registro na entidade ‘Alimentacao’, a coluna ‘idAnimal’ será nula quando ‘idLote’ não for, e vice-versa.

Quando os bovinos adoecem, ou como medida de prevenção, é efetuado algum tipo de diagnóstico, como um exame clínico e/ou aplicação de medicamentos. Esses fatos são gravados na entidade ‘Diagnostico’. Como pode ou não haver uso de medicamentos e custos externos ao confinamento, como honorários de um profissional veterinário, as colunas ‘idMedicamento’, ‘doses’ e ‘custoExtra’, são opcionais. Quando o diagnóstico for concretizado em um lote, o campo ‘idAnimal’ será nulo; e quando o diagnóstico for concretizado em um animal, o campo ‘idLote’ será atribuído nulo.

Quando vendido, cada animal tem seus dados de negociação armazenados na entidade ‘Venda’, que com a coluna ‘tipoAnimal’, permite especificar se esse foi vendido vivo, abatido, descartado ou em outra situação prevista.

Na entidade ‘Usuário’ estão registrados os dados e as credenciais do funcionários que terão acesso ao sistema. Em ‘Permissao’ e ‘UsuarioPermissao’ são armazenadas as permissões que os usuários têm. As variáveis necessárias ao funcionamento do sistema, como as utilizadas no cálculo da meta de engorda, são gravadas na entidade ‘Variavel’.

Algumas rotinas do banco de dados irão automatizar a manutenção das colunas ‘status’, ‘investimento’ e ‘ativo’ da entidade ‘Animal’. A adoção desses procedimentos refreia o fluxo de entrada e saída do banco de dados e contribui para melhoria de desempenho das consultas aos registros.

A coluna ‘status’ depende diretamente da coluna ‘meta’ e de registros de aferições de massa corporal. O *trigger* apresentado na Listagem de Código 1, executado a cada inserção na entidade ‘MassaCorporal’, compara a atual aferição do animal com sua meta e sua primeira massa corporal registrada para atribuir um *status* em percentagem de evolução. Existem também outros *triggers* de mesmo teor, um atualiza o *status* após exclusões de registros e outro impede que o primeiro registro de massa corporal de um animal seja excluído da base de dados.

```
CREATE FUNCTION ATUALIZA_STATUS_INSERT() RETURNS TRIGGER AS $$
DECLARE -- variáveis necessárias aos cálculos
valor_meta FLOAT8; -- meta do animal
p_massa FLOAT8; -- primeira massa corporal registrada
status_atual FLOAT8; -- status que será atribuído

BEGIN
      -- valoriza as variáveis
      SELECT INTO p_massa m.massa FROM massacorporal m WHERE
m.idmassacorporal = (SELECT MIN(mc.idmassacorporal) FROM massacorporal mc
WHERE mc.idanimal = NEW.idanimal);
      SELECT INTO valor_meta a.meta FROM animal a WHERE a.idanimal =
NEW.idanimal;
      SELECT INTO status_atual ((NEW.massa - p_massa) * 100) / (valor_meta
- p_massa);

      -- atualiza o status
      UPDATE animal SET status = status_atual WHERE idanimal = NEW.idanimal;
      -- informa sobre as mudanças
      RAISE NOTICE 'O status de engorda do animal % foi atualizado para %',
NEW.idanimal, status_atual;
      RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER TR_ATUALIZA_STATUS_INSERT AFTER INSERT ON massacorporal
FOR EACH ROW EXECUTE PROCEDURE ATUALIZA_STATUS_INSERT();
```

Listagem de Código 1 - *Trigger* que automatiza o cálculo de *status* de engorda

A coluna ‘investimento’ da entidade ‘Animal’ corresponde a soma do valor pago por um animal no momento da compra e seu consumo estimado de suprimentos. A cada inserção na entidade ‘Alimentacao’, o *trigger* detalhado na Listagem de Código 2 irá verificar os custos com o alimento servido e acumular o valor correspondente no investimento de cada animal envolvido. Um procedimento para tratar as exclusões nessa entidade também foi criado.

```
CREATE FUNCTION ATUALIZA_INVESTIMENTO_ALIMENTACAO_INSERT() RETURNS TRIGGER
AS $$

DECLARE -- variáveis necessárias aos cálculos
total FLOAT8; -- custo total da alimentação
qtd_animais FLOAT8; -- quantidade de animais do lote selecionado
custo_medio FLOAT8; -- custo médio simples para cada animal do lote

BEGIN

      -- valoriza a variável com o valor estimado do custo total da
alimentação
      SELECT INTO total (NEW.quantidade * a.valorkg) FROM alimento a WHERE
a.idalimento = NEW.idalimento;

      -- se a alimentação foi servida a um lote
      IF (NEW.idlote IS NOT NULL) THEN

            -- valoriza as variáveis
            SELECT INTO qtd_animais COUNT(idanimal) FROM animal a WHERE
a.idlote = NEW.idlote AND a.ativo = TRUE;
            SELECT INTO custo_medio (total / qtd_animais);

            -- acumula o custo médio no investimento de cada animal que
está no lote selecionado
            UPDATE animal SET investimento = (investimento + custo_medio)
WHERE idlote = NEW.idlote AND ativo = TRUE;

            -- informa sobre as mudanças
            RAISE NOTICE 'O valor do investimento de cada animal do lote %
foi atualizado', NEW.idlote;

      -- senão, a alimentação foi servida para um único animal
      ELSE
            -- acumula o custo total no investimento do animal que foi
selecionado
            UPDATE animal SET investimento = (investimento + total) WHERE
idanimal = NEW.idanimal;

            -- informa sobre as mudanças
            RAISE NOTICE 'O valor do investimento do animal % foi
atualizado', NEW.idanimal;
      END IF;

      RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER TR_ATUALIZA_INVESTIMENTO_ALIMENTACAO_INSERT AFTER INSERT ON
alimentacao
FOR EACH ROW EXECUTE PROCEDURE ATUALIZA_INVESTIMENTO_ALIMENTACAO_INSERT();
```

Listagem de Código 2 - *Trigger* que atualiza investimento conforme alimentações

Para as inserções na entidade ‘Diagnostico’ existe o *trigger* retratado na Listagem de Código 3 que faz a estimativa dos custos e acumula no investimento dos animais envolvidos. Da mesma forma, existe um procedimento que decrementa os investimentos quando um registro de diagnóstico é excluído do banco de dados.

```
CREATE FUNCTION ATUALIZA_INVESTIMENTO_DIAGNOSTICO_INSERT() RETURNS TRIGGER
AS $$

DECLARE -- variáveis necessárias aos cálculos
total FLOAT8; -- custo total do diagnóstico
qtd_animais FLOAT8; -- quantidade de animais do lote selecionado
custo_medio FLOAT8; -- custo médio simples para cada animal do lote

BEGIN
      -- se houve utilização de medicamento ou custo extra
      IF ((NEW.idmedicamento IS NOT NULL) OR (NEW.custoextra IS NOT NULL))
THEN
            -- valoriza a variável com o valor total estimado do diagnóstico
            IF (NEW.idmedicamento IS NOT NULL) THEN
                  SELECT INTO total ((NEW.doses * m.valordose) +
NEW.custoextra) FROM medicamento m WHERE m.idmedicamento =
NEW.idmedicamento;
            ELSE
                  SELECT INTO total NEW.custoextra;
            END IF;

            -- se o diagnóstico foi concretizado em um lote
            IF (NEW.idlote IS NOT NULL) THEN

                  -- valoriza as variáveis
                  SELECT INTO qtd_animais COUNT(a.idanimal) FROM animal a
WHERE a.idlote = NEW.idlote AND a.ativo = TRUE;

                  SELECT INTO custo_medio (total / qtd_animais);

                  -- acumula o custo médio no investimento de cada animal
que está no lote selecionado
                  UPDATE animal SET investimento = (investimento +
custo_medio) WHERE idlote = NEW.idlote AND ativo = TRUE;

                  -- informa sobre as mudanças
                  RAISE NOTICE 'O valor do investimento de cada animal do
lote % foi atualizado', NEW.idlote;

            -- senão, o diagnóstico foi aplicado em um animal
            ELSE
                  -- acumula o custo total no investimento do animal que
foi selecionado
                  UPDATE animal SET investimento = (investimento + total)
WHERE idanimal = NEW.idanimal;

                  -- informa sobre as mudanças
                  RAISE NOTICE 'O valor do investimento do animal % foi
atualizado', NEW.idanimal;

            END IF;
      END IF;

      RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER TR_ATUALIZA_INVESTIMENTO_DIAGNOSTICO_INSERT AFTER INSERT ON
diagnostico
FOR EACH ROW EXECUTE PROCEDURE ATUALIZA_INVESTIMENTO_DIAGNOSTICO_INSERT();
```

Listagem de Código 3 - *Trigger* que atualiza investimento conforme diagnósticos

A coluna ‘ativo’ da entidade ‘Animal’, utilizada em diversas consultas aos dados dos animais que ainda não foram vendidos, é automaticamente atualizada. Procedimentos do tipo *trigger* associados à entidade ‘Venda’ são responsáveis por essa atividade.

Existem ainda outras rotinas no banco de dados, como *procedures* e *views,* que também automatizam algumas tarefas e processam dados que demandam consultas mais complexas para serem alcançados. Tais procedimentos foram criados para viabilizar a contração do fluxo de comunicação entre a aplicação de *software* e o banco de dados e, consequentemente, proporcionar melhorias de desempenho em ambos os lados.

A Listagem de Código 4 apresenta um conjunto de 3 *views* que são associadas e utilizadas para retornar uma lista dos animais que estão em quarentena na data corrente, fato que acontece após a aplicação de um medicamento que requer prazo de carência.

```
CREATE VIEW qrta AS
SELECT d.iddiagnostico, d.idanimal, d.idmedicamento,
         m.nome AS nomemedicamento, m.carencia, d.data,
         d.descricao, a.brinco, a.sexo, a.pelagem
FROM diagnostico d
INNER JOIN medicamento m
ON d.idmedicamento = m.idmedicamento AND d.idanimal IS NOT NULL
INNER JOIN animal a
ON a.idanimal = d.idanimal AND a.ativo = TRUE
WHERE d.idmedicamento IN
(SELECT m.idmedicamento FROM medicamento m WHERE m.carencia > 0);

CREATE VIEW qrtb AS
SELECT q.iddiagnostico, q.idanimal, q.idmedicamento, q.nomemedicamento,
       q.carencia, q.data::TIMESTAMP::DATE AS datarealizada,
       ((q.data::TIMESTAMP::DATE) + q.carencia) AS dataalta,
         q.descricao, q.brinco, q.sexo, q.pelagem
FROM qrta q;

CREATE VIEW quarentena AS
SELECT * FROM qrtb WHERE dataalta >= CURRENT_DATE;
```

Listagem de Código 4 - Conjunto de *views* que retornam quarentenas ativas

4.3 IMPLEMENTAÇÃO DO SISTEMA

O primeiro passo do desenvolvimento do sistema foi a estruturação do banco de dados da aplicação, com seu conjunto de entidades, procedimentos e rotinas. O PgAdmin III foi utilizado para a manipulação do PostgreSQL, inicialmente em uma instância local para o desenvolvimento, e após a conclusão, em uma instância no modelo de *Database as a Service* da OpenShift.

A codificação do sistema foi feita em linguagem Java no ambiente de desenvolvimento integrado do Eclipse, em um projeto *web* do Apache Maven para a especificação do *JavaServer Faces*. O Mojarra foi escolhido como implementação da especificação do JSF e o OmniFaces como um conjunto auxiliar de utilitários.

Alguns *frameworks* foram empenhados para simplificar o desenvolvimento e proporcionar maior produtividade. O Hibernate ORM permitiu o mapeamento objeto-relacional e a persistência no banco de dados. O Hibernate Validator facilitou a validação de restrições de atributos e objetos, enquanto o Lombok automatizou a criação de métodos básicos de classes Java.

O conjunto de componentes do PrimeFaces foi utilizado para o desenvolvimento da interface gráfica de usuário, aliado a biblioteca do iText para geração de relatórios em formato PDF. O Weld proporcionou a injeção de dependências e contextos (CDI) nas classes e objetos do Java. O Spring Security viabilizou o controle de acesso dos usuários do sistema e suas permissões. O Apache Log4j propiciou a criação de *logs* de erros, atividades e mensagens do sistema em arquivos separados por data.

A organização da estrutura de pacotes do projeto é ilustrada na Figura 5.

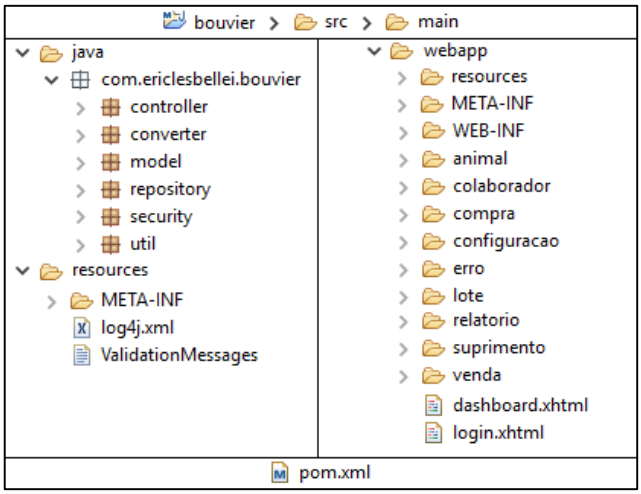

Figura 5 - Estrutura de pacotes do projeto
Fonte: Reprodução de tela.

As classes do pacote ‘controller’ são *beans* gerenciados pela CDI que atuam como *ManagedBeans* do JSF, no intermédio entre regras de negócio para acesso aos dados e as telas que fazem a interface gráfica de usuário, situadas dentro do diretório ‘webapp’. Uma das classes do pacote ‘controller’, detalhada na Listagem de Código 5, é responsável por atualizar objetos que contêm referências às folhas de estilo do *design* da interface gráfica que permitem ao usuário customizar suas cores e localização de componentes, conforme as ilustrações da Figura 9 e da Figura 10.

```
package com.ericlesbellei.bouvier.controller;

import javax.annotation.PostConstruct;
import javax.enterprise.context.SessionScoped;
import javax.inject.Named;
import lombok.*;
// outro imports

@Named
@SessionScoped
public class LayoutPreferences implements Serializable {

    private static final long serialVersionUID = 1L;

    @Getter
    private Map<String,String> themeColors;

    @Getter @Setter
    private String theme = "teal";

    @Getter
    private String menuClass = null;

    @Getter @Setter
    private String profileMode = "overlay";

    @Setter
    private String menuLayout = "static";

    @PostConstruct
    public void init() {
        themeColors = new HashMap<String,String>();
        themeColors.put("blue", "#03A9F4");
        themeColors.put("green", "#4CAF50");
        themeColors.put("teal", "#009688");
    }

    public String getMenuLayout() {
        if(this.menuLayout.equals("static"))
            return "menu-layout-static";
        else if(this.menuLayout.equals("overlay"))
            return "menu-layout-overlay";
        else if(this.menuLayout.equals("horizontal"))
            return "menu-layout-static menu-layout-horizontal";
        else
            return "menu-layout-static";
    }
```

```
    public void setLightMenu() {
        this.menuClass = null;
    }

    public void setDarkMenu() {
        this.menuClass = "layout-menu-dark";
    }
}
```

Listagem de Código 5 - Classe controladora da customização da interface gráfica

No pacote ‘converter’ estão as classes que atuam como conversores do JSF, mas para os objetos Java personalizados que irão ser carregados em caixas de seleção na interface gráfica, como lotes, alimentos, medicamentos etc.

As classes do pacote ‘model’ representam as entidades do banco de dados da aplicação. As classes do pacote ‘repository’ contêm métodos para consulta e manipulação de informações no banco de dados, alinhados às regras de negócio. A Listagem de Código 6 exemplifica o repositório do cadastro de usuários do sistema. Todas as consultas irão rejeitar o usuário com número identificador igual a 1, pois este foi criado para ser o *SysAdmin* padrão, utilizado para cadastrar o primeiro usuário efetivo do sistema.

```
package com.ericlesbellei.bouvier.repository;

import javax.inject.Inject;
import com.ericlesbellei.bouvier.model.Usuario;
import com.ericlesbellei.bouvier.util.jpa.Transactional;
//outros imports

public class UsuarioRepository implements Serializable{
     private static final long serialVersionUID = 1L;

        @Inject
        private EntityManager em;

        public Usuario getById(Long id){
           if(id != 1)
              return em.find(Usuario.class, id);
           else
              return null;
        }

        public List<Usuario> getAll(){
           return em.createQuery("SELECT u FROM Usuario u WHERE (u.idusuario
!= 1) ORDER BY u.nome ASC", Usuario.class).getResultList();
        }

        public List<Usuario> getByNome(String nome){
           String sql = "SELECT u FROM Usuario u WHERE UPPER(u.nome) LIKE
:n AND (u.idusuario != 1) ORDER BY u.nome";
           TypedQuery<Usuario> query = em.createQuery(sql, Usuario.class);
           query.setParameter("n", "%" + nome.trim().toUpperCase() + "%");
           return query.getResultList();
        }
```

```
        @Transactional
        public Usuario salvar(Usuario c){
             return em.merge(c);
        }

        @Transactional
        public void excluir(Usuario c) throws Exception{
             c = this.getById(c.getIdusuario());
             em.remove(c);
             em.flush();
        }
}
```

Listagem de Código 6 - Repositório de cadastros de usuários

O pacote ‘security’ contém as classes utilizadas para filtrar os usuários que fazem *login* e suas permissões de visualização e utilização de dados e informações no sistema, conforme atribuído pelo administrador.

No pacote ‘util’ estão classes e interfaces utilitárias, como as produtoras da anotação ‘@Transactional’ — que intercepta métodos de acesso ao banco de dados para iniciar, tratar e finalizar transações — e a classe que manipula as exceções que ocorrem no sistema e as armazena no *log* diário. Outra classe, apresentada na Listagem de Código 7, provisiona ao CDI injetar dependências de objetos Java do tipo ‘EntityManager’.

```
package com.ericlesbellei.bouvier.util.jpa;

import javax.enterprise.context.*;
import javax.enterprise.inject.*;
import javax.persistence.*;

@ApplicationScoped
public class EntityManagerProducer {

      private EntityManagerFactory factory;

      public EntityManagerProducer() {
            factory = Persistence.createEntityManagerFactory("BouvierPU");
      }

      @Produces
      @RequestScoped
      public EntityManager createEntityManager() {
            return factory.createEntityManager();
      }

      public void closeEntityManager(@Disposes EntityManager manager) {
            manager.close();
      }
}
```

Listagem de Código 7 - Classe que permite CDI de objetos *EntityManager*

No diretório ‘resources’ estão alguns arquivos de configurações, como de conexão à base de dados, mensagens de validação e criação de *logs* da aplicação.

O diretório ‘webapp’ contém arquivos responsáveis pela criação da interface gráfica e configuração do conteúdo *web*, agrupados por características em comum, dentro de diretórios secundários. Um *template*, detalhado na Listagem de Código 8, foi criado para servir como base para as páginas do sistema, e assim, evitar a repetição de código. O *template* inclui a estrutura básica de qualquer tela do sistema, composta pela barra superior, menus de navegação, uma janela *pop-up* para confirmação de ações e um componente que exibe ícone de *status* de carregamento enquanto o sistema executa chamadas *Asynchronous Javascript and XML* (AJAX).

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">

<h:head>
      <f:facet name="first">
            <meta http-equiv="Content-Type" content="text/html;
             charset=UTF-8" />
            <meta name="theme-color"
             content="#{layoutPreferences.themeColors[layoutPreferences.
             theme]}" />
      </f:facet>
      <title>Bouvier</title>
      <h:outputScript name="js/nanoscroller.js" library="ultima" />
      <h:outputScript name="js/layout.js" library="ultima" />
      <h:outputScript name="js/ripple.js" library="ultima" />
      <h:outputScript name="js/swipe.js" library="ultima" />
      <link rel="shortcut icon"
       href="#{resource['ultima:images/favicon.png']}" type="img/png" />

      <!-- FRAGMENTO PARA CABEÇALHO DO HTML -->
      <ui:insert name="head" />
</h:head>

<h:body styleClass="main-body layout-compact">

      <div class="layout-wrapper #{layoutPreferences.menuLayout}">
            <ui:include src="./topbar.xhtml" />
            <ui:include src="./menu.xhtml" />

            <div class="layout-main">
                  <!-- FRAGMENTO PARA CORPO DO HTML -->
                  <ui:insert name="content" />
            </div>
      </div>

      <h:form>
            <p:confirmDialog global="true" >
                  <p:commandButton value="Sim" type="button"
                   styleClass="ui-confirmdialog-yes green-btn" />
```

```
                    <p:commandButton value="Não" type="button"
                     styleClass="ui-confirmdialog-no red-btn" />
              </p:confirmDialog>
       </h:form>

       <p:ajaxStatus class="ajax-status">
              <f:facet name="start">
                    <i class="fa fa-refresh fa-spin fa-3x fa-fw
                     ajax-loader" aria-hidden="true" />
              </f:facet>

              <f:facet name="complete">
                    <h:outputText value="" />
              </f:facet>
       </p:ajaxStatus>

       <h:outputStylesheet name="css/nanoscroller.css" library="ultima" />
       <h:outputStylesheet name="css/animate.css" library="ultima" />
       <h:outputStylesheet name="css/ripple.css" library="ultima" />
       <h:outputStylesheet name="css/layout-
        #{layoutPreferences.theme}.css" library="ultima" />
</h:body>
</html>
```

Listagem de Código 8 - *Template* das páginas do sistema

4.4 APRESENTAÇÃO DO SISTEMA

Alinhado à proposta do sistema, foi eleito para seu nome ‘Bouvier’, termo da língua francesa que significa ‘vaqueiro, pastor, aquele que cuida de um rebanho’.

A interface gráfica de usuário foi projetada para adaptar-se às diferentes resoluções de tela e dispositivos que podem ser utilizados como meio de acesso ao sistema. Dinamicamente o conteúdo da interface é estruturado e apresentado conforme as restrições da tela em que precisa ser exibido, sem prejuízo à sua experiência de usabilidade. O *design* da interface é alicerçado nas características de responsividade das folhas de estilo do Ultima Theme do PrimeFaces. Para ilustrar tais características, as figuras deste texto que representam capturas de tela do sistema demonstram a estruturação do mesmo conteúdo quando exibido em um navegador de máquina *desktop* e um navegador de dispositivo móvel.

O mapa hierárquico de navegação apresentado na Figura 6 esquematiza a estrutura de acesso a todas as telas de funcionalidades do sistema. Essa estrutura é exibida somente quando um usuário com permissões de administrador efetua acesso. Para usuários com permissões específicas, como funcionários e supervisores, as telas não compatíveis com suas permissões ficam automaticamente ocultas e bloqueadas.

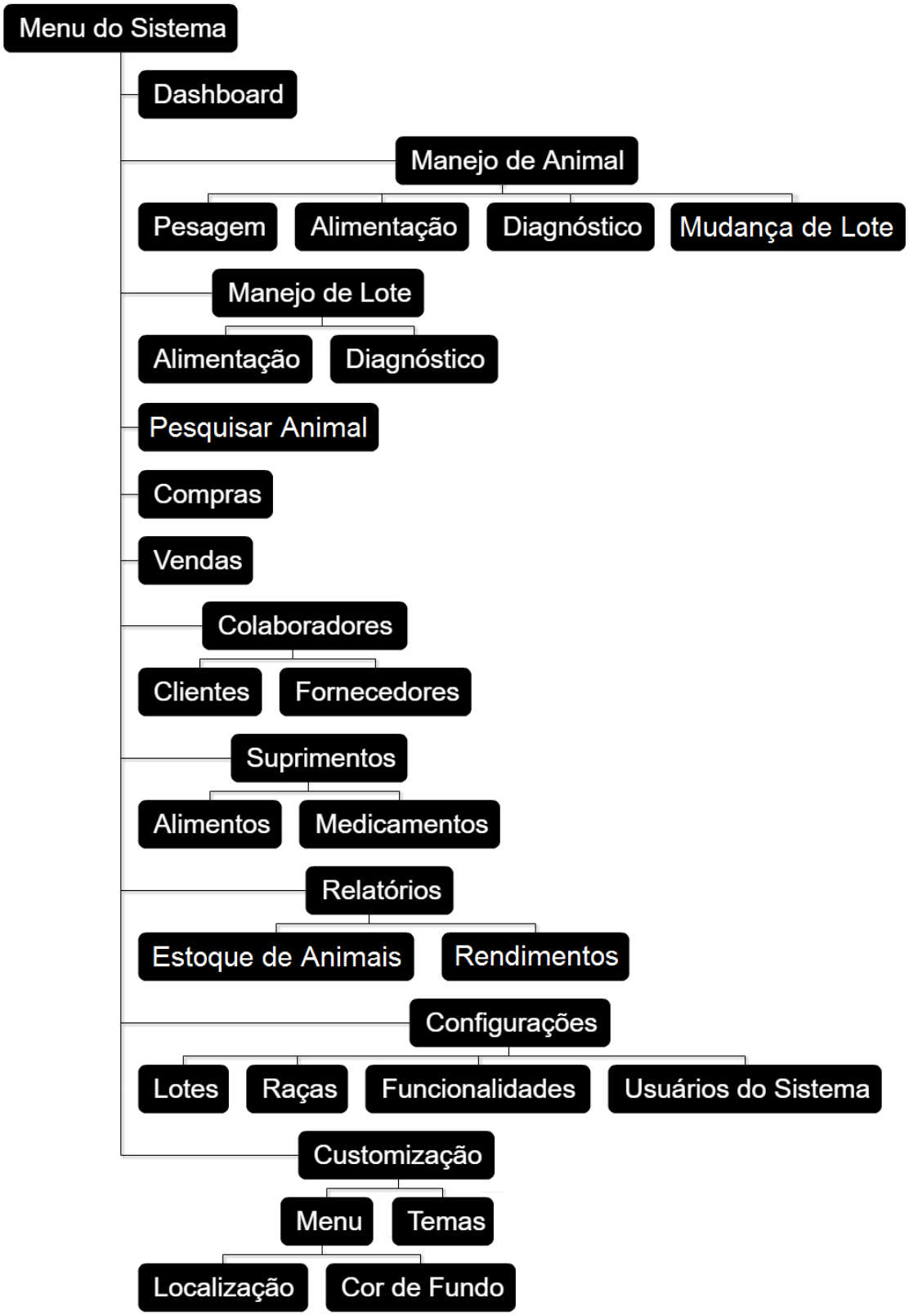

Figura 6 - Mapa hierárquico de navegação do sistema
Fonte: Autoria própria.

O acesso ao sistema é concretizado por meio de validação de usuário. A Figura 7 apresenta a tela inicial do sistema que o usuário utiliza para efetuar *login*.

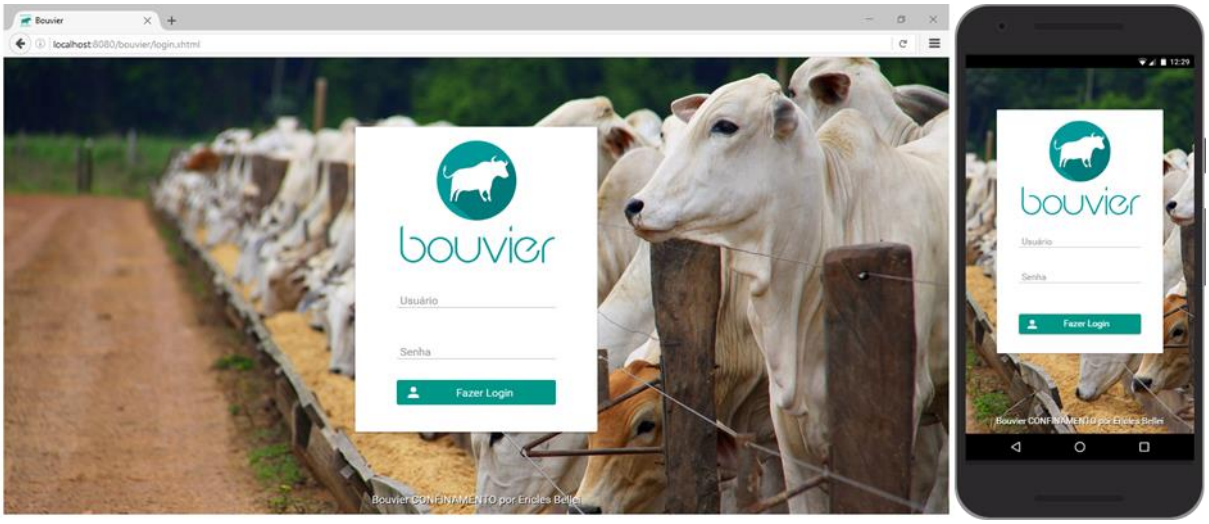

Figura 7 - Tela de *login*
Fonte: Reproduções de telas.

Após o *login* o sistema exibe a tela de *dashboard*, ilustrada na Figura 8. Essa tela contêm algumas informações pontuais sobre a situação do confinamento, como gráficos sobre a composição do estoque e uma tabela com os animais que estão em quarentena na data corrente. Na mesma tela é exibido também outra tabela com sugestões de venda de animais com o *status* de engorda mais adiantado, além de uma lista com os alimentos mais consumidos, quantidade servida e gasto estimado com cada um nos últimos 30 dias.

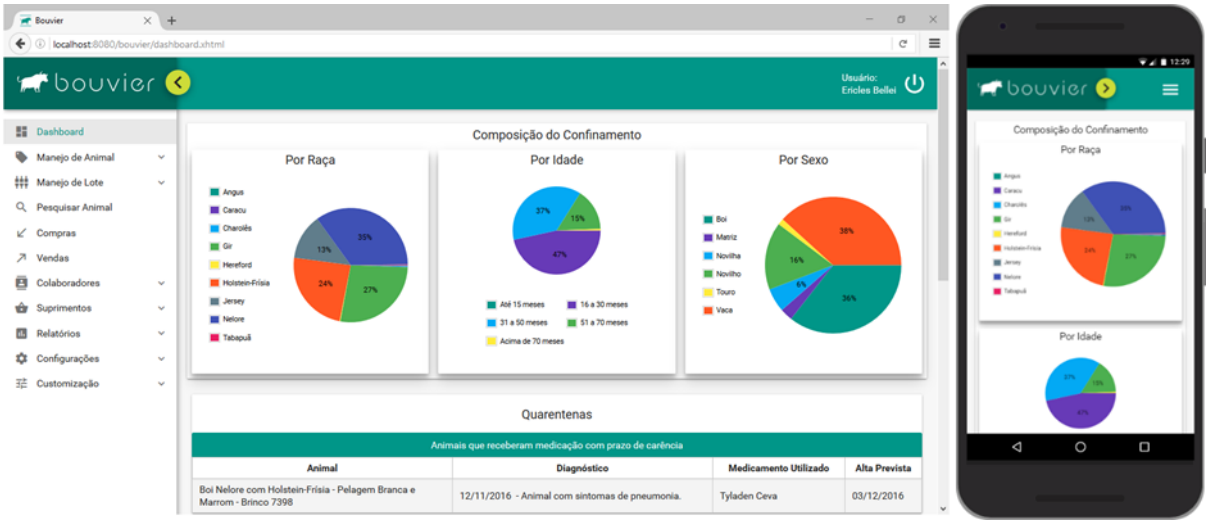

Figura 8 - Tela de *dashboard*
Fonte: Reproduções de telas.

Conforme a Figura 8, o *layout* do sistema está setorizado em três partes: a barra superior, a extremidade esquerda — onde há um menu baseado no mapa hierárquico da Figura 6 — e a extremidade direita, onde é exibido o conteúdo das

páginas. Em dispositivos que possuem a resolução de tela mais limitada, o menu que fica na extremidade esquerda torna-se deslizante e é exibido somente após o toque/clique no botão da barra superior que desempenha essa funcionalidade.

A barra superior é sempre fixa e visualizada, indiferentemente da resolução da tela onde é exibida. Ela apresenta um botão que exibe ou oculta o menu da extremidade esquerda, o nome do usuário que fez *login* e um botão para o *logout*.

A Figura 9 demonstra algumas variantes de customização e o comportamento da interface gráfica do sistema quando o menu da extremidade esquerda torna-se deslizante e parte da barra superior, que exibe o botão de *logout* e o nome de usuário, fica agrupada na lateral direita.

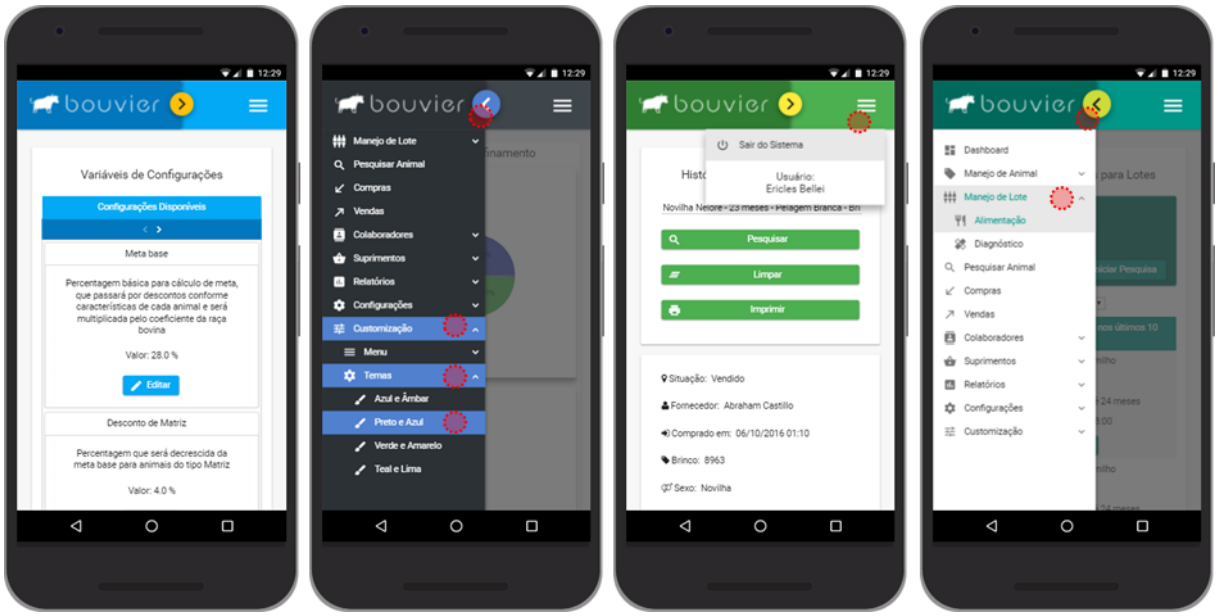

Figura 9 - Variantes e comportamento da interface gráfica em telas menores
Fonte: Reproduções de telas.

Por padrão o menu fica na extremidade esquerda, mas, o usuário tem a opção de colocá-lo na parte superior da tela, conforme as reproduções da Figura 10.

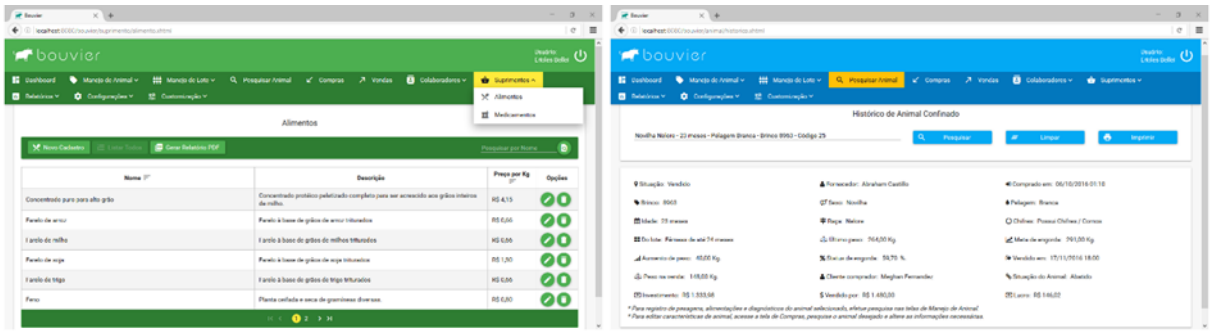

Figura 10 - Menu de navegação na parte superior da tela
Fonte: Reproduções de telas.

Para efetuar *logout*, o usuário clica no botão correspondente da barra superior e uma janela *pop-up* é exibida para confirmar sua opção, conforme a Figura 11.

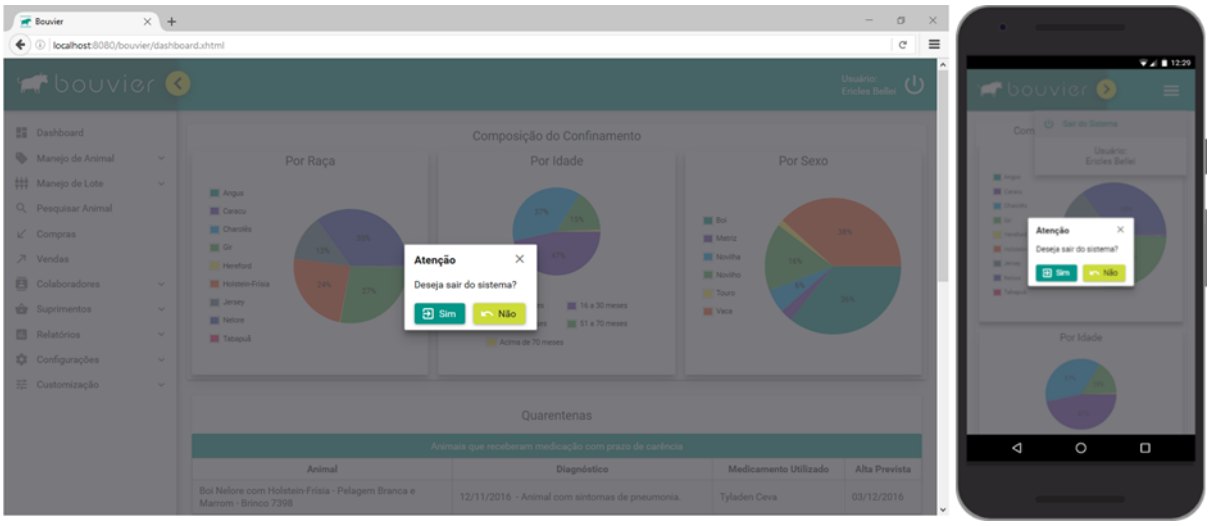

Figura 11 - Janela *pop-up* para confirmação de *logout*
Fonte: Reproduções de telas.

As telas para visualização e registros de manejos individuais e de manejos de lotes exibem de maneira tabular informações sobre o histórico dos últimos 10 dias com o manejo selecionado. Os dados exibidos podem ser classificados de maneira crescente ou decrescente, enquanto as colunas podem ser arrastadas e reorganizadas na ordem em que o usuário preferir. A Figura 12 exemplifica essas telas com o cadastro de alimentações servidas aos lotes.

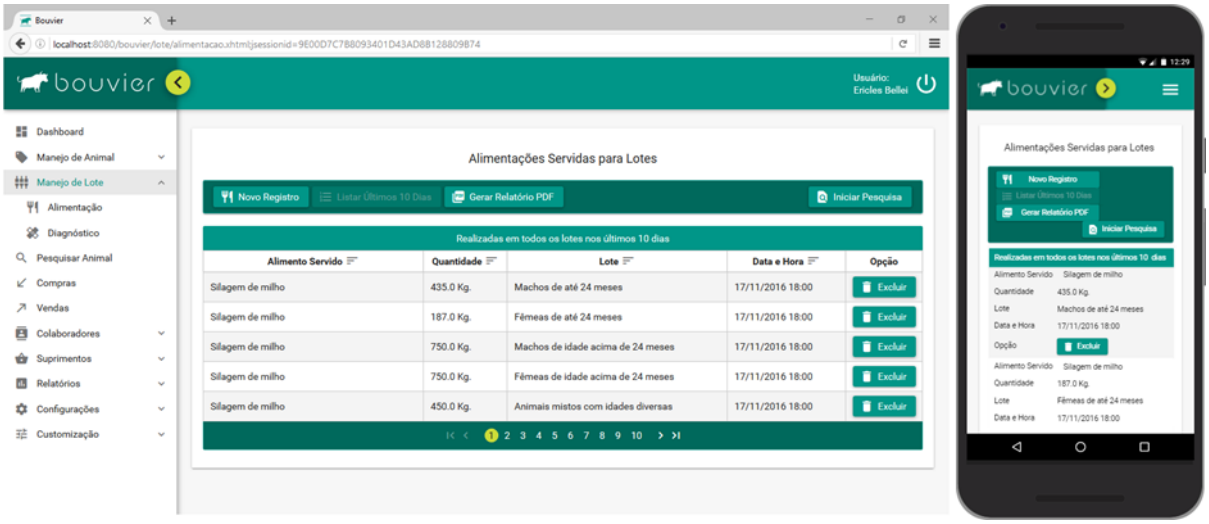

Figura 12 - Tela de cadastro de alimentações servidas aos lotes
Fonte: Reproduções de telas.

A tela possui uma barra com botões que dão acesso à algumas opções, como o registro de um novo manejo, a listagem dos dados dos últimos 10 dias, a criação de

um relatório em formato PDF e a pesquisa baseada em filtros. A Figura 13 e a Figura 14 ilustram, respectivamente, as janelas *pop-up* que são abertas quando o usuário precisa efetuar um novo registro e quando precisa efetuar uma pesquisa.

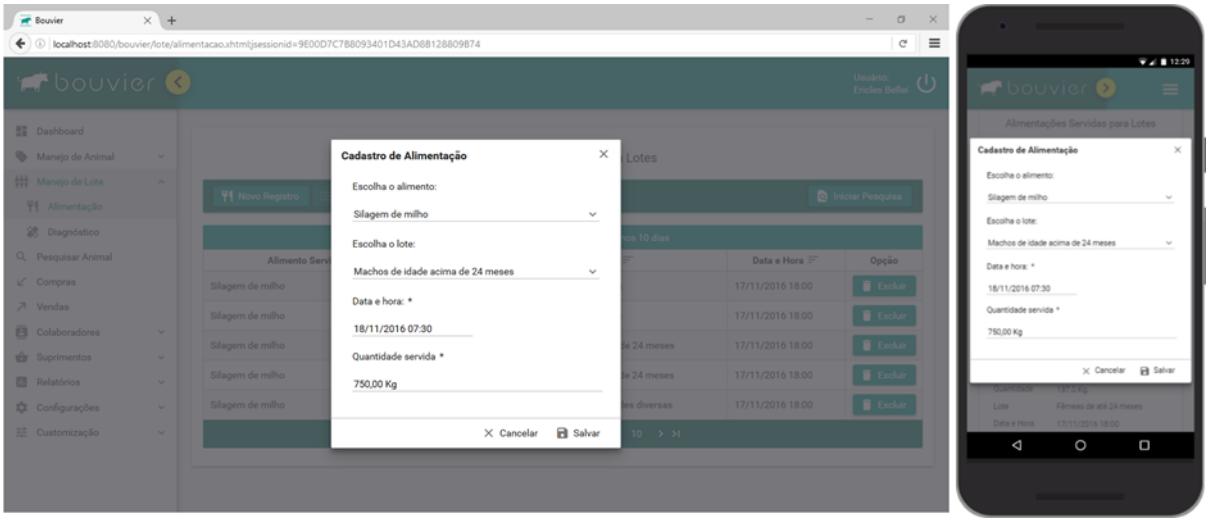

Figura 13 - Janela *pop-up* de cadastro de alimentação de lote
Fonte: Reproduções de telas.

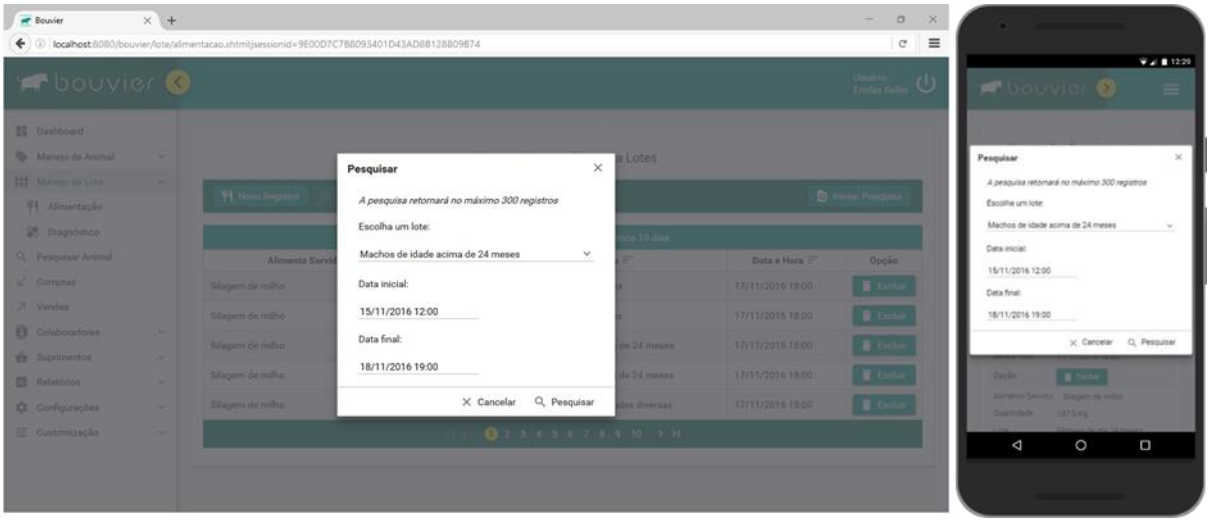

Figura 14 - Janela *pop-up* de pesquisa de alimentação de lote
Fonte: Reproduções de telas.

As operações de inclusão, edição e exclusão, quando concluídas, são informadas ao usuário com uma mensagem temporária no canto superior direito da tela, indicando erro ou sucesso, conforme demonstra o trecho da Figura 15.

Figura 15 - Mensagens de informação ao usuário
Fonte: Trechos de reproduções de telas.

As telas dentro de ‘Manejo de Animal’ (‘Pesagem’, ‘Alimentação’, ‘Diagnóstico’ e ‘Troca de Lote’) e as telas dentro de ‘Manejo de Lote’ (‘Alimentação’ e ‘Diagnóstico’) têm funcionamento essencialmente similar: os dados são apresentados de maneira tabular e com opção para exclusão; o cadastro de um novo manejo é efetuado em uma janela *pop-up*; a pesquisa é efetuada em uma janela *pop-up*, tem parâmetros como lote ou animal escolhido, data inicial e data final do intervalo pesquisado, e é apresentada diretamente na tabela; é possível exportar os dados da tabela para um relatório em formato PDF.

Análogas às telas de manejos, as telas de ‘Vendas’, ‘Colaboradores’ (‘Clientes’ e ‘Fornecedores’) e ‘Suprimentos’ (‘Alimentos’ e ‘Medicamentos’) também apresentam os dados em tabelas e possibilitam pesquisas, mas, com filtros diferentes para cada caso, como cliente comprador nas vendas, nome e descrição de suprimentos, nome de clientes e fornecedores etc. Essas telas também permitem a inclusão e edição de registros em uma janela *pop-up*, e exibem um pedido de confirmação quando o usuário deseja excluir um registro, ilustrado na Figura 16.

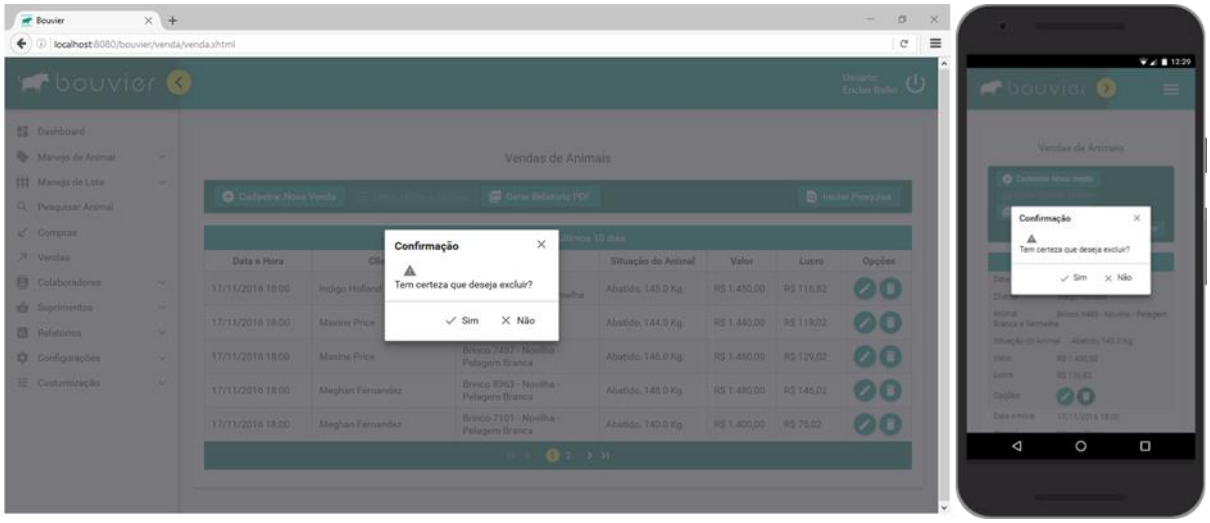

Figura 16 - Janela para confirmação de exclusão de registro
Fonte: Reproduções de telas.

Em todas as telas com cadastros, edições ou qualquer espaço de formulário para entrada de dados do usuário, são feitas validações, em sua maioria, no próprio *browser*, visando a garantia de integridade das informações armazenadas no sistema. Para viabilizar tais validações, diversas técnicas são empenhadas, como máscaras de dados, campos para entrada de datas que exibem calendários, caixas de auto completar, caixas de seleção com lotes, pelagens e sexos de animais, campos específicos para números inteiros e números decimais etc.

O cadastro de compra de animal é feito em um formulário reproduzido na Figura 17. O campo de data abre um calendário em que o usuário seleciona o dia e a hora em que a compra foi concretizada. O campo de fornecedor é auto completado quando o usuário começa a digitar o nome de um fornecedor já cadastrado. Os campos de sexo, lote, raças e pelagem são caixas de seleção que carregam dados previamente cadastrados. A idade do animal é escolhida ao arrastar o *slider* disponível. Uma caixa de checagem é utilizada para indicar se o animal possui cornos ou chifres. A meta do animal pode ser calculada e sugerida pelo sistema, com base nas características do animal em cadastro e em parâmetros registrados nas configurações do sistema, como raças, sexo, idade e peso.

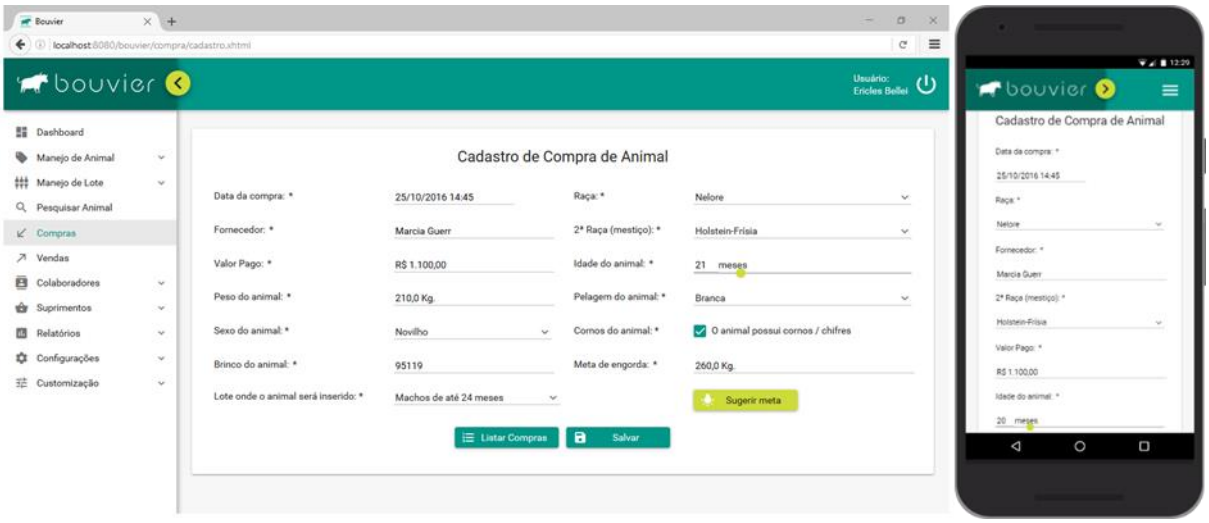

Figura 17 - Tela de cadastro de compra de animal
Fonte: Reproduções de telas.

A tela de ‘Lotes’ e a tela de ‘Raças’ bovinas estão em ‘Configurações’ por serem acessadas com menor frequência. A tela de ‘Funcionalidades’ permite ao administrador configurar as percentagens que são utilizadas pelo sistema como parâmetros para o cálculo e sugestão da meta de engorda dos animais. Entre essas variáveis estão a percentagem básica de engorda e os decréscimos que são feitos conforme o sexo e a faixa etária do bovino.

Os usuários do sistema e suas permissões são cadastrados pelo administrador na tela ilustrada na Figura 18. Cada usuário pode ser detentor de uma ou múltiplas permissões. Administradores têm acesso geral às funcionalidades do sistema. Funcionários podem acessar ‘Pesquisar Animal’, ‘Manejo de Animal’ e ‘Manejo de Lote’. Supervisores podem visualizar as telas de ‘Pesquisar Animal’, ‘Suprimentos’ e ‘Colaboradores’.

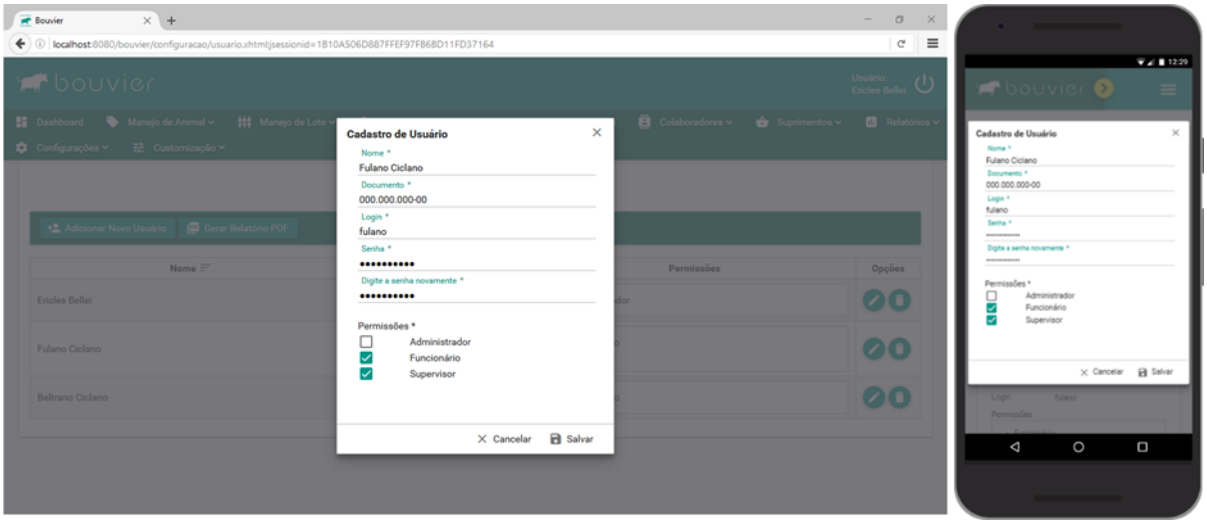

Figura 18 - Tela de cadastro de usuários do sistema
Fonte: Reproduções de telas.

O histórico de qualquer animal que passou pelo confinamento pode ser encontrado na tela de ‘Pesquisar Animal’. O usuário digita o número do brinco e seleciona um animal encontrado na lista mostrada pelo sistema. Todas as características zootécnicas, origem, destino, datas importantes, informações sobre desempenho, valores de investimento e lucro podem ser visualizados, conforme a reprodução da Figura 19.

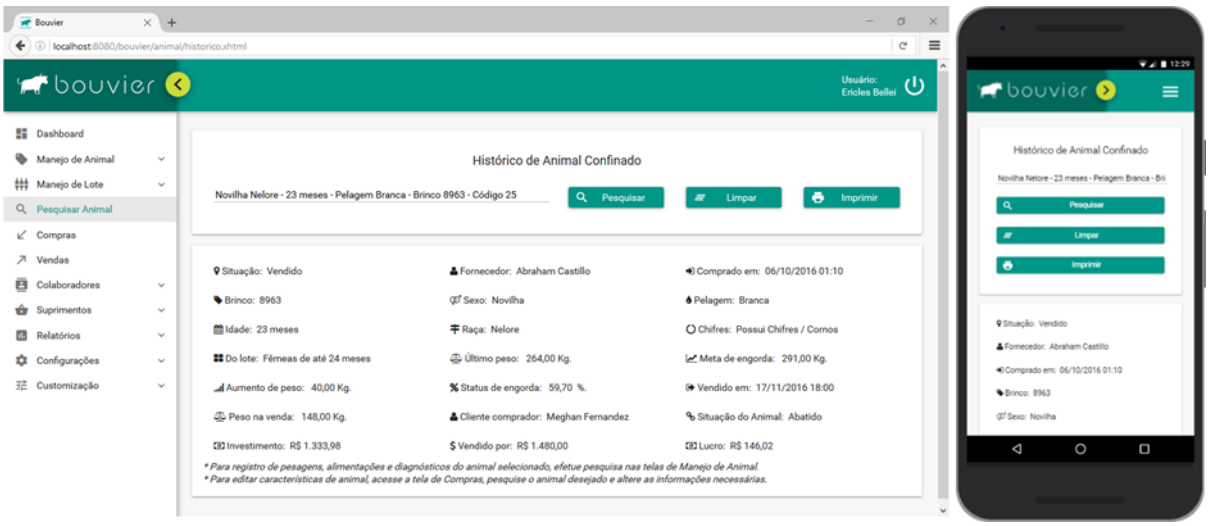

Figura 19 - Tela para pesquisa de histórico de animal confinado
Fonte: Reproduções de telas.

O relatório de ‘Estoque de Animais’, ilustrado na Figura 20, exibe em uma tabela todos os animais que estão no confinamento, incluindo informações individuais como *status*, meta, peso ganho desde a entrada e o valor de seu investimento atualizado. As colunas da tabela podem ser classificadas em ordem crescente ou

decrescente e é possível listar os animais que estão em baias individuais. O sistema proporciona pesquisa no estoque, conforme a Figura 21, com diversos filtros, como intervalo de datas de compra, fornecedor, raça, sexo, faixa etária e lote.

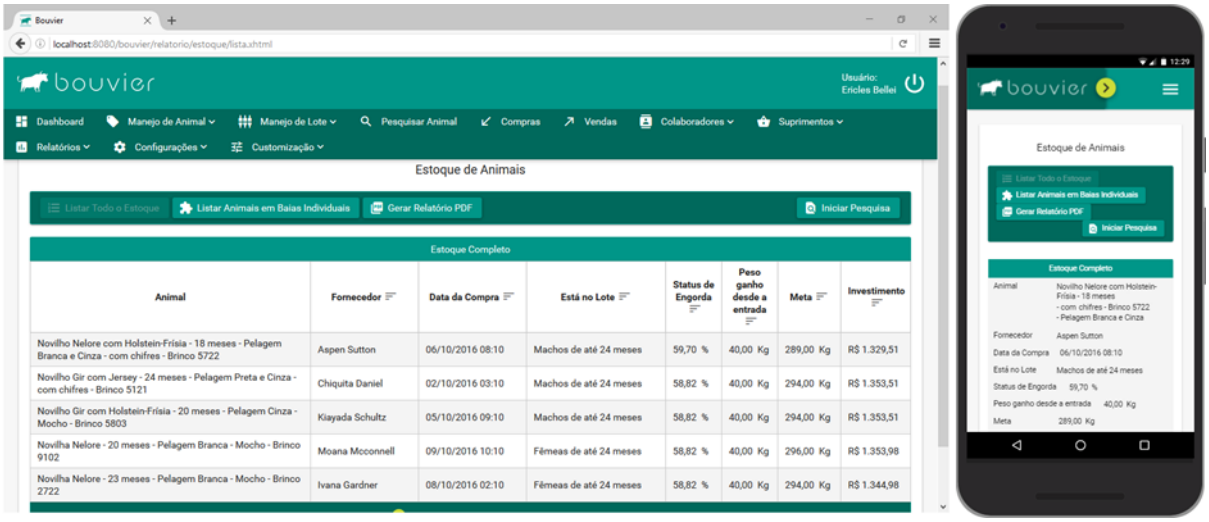

Figura 20 - Tela de relatório de estoque de animais
Fonte: Reproduções de telas.

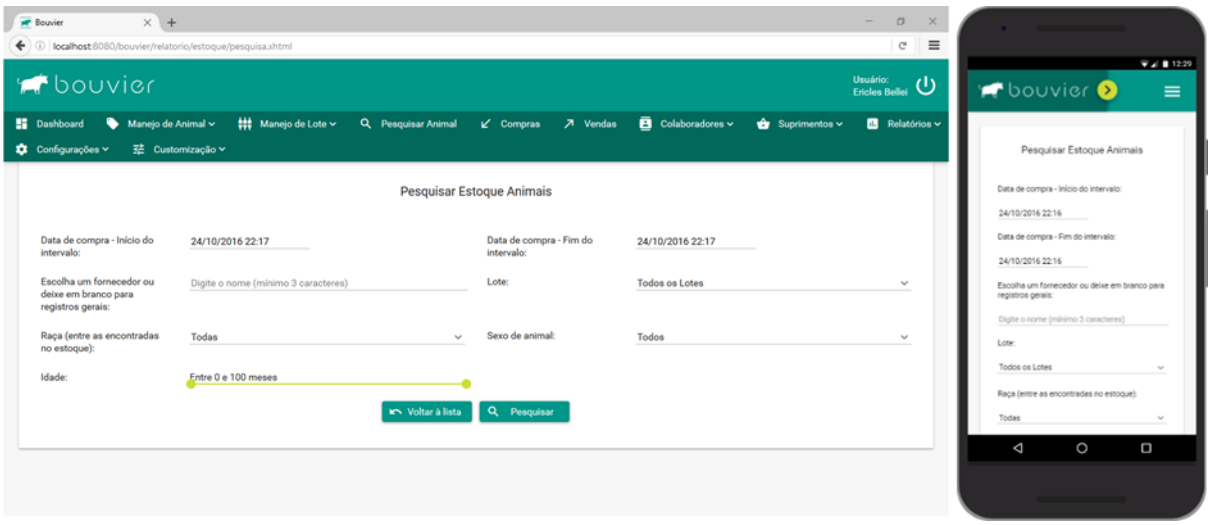

Figura 21 - Tela de pesquisa de estoque de animais
Fonte: Reproduções de telas.

As vendas de bovinos e seus rendimentos podem ser analisados no relatório de ‘Rendimentos’, apresentado na Figura 22. O relatório aponta o animal vendido e sua situação, o último peso aferido no confinamento, o peso na venda, a proporção da conversão para carcaça nos abates, o valor do investimento, da venda e do lucro proporcionado por cada um. O relatório exibe também a soma das vendas, dos lucros e a margem de lucro média exibida na tabela.

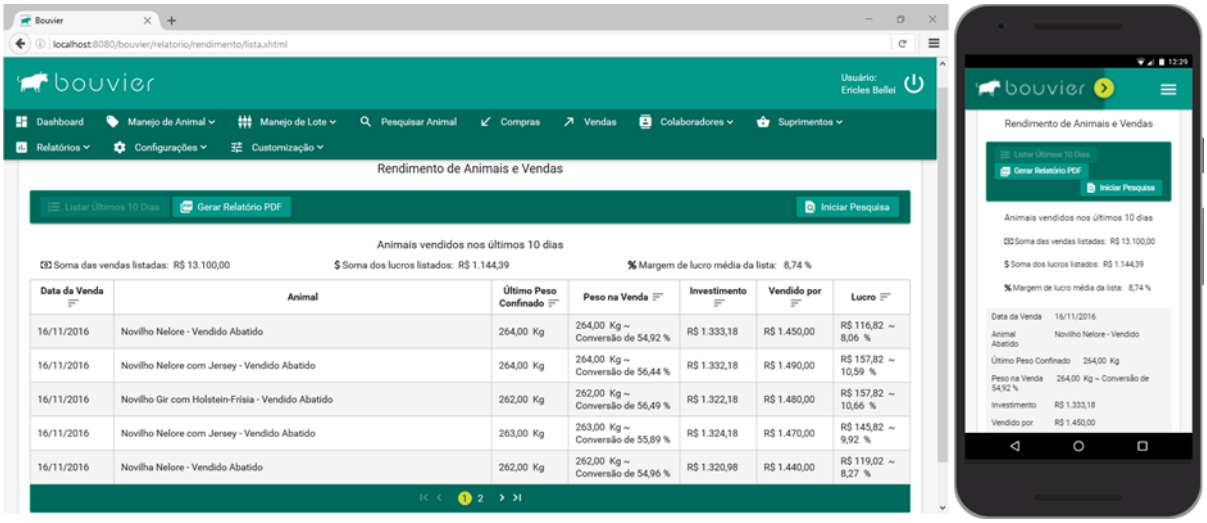

Figura 22 - Relatório de rendimento de vendas
Fonte: Reproduções de telas.

O relatório de ‘Rendimentos’ viabiliza também uma pesquisa, ilustrada na Figura 23, que permite filtros por intervalo de datas, sexo, raça e situação do animal na venda. Caso a pesquisa retorne resultados, o sistema redireciona o usuário para a tela da Figura 22, com os resultados obtidos. Caso contrário, o usuário é informado com uma mensagem, demonstrada no trecho de captura da Figura 24.

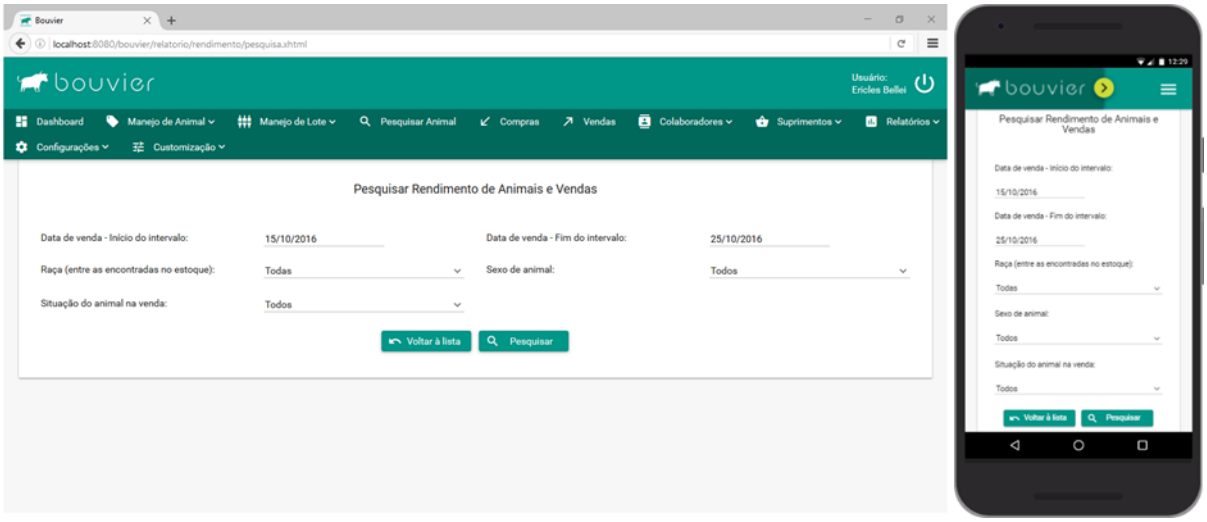

Figura 23 - Tela de pesquisa de rendimento de vendas
Fonte: Reproduções de telas.

Figura 24 - Mensagem informativa de pesquisa sem resultados
Fonte: Trechos de reproduções de telas.

5 CONCLUSÃO

O objetivo deste trabalho foi viabilizar um sistema computacional voltado a um confinamento de gado de corte. Procedeu-se com a execução e explanação de fases de desenvolvimento de *software*, como levantamento de requisitos, análise, projeto, escolha de tecnologias e ferramentas auxiliares, implementação e testes.

Esse sistema visa a abstração de informações primordiais para a administração de um estabelecimento que confina gado de corte, mediante o registro, cruzamento e análise gerencial dos dados derivados das atividades concretizadas. Abrange-se acontecimentos de negócio, fornecedores, clientes, manutenção de rebanho, evolução em termos de ganho de massa corporal, interação medicamentosa e consumo de insumos.

A solução pretendida foi desenvolvida como um aplicativo *web* em linguagem Java, com o PrimeFaces na implementação do JSF, hospedado na nuvem e com a persistência de dados efetuada em um banco de dados relacional, utilizado como um serviço, dentre os diversos tipos específicos de serviços de computação em nuvem.

A tecnologia de computação em nuvem emerge como tendência e desafia muitas concepções tradicionais da área porque faz repensar a forma de organização de programação, ao prometer aumento de produtividade aliado à economia de tempo e de custos. O PrimeFaces disponibiliza um vasto conjunto de componentes ricos que otimizam e aceleram o desenvolvimento de sistemas, respaldado pela ampla documentação disponível em sua comunidade.

A concretização do projeto do sistema evidencia que os objetivos para este trabalho foram alcançados. As funcionalidades implementadas e testadas legitimam a precisão e coerência dos requisitos coletados e se mostram capazes de proporcionar avanços nos processos de gestão de negócios, acompanhamento de tarefas e conhecimento de resultados no contexto abordado.

Algumas melhorias, como reforços em aspectos de segurança e inclusão de novos relatórios e funcionalidades, são preconizadas como trabalhos futuros. Uma ideia mais ambiciosa cogitada para os próximos projetos abarca a integração do sistema com equipamentos típicos do ambiente de confinamento de gado, como balanças eletrônicas e brincos de bovinos com identificação por radiofrequência, a fim de propiciar a automação de algumas atividades desse ambiente.

REFERÊNCIAS

AGRISOFT TI-AGRO. **Módulo Rebanho.** Disponível em: <http://www.agrisoft.com.br/solucoes/software-modulo-rebanho>. Acesso em: 08 jan. 2016.

ASSOCIAÇÃO BRASILEIRA DAS INDÚSTRIAS EXPORTADORAS DE CARNES. **Exportações de carne bovina brasileira atingem US$ 5,9 bilhões em 2015**. 2015. Disponível em: < http://www.abiec.com.br/news_view.asp?id=%7BD740B5F0-3AE2-4A5C-AF0C-50E206FF8751%7D >. Acesso em: 13 fev. 2016.

BEREA, Diego. **A ISO 20000 e a computação em nuvem**. 2012. Disponível em: < http://www.modulo.com.br/index.php?option=com_content&task=view&id=2976&Itemid=211>. Acesso em: 17 abr. 2016.

BORGES, Helder P.; DE SOUZA, José N.; SCHULZE, Bruno; MURY, Antonio R. **Computação em nuvem.** 2011. Disponível em < http://livroaberto.ibict.br/bitstream/1/861/1/COMPUTA%C3%87%C3%83O%20EM%20NUVEM.pdf>. Acesso em: 25 abr. 2016.

BRABOV. **Sobra a Brabov**. 2016. Disponível em: <http://brabov.com.br/sobre/>. Acesso em: 08 jan. 2016.

CARDOSO, Esther Guimarães. Confinamento de Bovinos**.** In: **Suplementação em Pasto e Confinamento de Bovinos.** Campo Grande: Embrapa Gado de Corte, 2000.

CARISSIMI, Alexandre. Desmistificando a Computação em Nuvem. **15° Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul.** SBC e CRAD/RS, 2015.

FERRAZ, José B. S; ELER, Joanir P. Melhoramento genético para aumento de produtividade em gado de corte no Brasil: a história, o presente e o futuro. In: PIRES, Alexandre Vaz. **Bovinocultura de corte**. Piracicaba: FEALQ, 2010. v. II, cap. 38, p. 763-784.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO/IEC 20000-1: 2011. **Information technology — Service management — Part 1:** Service management system requirements. 2011. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso-iec:20000:-1:ed-2:v1:en>. Acesso em: 15 abr. 2016.

MOLLAH, Muhammad B.; ISLAM, Kazi R.; ISLAM, Sikder S. Next Generation of Computing through Cloud Computing Technology. **IEEE 25th Canadian Conference on Electrical and Computer Engineering.** 2012.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **NIST Cloud Computing Program.** 2010. Disponível em < http://www.nist.gov/itl/cloud/index.cfm>. Acesso em: 03 jun. 2016.

PORTAL BRASIL. **Rebanho bovino brasileiro cresce e chega a 212,3 milhões de cabeças de gado**. 2015. Disponível em: < http://www.brasil.gov.br/economia-e-emprego/2015/10/rebanho-bovino-brasileiro-cresce-e-chega-a-212-3-milhoes-de-cabecas-de-gado >. Acesso em: 03 abr. 2016.

Red Hat. **OpenShift Container Platform.** 2016. Disponível em < https://www.redhat.com/en/technologies/cloud-computing/openshift>. Acesso em: 29 jul. 2016

RALLY DA PECUÁRIA 2015. **Rebanho confinado deve crescer meio milhão de cabeças com perspectiva de melhora de preços**. 2015. Disponível em: < http://www.rallydapecuaria.com.br/noticias/rebanho-confinado-deve-crescer-meio-milhao-de-cabecas-com-perspectiva-de-melhora-de-precos>. Acesso em: 11 jun. 2016.

SCALEDB. **Database-as-a-Service (DBaaS).** 2014. Disponível em: < http://www.scaledb.com/dbaas-database-as-a-service.php>. Acesso em: 19 jun. 2016.

SOMMERVILLE, Ian. **Engenharia de Software**. 6ª ed. São Paulo: Pearson, 2004.

SOUSA, Flávio R. C.; MOREIRA, Leonardo O.; DE MACÊDO, José A. F.; MACHADO, Javam C. Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios. In: **Tópicos em Sistemas Colaborativos, Interativos, Multimídia, Web e Bancos de Dados**. Belo Horizonte: SBC e Universidade Federal de Minas Gerais, 2010, cap. 4, p.101-130.

THIAGO, Luiz R. L. de S. Confinamento de bovinos. In: **Coleção Criar.** Brasília: EMBRAPA-SPI, 1996. Disponível em: < http://ainfo.cnptia.embrapa.br/digital/bitstream/item/11893/2/00013740.pdf >. Acesso em: 23 nov. 2016.

ZHANG, Jin. **Database in the cloud**. 2011. Disponível em: <http://www.ibm.com/developerworks/data/library/dmmag/DMMag_2011_Issue2/cloudDBaaS/>. Acesso em: 10 mar. 2016.


UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PATO BRANCO