Sistema para Gerenciamento de Granja de Suínos - OINC Suínos

Alcides Antonio Lorenski Neto[1]
Felipe Eduardo Bohnen[2]
Miguel Schneiders Flach[3]
Murilo Morosini[4]
Roney Bieger Anschau[5]
Otília Donato Barbosa[6]
Roberson Junior Fernandes Alves[7]

Resumo

Este projeto apresenta o desenvolvimento do sistema OINC Suínos, criado para facilitar e organizar a gestão de granjas de suínos. O trabalho começou com o levantamento de requisitos, onde foi possível entender as necessidades reais da suinocultura, e avançou para a modelagem do banco de dados e criação de diagramas UML que ilustram o funcionamento do sistema. Utilizando ferramentas como *Visual Paradigm, Eclipse* e *DBeaver*, foram construídos e testados os principais recursos do sistema. O resultado é uma solução prática, eficiente e adaptada às rotinas diárias, ajudando no manejo e na tomada de decisões dentro das granjas.

Palavras-chave: Gestão de granjas. Suinocultura. Banco de dados. Diagramas UML. Visual Paradigm. DBeaver.

[1] Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. alcidesantonio08@hotmail.com
[2] Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. felipebohnenf@gmail.com
[3] Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. miguelsflach65@gmail.com
[4] Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. murilomorosini@gmail.com
[5] Graduando do Curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. roneyronyy@hotmail.com
[6] Docente do curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. otilia.barbosa@unoesc.edu.br
[7] Docente do curso de Ciência da Computação - Unoesc Campus São Miguel do Oeste. roberson.alves@unoesc.edu.br

1. INTRODUÇÃO

No âmbito da produção de suínos, diversas questões exigem atenção detalhada, destacando-se o controle das atividades, a alimentação e nutrição, o manejo adequado e a sustentabilidade das práticas adotadas, entre outras.. Segundo ABPA (2023), em 2022, a produção brasileira de carne suína, em toneladas, foi de 4,983 milhões, um avanço sobre 2019, cuja demanda foi de 3,983 milhões, indicando um fortalecimento e um crescimento contínuo.

De acordo com Costa, Costa e Rohr (2016, p. 10), cada granja possui suas peculiaridades, onde as práticas de controle e manejo devem ser levadas em conta considerando o bem-estar animal e o produto final, agregando no fortalecimento do setor e dos serviços envolvendo a suinocultura e execução da mesma.

No contexto do animal e do monitoramento, torna-se necessário a construção e o planejamento de um sistema de gerenciamento de granja, com o objetivo de facilitar, agilizar e automatizar os processos envolvidos. Tal modelo proposto tem como princípio o cadastro do suíno, a separação do grupo do bicho, cadastro de funcionários, controle de vacinas, insumos, alimentação e relatórios gerados conforme a necessidade.

Sendo assim, busca-se a eficácia dinâmica de todas as informações do plantel, de modo a facilitar a fiscalização e a gestão diária, gerando estatísticas confiáveis, pesquisas detalhadas e a orientação de investimentos, promovendo a saúde dos animais e a sustentabilidade das operações.

2. DESENVOLVIMENTO

2.1 Levantamento de requisitos

Sobre a coleta de requisitos, Mendonça (2014) destaca: “O levantamento de requisitos desempenha um papel importante na construção de um sistema de informação, pois é o início para toda a atividade de desenvolvimento de software”.

Essa fase inicial do desenvolvimento de software visa captar integralmente as necessidades do cliente e do objetivo envolvido. Em outras palavras, é o momento de compreender exatamente o que deve ser desenvolvido, esclarecer dúvidas e obter uma visão clara de como o sistema operará (Mendonça, 2014).

Pretendendo alcançar os objetivos almejados pelo OINC Suínos, como manter uma boa estruturação de dados e atender aos desejos dos interessados, não apenas foi definido os objetivos e funcionalidades esperadas, mas também estabelecidas as bases para um projeto bem-sucedido, alinhado com as expectativas e requisitos do usuário final.

2.1.1 Experiência em Granja

Buscando saber mais sobre as necessidades envolvidas na suinocultura e no trabalho diário da mesma, o desenvolvedor do projeto, Felipe Eduardo Bohnen, auxiliou nesse quesito, trazendo para o desenvolvimento suas experiências, ideias e requisitos na área como trabalhador, de modo a facilitar a compreensão mútua. Além disso, a entrevista com Jandir Flach, funcionário e gerente da Granja de Suínos Baumgratz, em Tunápolis – SC, trouxe importantes contribuições para alinhar o projeto às necessidades reais na suinocultura baseando-se na experiência e no manejo do entrevistado.

2.2 Modelo Relacional

Com todas as informações necessárias em mãos para o projeto OINC Suínos, iniciou-se a construção do modelo relacional, que é uma representação abstrata do banco de dados. Isso envolve transformar os requisitos em um mapa estrutural do banco de dados, especificando tabelas, suas conexões, colunas e os tipos de dados a serem armazenados. Com isso em mente, foram projetadas as tabelas necessárias para o funcionamento de uma granja, dando forma ao modelo relacional.

2.2.1 Construção do Modelo Relacional

Foi desenvolvido um modelo concreto de banco de dados utilizando o Visual Paradigm pelas seguintes ordens:

1. Criação das tabelas e suas colunas: Foram definidas as tabelas que compõem o banco de dados, especificando cada coluna necessária.
2. Definição dos tipos de dados das colunas: Foi determinado o tipo de dado apropriado para cada coluna, assegurando o correto armazenamento das informações.

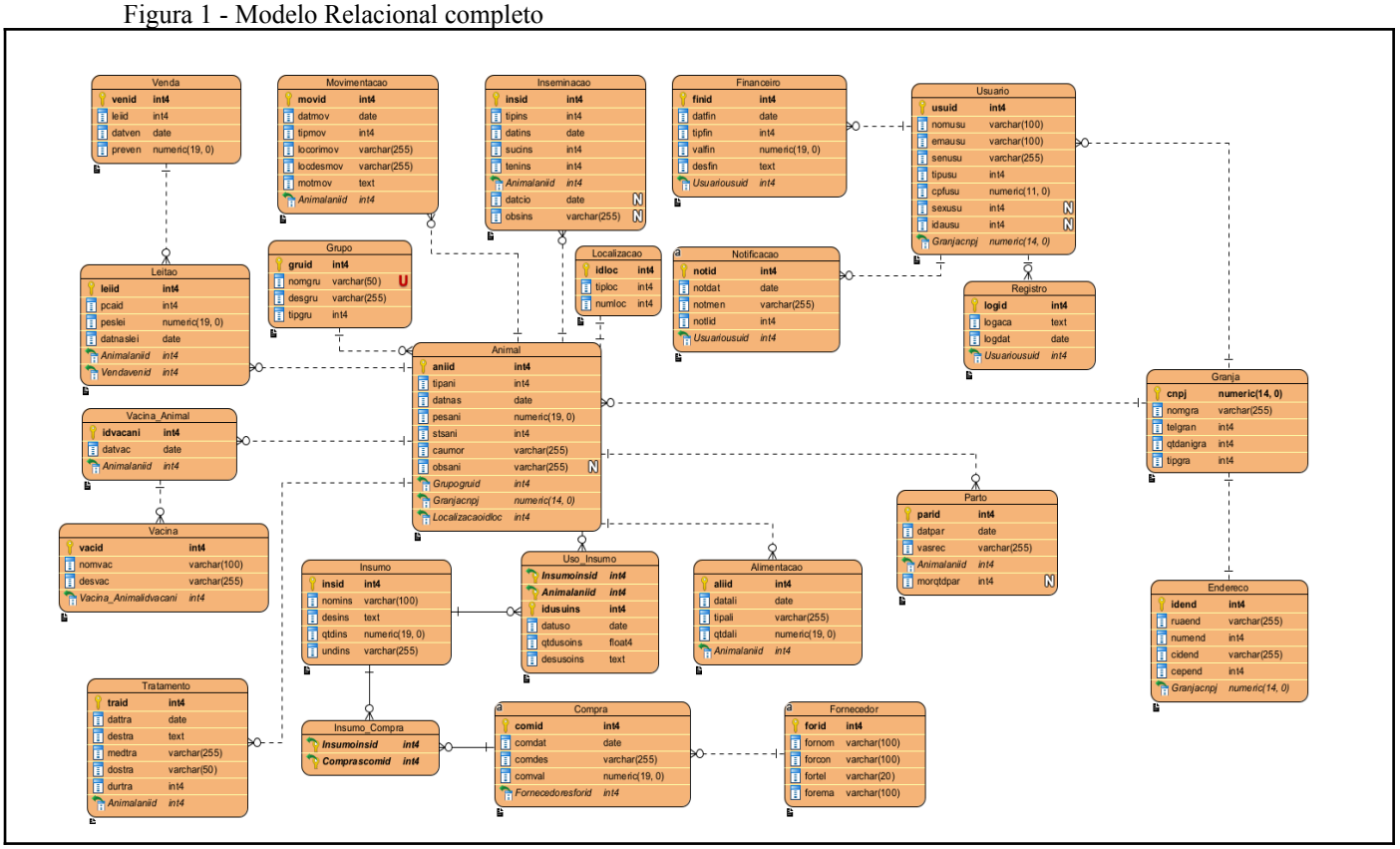
3. Identificação das chaves primárias: Identificou-se as colunas que atuariam como chaves primárias, garantindo a identificação única de cada registro.
4. Configuração das relações entre as tabelas: Estabeleceram-se as conexões entre as tabelas, mantendo a integridade referencial dos dados.
5. Documentação detalhada: Foi criada uma documentação completa para cada tabela e suas colunas, facilitando o entendimento e a manutenção futura do banco de dados.

Essa fase do projeto exigiu um tempo significativo, pois define a base estrutural que será utilizada no banco de dados. O planejamento e a documentação cuidadosa dessa estrutura foram essenciais para garantir que o banco de dados fosse robusto, eficiente e de fácil manutenção, contribuindo de maneira significativa para o sucesso do projeto OINC Suínos.

A utilização do *Visual Paradigm* permitiu uma visualização clara e compartilhada do modelo de dados. Essa abordagem colaborativa foi crucial para identificar e resolver potenciais problemas de design antes da implementação, evitando atrasos e garantindo um fluxo de desenvolvimento mais eficiente.

Após a conclusão deste processo, foi possível obter o seguinte resultado:

Figura 1 - Modelo Relacional completo



Fonte: Os autores.

2.2.2 Normalização das Tabelas

Após a construção do modelo relacional, foi necessário realizar um processo de validação de estrutura, ou seja, verificar se o mesmo está livre de anomalias, possíveis redundâncias e demais problemas que podem vir a surgir durante os processos de inserção, exclusão, modificação e consulta de seus dados armazenados. Foi feita a aplicação das 3 normas formais, um conjunto de regras destinadas à normalização de um banco de dados.

Foi utilizada a definição para o nosso projeto com os seguintes conceitos:

1. **Primeira Forma Normal**: Verifica-se se cada coluna da tabela contém apenas um único valor, evitando campos multivalorados
2. **Segunda Forma Normal**: Confirma-se que a tabela está em conformidade com a Primeira Forma Normal e que os atributos não chave dependem totalmente da chave primária, e não apenas de parte dela.
3. **Terceira Forma Normal**: Garante-se que a tabela atende à Segunda Forma Normal e que todos os atributos não chave dependem exclusivamente da chave primária, sem dependências transitivas de outros atributos não chave.

2.3 Diagramas UML

Segundo Guedes (2011), "A UML – *Unified Modeling Language* ou Linguagem de Modelagem Unificada – é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos". Dessa maneira, a UML, por não ser uma linguagem de programação, adota um padrão para visualizar, especificar, construir e documentar sistemas de software de forma clara e compreensível de modo a auxiliar os engenheiros de software a definirem as características do projeto.

No OINC Suínos, a utilização de diagramas UML serviu como base e apoio no que diz respeito à dinâmica de processos, estruturação lógica, atividade e sequência na troca de informações no sistema, proporcionando uma visão clara e detalhada do funcionamento e das interações entre os componentes envolvidos.

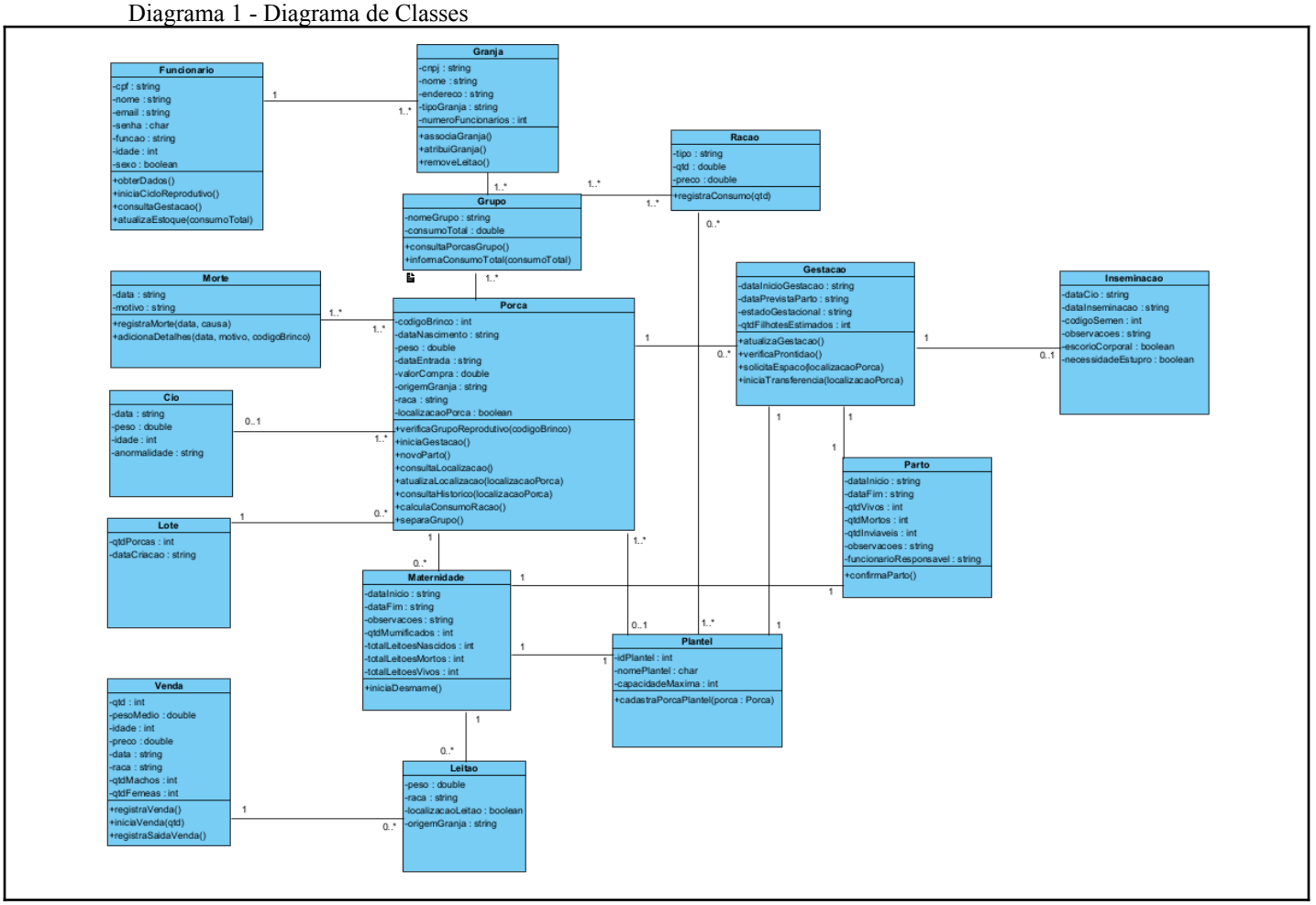
Foram feitos os diagramas de acordo com a prioridade das ações e da dinâmica do sistema, colocando como prioridade aqueles que, com base na lista de requisitos e no decorrer do desenvolvimento do projeto, foram considerados essenciais para o funcionamento inicial e para atender às necessidades principais do sistema.

2.3.1 Diagrama de Classes

Como o próprio nome diz, o diagrama de classes tem como intuito definir a estrutura das classes utilizadas pelo sistema de modo a determinar os atributos e métodos de cada classe e o relacionamento entre elas.

A seguir, apresenta-se o diagrama de classes desenvolvido para o sistema:

Diagrama 1 - Diagrama de Classes



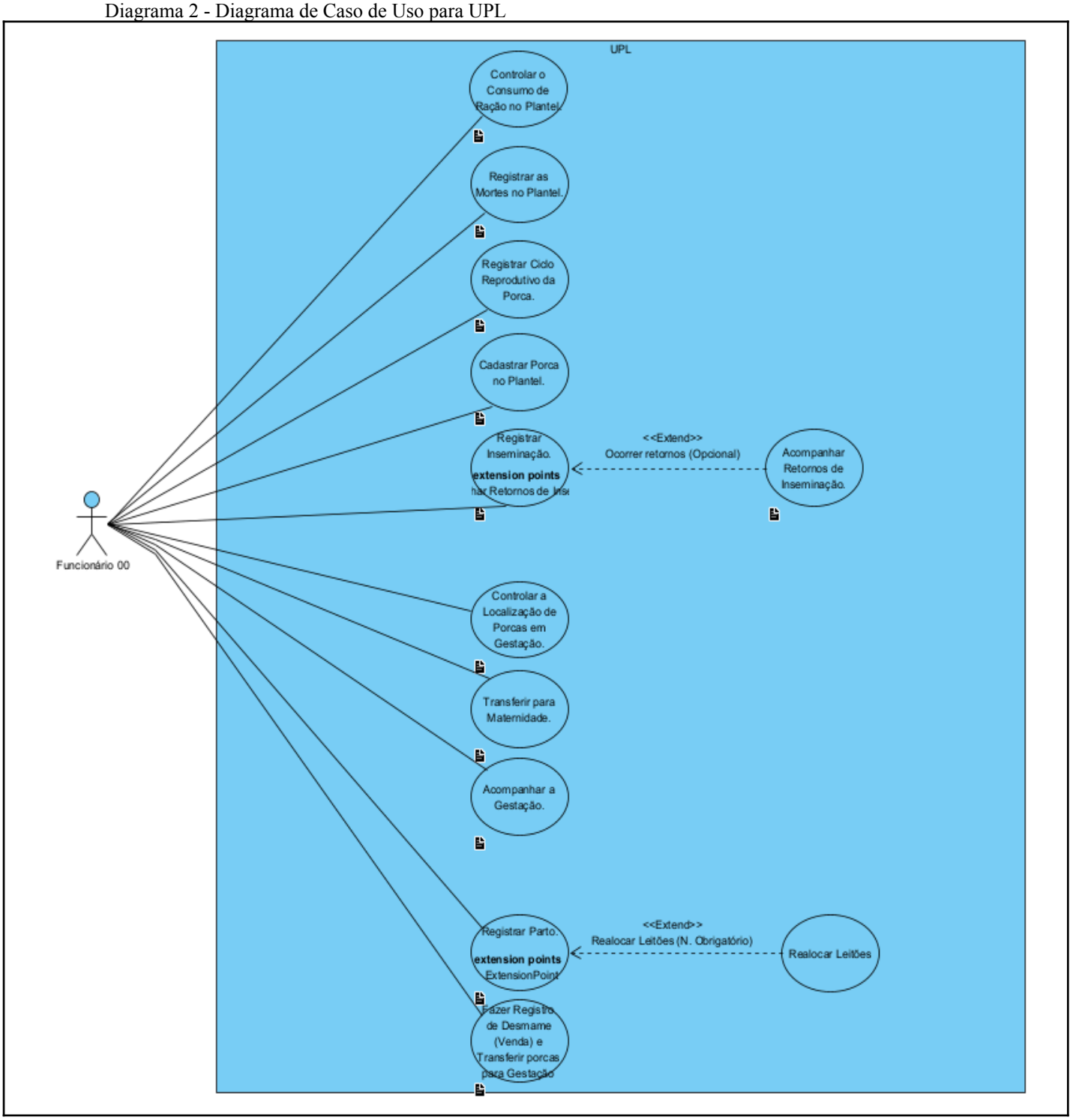
Fonte: Os autores.

2.3.2 Diagrama de Casos de Uso

Utilizado na análise de requisitos e durante todo o processo de modelagem, o diagrama de casos de uso mostra como sistema irá se comportar, procurando identificar aqueles que irão interagir diretamente com o sistema, conhecidos como atores e os casos de uso, que representam as funcionalidades do sistema para os atores, delimitando o escopo e facilitando a comunicação no projeto.

Ao decorrer da produção do trabalho, foram elaborados três diagramas de casos de uso, sendo eles para: UPL, Creche e Terminação. Segue, abaixo, um exemplo ilustrativo do diagrama para a UPL:

Diagrama 2 - Diagrama de Caso de Uso para UPL



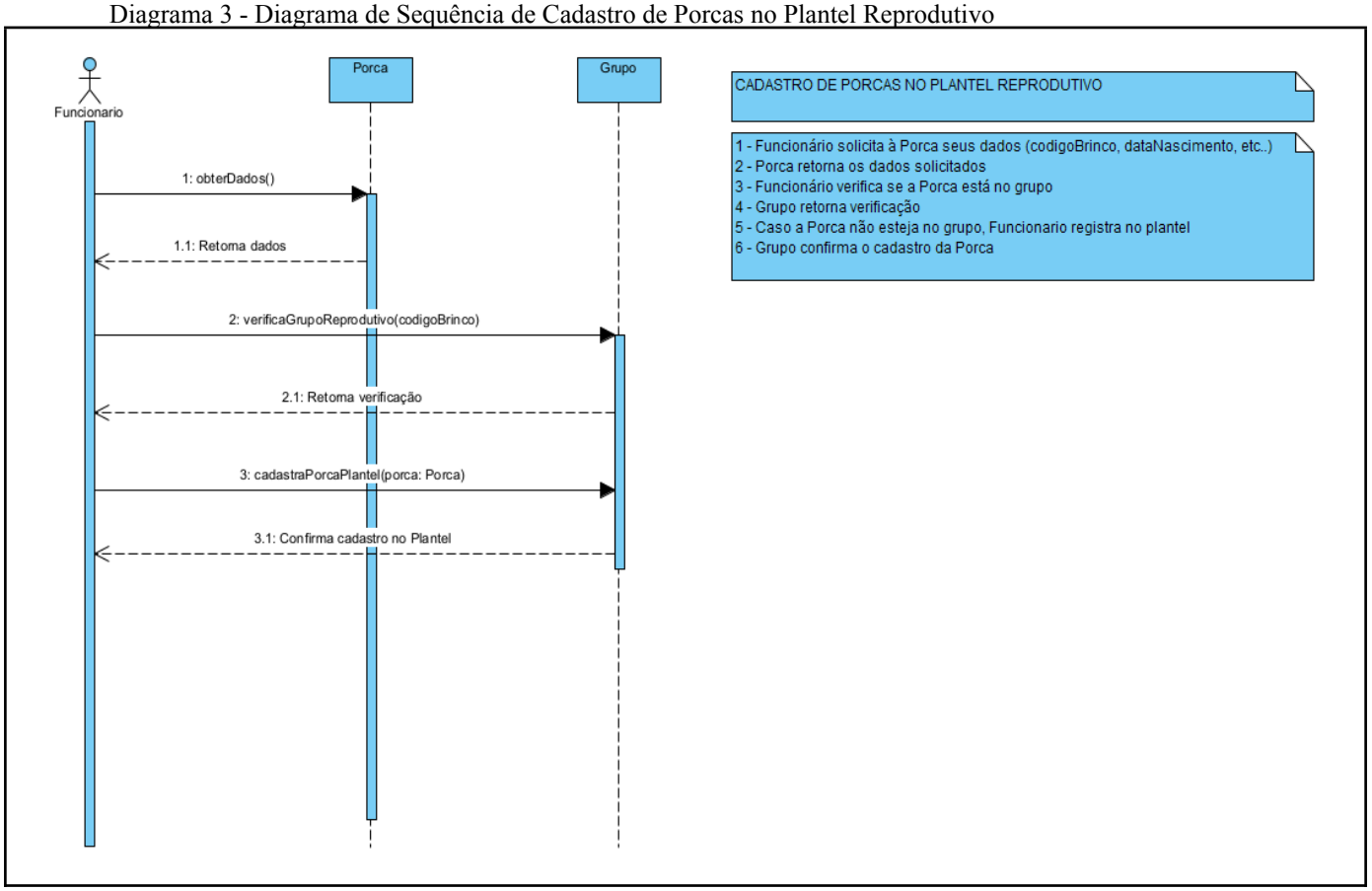
Fonte: Os autores.

2.3.3 Diagrama de Sequência

O diagrama de sequência é um modelo comportamental que destaca a sequência temporal das interações entre objetos em um processo específico. Nesse contexto, o diagrama identifica o evento inicial, geralmente desencadeado por um ator externo, e detalha como o processo ocorre até a sua conclusão. Isso é feito representando as mensagens trocadas e os métodos invocados entre os objetos. Ele é essencial para compreender e modelar a dinâmica dos processos, garantindo clareza na implementação do sistema e na comunicação entre desenvolvedores.

Neste projeto, foram elaborados 11 diagramas de sequência, ilustrando diferentes cenários e fluxos do sistema, como exemplificado a seguir na troca de mensagens entre objetos ao cadastrar as porcas no plantel:

Diagrama 3 - Diagrama de Sequência de Cadastro de Porcas no Plantel Reprodutivo



Fonte: Os autores.

2.3.4 Diagrama de Atividade

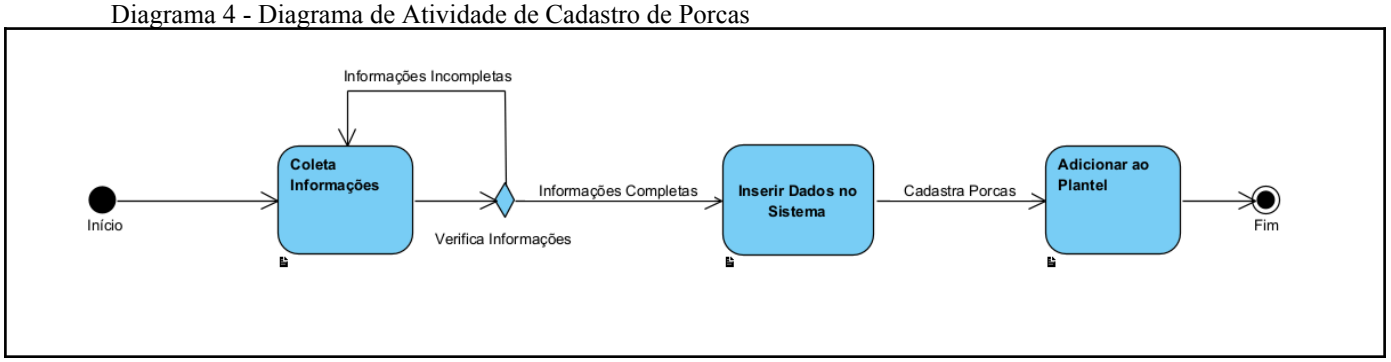
O diagrama de atividade é uma ferramenta essencial para modelar o fluxo de trabalho ou os processos de um sistema, com foco na sequência lógica e condicional das ações

realizadas. Ele é amplamente utilizado para representar processos de negócios, fluxos de operações ou atividades específicas dentro de um sistema.

Esse tipo de diagrama inicia identificando o ponto de partida do processo e detalha as atividades realizadas, decisões condicionais e os caminhos alternativos até sua conclusão. Ele é útil para entender o comportamento de sistemas complexos e propor melhorias nos fluxos representados.

Ao longo deste projeto, foram criados seis diagramas de atividade para representar de maneira detalhada os principais processos. Um exemplo pode ser observado a seguir ao realizar a atividade de cadastrar as porcas no plantel reprodutivo:

Diagrama 4 - Diagrama de Atividade de Cadastro de Porcas



Fonte: Os autores.

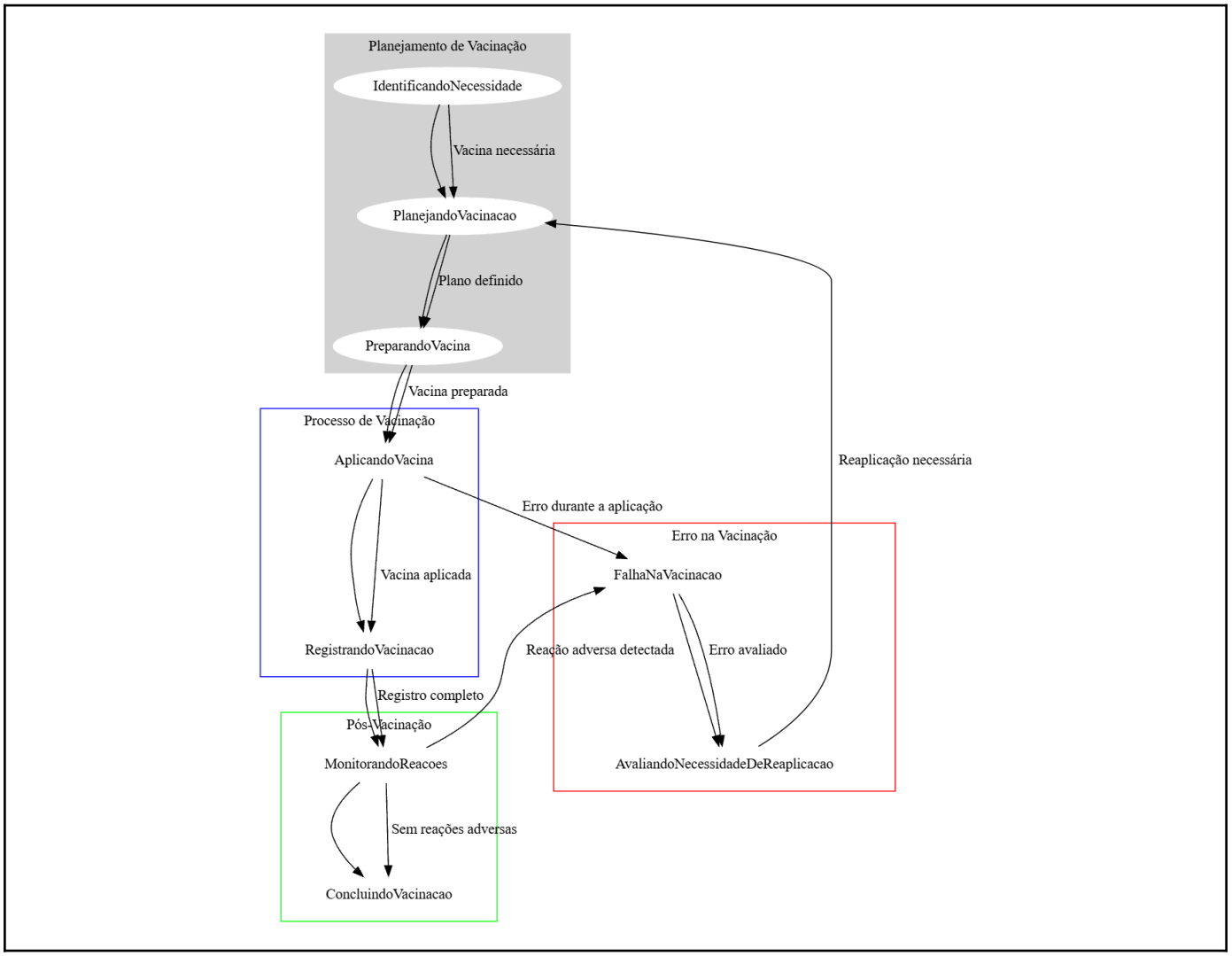
2.3.5 Diagrama de Estados

De modo a demonstrar o comportamento de um elemento por meio de transições, o diagrama de estados auxilia no entendimento da mudança de estado em resposta a eventos ou condições específicas, mostrando como as interações evoluem ao longo do tempo de maneira detalhada, evidenciando o fluxo de controle e dinâmica do projeto.

Também é baseado, dependendo da aplicação, em casos de uso específicos, como, por exemplo, na criação ou manipulação de instâncias de uma classe, onde o comportamento e as transições entre estados podem ser observados e mapeados de acordo com os eventos ou ações realizadas.

Foram desenvolvidos quatro diagramas de estados durante o desenvolvimento. Um exemplo é o diagrama que ilustra a transição de estados do planejamento de vacinação, desde a criação até a conclusão, considerando eventos como aplicações e reações adversas:

Diagrama 5 - Diagrama de Estados do Planejamento de Vacinas



Fonte: Os autores.

2.4 Banco de Dados

2.4.1 Geração de Scripts

A sequência do desenvolvimento se dá pela criação dos scripts de construção do banco de dados físico. Os scripts consistem em comandos escritos em SQL (*Structured Query Language*), que são utilizados para instruir o SGBD (Sistema Gerenciador de Banco de Dados) sobre como criar e manipular o banco de dados, suas tabelas e seus dados.

Para executar essa tarefa, a ferramenta *Visual Paradigm* foi utilizada, cuja finalidade foi de gerar scripts a partir de um modelo relacional já criado. Os comandos gerados foram exportados para o DBeaver, ferramenta utilizada para manipular o banco de dados, juntamente com o Eclipse, onde foi possível integrar a aplicação Java com o banco de dados, executar consultas, testar funcionalidades da API e verificar a consistência dos dados armazenados.

2.4.2 Comandos de Criação

São os comandos necessários para a criação do banco de dados, suas tabelas, colunas e documentações. Foram implementados índices e views para otimizar as consultas, facilitar o acesso às informações e garantir integridade referencial. Além disso, foram aplicadas *constraints* para assegurar a validade dos dados, como a integridade de chaves primárias, chaves estrangeiras, e a exclusividade de registros.

É possível visualizar abaixo um exemplo da criação da tabela Animal:

Script 1 - Criação da Tabela Animal

```
CREATE TABLE Animal (
  aniid            SERIAL NOT NULL,
  tipani           int4 NOT NULL CHECK(tipani = 1 or tipani = 2 or tipani = 3),
  datnas           date NOT NULL,
  pesani           numeric(19, 0) NOT NULL,
  stsani           int4 NOT NULL CHECK(stsani = 1 or stsani = 2),
  caumor           varchar(255) NOT NULL,
  obsani           varchar(255),
  Grupogruid       int4 NOT NULL,
  Granjacnpj       numeric(14, 0) NOT NULL,
  Localizacaoidloc int4 NOT NULL,
  PRIMARY KEY (aniid)
);
```

Fonte: Os autores.

2.4.3 Comandos de Inserção de Dados

Os comandos de inserção de dados em um banco de dados SQL são utilizados para adicionar novos registros nas tabelas e, seguindo a criação do banco de dados, é necessário inserir informações para garantir a integridade e a consistência dos dados, facilitando a recuperação e análise futura. Além disso, é crucial definir regras de validação para assegurar que cada registro esteja correto e completo.

O comando SQL utilizado para isso é o *INSERT*, que aloca os valores na tabela Granja, como demonstrado no exemplo abaixo:

Script 2 - Inserção de dados na tabela Granja

```sql
INSERT INTO Granja (cnpj, nomgra, telgran, qtdanigra, tipgra)
VALUES (12345678901234, 'Granja Principal', 12345678, 100, 1);

INSERT INTO Granja (cnpj, nomgra, telgran, qtdanigra, tipgra)
VALUES (98765432109876, 'Granja Secundária', 87654321, 50, 2);
```

Fonte: Os autores.

2.4.4 Comandos de Consultas

Os comandos de consulta em SQL são usados para buscar e recuperar dados de um banco de dados, com o *SELECT* sendo o principal comando. Junto a ele, foram utilizadas junções (*joins)* para combinar dados de várias tabelas, permitindo consultas mais complexas. Além disso, as *views* também foram utilizadas como consultas armazenadas, que simplificam e otimizam a recuperação de dados.

Pode ser conferido logo abaixo um script de consulta envolvendo total de animais por granja:

Script 3 - Consulta de Total de Animais por Granja

```sql
CREATE VIEW total_animais_por_granja AS
SELECT
    g.cnpj AS cnpj_granja,
    g.nomgra AS nome_granja,
    COUNT(a.aniid) AS total_animais
FROM
    Granja g
LEFT JOIN
    Animal a ON g.cnpj = a.Granjacnpj
GROUP BY
    g.cnpj, g.nomgra;
```

Fonte: Os autores.

2.4.5 Políticas de Acesso

As políticas de acesso de um banco de dados são constituídas por uma série de regras e mecanismos de controle que determinam quem tem a permissão para acessar, visualizar ou modificar os dados armazenados. Essas políticas englobam processos de autenticação para confirmar a identidade dos usuários, mecanismos de autorização para definir os níveis de permissão, e sistemas de monitoramento para registrar todas as atividades realizadas no banco de dados.

O principal objetivo dessas políticas é assegurar a proteção e integridade das informações, garantindo que somente usuários autorizados possam ter acesso, de acordo com as necessidades da organização e em conformidade com as regulamentações de proteção de dados.

Pode ser visualizado, no exemplo a seguir, o uso das permissões de acesso no usuário *admin_user*:

Script 4 - Gerenciamento de Permissões de Acesso

```
CREATE ROLE admin;
CREATE ROLE granja_manager;

CREATE USER admin_user WITH PASSWORD 'admin123';
CREATE USER manager_user WITH PASSWORD 'manager123';

GRANT admin TO admin_user;
GRANT granja_manager TO manager_user;


GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO granja_manager;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO admin;
```

Fonte: Os autores.

2.4.6 Regras de Integridade

Segundo Camargo (2001, p. 23), “A restrição de integridade é uma condição especificada no esquema do banco de dados, com o objetivo de restringir a entrada de dados não desejáveis a uma instância do banco de dados”. Manter a consistência em uma base de arquivos assegura a validação de integridade dos dados, impedindo que as informações inconsistentes ou inválidas sejam gravadas no banco de dados, garantindo a qualidade e a

precisão dos dados. Com a adição de gatilhos (*triggers)* nas restrições de integridade, foi possível automatizar a validação e a aplicação das regras de integridade durante o projeto.

No exemplo abaixo, a função verificar_peso_leitao() é responsável por validar o peso mínimo permitido para o registro ou atualização de um leitão na tabela Leitao juntamente com um *trigger:*

Script 5- Validação de peso do Leitao

```
CREATE OR REPLACE FUNCTION verificar_peso_leitao()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.peslei < 5 THEN
        RAISE EXCEPTION 'Peso inválido para o leitão. Deve ser maior que 5kg.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_peso_leitao
BEFORE INSERT OR UPDATE ON Leitao
FOR EACH ROW
EXECUTE FUNCTION verificar_peso_leitao();
```

Fonte: Os autores.

2.4.6 Regras de Negócio

Regras de negócio do sistema são diretrizes, restrições ou condições específicas que definem como um sistema deve operar em um contexto de negócio. Elas representam os processos, políticas e práticas organizacionais que o sistema precisa refletir, automatizar ou respeitar para atender às necessidades da organização ou dos usuários.

Por exemplo, o procedimento atualizar_status_granja() é uma função no banco de dados que, ao receber o CNPJ de uma granja como parâmetro, atualiza seu status (tipgra) para o valor 2 na tabela Granja. Ele automatiza a mudança de status, garantindo consistência nos dados ao realizar a operação diretamente no banco com base no identificador fornecido:

Script 6 - Regra de Negócio

```
CREATE OR REPLACE PROCEDURE atualizar_status_granja(cnpj_granja NUMERIC)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE Granja
    SET tipgra = 2
    WHERE cnpj = cnpj_granja;
END;
$$;
```

Fonte: Os autores.

2.5 Programação

O projeto de desenvolvimento da aplicação foi realizado utilizando uma série de tecnologias de desenvolvimento Java com conhecimentos obtidos durante as aulas de Programação II.

Para interagir diretamente com o banco de dados, foi utilizado JDBC (Java Database Connectivity), possibilitando a realização de operações como consultas, inserções, atualizações e exclusões de dados das tabelas do banco de dados.

O Extensible Markup Language (XML) foi empregado para configurar o arquivo persistence.xml do Java Persistence API (JPA) e para definir as configurações do Hibernate. Este arquivo contém informações sobre a conexão com o banco de dados e as entidades mapeadas.

Foi implementado o padrão Objeto de Acesso a Dados (DAO) para separar a lógica de acesso aos dados das outras lógicas da aplicação. Foram criadas classes DAO específicas para cada entidade, responsáveis por operações de CRUD.

A utilização da Java Persistence API (JPA) concedeu a permissão de mapear entidades Java para as tabelas do banco de dados, por meio da definição das anotações JPA apropriadas em cada entidade, garantindo assim um mapeamento preciso e correto.

O Hibernate foi utilizado como provedor de implementação do JPA, sendo ele configurado no projeto através do Maven, utilizando dependências adequadas no arquivo pom.xml, o que facilitou o gerenciamento do mapeamento objeto-relacional e operações de persistência.

Para criar a API RESTful da aplicação, foi empregado Spring Boot. Com o Spring Boot, foi realizado o desenvolvimento de endpoints para operações de CRUD e integração com o frontend. O uso do Spring Boot simplificou o desenvolvimento e a configuração do projeto.

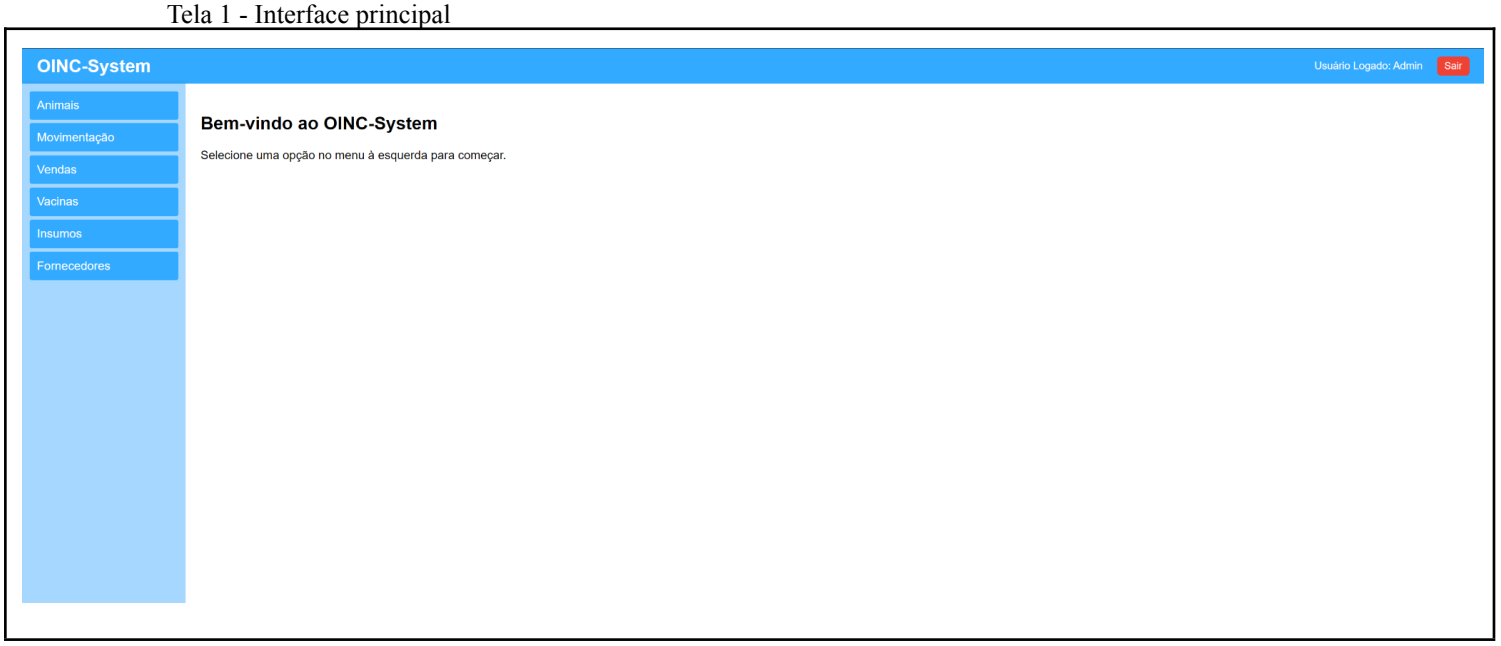
As telas do sistema foram desenvolvidas utilizando as linguagens HTML, CSS e JavaScript, que foram combinadas para criar interfaces de cadastro e outras funcionalidades. A seguir, é possível visualizar algumas telas do sistema e um código *model* do sistema:

Código 1 - Model da Granja

```java
@Entity
@Table(name = "Granja")
public class Granja {

    @Id
    @Column(name = "cnpj")
    private Long cnpj;

    @Column(name = "nomgra", nullable = false)
    private String nomgra; // Nome atualizado para corresponder ao banco

    @Column(name = "telgran", nullable = false)
    private Integer telgran;

    @Column(name = "qtdanigra", nullable = false)
    private Integer qtdanigra;

    @Column(name = "tipgra", nullable = false)
    private Integer tipgra;

    @OneToMany(mappedBy = "granja", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    private List<Usuario> usuarios;
```

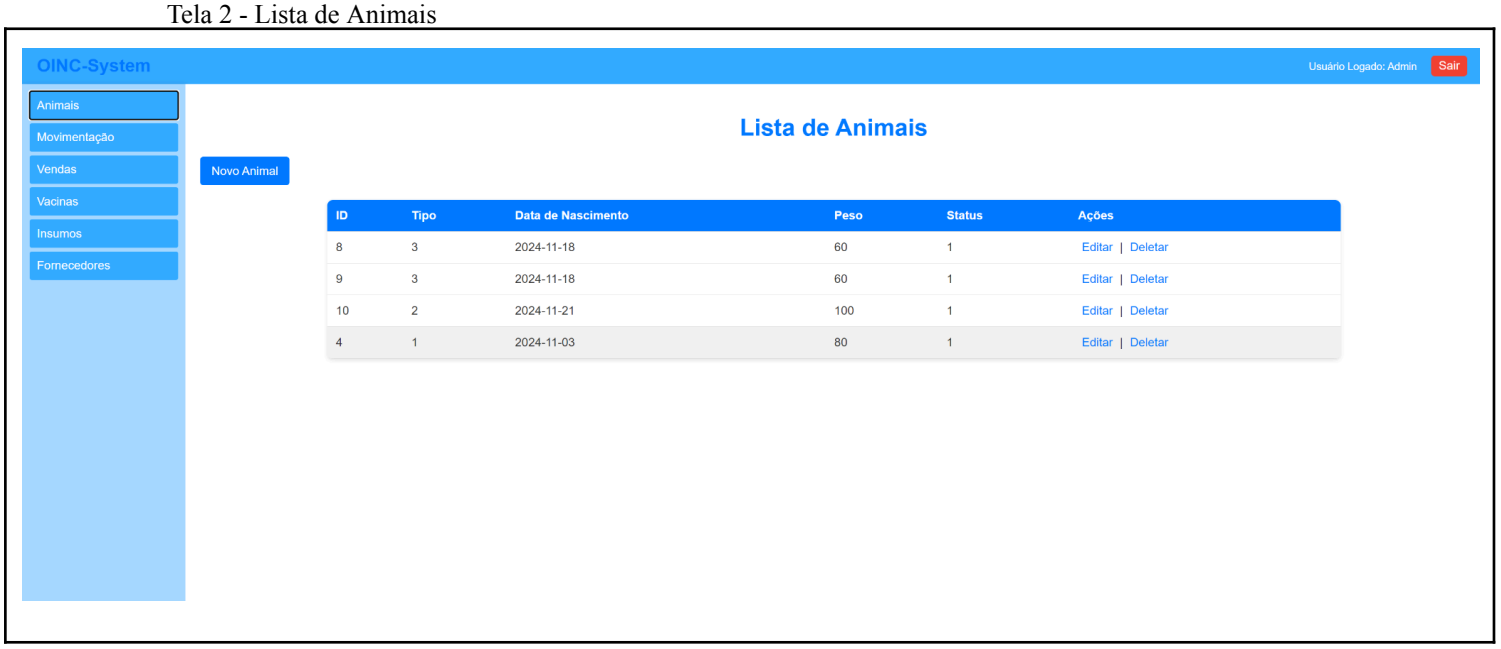
Fonte: Os autores.

Tela 1 - Interface principal



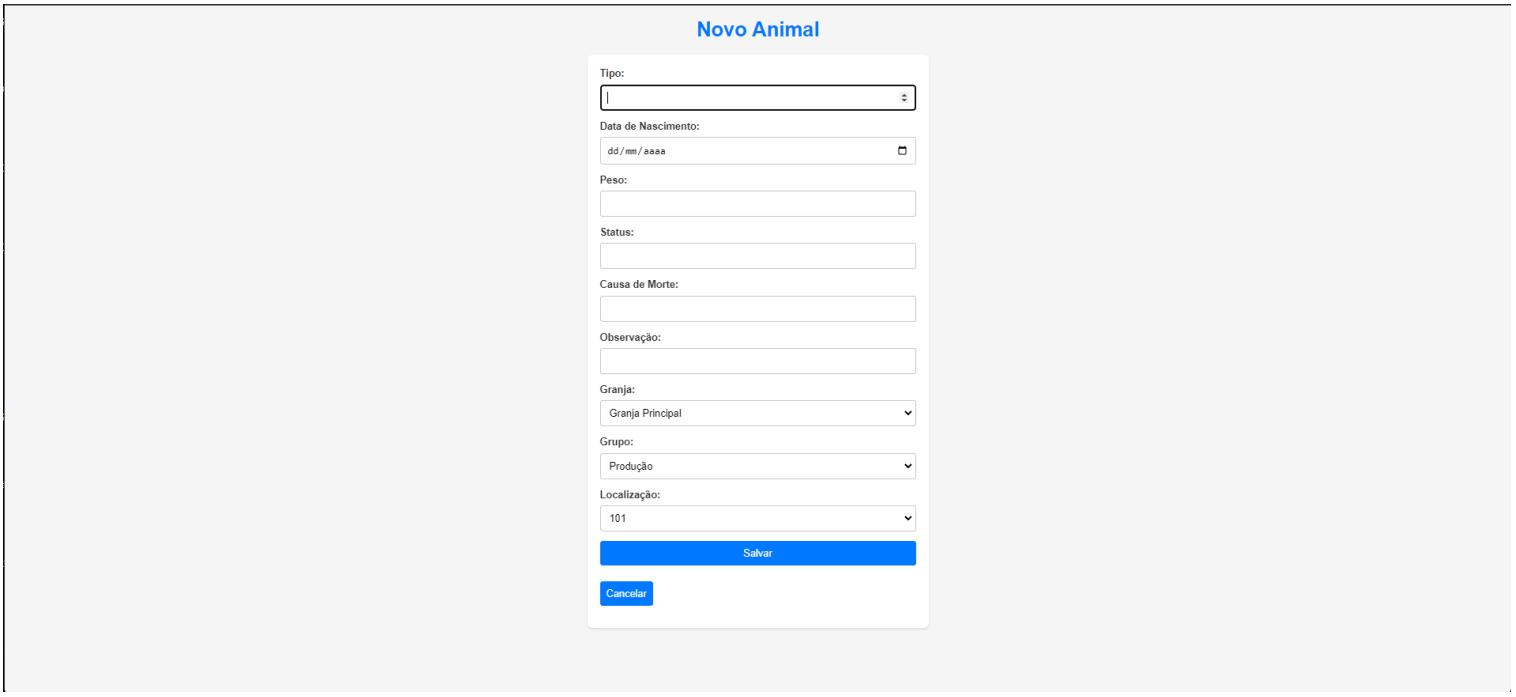
Fonte: Os autores.

Tela 2 - Lista de Animais



Fonte: Os autores.

Tela 3 - Cadastro de Animais



Fonte: Os autores.

3. CONCLUSÃO

A criação e implementação de um sistema de banco de dados para o manejo de uma granja de suínos marca uma importante evolução na gestão suinícola. Este projeto não apenas facilita a organização e o monitoramento das operações diárias, mas também proporciona uma ferramenta poderosa para a tomada de decisões estratégicas e melhoria contínua.

Com a centralização de informações sobre a saúde, alimentação, reprodução e outros aspectos essenciais dos suínos, os gestores podem identificar rapidamente problemas, otimizar recursos e melhorar a produtividade. O acesso a dados históricos e a capacidade de realizar análises detalhadas oferecem uma base sólida para o planejamento futuro, garantindo uma gestão mais proativa e eficaz.

Além disso, este sistema também reflete um compromisso com a sustentabilidade ambiental e o bem-estar animal, alinhando-se com as expectativas modernas de produção responsável.

Em conclusão, o projeto para gerenciamento de granjas de suínos é um passo crucial para modernizar e fortalecer a operação suinícola. Ao proporcionar uma gestão mais eficiente e informada, o sistema contribui significativamente para a sustentabilidade econômica e a competitividade do agronegócio, posicionando a granja para enfrentar os desafios e aproveitar as oportunidades do futuro.

REFERÊNCIAS

ABPA, **Associação Brasileira de Proteína Animal.** Relatório Anual 2023. p. 72-75. Disponível em: https://abpa-br.org/wp-content/uploads/2023/04/Relatorio-Anual-2023.pdf. Acesso em: 16 nov. 2024.

CAMARGO, Luiz Carlos. **Restrições de integridade e regras ativas em banco de dados distribuídos.** 2001. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2001. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/79996/182871.pdf?sequence=1&isAllowed=y. Acesso em: 20 nov. 2024.

COSTA, Filipe A. Dalla; COSTA, Osmar A. Dalla; ROHR, Stefan Alexander. **Bem-estar animal na produção de suínos:** práticas de manejo e características das instalações nas granjas**.** Brasília: Associação Brasileira dos Criadores de Suínos (ABCS), 2016. Disponível em: https://ainfo.cnptia.embrapa.br/digital/bitstream/item/143588/1/original8101.pdf. Acesso em: 16 nov. 2024.

GUEDES, Gilleanes T. A. **UML 2:** uma abordagem prática. São Paulo: Novatec, 2011. p. 19, 30-37.

MENDONÇA, Ricardo A. Ribeiro de. **Levantamento de requisitos no desenvolvimento ágil de software**. Cloudfront.net, 2014. Disponível em: https://acesse.dev/levantamento-de-requisitos. Acesso em: 18 nov. 2024.