Sistema para Rastreabilidade de Rebanhos

Cássio Jhonatan Gamarra Olegário[1] Alessandro André Mainardi de Oliveira[1]

[1]Ciência da Computação – UFN
Santa Maria, RS – Brasil

cassio.gamarra@ufn.edu.br, alessandromainardi.o@gmail.com

Abstract. *This work presents a development of a mobile application using the React Native framework and the Javascript language, in addition to the SQLite integrated database engine, which allows greater agility when sending and receiving information from a stick-shaped device, responsible for reading of radio frequency identifiers inserted in cattle. A web application with a React library and Javascript language, allowing information management and reporting, as well as a REST API with ExpressJS framework and Javascript language, in addition to the PostgreSQL database manager. For the development of all applications, the agile FDD methodology was used, following its best practices. All applications are intended to provide the rancher with useful information, enabling better traceability of his herd.*

Resumo. *Este trabalho apresenta o desenvolvimento de um aplicativo móvel utilizando o framework React Native e a linguagem Javascript, além do motor de banco de dados integrado SQLite, que permite maior agilidade ao enviar e receber informações de um dispositivo em forma de bastão, responsável pela leitura de identificadores de rádio frequência inseridos em bovinos. Uma aplicação web com a biblioteca React e a linguagem Javascript, permitindo o gerenciamento de informações e geração de relatórios, além de uma API REST com o framework ExpressJS e a linguagem Javascript, além do gerenciador de banco de dados PostgreSQL. Para o desenvolvimento de todas as aplicações, foi utilizada a metodologia ágil FDD, seguindo suas boas práticas. Todas as aplicações têm como objetivo proporcionar ao pecuarista informações úteis, que possibilitem uma melhor rastreabilidade de seu rebanho.*

1. Introdução

Nos últimos anos, foi possível perceber que a procura por carne de qualidade aumentou muito, as chamadas boutiques de carnes foram ganhando destaque no mercado e os requisitos para uma carne ser certificada se tornaram necessários para manter a qualidade da mesma [Gottems 2020].

Segundo Mendes (2011) em conversa com produtores, a maioria afirmou: "os produtos de *software* oferecidos são complexos e caros, pois tentam englobar várias etapas da produção, o que leva ao excesso de funções nos produtos (algumas vezes desnecessárias) que, apesar disso, não diz respeito à real situação do produtor". Com isso, o pecuarista não consegue utilizar o *software* de modo eficiente e, aqueles que o utilizam, complementarmente, utilizam planilhas agregadas. Foi apontada a falta de foco no cliente por parte dos desenvolvedores para ocorrência desse fato. Além disso,

ainda falta clareza na relação custo/benefício, por parte dos pecuaristas, das tecnologias hoje disponíveis: *chips*, brincos, códigos de barra, *software*, etc. Assim, é preciso que parta dos produtores a procura pela tecnologia (rastreamento de animais, *software* e entre outras).

Ainda segundo Mendes (2011), o Ministério da Agricultura, Pecuária e Abastecimento (MAPA) dá indícios de que acredita que a questão da rastreabilidade é um problema do produtor, que necessita de uma conscientização de que produz um alimento, já que o quesito segurança alimentar vem ganhando importância nos últimos anos, em virtude da presença de um mercado consumidor mais exigente.

Com isso, a rastreabilidade de rebanhos tem se mostrado cada vez mais importante nesse processo, tendo em vista que com ela é possível saber a origem do produto do campo até o consumidor. É um conjunto de medidas que possibilita controlar e monitorar todas as movimentações nas unidades, de entrada e saída, com o objetivo de produzir com qualidade de origem garantida [Silva 2005].

Utilizando *frameworks* para criação de aplicativos móveis, web e API's, é possível desenvolver uma plataforma completa, que permite que o pecuarista tenha total controle de seu rebanho, sabendo todo o histórico de um animal, desde seu nascimento até a venda do mesmo.

O projeto tem como objetivo apresentar ao produtor rural uma gama de ferramentas que auxiliem no controle de seus rebanhos, pois de acordo com Mendes (2011) a maioria das soluções presentes no mercado são defasadas ou pouco acessíveis. A criação de um aplicativo móvel proporcionou uma maior versatilidade ao Bastão de Leitura desenvolvido por Carvalho et al. (2020), facilitando a utilização do mesmo, não sendo mais necessário conectar via USB e editar o arquivo de texto que contém os dados e alertas, além disso pode deu continuidade ao trabalho de Boeira e Cantarelli (2016), que sugeriram a criação de uma aplicação móvel para facilitar a coleta de informações nas atividades cotidianas da criação de bovinos.

1.1. Objetivo Geral

Este trabalho teve por objetivo o desenvolvimento de uma aplicação móvel, uma aplicação web e uma API REST, sendo que a aplicação móvel tem como propósito facilitar o envio e recebimento de dados do Bastão de Leitura de Brincos RFID.

1.2. Objetivos Específicos

- Desenvolvimento de uma aplicação móvel utilizando a linguagem de programação Javascript e o *framework* React Native, além do motor de banco de dados integrado SQLite, para comunicação com o bastão de leitura RFID e funcionando *on-line* e *off-line*;
- Desenvolvimento de uma aplicação web, utilizando a linguagem de programação Javascript e a biblioteca React;
- Desenvolvimento de uma API REST, utilizando a linguagem de programação Javascript juntamente com o *framework* ExpressJS, além da utilização do Sistema de Gerenciamento de Banco de Dados (SGBD) PostgreSQL;
- Utilizar a metodologia *Feature Driven Development* (FDD) para a realização da análise e projeto das aplicações.

2. Referencial Teórico

Nessa seção, serão apresentados conceitos que serão utilizados para o desenvolvimento deste trabalho.

2.1. Pecuária e Rastreabilidade

O Brasil é considerado um país industrializado, ao mesmo tempo em que ocupa um dos primeiros lugares em produção agrícola e pecuária [Freitas 2021]. A Pecuária, atividade pertencente ao setor primário da economia, é uma das principais áreas em termos de produção de riqueza no país, estando cada vez mais interligada ao meio industrial e mais dependente das transformações nas técnicas e nos recursos tecnológicos [Procreare 2017].

Além disso, a pecuária exerce uma grande relevância nas exportações brasileiras, além de abastecer o mercado interno. No caso dos bovinos, além da carne, são extraídas outras matérias-primas, como o couro (produção de calçados), pele (vestuário), ossos (fabricar botões) e muitos outros [Freitas 2021].

Rastreabilidade é um mecanismo que permite identificar a origem do produto desde o campo até o consumidor, podendo este ter sido, ou não, transformado ou processado. É um conjunto de medidas que possibilita controlar e monitorar todas as movimentações nas unidades, de entrada e de saída, com o objetivo de produzir com qualidade e com origem garantida [Silva 2005].

O Sistema Brasileiro de Identificação Individual de Bovinos e Búfalos (SISBOV) é o sistema oficial de identificação individual de bovinos e búfalos, sendo que a adesão, pelos produtores rurais, é voluntária, exceto quando definida sua obrigatoriedade em ato normativo próprio, ou exigida por controles ou programas sanitários oficiais. Atualmente, a Instrução Normativa MAPA nº 51, de 1 de outubro de 2018, aprova, na forma de seu Anexo III, a norma operacional que é utilizada para embasar a certificação oficial brasileira para países que exijam a rastreabilidade individual de bovinos e búfalos, até que haja a homologação pelo MAPA e a implementação de protocolo de rastreabilidade de adesão voluntária que trata o art. 7º do Decreto nº 7.623, de 22 de novembro de 2011 [MAPA 2017].

2.3. O *framework* React Native e a Linguagem Javascript

Criado pelo Facebook em 2015 sobre a licença MIT, o React Native é um Framework para desenvolvimento de aplicativos móveis multiplataforma. Baseado no React, framework JS para desenvolvimento web, o React Native possibilita a criação de aplicações móvel multiplataforma (Android e iOS) utilizando apenas Javascript. Porém, diferente de outros frameworks com esta mesma finalidade (Cordova, por exemplo), todo o código desenvolvido com o React Native é convertido para linguagem nativa do sistema operacional, o que torna o app muito mais fluido. [Andrade 2021].

JavaScript (às vezes abreviado para JS) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas web. O JavaScript é uma linguagem baseada em protótipos, multiparadigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (como por exemplo a programação funcional) [Mozilla 2021].

Além dessas tecnologias, também é importante a utilização do banco de dados SQLite, permitindo armazenar todos os dados no dispositivo, durante seu uso *off-line*, pois o SQLite é um mecanismo de banco de dados SQL embutido. Ao contrário da maioria dos outros bancos de dados SQL, o SQLite não tem um processo de servidor separado. O SQLite lê e grava diretamente em arquivos [SQLite 2021].

2.4. A biblioteca React

React é uma biblioteca Javascript criada pelo Facebook, voltada para a construção da interface de aplicações. Seu foco é em aplicações web de código-fonte aberto e *front-end* [Dawson 2014]. A plataforma React provê recursos para o desenvolvimento de *Single Page Applications* (SPA). Ou seja, uma única página web, com o objetivo de fornecer a experiência ao usuário parecida com a de um aplicativo de desktop, onde o código é carregado na página única, de forma dinâmica.

2.5. O *framework* ExpressJS e o NodeJS

Tendo sua versão inicial lançada no ano de 2010, o Express.js (ou somente Express) é um *framework* para o desenvolvimento de aplicações JavaScript com o Node.js. De código aberto, sobre a licença MIT, o Express.js foi desenvolvido para otimizar a construção de aplicações web e APIs, tornando-se um dos *frameworks* mais populares da internet e que utiliza o Node para execução do javascript como linguagem de *back-end* [Andrade 2021].

Como um ambiente de execução JavaScript assíncrono orientado a eventos, o Node.js é projetado para desenvolvimento de aplicações escaláveis de rede. Os usuários do Node.js não precisam se preocupar com *deadlock* de processos, pois não existem *locks*. Quase nenhuma função no Node.js realiza diretamente operações de E/S, por essa razão o processo nunca bloqueia. Por não existirem operações bloqueantes, sistemas escaláveis são razoavelmente fáceis de serem desenvolvidos em Node.js [Foundation 2021].

2.6. API REST

API REST, também chamada de API RESTful, é uma interface de programação de aplicações que segue conformidade com as restrições da arquitetura REST. A sigla REST significa *Representational State Transfer* (Transferência Representacional de Estado, em português). Uma interface de programação de aplicações (API) é um conjunto de definições e protocolos para criar e integrar *softwares* de aplicações. Às vezes, as APIs são referidas como um contrato entre um provedor e um usuário de informações, estabelecendo o conteúdo exigido pelo consumidor (a chamada) e o conteúdo exigido pelo produtor (a resposta) [RedHat 2021].

2.7. Sistema Gerenciador de Banco de Dados PostgreSQL

PostgreSQL é um sistema de banco de dados relacional de código aberto que usa e estende a linguagem SQL [PostgreSQL 2021]. Pode ser facilmente integrado em aplicações NodeJS que utilizam o *framework* ExpressJS através da biblioteca node-postgres, além disso é possível utilizar *schemas*, que facilitam a separação de dados de clientes em uma mesma base de dados.

3. Trabalhos Correlatos

Nesta seção são apresentados os trabalhos relacionados ao sistema proposto, os quais possuem algumas características semelhantes ao presente trabalho, com o intuito de contribuir como base de conhecimento para a elaboração e organização desse projeto.

3.1. Desenvolvimento de um software para pecuária

Boeira e Cantarelli (2016), desenvolveu um *software* para gestão de pecuária. O sistema foi desenvolvido com a linguagem de programação PHP e o sistema gerenciador de banco de dados MySQL. O sistema permite o gerenciamento deusuários, grupos de acesso, informações administrativas e financeiras, gerenciamento de animais, lançamentos de medicamentos e processos necessários para a criação de bovinos, além disso um relatório sobre os animais.

Foi destacado no trabalho a relevância que uma aplicação móvel possui para o trabalho, uma vez que facilita a coleta de informações nas atividades cotidianas da criação de bovinos.

3.2. Uso de tecnologia na rastreabilidade do rebanho de corte

Trigo et al. (2018), apresenta uma proposta de uma solução para os produtores de gado de corte, de pequeno ou médio porte, visando o acesso a todas as informações necessárias sobre o rebanho, desde a quantidade de animais, pesos e vacinas, até o tamanho da área de confinamento, através de um aplicativo.

Segundo os autores, o aplicativo foi desenvolvido exclusivamente para uso da empresa Agro Sky, possuindo conexão com Drones e microchips implantados nos animais, além disso, para utilizar todos os serviços, seria necessário o pagamento de uma mensalidade de acordo com o tamanho do rebanho, propriedade e telas acessadas

3.3. Bastão de leitura de brinco eletrônico RFID

Carvalho et al. (2020), desenvolveu um leitor RFID em formato de bastão, que realiza a leitura de brincos que identificam os bovinos. O usuário tem a possibilidade de ler um brinco durante o manejo dos animais, onde os dados são exportados para um sistema ou tabelas digitais. No momento do manejo, o sistema irá informar se o bovino possui observações salvas no sistema, possibilitando um maior cuidado do rebanho por parte de seus criadores.

O módulo *bluetooth* possibilita que a importação e exportação de dados seja feita de modo sem fio. Possibilita também a criação de um aplicativo, assim o bastão seria construído apenas com módulo *bluetooth*, leitor RFID e Arduino, nesse caso, quando feita a leitura de um brinco, o leitor enviará o número para o aplicativo, possibilitando o gerenciamento dos dados no dispositivo móvel.

No trabalho, os autores destacam que seria relevante a criação de um aplicativo para dispositivos móveis, onde possa ser feito o gerenciamento dos dados referente aos animais, além de se conectar com algum servidor para exportação dos dados.

3.4. Considerações sobre os trabalhos correlatos

O trabalho de Boeira e Cantarelli (2016), que foi o desenvolvimento de um sistema para gerenciamento de propriedades rurais, propõe como trabalho futuro um aplicativo para

dispositivos móveis, facilitando assim a coleta de informações nas atividades cotidianas da criação de bovinos.

O trabalho de Trigo et al. (2018), visou o desenvolvimento de um aplicativo, porém com o foco em uma única empresa, o que dificulta o uso por parte do público alvo,que é o produtor de gato de corte, de pequeno ou médio porte, além disso, não foram encontradas evidências de que o trabalho teve prosseguimento e conclusão.

Já o trabalho de Carvalho et al. (2021), desenvolveu um leitor RFID em formato de bastão, onde existe uma possibilidade de conexão via *bluetooth* para envio e recebimento de dados, além disso, é proposto como trabalho futuro o desenvolvimento de um aplicativo que se conecte com o bastão, com o objetivo de gerenciar os dados referentes aos animais e se conectar com um servidor para realizar a exportação dos dados.

A proposta então foi dar continuidade ao trabalho de Boeira e Cantarelli (2016) e Carvalho et al. (2021), para isso desenvolvendo um aplicativo móvel que realize a comunicação com o leitor RFID e exporte os dados para um servidor. A Figura 1 demonstra a comunicação entre o bastão e o aplicativo móvel.

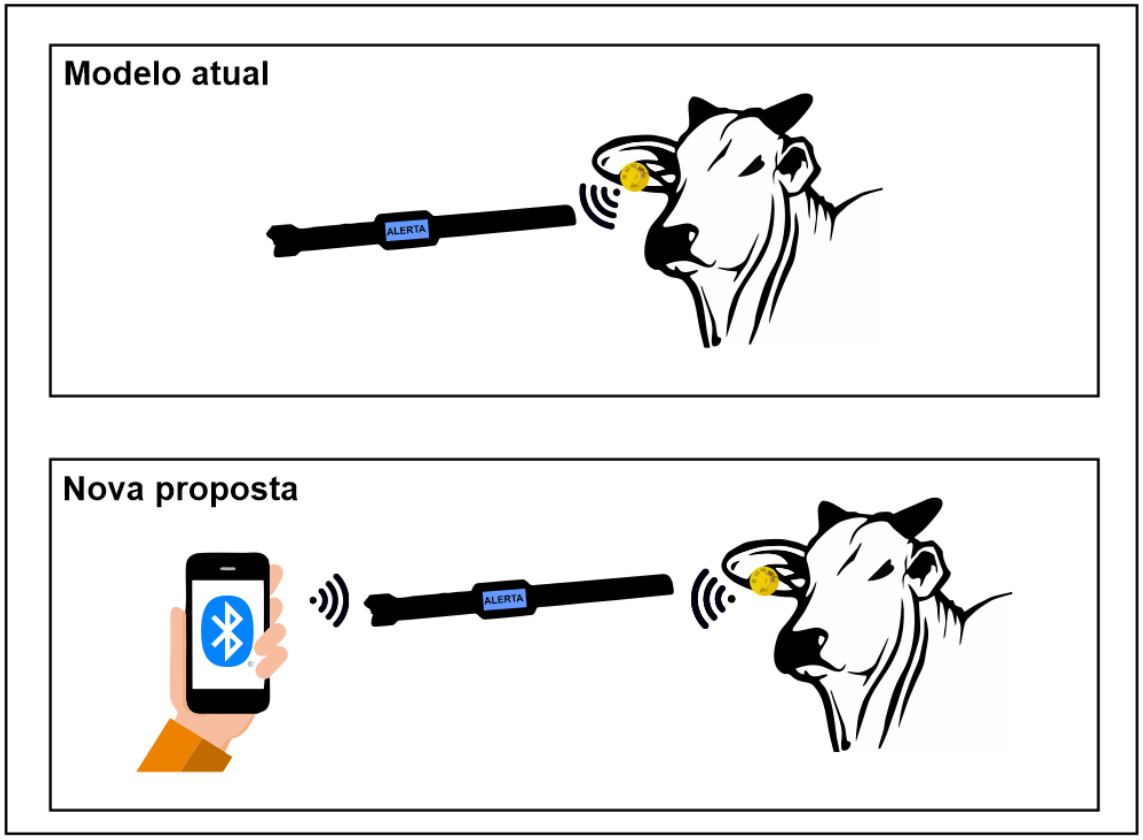

Figura 1. Proposta para a aplicação.

4. Metodologia

Todo projeto precisa de planejamento e organização para que se possa minimizar os futuros erros que podem ocorrer. Na criação de sistemas esse planejamento pode ser feito através de documentação, é necessário utilizar uma metodologia adequada. Para

conseguir um bom resultado no desenvolvimento deste projeto, vai ser utilizada a metodologia *Feature Driven Development* (FDD).

4.1. *Feature-Driven Development* (FDD)

Desenvolvida por Nebulon Jeff De Luca, a metodologia foi criada em 1997 e surgiu a partir do gerenciamento de projetos de Jeff De Luca, onde o objetivo da metodologia é buscar o desenvolvimento da aplicação por funcionalidades, de maneira que seja utilizada para projetos que estão em fase inicial ou aqueles que já possuem codificação [Rocha 2013].

O FDD busca o desenvolvimento por funcionalidade, ou seja, por um requisito funcional do sistema. É prático para o trabalho com projetos iniciais ou projetos com codificações existentes e conta com cinco processos:

- Desenvolver um modelo abrangente: Consiste na definição dos requisitos do domínio do sistema para sua construção e documentação da especificação de suas funcionalidades;
- Construção da lista de funcionalidades: Momento onde cada funcionalidade é revisada pelo cliente (Apêndice A);
- Planejamento por funcionalidades: Consiste em criar uma ordem sequencial onde as funcionalidades são executadas de acordo com sua prioridade (Apêndice B);
- Detalhamento por funcionalidades: etapa que visa possibilitar o entendimento do fluxo de execução do sistema (Apêndice C), suas atividades e respectivos tratamentos;
- Construir por funcionalidades: Consiste na implementação das funcionalidades requeridas pelo sistema, obedecendo, também, uma ordem prioridade (Apêndice D).
- Protótipos de interface: Consiste na construção de modelos de como as telas devem ser apresentadas (Apêndice E).

4.2. Desenvolvimento de um modelo abrangente

O primeiro passo para entender como a aplicação será criada é desenvolver uma visão geral do trabalho. Nesse processo, é realizado um estudo sobre o escopo do sistema e, a partir disso, são realizados estudos com maior riqueza de detalhes no que abrange o domínio do negócio, observando cada área a ser feita a modelagem, desenvolvendo o Diagrama de Domínio, representado na Figura 2, o qual deve ser uma visão geral do projeto, mostrando a área de domínio da aplicação e revelando, graficamente, essa visão do *software* [Retamal 2008]. No diagrama de domínio, as entidades na cor amarela são comuns as aplicações web, móvel e API REST. As entidades em azul são exclusivas da aplicação móvel, voltadas para gerenciamento de configurações.

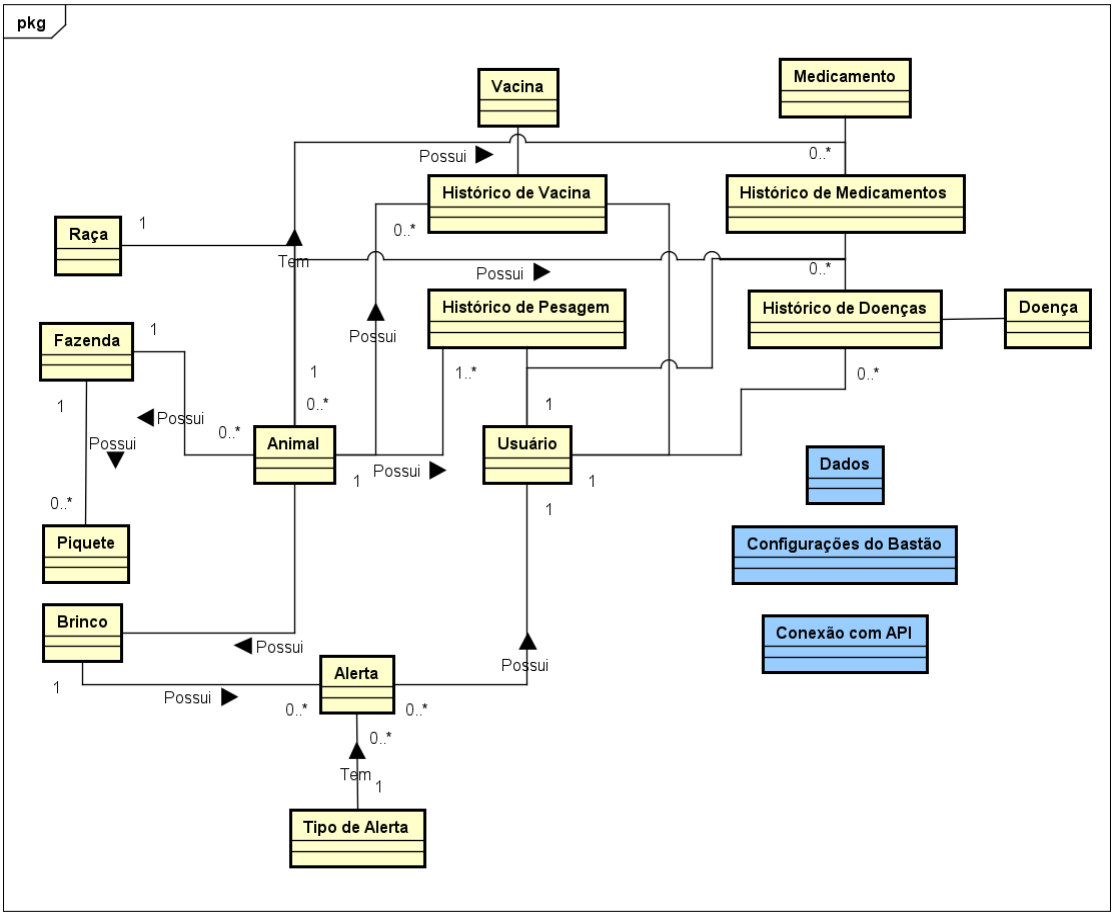

Figura 2. Diagrama de domínio.

4.3. Construindo por funcionalidade

Essa atividade é executada somente uma vez para cada funcionalidade, com o objetivo de produzir um Pacote de Arquitetura por funcionalidade. Foram desenvolvidos o Diagrama de Atividade, o Diagrama de Classe e o Diagrama de Entidade Relacional (Apêndice D).

No processo de desenvolvimento do sistema, os dois pontos mais importantes são a conexão do Bastão de Leitura com o aplicativo móvel através do bluetooth. A Figura 3 apresenta a função verificaConexao responsável pela conexão com o Bastão de Leitura.

```
async function verificaConexao(item) {
  if(item.name !== 'HC-06') {
    Toast.show({
      type: "error",
      text1: 'Por favor, selecione o bastão (HC-06).',
      position: 'bottom'
    });
    return;
  } else {
    startLoading();
    try {
      const connection = await item.isConnected();
      if (!connection) {
        connection = await item.connect({ DELIMITER: '\r' });
      }
      const bastao = Bastao.getInstance();
      bastao.setBastao(item);
      Toast.show({
        type: "success",
        text1: 'Bastão conectado.',
        position: 'bottom'
      });
      stopLoading();
    } catch (error) {
      stopLoading();
    }
  }
}
```

Figura 3. Função responsável pela conexão com o bastão de leitura.

Também é importante a classe responsável pela sincronização entre aplicação web e móvel, através da API REST, onde a base de dados da API e da aplicação móvel são atualizadas. A Figura 4 apresenta o trecho de código responsável pela sincronização na aplicação móvel, onde após o retorno da API, as tabelas são recriadas, após a criação, os dados retornados pela API são inseridos em suas respectivas tabelas.

```
if(callBackPost && callBackPost.data.statusCode === 200) {
  try {
    SQLiteManager.dropTables();
    SQLiteManager.createTablesFromSchema();
    if(token) {
      try {
        const getDados = await api.get('/sincronizar', {
          headers: { Authorization: "Bearer " + token }
        });

        for(let animal of getDados.data.animais) {
          await SQLiteManager.addTableAnimal(animal);
        }
```

Figura 4. Trecho de código responsável pela sincronização da aplicação móvel.

Já a Figura 5 apresenta o trecho de código responsável pela sincronização na API REST, onde são montados *arrays* para a inserção de medicamentos.

```
const valuesHistoricoMedicamentos = [];
for(let item of historicoMedicamentos) {
  valuesHistoricoMedicamentos.push([idFuncionario, item.ID_ANIMAL, item.ID_MEDICAMENTO, item.OBSERVACAO, item.DATA, item.HORA])
}
const insertHistoricoMedicamentos = format(`
  INSERT INTO ${schema}.HISTORICO_MEDICAMENTOS
    (ID_FUNCIONARIO, ID_ANIMAL, ID_MEDICAMENTO, OBSERVACAO, DATA, HORA)
  VALUES %L`, valuesHistoricoMedicamentos);

pool.connect( async (err, client, done) => {
  if (err) throw err;

  if(valuesHistoricoMedicamentos.length > 0) await client.query(insertHistoricoMedicamentos);

  done();
```

Figura 5. Trecho de código responsável pela sincronização da API.

5. Resultados

Como resultado do presente trabalho, tem-se a construção da aplicação web para gerenciamento dos dados do rebanho, a aplicação móvel trabalhando em conjunto com o bastão e a API REST realizando o intercâmbio de informações entre os dois sistemas citados.

É importante ressaltar que durante o desenvolvimento, foi necessário substituir o *framework* Flutter pelo *framework* React Native. Essa mudança se deve principalmente ao fato de que como o React Native utiliza como linguagem o Javascript, todas as aplicações (web, móvel e api), possuem a mesma linguagem, facilitando o desenvolvimento.

Todas as interfaces de cadastro e listagem possuem o mesmo padrão, facilitando o uso por parte do administrador do sistema. Devido ao grande número de interfaces, serão apresentadas as mais importantes para o sistema web, que são a listagem de animais, cadastro de animais e cadastro de brincos e as mais importantes para aplicação móvel, que são a conexão com o Bastão de Leitura RFID e as rotinas com animais. A Figura 6 apresenta a tela inicial da aplicação web, contendo todos menus que o administrador possui acesso.

Figura 6. Interface inicial da aplicação web.

Já a Figura 7 apresenta a interface de gerenciamento de animais, exibindo uma lista com os animais cadastrados no sistema e com as opções de adicionar, editar ou excluir um animal.

Figura 7. Interface de gerenciamento de animais da aplicação web.

A Figura 8 apresenta a interface de cadastro de um novo animal, onde os campos com asterisco são obrigatórios, caso o usuário tente salvar os dados sem preencher esses campos, a aplicação retorna uma mensagem de erro em cada campo.

Figura 8. Interface de cadastro de animal da aplicação web.

A Figura 9 apresenta a interface de cadastro de um novo brinco, onde os campos com asterisco são obrigatórios. O usuário pode selecionar um animal na hora de cadastrar um brinco ou apenas cadastrar o código RFID ou código visual.

Figura 9. Interface de cadastro de brincos da aplicação web.

A Figura 10 apresenta algumas interfaces do aplicativo móvel, sendo elas a interface de registro de rotinas e a interface de conexão com o bastão. Ao aproximar o leitor de um brinco, é possível adicionar um novo alerta ou realizar o registro de peso do animal.

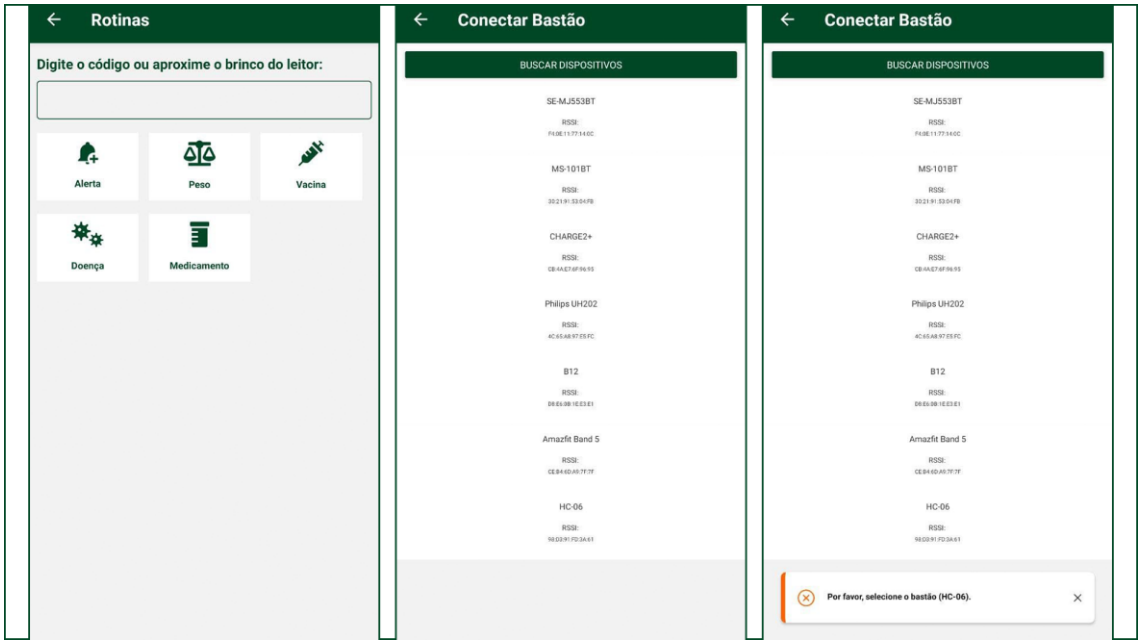

Figura 10. Interfaces da aplicação móvel.

Devido ao momento de pandemia, o sistema foi testado apenas localmente, fazendo a leitura de brincos e enviando os dados via *bluetooth*. Para este fim, foi necessário remover alguns trechos de códigos do Arduino contido no bastão, tornando mais específico para a conexão via *bluetooth.* Além disso foram feitos testes de sincronização e *deploy* da aplicação em um servidor web.

6. Conclusão

Este trabalho possibilitou o desenvolvimento e a implementação de um sistema para pecuária, desenvolvendo um aplicativo móvel com o objetivo de fornecer mais versatilidade ao Bastão de Leitura de Brincos RFID, desenvolvido por Carvalho et al. (2020) e ao mesmo tempo dando continuidade ao trabalho de Boeira e Cantarelli (2016).

Em relação aos trabalhos relacionados, eles foram essenciais para a construção deste trabalho, pois sendo este uma continuação dos mesmos, ajudaram a entender o que seria necessário ser implementado e quais ferramentas seriam mais eficientes para este fim.

Como metodologia para desenvolvimento do trabalho, foi escolhida a FDD por ser uma metodologia ágil que se destaca pelo desenvolvimento por funcionalidade, entregas frequentes e permitindo modificações casuais no projeto, caso necessário.

Para atender as necessidades do projeto, foi desenvolvida uma API REST, que permite a integração da aplicação web com a aplicação móvel, de uma forma centralizada. A aplicação móvel proporcionou uma maior versatilidade ao bastão de leitura de brincos RFID, o sistema recebe a leitura dos brincos através do *bluetooth*, e a partir disso é possível visualizar de forma mais clara quais brincos foram lidos pelo mesmo, facilitando a geração de alertas, registro de alteração de peso, vacinas, medicamentos e doenças. Além disso, é possível enviar os dados para um servidor, armazenando-os em uma base de dados. A aplicação web permite um gerenciamento do

rebanho e também a geração de relatórios, proporcionando uma melhor rastreabilidade do rebanho, pois é possível ter acesso a todo o histórico dos animais.

O sistema desenvolvido apresenta as principais funcionalidades necessárias para o gerenciamento de um rebanho, sendo necessários apenas ajustes para correção de possíveis *bugs* que possam ocorrer.

Como sugestão de trabalhos futuros que poderão ser desenvolvidos para buscar ainda mais controle para o pecuarista, como a criação de relatórios avançados, refatoração de alguns pontos do projeto, trazendo um código mais limpo e de mais fácil manutenção. Integração com balanças que permitam a conexão via *bluetooth* para um registro mais preciso de pesagem. Além disso também seria interessante testar o aplicativo em uma propriedade, podendo gerar relatórios e dados reais.

Referências

Andrade, A. P. (2021). "O que é o express.js?" https://www.treinaweb.com.br/blog/o-que-e-o-express-js, janeiro.

Andrade, Ana Paula de. (2021). O que é o React Native. https://www.treinaweb.com.br/blog/o-que-e-o-react-native.

Boeira, Uriel S. e Cantarelli, G. S. (2016). "Desenvolvimento de um software para pecuária." https://www.ufn.edu.br/eventos/maiseventos/Edicaoanterior.aspx?qtd=6259, outubro.

Brasil (2011). Decreto nº 7.623, de 22 de novembro de 2011. http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/decreto/d7623.htm. novembro.

Carvalho, Guilherme L., Oliveira, Alessandro A. M., Olegário, Cássio J. G., Pereira, Gustavo G., Hartmann, Frederico G. e Frigo Junior, J. M. (2020). Engenharia no Século XXI, Capítulo 21. Ed. Poisson. https://poisson.com.br/2018/produto/engenharia-no-seculo-xxi-volume-20/. outubro.

Dawson, C. (2014). Javascript's history and how it led to reactjs. https://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/. julho.

Foundation, O. (2021). Sobre node.js. https://nodejs.org/pt-br/about/. janeiro.

Freitas, E. d. (2021). Pecuária brasileira. https://brasilescola.uol.com.br/brasil/pecuaria.htm. maio.

Gottems, L. (2020). 4 tendências da indústria da carne em 2021. https://www.agrolink.com.br/noticias/4-tendencias-da-industria-da-carne-em-2021--confira_443849.html. dezembro.

Guedes, M. (2019). O que é dart? https://www.treinaweb.com.br/blog/o-que-e-dart/. outubro.

MAPA Ministério da Agricultura, Pecuária e Abastecimento, (2018). Instrução normativa nº 51, de 1 de outubro de 2018. https://www.in.gov.br/web/guest/materia/-/asset_publisher/Kujrw0TZC2Mb/content/id/44306336/do1-2018-10-08-instrucao-normativa-n-51-de-1-de-outubro-de-2018-44306204. outubro.

MAPA Ministério da Agricultura, Pecuária e Abastecimento, (2021). Sisbov.

https://www.gov.br/agricultura/pt-br/assuntos/sanidade-animal-e-vegetal/saude-animal/rastreabilidade-animal/sisbov. abril.

Mendes, C. I. C., Oliveira, D. R. M. dos S. e Santos, A. R. d. (2011). Estudo do mercado brasileiro de software para o agronegócio. https://www.embrapa.br/busca-de-publicacoes/-/publicacao/924562/estudo-do-mercado-brasileiro-de-software-para-o-agronegocio.

Mozilla (2021). Javascript. https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. abril.

PostgreSQL (2021). What is postgresql? https://www.postgresql.org/about/. abril.

Procreare (2017). Pecuária no brasil. https://procreare.com.br/pecuaria-no-brasil/. março.

RedHat (2021). O que é api rest? https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. abril.

Retamal, A. M. (2008). Feature-driven development: descrição dos processos. http://www.featuredrivendevelopment.com/. novembro.

Rezende, D. A. (2005). Engenharia de Software e Sistemas de Informação. 3ª edição. Ed. Brasport.

Rocha, F. G. (2013). Introdução ao fdd - feature driven development. https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971. novembro.

Silva, F. G., Hoentsch, S. C. P. e Silva, L. (2009). Uma análise das Metodologias Ágeis FDD e Scrum sob a Perspectiva do Modelo de Qualidade MPS. BR, volume 5 of 12. Scientia Plena.

Silva, R. d. O. P. e. (2005). Rastreabilidade nas cadeias de carnes. http://www.iea.sp.gov.br/out/verTexto.php?codTexto=2509. junho.

SQLite (2021). About sqlite. https://www.sqlite.org/about.html.

Trigo, I. A., Yada, Marcela M., Lourençano, Ludmila da S. e Lima, Y. K. (2018). Uso de tecnologia na rastreabilidade do rebanho de corte. https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/464. dezembro

Apêndice

Apêndice A. Construção da lista de funcionalidades

Neste processo, com base no domínio da aplicação, é realizado o levantamento dos requisitos necessários para o cumprimento das necessidades do cliente [Silva 2009]. O resultado que se obtém são os quadros listados abaixo, onde no Quadro 1 estão as Regras de Negócio (RN).

Quadro 1. Regras de Negócio

ID	DESCRIÇÃO
RN01	Deve existir uma aplicação móvel.
RN02	Deve existir uma aplicação web.
RN03	Deve existir uma API REST, para integração entre site e o aplicativo móvel
RN04	Deve existir uma API para integração entre o site e o aplicativo móvel
RN05	O aplicativo móvel deve se comunicar via *bluetooth* com o Bastão de Leitura de Brincos RFID, permitindo a manipulação do arquivo de alertas
RN06	O cadastro de proprietários deve ser apenas na aplicação web
RN07	O gerenciamento de fazendas deve ser feito apenas por proprietários, na aplicação web
RN08	O gerenciamento de piquetes deve ser feito apenas por proprietários, na aplicação web
RN09	O gerenciamento de funcionários deve ser feito apenas por proprietários, na aplicação web
RN10	O gerenciamento de animais deve ser feito apenas por proprietários, na aplicação web
RN11	O gerenciamento de vacinas deve ser feito apenas na aplicação web
RN12	O gerenciamento de medicamentos deve ser feito apenas na aplicação web
RN13	O gerenciamento de doenças deve ser feito apenas na aplicação web
RN14	O gerenciamento de tipos de alertas deve ser feito apenas na aplicação web
RN15	Os brincos podem ser RFID ou visuais, sendo possível cadastrar o código RFID ou visual no sistema
RN16	Os históricos devem armazenar o identificador do animal, permitindo reaproveitamento do brinco
RN17	Deve ser possível gerar relatórios na aplicação web

O Quadro 2 contém os Requisitos Funcionais (RF), que podem ser definidos como as funções ou atividades que o sistema deve realizar [Rezende 2005].

Quadro 2. Requisitos Funcionais

ID	DESCRIÇÃO	RELEVÂNCIA	COMPLEXIDADE
RF01	**Requisitos do aplicativo móvel**		
RF01.1	Permitir o gerenciamento de animais	Essencial	Média
RF01.2	Permitir o gerenciamento de brincos	Essencial	Média
RF01.3	Permitir o gerenciamento de alertas	Essencial	Média
RF01.4	Permitir o gerenciamento de históricos de pesagem, vacinas, medicamentos e doenças	Essencial	Média
RF01.5	Permitir a comunicação *bluetooth* com o bastão de leitura RFID	Essencial	Alta
RF01.6	Permitir o gerenciamento do arquivo de alertas do bastão de leitura RFID	Essencial	Alta
RF02	**Requisitos da aplicação web**		
RF02.1	Permitir o gerenciamento de proprietários	Essencial	Média
RF02.2	Permitir o gerenciamento de fazendas	Essencial	Média
RF02.3	Permitir o gerenciamento de piquetes	Essencial	Média
RF02.4	Permitir o gerenciamento de funcionários	Essencial	Média
RF02.5	Ao cadastrar proprietários, o sistema deve inserir um usuário ligado ao proprietário, com o nível de acesso 2 (administrador) com o status 0 (inativo), solicitando a confirmação do cadastro através do e-mail. Após a confirmação do e-mail, o status deve ser alterado para 1 (ativo).	Essencial	Média
RF02.6	Ao cadastrar funcionários, o sistema deve inserir um usuário	Essencial	Média

	de nível 2 (funcionário) com o status 1(ativo).		
RF02.7	Permitir o gerenciamento de animais e raças.	Essencial	Média
RF02.8	Permitir o gerenciamento de vacinas.	Essencial	Média
RF02.9	Permitir o gerenciamento de medicamentos.	Essencial	Média
RF02.10	Permitir o gerenciamento de doenças.	Essencial	Média
RF02.11	Permitir o gerenciamento de tipos de alertas e alertas.	Essencial	Média
RF02.12	Permitir a visualização de históricos de pesagem, vacinas, medicamentos e doenças.	Essencial	Média
RF02.13	Permitir a geração de relatórios com base nos histórico	Essencial	Média
RF03	**Requisitos da API**		
RF03.1	Integração entre site e aplicativo móvel	Essencial	Média
RF03.2	Gerenciar as sessões web através do uso de *tokens*	Essencial	Média
RF03.3	Deve prover todas as opções de gerenciamento do aplicativo móvel e aplicação web	Essencial	Média

No Quadro 3 os Requisitos Não Funcionais (RNF), que são aqueles que não interferem diretamente no desenvolvimento do sistema [Rezende 2005].

Quadro 3. Requisitos Não Funcionais

ID	DESCRIÇÃO	RELEVÂNCIA	COMPLEXIDADE
RNF01	Utilizar a linguagem de programação Dart	Essencial	Média
RNF02	Utilizar o *framework* Flutter	Essencial	Média
RNF03	Utilizar SQLite na aplicação móvel	Essencial	Baixa
RNF04	Utilizar a linguagem de programação Javascript	Essencial	Baixa

RNF05	Utilizar a biblioteca React	Essencial	Baixa
RNF06	Utilizar o *framework* ExpressJS	Essencial	Baixa
RNF07	Utilizar o SGBD PostgreSQL	Essencial	Baixa
RNF08	Utilizar o protocolo *bluetooth* para a comunicação entre bastão e aplicativo móvel	Essencial	Alta

Apêndice B. Planejamento por funcionalidade

Na etapa de planejamento por funcionalidade é definida a ordem da implementação dos requisitos funcionais, baseada na complexidade e dependências desses, estimando-se um tempo de desenvolvimento para cada funcionalidade do *software*.

No Quadro 4, é demonstrado o Planejamento por Funcionalidade, onde é possível ver o cálculo estimado de horas para a implementação de cada funcionalidade, sendo que foram agrupados as funcionalidades da aplicação web e API REST, para maior agilidade durante o desenvolvimento.

Quadro 4. Planejamento por Funcionalidade

ORDEM	**FUNCIONALIDADE**	**TEMPO ESTIMADO DE DESENVOLVIMENTO**
-	**Aplicação web e API REST**	**-**
1	Gerenciamento de Usuários (proprietários e funcionários)	40 horas
2	Gerenciamento de Fazendas e Piquetes	30 horas
3	Gerenciamento de Animais e Raças	30 horas
4	Gerenciamento de Brincos, Alertas e Tipos de Alerta	30 horas
5	Gerenciamento de medicamentos, doenças e vacinas	30 horas
6	Gerenciamento de históricos (medicamentos, pesagem, doenças e vacinas)	60 horas
-	Aplicação Móvel	-
7	Gerenciamento de históricos (medicamentos,	120 horas

	pesagem, doenças e vacinas)	
8	Envio e recebimento de arquivos via *bluetooth*	120 horas

Apêndice C. Detalhamento por funcionalidade

Nesta parte o propósito é construir o Diagrama de Caso de Uso, o mesmo tem como finalidade descrever as principais funcionalidades do sistema, mostrando qual a interação dessas funcionalidades com os usuários, não dando importância para detalhes técnicos que expliquem como o sistema faz o processo. Na Figura 11 é mostrado o caso de uso do sistema.

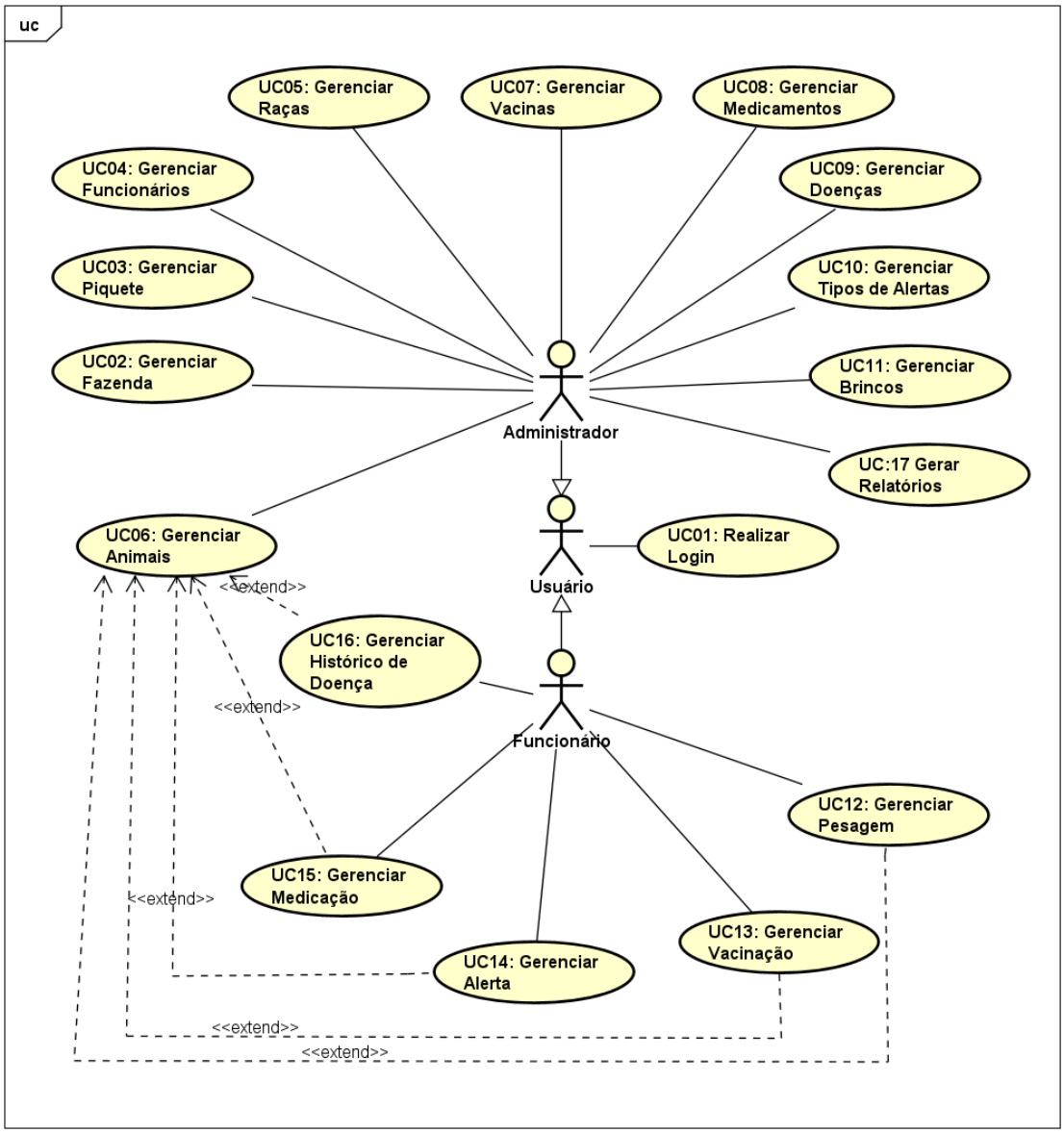

Figura 11. Diagrama de caso de uso.

Logo abaixo, são expostos dois descritivos de caso de uso, considerando os mais relevantes para o desenvolvimento da aplicação. No Quadro 5, é apresentado o descritivo de caso de uso da funcionalidade Gerenciar Animal, que é composta pelos processos de cadastro, edição, alteração, exclusão e visualização de um animal

Quadro 5. Descritivo de caso de uso: Gerenciar Animal

Identificação	UC06
Caso de uso	Gerenciar Animal
Descrição	Descrição do processo de gerenciamento de animais, com Cadastro, Visualização, Edição e Exclusão
Atores Principais	Administrador
Atores Secundários	N/A
Pré-condição	O usuário administrador deve estar logado no sistema.
Pós-condição	Animal cadastrado, alterado, excluído ou consultado com sucesso.
Fluxo Principal	
	FP01 - O sistema exibe uma tabela contendo os animais cadastrados.
	FP02 - O ator pode clicar no botão "Incluir"para adicionar um novo animal[FA1] ou no botão "Visualizar", para ver detalhes de um animal [FA2] ou no botão "Editar", para realizar alterações no cadastro do animal[FA3] ou clicar no botão "Excluir", para excluir um animal[FA4].
Fluxo Alternativo	
	FA1 - Clicar em Incluir
	FA1.1 - O ator preenche todos os dados de um animal.
	FA1.2 - O sistema valida os dados do animal[E01] e após isso grava os dados.
	FA1.3 - O sistema retorna ao [FP01].
	FA2 - Clicar em Visualizar
	FA2.1 - O ator visualiza as informações detalhadas de um animal.
	FA2.2 - O sistema permite que o usuário edite as informações [FA3]. FA2.3 - Se o usuário clicar em "Voltar", o sistema retorna ao [FP01].
	FA3 -Clicar em Editar
	FA3.1 - O ator pode alterar os dados de um animal.
	FA3.2 - O sistema valida os dados do animal[E01] e após isso grava os dados.

	FA3.3 - O sistema retorna ao [FP01].
	FA4 - Clicar em Excluir
	FA4.1 - O ator confirma se deseja excluir o animal[E02].
	FA4.2 - O sistema valida os dados do animal[E02] e após isso exclui os dados.
	FA4.3 - O sistema retorna ao [FP01].
Fluxo de Exceções	
	E01 - Dados inválidos
	E01.1 - O sistema informa a inconsistência dos dados.
	E01.2 - O sistema retorna para o fluxo correspondente.
	E02 - Cancelamento de Exclusão
	E02.1 - O ator clica na opção "Não".
	E02.2 - O sistema verifica que o animal não pode ser excluído e exibe uma mensagem.
	E02.3 - O sistema retorna para o fluxo correspondente.
Regtras de Negócio	RN10
Caso de Uso ou Cenário Incluído	N/A
Ponto de Extensão	Aplicação web.
Observação	Ao tentar excluir um animal, devem ser feitas consistências para evitar que o mesmo seja excluído se possuir históricos. Nesses casos, deve-se solicitar ao ator se o mesmo deseja manter os dados apenas deixar o animal como inativo ou se realmente deseja excluir todos os dados.

No Quadro 6, é apresentado o descritivo de caso de uso da funcionalidade Gerenciar Alertas, que é a principal funcionalidade da aplicação móvel, quando conectada ao bastão de leitura RFID.

Quadro 6. Descritivo de caso de uso: Gerenciar Alerta

Identificação	UC14
Caso de uso	Gerenciar Alerta
Descrição	Descrição do processo de gerenciamento de alertas, com Cadastro, Visualização, Edição e Exclusão
Atores Principais	Funcionário
Atores Secundários	Administrador

Pré-condição	O usuário deve estar logado no sistema.
Pós-condição	Alerta cadastrado, alterado, excluído ou consultado com sucesso.
Fluxo Principal	
	FP01 - O sistema exibe uma tabela contendo os alertas cadastrados.
	FP02 - O ator pode clicar no botão "Incluir"para adicionar um novo alerta[FA1] ou no botão "Visualizar", para ver detalhes de um alerta[FA2] ou no botão "Editar", para realizar alterações no cadastro do alerta[FA3] ou clicar no botão "Excluir", para excluir um alerta[FA4]
Fluxo Alternativo	
	FA1 - Clicar em Incluir
	FA1.1 - O ator preenche todos os dados de um alerta.
	FA1.2 - O sistema valida os dados do alerta [E01] e após isso grava os dados.
	FA1.3 - O sistema retorna ao [FP01].
	FA2 - Clicar em Visualizar
	FA2.1 - O ator visualiza as informações detalhadas de um alerta.
	FA2.2 - O sistema permite que o usuário edite as informações [FA3]. FA2.3 - Se o usuário clicar em "Voltar", o sistema retorna ao [FP01].
	FA3 -Clicar em Editar
	FA3.1 - O ator pode alterar os dados de um alerta.
	FA3.2 - O sistema valida os dados do alerta[E01] e após isso grava os dados.
	FA3.3 - O sistema retorna ao [FP01].
	FA4 - Clicar em Excluir
	FA4.1 - O ator confirma se deseja excluir o alerta[E02].
	FA4.2 - O sistema valida os dados do alerta[E02] e após isso exclui os dados.
	FA4.3 - O sistema retorna ao [FP01]
Fluxo de Exceções	
	E01 - Dados inválidos
	E01.1 - O sistema informa a inconsistência dos dados.

	E01.2 - O sistema retorna para o fluxo correspondente.
	E02 - Cancelamento de Exclusão
	E02.1 - O ator clica na opção "Não".
	E02.2 - O sistema verifica que o alerta não pode ser excluído e exibe uma mensagem.
	E02.3 - O sistema retorna para o fluxo correspondente.
Regtras de Negócio	RN05
Caso de Uso ou Cenário Incluído	N/A
Ponto de Extensão	Aplicação web e Aplicação móvel
Observação	Ao incluir um alerta, o ator pode sincronizar os dados com o bastão de leitura rfid. Caso o bastão não esteja conectado, é necessário solicitar que o usuário conecte o bastão e tente novamente. Caso o *bluetooth* esteja desligado, o aplicativo deve exibir a opção de ligar e efetuar a conexão

Apêndice D. Construindo por funcionalidade

Essa atividade é executada somente uma vez para cada funcionalidade, com o objetivo de produzir um Pacote de Arquitetura por funcionalidade. Foram desenvolvidos o Diagrama de Atividade, o Diagrama de Classe e o Diagrama de Entidade Relacional [Retamal 2008].

A Figura 12, representa o diagrama de atividade referente ao caso de uso de gerenciar animais, nela estão representados os processos básicos de acesso ao sistema feito por um administrador e a inserção ou edição de um novo animal.

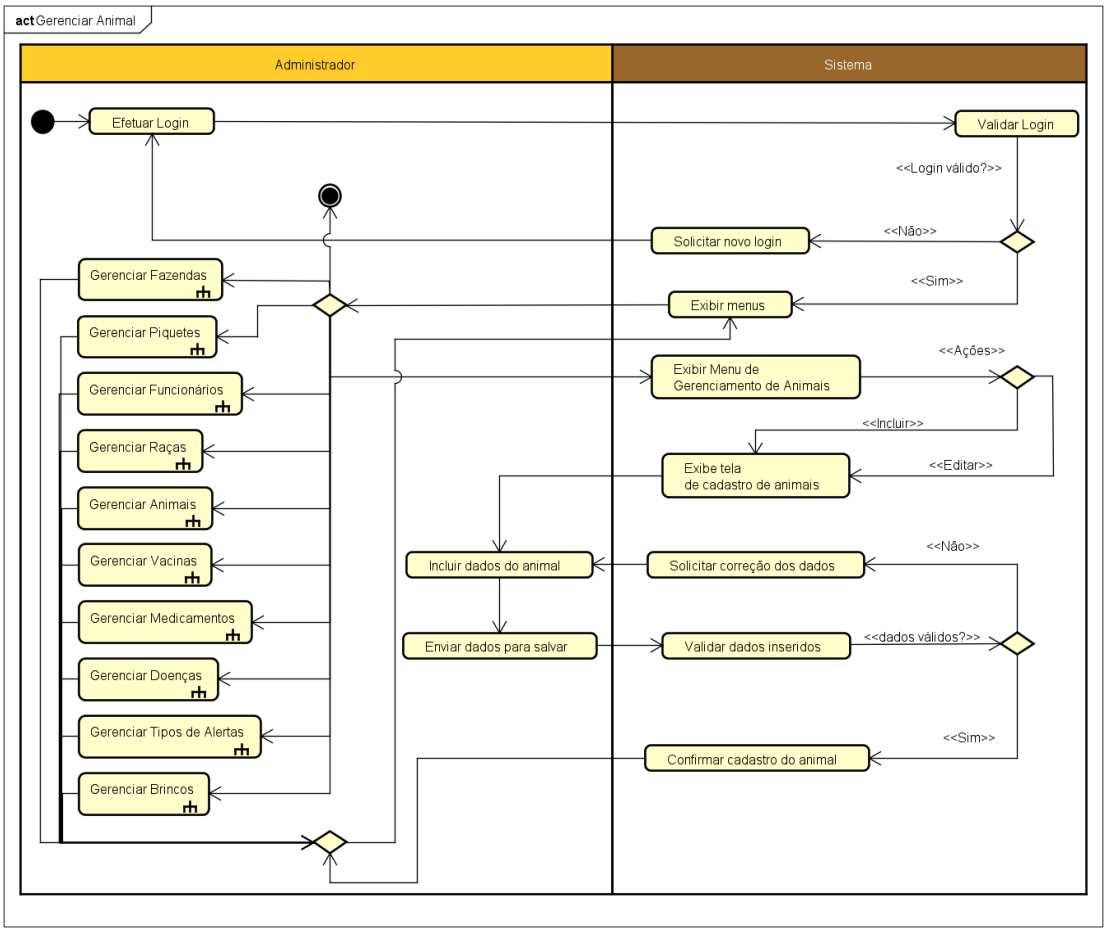

Figura 12. Diagrama da Atividade Gerenciar Animal.

Após um administrador do sistema efetuar o login, caso o mesmo seja válido, é exibido uma tela com ações que são possíveis de realizar no sistema. Ao selecionar a opção "Gerenciar Animais", é exibida uma tela com todos os animais cadastrados no sistemas, além das opções de incluir um novo animal ou editar um existente. Ao clicar em incluir, é apresentada a tela de cadastro, onde o administrador deve inserir todos os dados obrigatórios, e após isso, os dados são salvos.

A Figura 13 representa as classes que serão implementadas na API REST, aplicação web e aplicação móvel, representando os objetos e suas características, sendo que algumas classes são exclusivas da aplicação móvel.

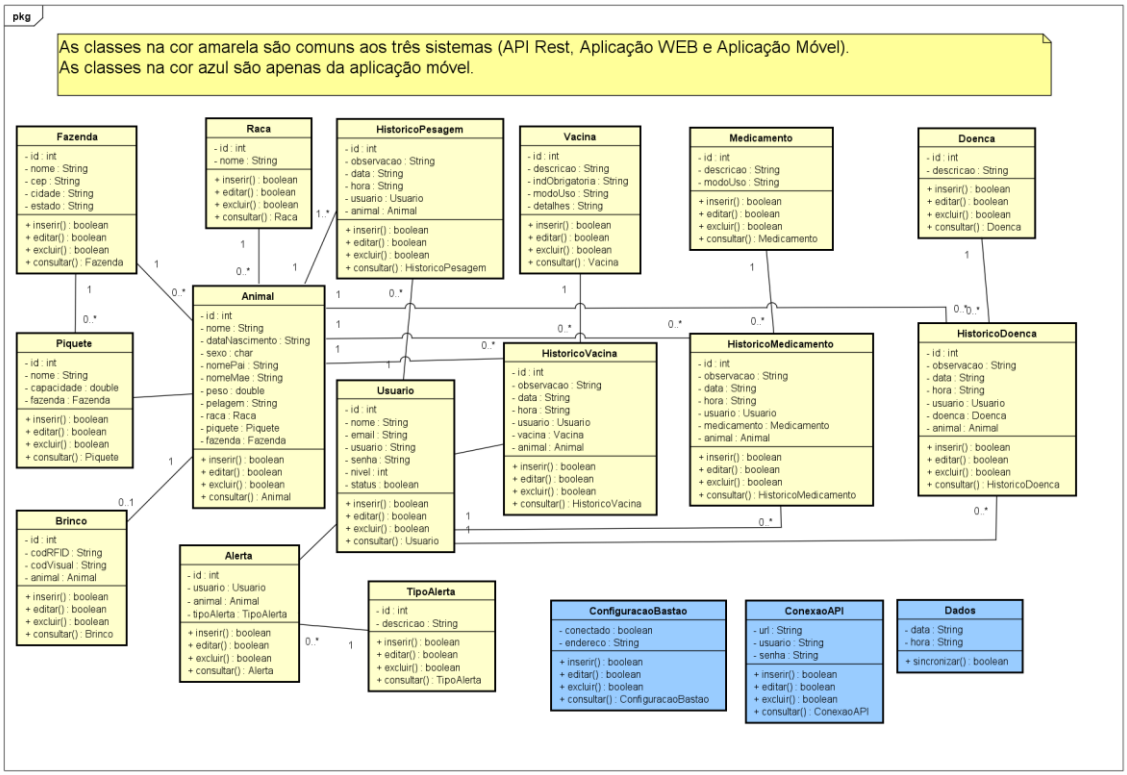

Figura 13. Diagrama de classes da aplicação proposta.

O Diagrama de Entidade Relacionamento (DER) representa a estrutura do banco de dados, apresentando quais as entidades existentes e como se relacionam, especificando o *software*. A Figura 14 representa o DER do banco de dados da API REST, no qual são registrados os dados da aplicação web e do aplicativo móvel.

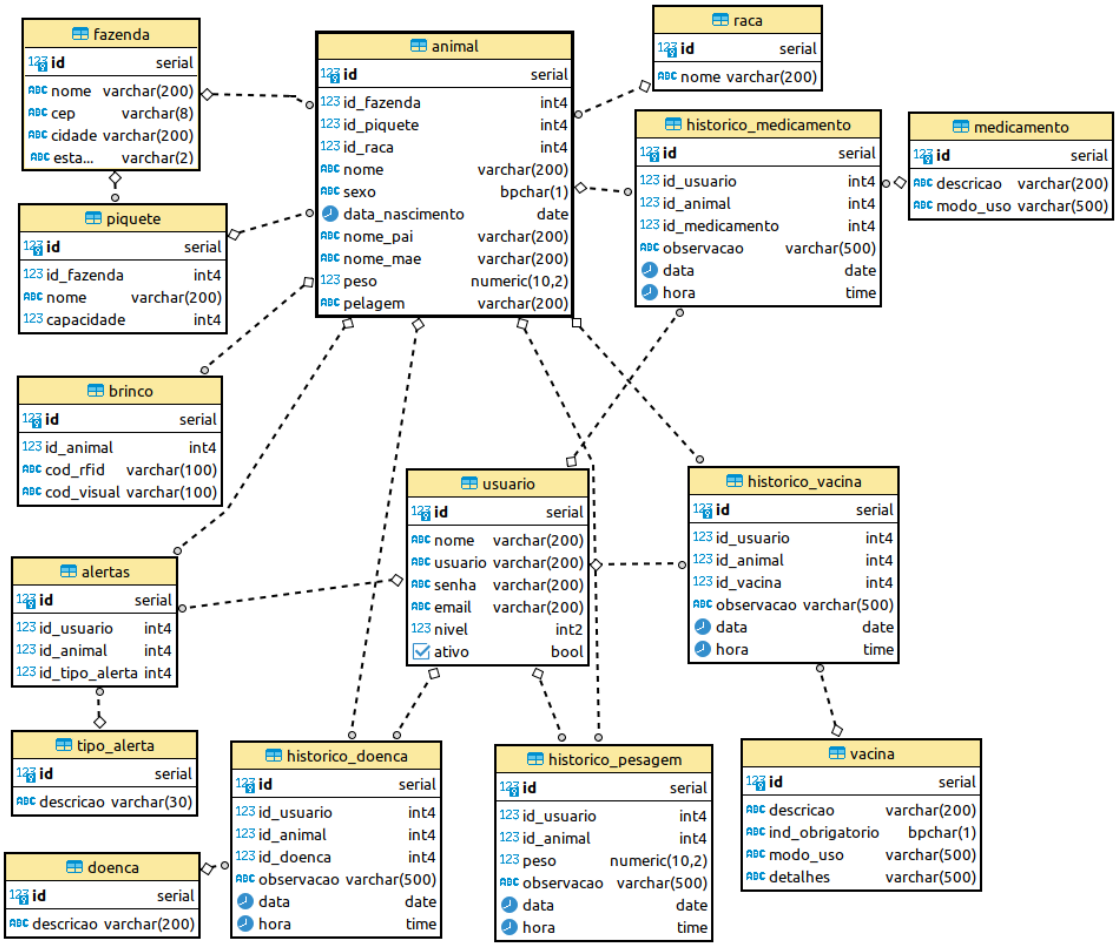

Figura 14. Diagrama de Entidade Relacionamento da API REST.

Apêndice E. Protótipos de interfaces

A Figura 15 apresenta a interface inicial da aplicação web, que contém todos os atalhos do sistema. A ideia é prover uma navegação simples e amigável, onde o usuário encontre tudo de forma fácil e rápida.

Figura 15. Protótipo da interface inicial da aplicação web.

Já a Figura 16 apresenta a interface de gerenciamento de animais, exibindo uma lista com os animais cadastrados no sistema e com as opções de adicionar, editar ou excluir um animal.

Figura 16. Protótipo da interface de gerenciamento de animais da aplicação web.

A Figura 17 apresenta a interface de cadastro de um novo animal, onde os campos com asterisco são obrigatórios.

Figura 17. Protótipo da interface de cadastro de animal da aplicação web.

A Figura 18 apresenta as interfaces do aplicativo móvel, sendo elas a tela inicial, o registro de alertas e registro de peso. Ao aproximar o leitor de um brinco, é possível adicionar um novo alerta ou realizar o registro de peso do animal.

Figura 18. Protótipos de interface da aplicação móvel.