UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

William Felipe de Almeida Borges

UMA PROPOSTA DE CLASSIFICAÇÃO DE ANÚNCIOS PARA UM SISTEMA DE COMERCIALIZAÇÃO DE BOVINOS

Santa Maria, RS
2022

William Felipe de Almeida Borges

UMA PROPOSTA DE CLASSIFICAÇÃO DE ANÚNCIOS PARA UM SISTEMA DE COMERCIALIZAÇÃO DE BOVINOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADOR: Prof. Giovani Rubert Librelotto

517
Santa Maria, RS
2022

End. Eletr.: wfborges@inf.ufsm.br

William Felipe de Almeida Borges

UMA PROPOSTA DE CLASSIFICAÇÃO DE ANÚNCIOS PARA UM SISTEMA DE COMERCIALIZAÇÃO DE BOVINOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 19 de agosto de 2022:

Giovani Rubert Librelotto, Dr. (UFSM)
(Presidente/Orientador)

Rafael Teodósio Pereira, Dr. (UFSM)

Jonas Bulegon Gassen, Dr. (UFSM)

Santa Maria, RS
2022

RESUMO

UMA PROPOSTA DE CLASSIFICAÇÃO DE ANÚNCIOS PARA UM SISTEMA DE COMERCIALIZAÇÃO DE BOVINOS

AUTOR: William Felipe de Almeida Borges
ORIENTADOR: Giovani Rubert Librelotto

No ano de 2020 com o inicio da pandemia diversos setores tiveram que adaptar a forma como efetuavam a comercialização de seus produtos para que seus negócios não viessem a fechar. A solução encontrada foi a digitalização das empresas por meio do *e-commerce*. Porém ainda existem diversos setores que estão em expansão tecnológica e ainda possuem uma carência por plataformas para a comercialização de seus produtos. Depois de analisada a região de Santa Maria em busca de um setor para implantação de um *e-commerce* utilizando o *marketplace* como modelo de negócio, foi observado a grande dificuldade que os pecuaristas enfrentam na hora de anunciar seu rebanho para a venda. Percebendo a grande importância deste setor para a economia da região, e buscando melhorar a experiência do pecuarista na hora de comprar e vender gado foi idealizado a plataforma descrita neste trabalho. O intuito da plataforma é de corrigir algumas inconsistências do processo de negociação, possibilitando a centralização do usuários através de um sistema que atue exclusivamente no setor e que permita os usuários anunciarem e buscarem por anúncios de forma fácil e prática.

Palavras-chave: *Marketplace*, *E-commerce*, Bovinos, Plataforma

ABSTRACT

A ADVERTISEMENT CLASSIFICATION PROPOSAL FOR A CATTLE COMMERCIALIZATION SYSTEM

AUTHOR: William Felipe de Almeida Borges
ADVISOR: Giovani Rubert Librelotto

In the year 2020, with the beginning of the pandemic, several sectors had to adapt the way they marketed their products so that their businesses would not close. The solution found was the digitization of companies through e-commerce. However, there are still several sectors that are in technological expansion and still have a lack of platforms for the commercialization of their products. After analyzing the region of Santa Maria in search of a sector to implement an e-commerce using the marketplace as a business model, it was observed the great difficulty that ranchers face when advertising their herd for sale. Realizing the great importance of this sector for the region's economy, and seeking to improve the cattle rancher's experience when buying and selling cattle, the platform described in this work. The purpose of the platform is to correct some inconsistencies in the negotiation process, enabling the centralization of users through a system that operates exclusively in the sector and that allows users to advertise and search for ads in an easy and practical way.

Keywords: Marketplace, E-commerce, Cattle, Platform

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

$B2B$	Business-to-Business
$B2C$	Business-to-Consumer
$C2C$	Consumer-to-Consumer
$IBGE$	Instituto Brasileiro de Geografia e Estatística
$Sebrae$	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
$INPE$	Instituto Nacional de Pesquisas Espaciais
$Embrapa$	Empresa Brasileira de Pesquisa Agropecuária
API	Application Programming Interface
$HTTP$	HyperText Transfer Protocol
$REST$	Representational State Transfer
$JSON$	JavaScript Object Notation
XML	Extensible Markup Language
MVC	Model-View-Controller
$CRUD$	Create, Read, Update, Delete
$MERN$	MongoDB, Express, React, Node
$NoSQL$	Not Only SQL
SQL	Structured Query Language
$BSON$	Binary JSON
MQL	MongoDB Query Language
URL	Uniform Resource Locator
$HTML$	HyperText Markup Language
SPA	Single Page Application
NPM	Node Package Manager
CSS	Cascading Style Sheet
URI	Uniform Resource Identifier

SUMÁRIO

1 INTRODUÇÃO

Durante o ano de 2020 devido a pandemia de *Covid-19* foi observado um crescimento anormal na busca por ferramentas que possibilitassem à execução de tarefas com o mínimo de contato ou interação com outros indivíduos. Dessa forma, um dos setores que mais cresceu durante o período de isolamento, foi o de *e-commerce*.

Em um levantamento feito pela *Mastercard SpeendingPulse*[1] as vendas realizadas por plataformas virtuais tiveram um aumento de 75% no mês de maio. Comparados ao mesmo período do ano anterior (2019), a média dos meses de março, abril e maio foi de mais de 48% (ALVES, 2020).

O setor de comércio virtual possui diversas subdivisões, onde cada modelo explora um nicho específico. No Brasil, atualmente são utilizados 3 principais modelos: B2B (*Business-to-Business*), B2C(*Business-to-Consumer*) e C2C (*Consumer-to-Consumer*) ou *marketplace*.

Em decorrência da pandemia houve um importante crescimento do modelo de negócio de *marketplace* dentro do setor de comércio virtual. Devido a este aumento, o faturamento anual passou de 35% para 51% do valor total arrecadado pelo setor (PINHO, 2021).

Um dos principais fatores que fazem com que o modelo de negócios do tipo C2C ou *marketplace* continue aumentando, é o baixo custo para se começar a vender e ter lucros. Muitas vezes as dificuldades para o comerciante em criar um *e-commerce* próprio acaba atrasando o processo de inserção no comércio virtual. Diante disso, a facilidade em ter acesso a uma ferramenta com diversos recursos integrados (ex: Sistemas de pagamento, sistemas de *marketing*, sistemas antifraude, etc) e o baixo custo para manter os anúncios, torna-se um atrativo para novos usuários utilizarem essas plataformas (CHAUSSARD, 2019).

Baseado nos dados citados acima, foi feito uma observação na localidade de Santa Maria e também em cidades próximas, com intuito de buscar um nicho pouco explorado e que possibilitasse a fácil inclusão de uma plataforma de *e-commerce* utilizando o modelo de negócios C2C/*marketplace*, com a finalidade de aproveitar o momento de expansão dessa tecnologia para auxiliar os comerciantes da região.

Após um período de analises e pesquisas, pode-se perceber um grande público que obtém a sua renda do agronegócio, setor que está passando por um grande processo de absorção de tecnologias nos últimos anos. Observando um pouco dessa área foi detectado uma maior carência por tecnologias na parte da pecuária, onde atualmente boa

[1] Indicador macroeconômico de vendas no varejo em todos os tipos de pagamento em certos mercados globais. Os relatórios baseiam-se na atividade agregada de vendas na rede de pagamentos da Mastercard, combinada com estimativas baseadas em pesquisas para determinadas outras formas de pagamento, como dinheiro e cheque.

parte dos anúncios para negociações envolvendo bovinos é feito utilizando-se de meios como: programas de rádio, classificados de jornal, *Facebook*, *Instagram* e muitas vezes empregando o *marketing* “passa-a-palavra” ou mais conhecido como *marketing* “boca a boca”.

A grande variedade nos meios de comunicação utilizados para anunciar os bovinos acaba por descentralizar os negociantes, na maioria das vezes ocasionando uma grande lacuna de tempo desde a data em que o anúncio é realizado, até a conclusão da venda. Esse intervalo de tempo se sucede por diversas inconsistências durante o processo, como por exemplo: o vendedor precisa divulgar em diversos meios para propagar o anúncio para o maior número possível de compradores da região, enquanto o cliente precisará buscar em diversos locais até conseguir localizar a mercadoria que deseja adquirir.

A partir das falhas analisadas no processo do comércio de bovinos, este trabalho tem como objetivo principal a criação de um *marketplace* para compra e venda de bovinos, que busca aprimorar a experiência do usuário, tornando o processo de negociação mas ágil e centralizado, permitindo uma maior confiabilidade e segurança através da classificação anúncios e usuários.

O *marketplace* desenvolvido permitirá que os usuários criem os anúncios de maneira simples e padronizada por meio de uma interface gráfica de fácil manuseio, precisando apenas de uma conexão com a *internet* e estar cadastrado no sistema. A plataforma também possibilitará a utilização de filtros durante as buscas por anúncios, afim de direcionar o usuário para o produto que ele procura em um menor tempo.

No decorrer deste trabalho serão abordadas as tecnologias e os conceitos utilizados para criação da plataforma, tal qual como todas as funcionalidades desenvolvidas. Também será feita uma conclusão sobre as vantagens e desvantagens da plataforma, bem como uma descrição sobre futuras implementações.

O presente trabalho está estruturado da seguinte forma. O Capítulo 2 irá apresentar uma revisão bibliográfica dos conceitos, ferramentas e problemáticas que serão abordadas no decorrer do trabalho. O Capítulo 3 apresentará algumas das principias plataformas que atuam na comercialização de bovinos. O Capítulo 4 apresentará a estrutura formalizada do sistema de comercialização de bovinos, a proposta de classificação desenvolvida e também será apresentado o passo a passo para a construção da plataforma. No Capítulo 6 será apresentada a conclusão do trabalho, descrevendo a importância do desenvolvimento e destacando as próximas funcionalidades a serem implementadas no sistema.

2 REFERENCIAL TEÓRICO

Este capítulo retratará conteúdos que serão de grande importância no decorrer do desenvolvimento da monografia, sejam eles relacionados as ferramentas, métodos ou conhecimentos sobre o assunto principal. Estes conceitos permitirão um melhor entendimento sobre o trabalho.

2.1 *E-COMMERCE*

Pode-se descrever como *e-commerce*, todo e qualquer modelo de negócio em que o processo de compra e venda é realizado de forma *online*. Neste modelo o usuário consegue anunciar, buscar, comprar e efetuar pagamentos por meio de uma plataforma digital.

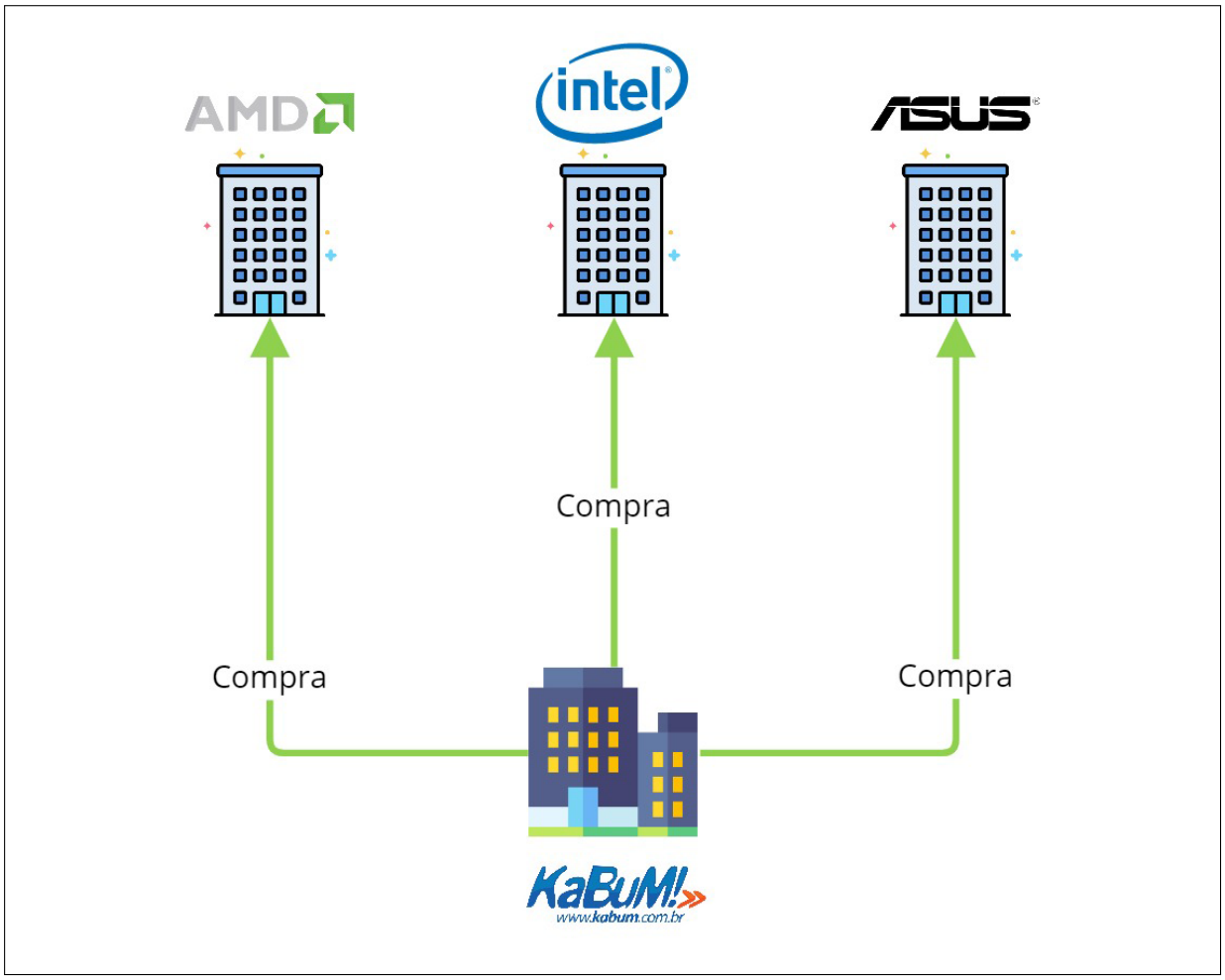
As plataformas de *e-commerce*, assim como os comércios físicos, tem como objetivo permitir a venda de produtos e serviços dos mais variados seguimentos. Porém o grande diferencial é que, por funcionar de forma totalmente *online*, as plataformas de comércio virtual proporcionam maior comodidade e praticidade aos consumidores (PAGAR.ME, 2021b).

O setor de *e-commerce* possui diversos modelos de negócio, cada um deles explorando um nicho especifico, abaixo pode-se visualizar alguns exemplos dos principais utilizados atualmente:

- B2B (Business-to-Business): O *e-commerce* B2B é um modelo no qual empresas negociam diretamente com outras empresas. Geralmente as negociações dentro deste modelo envolvem grandes volumes de mercadorias ou serviços. Essas são algumas das principais plataformas do segmento B2B no país (PIRES, 2021):
  - Magento
  - Vtex
  - Salesforce
  - Infracommerce
  - SAP Hybris
  - Oracle B2B commerce
  - Shopify
  - F1commerce
  - Flexy

Na Figura 2.1, pode-se visualizar o funcionamento básico de um modelo de negócios B2B. Este modelo tem como exemplo a loja de tecnologia *Kabum*, que possui diversos fornecedores de componentes, das mais variadas marcas. Esse processo de compra da *Kabum* direto com outras empresas é o que caracteriza o modelo de negócio.

Figura 2.1 – Exemplo do funcionamento de um modelo de negócio B2B

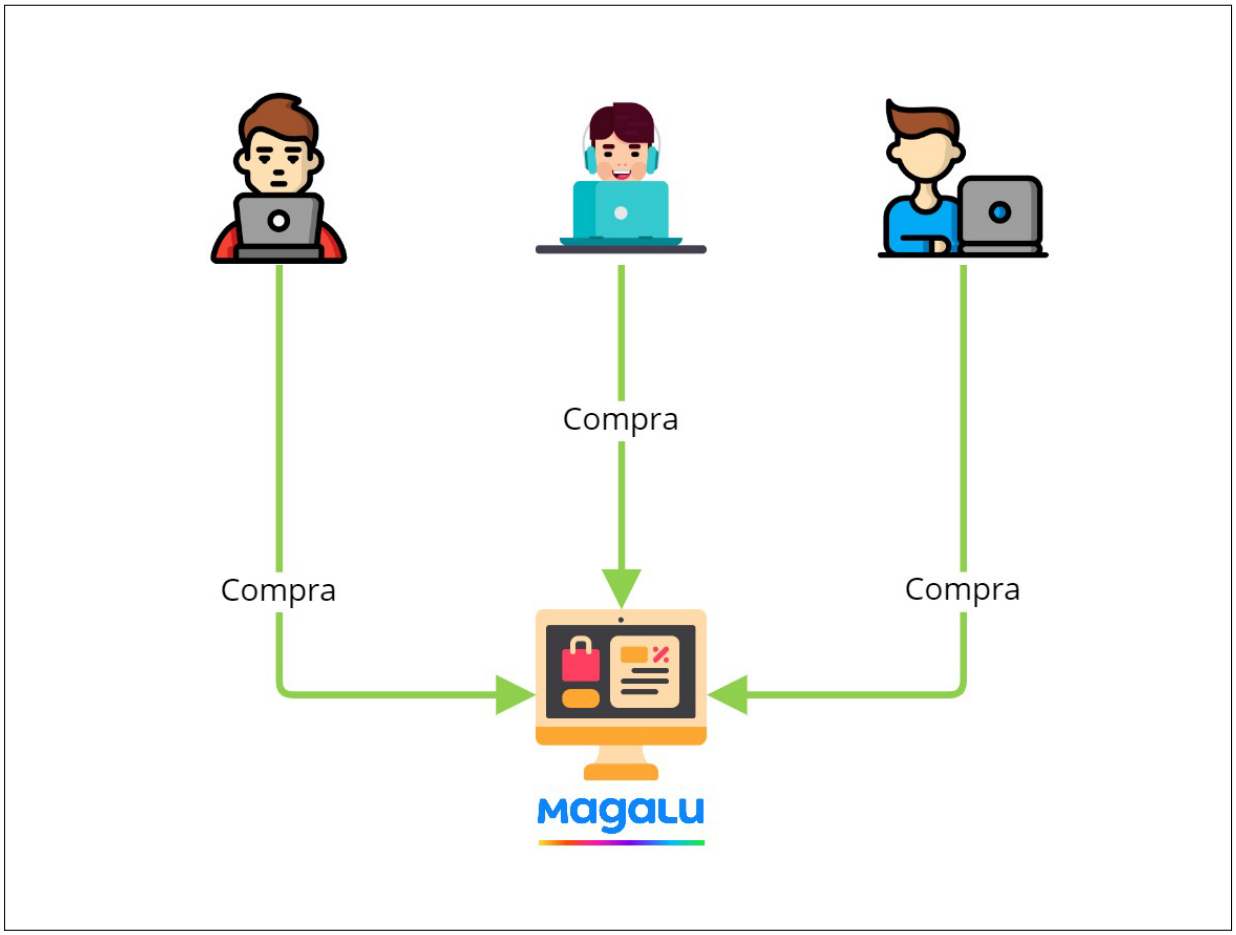


Fonte: Autor do trabalho

- B2C (Business-to-Consumer): No modelo B2C as empresas comercializam produtos e serviços diretamente com o consumidor final. A negociação neste modelo pode envolver mercadorias e serviços da própria empresa, bem como, de fornecedores. A principais plataformas que praticam o B2C atualmente são:
  - Magazine Luiza (Magalu)
  - Grupo B2W (Americas, Submarino, Shoptime, Sou barato)
  - Kabum
  - Dell

A ilustração presente na Figura 2.2 demonstra o modelo de negócio B2C. O exemplo se refere ao funcionamento de uma das maiores plataformas do segmento no Brasil, a *Magazine Luiza* ou *Magalu*. Possuindo uma imensa variedade de produtos a empresa permite que usuários comuns efetuem o processo de compra diretamente em sua plataforma. Essa ligação entre a empresa e o consumidor final é o que configura um modelo do tipo B2C.

Figura 2.2 – Exemplo do funcionamento de um modelo de negócio B2C



Fonte: Autor do trabalho

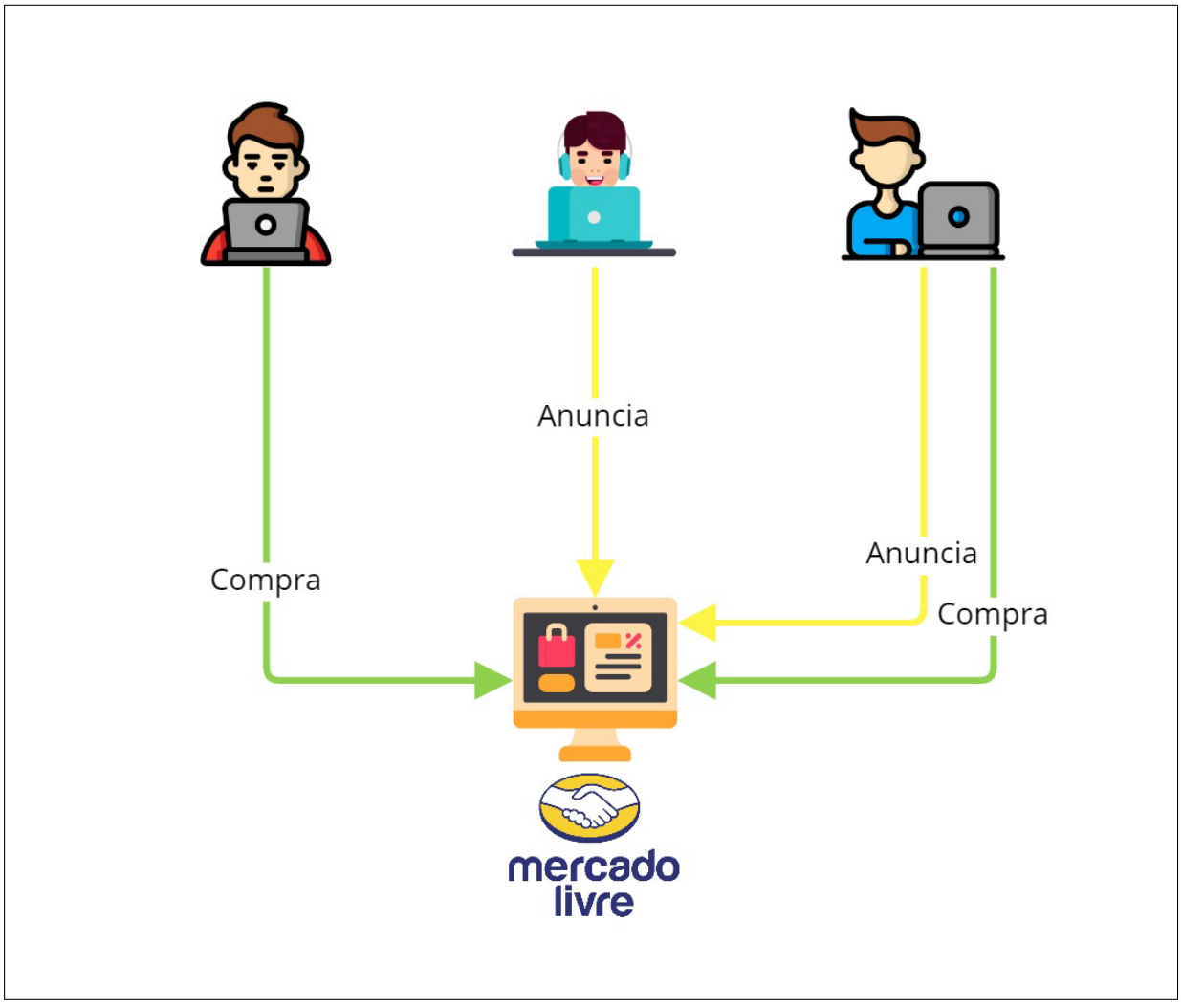
- C2C (Consumer-to-Consumer): esse modelo, mais conhecido como *marketplace*, tem como principal característica a comercialização de produtos e serviços entre consumidores. Este modelo permite que pequenos e médios comerciantes possam efetuar vendas no meio digital, sem a necessidade de uma loja virtual para cada vendedor. As plataformas mais famosas que utilizam este modelo são:
  - Mercado Livre
  - OLX
  - Enjoei

Vale ressaltar que empresas como *Magalu*, *Americas* e *Netshoes* também praticam o modelo de negócio C2C, onde os comerciantes conseguem utilizar os *sites* das empresas como uma vitrine para seus produtos. Isto acaba proporcionando uma maior visibilidade e confiabilidade para os anúncios.

A Figura 2.3 ilustra o perfil de um modelo de negócio do tipo C2C, como exemplo de demonstração utilizou-se a base de funcionamento do *Mercado Livre*. Este modelo de negócio permite que pessoas físicas comercializem diretamente com pessoas físicas, utilizando-se da plataforma apenas como um intermediador. Essa liberdade de permitir o fácil comércio entre usuários comuns é o que define o C2C.

Uma lista feita em 2022 revela que a plataforma do *Mercado Livre* além de ser uma das mais conhecidas do país, acabou de se tornar a companhia mais valiosa da América Latina (LIMA, 2021).

Figura 2.3 – Exemplo do funcionamento de um modelo de negócio C2C



Fonte: Autor do trabalho

2.1.1 Crescimento e consolidação do *e-commerce* no Brasil

No ano de 2020 com início da pandemia, houve um crescimento anormal na busca por ferramentas tecnológicas relacionadas a compra e venda de produtos de forma *online*. Esse crescimento tecnológico sucedeu-se devido as restrições impostas pelo ministério da saúde.

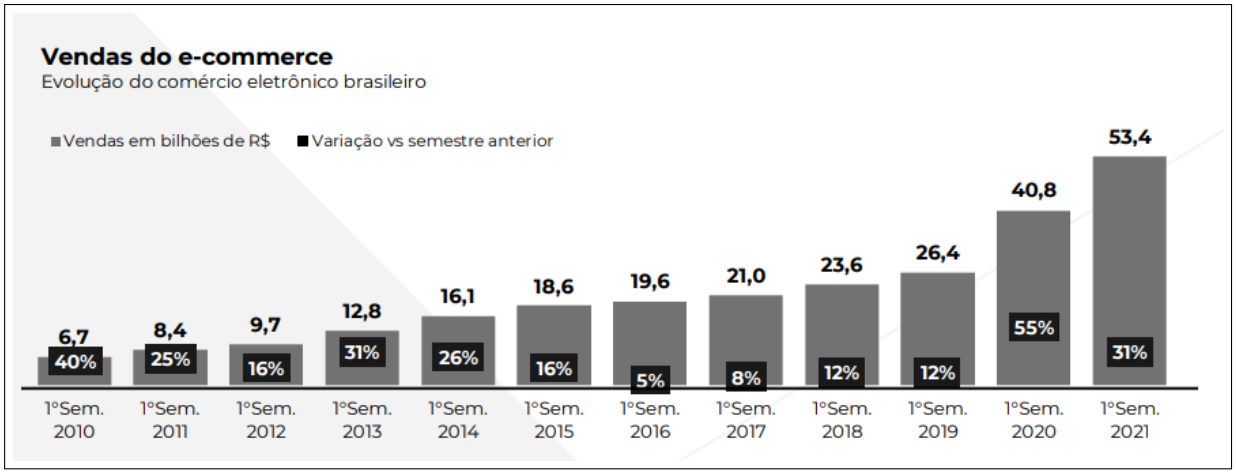
Com o avanço da pandemia, em um levantamento realizado pelo IBGE, 522,7 mil empresas de um total de 1,3 milhão encerraram suas atividades, temporária ou definitivamente, na primeira quinzena de junho. Essa pesquisa também aponta que as empresas que mais sofreram com a pandemia foram as de pequeno porte, que tiveram um total de 518,4 mil (99,2%) empresas que encerraram as atividades (INDIO, 2020).

Sem previsões para o término da pandemia, e com a queda nos lucros, muitas empresas tiveram que buscar soluções para contornar o problema. A principal alternativa foi de transformar seus negócios digitalmente, por meio do *e-commerce*.

Em um relatório referente a 44 ° edição da *Webshopers*[1], como pode-se observar na Figura 2.4, o faturamento das vendas relacionadas ao *e-commerce* manteve um crescimento linear de 2010 até o ano de 2019, porém com a chegada da pandemia o faturamento de 2020 teve a maior varição registrada nos últimos 10 anos (NIELSEN, 2021).

O aumento de 55% no faturamento de 2020 tem como principal determinante o isolamento social, que fez com que a população se adaptasse a situação utilizando-se das plataformas de *e-commerce*, que permitiam comprar produtos e serviços de forma *online*, recebendo-os na porta de casa sem desrespeitar as restrições impostas pelo governo.

Figura 2.4 – Evolução no faturamento do *e-commerce* brasileiro



Fonte: Adaptado de (NIELSEN, 2021)

Analisando o ano de 2021, percebe-se que o *e-commerce* se consolidou no mercado, tornando-se uma das principais fontes de renda do país, pois permitiu durante a pan-

[1] Realizado pela Ebit | Nielsen desde 2001, o Webshoppers é o estudo de maior credibilidade sobre o comércio eletrônico brasileiro e a principal referência para os profissionais do segmento

demia com que pequenos e médios comerciantes, além de evitar o fechamento de suas empresas, pudessem alcançar clientes em todo o território nacional, algo quase impossível sem a digitalização de seus negócios.

2.2 AGRICULTURA DIGITAL NO BRASIL

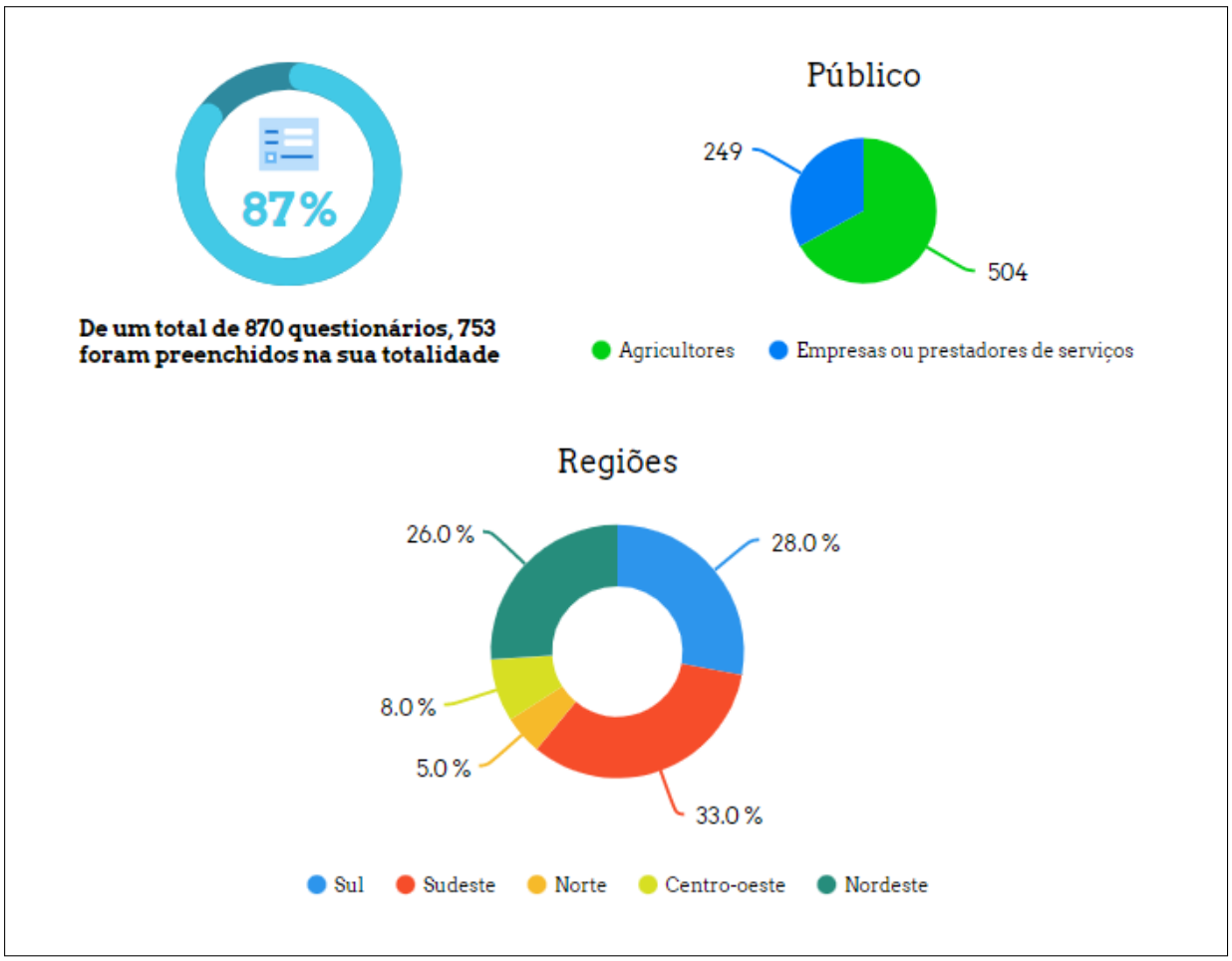
No presente cenário, com a expansão da digitalização, ainda pode-se observar uma grande carência por tecnologia na agricultura do país. Em 2020, as equipes do Sebrae, INPE e Embrapa realizaram um levantamento, com o objetivo de traçar um panorama sobre as tecnologias utilizadas na agricultura, bem como os principais desafios e expectativas do setor.

O questionário utilizado para fazer o levantamento dos dados ficou disponível de forma pública na plataforma *LimeSurvey* entre os dias 17 de abril e 2 de junho (totalizando 45 dias). A pesquisa focou em dois principais grupos: Agricultores (foco em pequenos e médios produtores rurais) e empresários e prestadores de serviços (foco em atuação ou interessados em agricultura digital - empresas, *startups*, técnicos e extensionistas) (BOLFE et al., 2020).

A Figura 2.5 apresenta alguns dados referentes a aplicação da pesquisa, a consulta obteve um total de 870 questionários respondidos, destes 753 foram preenchidos em sua totalidade, os quais foram utilizados para análise dos dados. Dos questionários analisados, 504 foram respondidos por agricultores, enquanto 249 foram respondidos por empresas e prestadoras de serviços em agricultura digital.

A pesquisa teve abrangência nacional e obteve respostas de 556 municípios diferentes. Rio Grande do Sul (15,8%), Minas Gerais (14,5%), São Paulo (13,3%), Bahia(10%) e o Paraná (8,6%) foram os estados onde obteve-se o maior número de respostas. Pode-se observar na Figura 2.5 que 32,5% dos participantes da pesquisa foram do Sudeste; 28% do Sul; 26,2% do Nordeste; 8% do Centro-Oeste; e 5,3% do norte.

Figura 2.5 – Dados sobre os questionários preenchidos



Fonte: Adaptado de (BOLFE et al., 2020)

Dentre os 504 agricultores que participaram da pesquisa, 74% trabalham com agricultura, enquanto 54% trabalham com pecuária. Também pode-se ressaltar que 72% cultivam em áreas de até 50 hectares (ha), e são caracterizados como produtores de pequeno porte. Dentre os agricultores, 84,1% fazem o uso de pelo menos uma tecnologia digital para auxiliar no dia a dia, enquanto 15,9% não faz o uso de nenhuma tecnologia (BOLFE et al., 2020).

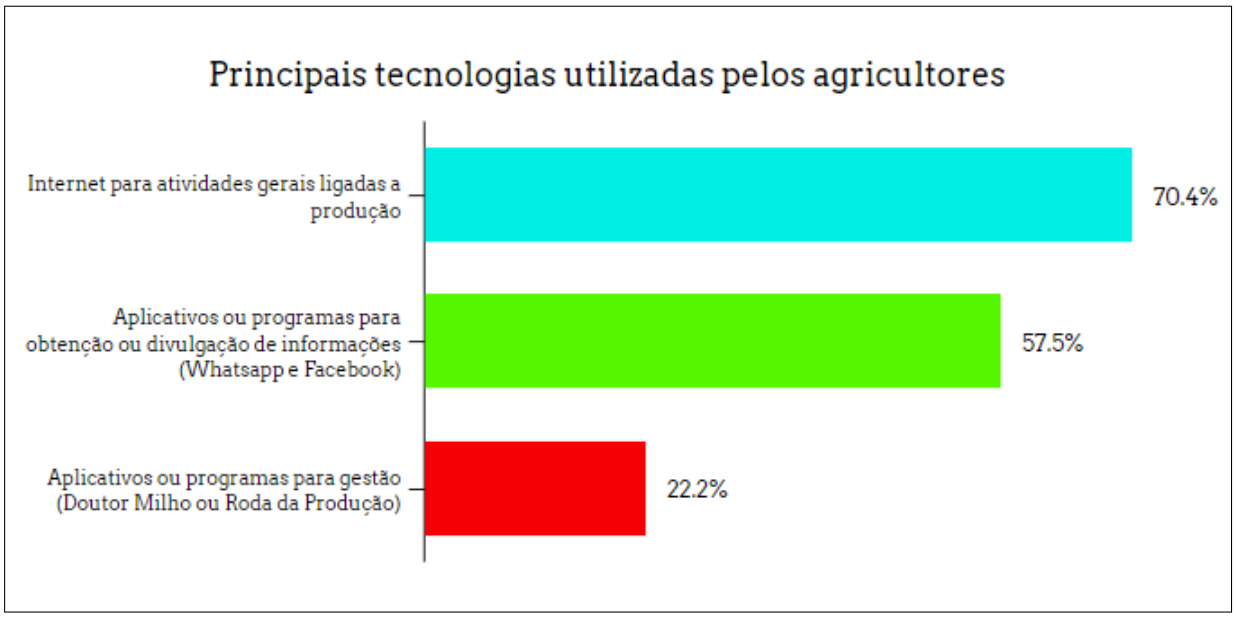
A Figura 2.6 demonstra as 3 principias tecnologias mais utilizadas pelos agricultores analisados na pesquisa. Como é possível visualizar abaixo, um grande grupo de agricultores (70%), utiliza-se da internet como principal ferramenta para efetuar buscas ligadas a produção, como por exemplo: cotação da saca da soja, preços de insumos, entre outros.

Também pode-se destacar o alto percentual referente ao uso de redes sociais (57,5%), como principal plataforma de obtenção ou divulgação de informações sobre a propriedade ou produção, isso revela carência ou inacessibilidade à ferramentas especializadas, acabando por muitas das vezes por não garantir confiabilidade sobre as informações obtidas ou repassadas pelo agricultor.

Também é possível analisar o baixo uso e conhecimento dos entrevistados relacio-

nado a ferramentas de gestão para a propriedade e a produção.

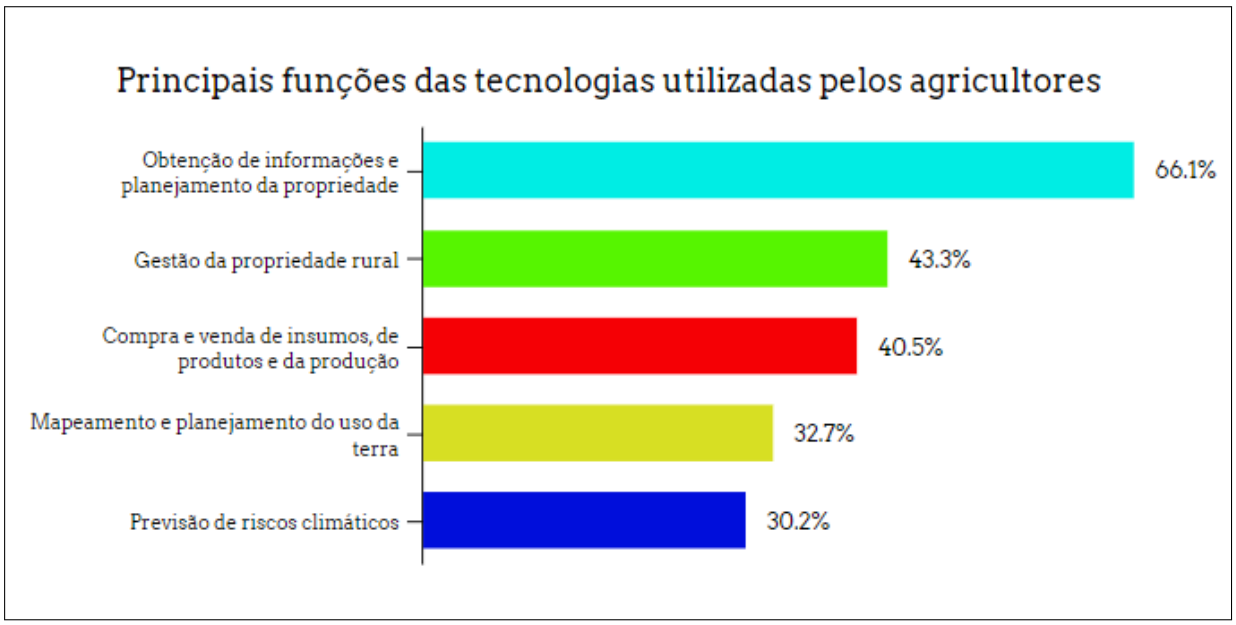
Figura 2.6 – Principais tecnologias utilizadas pelos agricultores



Fonte: Adaptado de (BOLFE et al., 2020)

Outro ponto de grande importância da pesquisa retrata quais as principais funções das tecnologias utilizadas na agricultura digital que conforme apresentados na Figura 2.7 são: a obtenção de informações e planejamento das atividades na propriedade (66,1%); gestão da propriedade rural (43,3%); compra e venda de insumos, de produtos e da produção (40,5%); mapeamento e planejamento do uso da terra (33,7%); e a previsão de riscos climáticos como geada, granizo, veranico e chuvas intensas (30,2%).

Figura 2.7 – Principais funções das tecnologias utilizadas pelos agricultores

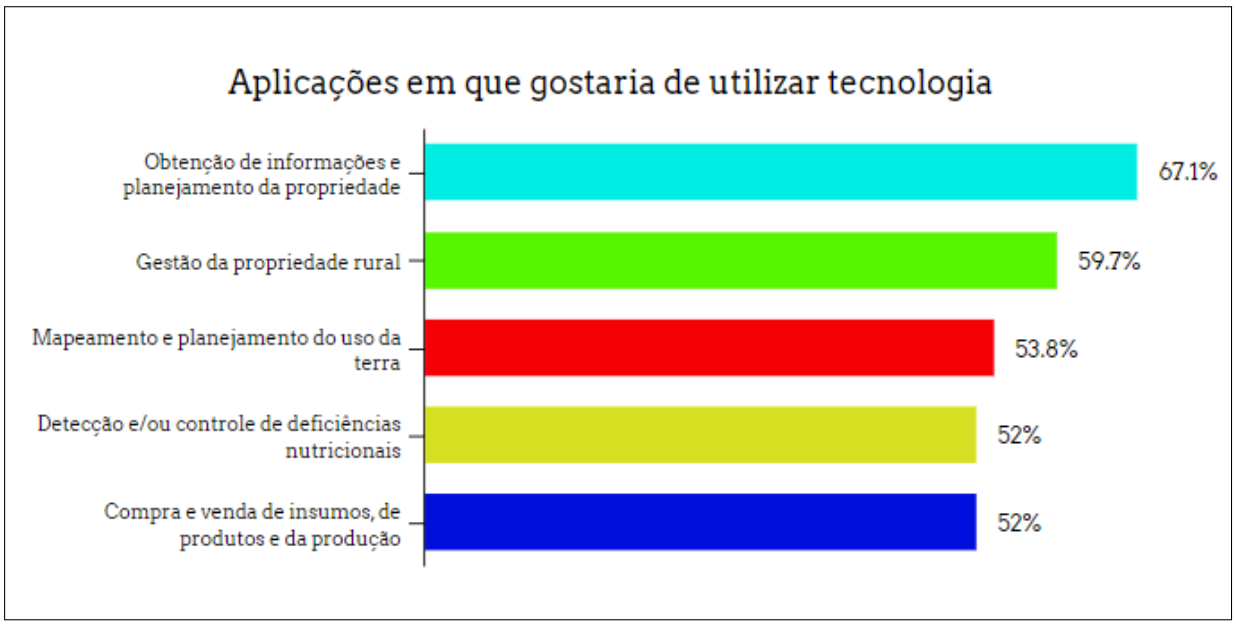


Fonte: Adaptado de (BOLFE et al., 2020)

A baixa adesão as ferramentas digitais muitas vezes sucede-se por diversos problemas. As principais dificuldades apontadas pelos agricultores são: valor do investimento para aquisição de máquinas, equipamentos ou aplicativos (67,1%), problemas ou falta de conexão em áreas rurais (47,8%), valor para contratação de prestadores de serviços especializados (44%) e a falta de conhecimento sobre quais tecnologias mais apropriadas para o uso na propriedade (40,9%) (BOLFE et al., 2020).

Pensando no futuro os entrevistados foram questionados sobre em quais aplicações gostariam de utilizar tecnologias digitais, e as principais respostas foram: obtenção de informações e planejamento da propriedade (67,1%), gestão da propriedade rural (59,7%), mapeamento e planejamento do uso da terra (53,8%), detecção e/ou controle de deficiências nutricionais (52%) e compra e vendas de insumos, produtos e produção (52%) (BOLFE et al., 2020). Podemos analisar visualmente esses dados por meio da Figura 2.8.

Figura 2.8 – Aplicações em que os agricultores gostariam de utilizar tecnologia



Fonte: Adaptado de (BOLFE et al., 2020)

Analisando a pesquisa como um todo, pode-se perceber que ainda há uma carência no setor da agricultura por ferramentas especializadas para cada tarefa ou aplicação. Essa falta de ferramentas dedicadas é observada quando quase todos os agricultores entrevistados utilizam-se das redes sociais e da internet como principais ferramentas para obtenção e disponibilização de informação, bem como para compra de produtos e venda da produção.

Esta falta de tecnologias especificas para cada tarefa só reforça o objetivo deste trabalho de criar um *marketplace*, que permita os pecuaristas efetuarem as negociações por meio de uma plataforma especializada, com intuito de centralizar os anúncios e maximizar as vendas.

2.3 MATERIAIS E MÉTODOS

2.3.1 API REST

Uma API (Interface de Programação de Aplicações) é um conjunto de definições e protocolos usados no desenvolvimento e na integração de aplicações. Às vezes, as APIs são descritas como um contrato entre um provedor e um usuário de informações, estabelecendo o conteúdo exigido pelo consumidor (a requisição) e o conteúdo exigido pelo produtor (a resposta) (REDHAT, 2020).

No ano de 2000, Roy Fielding, um dos principais autores do protocolo HTTP (o protocolo sobre o qual se baseia toda a infraestrutura da Web) apresentou sua tese de doutorado denominada de *"Architectural Styles and the Design of Network-based Software Architecture"*, onde foi introduzido o termo *REST*, que consistia de um estilo arquitetural para sistemas de hipermídias distribuídas. Simplificando, REST (abreviação de REpresentational State Transfer) é um estilo de arquitetura definido para ajudar a criar e organizar sistemas distribuídos (DOGLIO, 2018).

O estilo de arquitetura REST impõe algumas restrições que quando cumpridas permitem que uma a API se torne *RESTful*. Para que um sistema seja considerado *RESTful* ele deve atender aos seguintes critérios e restrições:

- *Client-Server Architecture*: O princípio por trás dessa restrição é a separação de interesses. Ele permite a separação do código *front-end* (representação e possível processamento das informações relacionado à interface do usuário) do código do lado do *back-end*, que deve cuidar do armazenamento e do processamento dos dados no lado do servidor. Essa restrição permite a evolução independente de ambos os componentes, oferecendo muita flexibilidade ao permitir que as aplicações cliente melhorem sem afetar o código do servidor e vice-versa (DOGLIO, 2018).

  As conexões entre o cliente e o servidor são feitas através de métodos HTTP (GET, POST, PUT, DELETE, PATCH) e as trocas de informações são executas por meio de uma representação de dados como JSON (mais utilizado atualmente), XML, textos, imagens, entre outros.

- *Stateless*: A comunicação entre cliente e servidor deve ser *stateless*, ou seja, cada solicitação feita do cliente deve conter todas as informações necessárias para que o servidor a entenda, sem tirar proveito de nenhum dado armazenado. Essa restrição representa várias melhorias para a arquitetura, como visibilidade, confiabilidade, escalabilidade e fácil implementação. Por um lado, os benefícios são obtidos pelo sistema, mas por outro lado, o tráfego de rede pode ser potencialmente prejudicado pela adição de uma pequena sobrecarga em cada solicitação de envio de informações de estado repetidas. Dependendo do tipo de sistema que está sendo implementado e da quantidade de informações repetidas, isso pode não ser uma compensação aceitável (DOGLIO, 2018).

- *Cacheable*: esta restrição propõe que cada resposta a uma solicitação deve ser definida explicitamente ou implicitamente como armazenável em *cache* (quando aplicável). Ao armazenar em *cache* as respostas, existem alguns benefícios óbvios que são adicionados à arquitetura: no lado do servidor, algumas interações (como uma solicitação de banco de dados, por exemplo) são completamente ignoradas enquanto o conteúdo é armazenado em *cache*. Do lado do cliente, percebe-se uma aparente

melhora de desempenho. A desvantagem desta restrição é a possibilidade dos dados armazenados em *cache* ficarem obsoletos. Essa restrição é opcional e pode ser aplicada de acordo com o sistema implementado (DOGLIO, 2018).

- *Uniform Interface*: ao manter uma interface uniforme entre os componentes, você simplifica o trabalho do cliente quando se trata de interagir com seu sistema. Outra grande vantagem é que a implementação do cliente é independente do servidor, portanto, ao definir uma interface padrão e uniforme para todos os serviços, acaba simplificando efetivamente a implementação de clientes independentes, fornecendo a eles um conjunto claro de regras a serem seguidas, como por exemplo:
  - os recursos solicitados sejam identificáveis e estejam separados das representações enviadas ao cliente;
  - os recursos possam ser manipulados pelo cliente por meio da representação recebida com informações suficientes para tais ações;
  - as mensagens auto-descritivas retornadas ao cliente contenham informações suficientes para descrever como processá-las;
  - hipertexto e hipermídia estão disponíveis. Isso significa que após acessar um recurso, o cliente pode usar *hiperlinks* para encontrar as demais ações disponíveis para ele no momento;

  Uma das desvantagens é que ao ter uma interface padronizada e uniforme para todas as interações com seu sistema pode prejudicar o desempenho quando existe uma forma de comunicação mais otimizada. Particularmente, o estilo REST foi projetado para ser otimizado para a *web*, portanto, quanto mais você se afasta disso, mais ineficiente a interface pode ser (DOGLIO, 2018).

- *Layered System*: ao separar os componentes em camadas e permitir que cada camada use apenas a de abaixo e comunique sua saída à acima, você simplifica a complexidade geral do sistema e mantém o acoplamento de componentes sob controle. Este é um grande benefício em todos os tipos de sistemas, especialmente quando a complexidade de tal sistema está sempre crescendo (por exemplo, sistemas com grandes quantidades de clientes, sistemas que estão evoluindo, etc.). A principal desvantagem dessa restrição é que, para sistemas pequenos, ela pode adicionar latência indesejada ao fluxo geral de dados, devido as diferentes interações entre as camadas (DOGLIO, 2018).

- *Code-on-demand*: é a única restrição opcional imposta pelo REST, o que significa que um arquiteto que usa REST pode optar por usar ou não essa restrição e obter suas vantagens ou sofrer suas desvantagens. Com essa restrição, o cliente pode

baixar e executar o código fornecido pelo servidor (como applets Java, scripts JavaScript, etc.). No caso de APIs REST, essa restrição parece desnecessária, porque o normal para um cliente de API é apenas obter informações de um *endpoint* e processá-las da maneira que for necessário; mas para outros usos de REST, como servidores web, um cliente (ou seja, um navegador) provavelmente se beneficiará dessa restrição (DOGLIO, 2018).

2.3.2 MVC

Padrões de projetos são soluções para problemas que alguém um dia teve e resolveu aplicando um modelo que foi documentado e que você pode adaptar integralmente ou de acordo com necessidade de sua solução (MACORATTI, 2004).

Atualmente existem diversos padrões de arquitetura e a principal vantagem do uso desses padrões está no reuso das soluções propostas para determinado problema. Abaixo estão listadas outras vantagens e desvantagens de se utilizar padrões de projetos (MARCIO, 2011):

- Aumento de produtividade;
- Uniformidade na estrutura do *software*;
- Redução de complexidade no código;
- As aplicações ficam mais fáceis de manter;
- Facilita a documentação;
- Estabelece um vocabulário comum de projeto entre desenvolvedores;
- Permite a reutilização de módulos do sistema em outros sistemas;
- É considerada uma boa prática utilizar um conjunto de padrões para resolver problemas maiores que, sozinhos, não conseguiriam;
- Ajuda a construir *softwares* confiáveis com arquiteturas testadas;
- Reduz o tempo de desenvolvimento de um projeto.

O MVC é uma sigla que vem do termo inglês *Model-View-Controller* e é um padrão de arquitetura de *software* responsável por contribuir na otimização entre as requisições feitas pelos comandos dos usuários. Com mais de 50 anos de formulação o MVC é divido em três componentes essenciais: os *models*, os *controllers* e as *views*. Porém podem

ser adicionadas outras camadas conforme as necessidades do projeto, mas um código com muitas camadas se torna muito confuso e por isso, o ideal é manter o padrão original (ZUCHER, 2020).

A utilização do padrão MVC trás como benefício isolar as regras de negócios da lógica de apresentação. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reuso das classes (ZUCHER, 2020).

A seguir será explicado como funciona cada camada do MVC, suas restrições e atribuições.

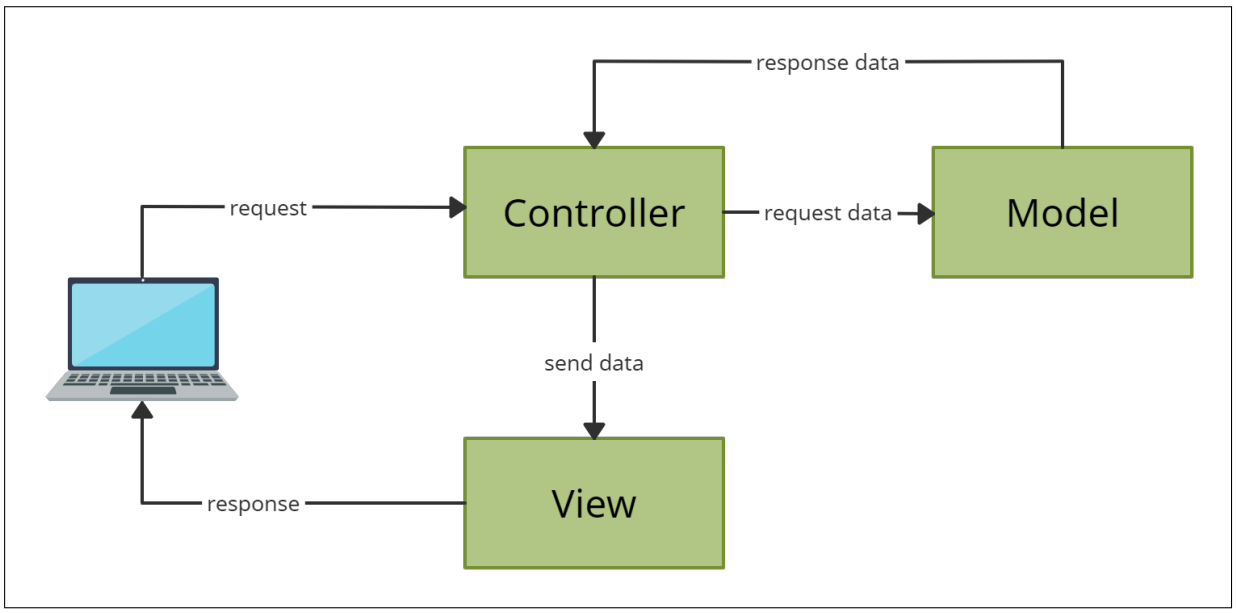
- *Model*: é a camada que possui a lógica da aplicação, sendo responsável pelas regras de negócios, persistências no banco de dados e as entidades do projeto. O *model* recebe as requisições vindas do *controller* e gera respostas a partir destas requisições. Esta camada é conhecida como o núcleo da aplicação e terá conhecimento apenas dos dados armazenados no sistema e da lógica sobre eles, também ficando responsável por realizar as operações de CRUD.

- *View*: é a camada responsável pela apresentação do sistema, que permite que os usuários possam interagir com a aplicação, submetendo dados através de formulários de entrada, cliques em elementos gráficos, entre outras ações. Também por meio das *views* o usuário poderá visualizar as respostas das suas requisições através de tabelas, textos, imagens, etc. A *view* não contém lógica de negócios, portanto todo o processamento é feito pela camada *model* e então a resposta é repassada para a *view* pelo *controller*.

- *Controller*: é a camada responsável por intermediar a conexão entre as *views* e os *models*. As requisições que vem da *view* são organizadas e tratadas pelo *controller*, depois de tratadas as requisições são repassadas para o *model* efetuar a persistência dos dados no banco. Caso as requisições feitas ao *model* possuam uma resposta, está será repassada para a view através do *controller*.

Na Figura 2.9 consegue-se observar uma representação do funcionamento do padrão de projetos MVC.

Figura 2.9 – Representação do funcionamento do padrão de projetos MVC



Fonte: Autor do trabalho

As principais vantagens e desvantagens de se utilizar o padrão de projetos MVC são (MARCIO, 2011):

- Vantagens
  - Separação muito clara entre as camadas de visualização e regras de negócios;
  - Manutenção do sistema se torna mais fácil;
  - Reaproveitamento de código, principalmente da camada de modelo, que pode ser reutilizada em outros projetos;
  - As alterações na camada de visualização não afetam as regras de negócios já implementadas na camada de modelo;
  - Permite o desenvolvimento, testes e manutenção de forma isolada entre as camadas;
  - O projeto passa a ter uma melhor organização em sua arquitetura;
  - Torna o entendimento do projeto mais fácil para novos programadores que não participaram de sua criação.
- Desvantagens
  - Em sistemas de baixa complexidade, o MVC pode criar uma complexidade desnecessária;
  - Exige muita disciplina dos desenvolvedores em relação à separação das camadas;

  - Requer um tempo maior para modelar o sistema;

2.3.3 MERN *stack*

MERN é uma *stack* de desenvolvimento e sua sigla faz referência as quatro principais tecnologias que a compõem: *MongoDB, Express.js, React.js, Node.js*. A utilização desta *stack* torna mais rápido e fácil o desenvolvimento, onde cada uma dessas 4 tecnologias poderosas fornece uma estrutura de ponta a ponta para os desenvolvedores trabalharem e cada uma dessas tecnologias desempenha um papel importante no desenvolvimento de aplicativos da *web* (JASRAJ, 2021). A seguir serão apresentadas cada uma das tecnologias que fazem parte da *stack*.

- *MongoDB*: é um banco de dados NoSQL orientado a documentos, que foi projetado para armazenar dados JSON nativamente (ele tecnicamente usa uma versão binária do JSON chamada BSON). Todo documento adicionado no *MongoDB* é composto por um conjunto de dados que segue o padrão JSON de chave-valor. A principal vantagem de se utilizar *MongoDB* nesta *stack* é que desde a sua linha de comando até a sua linguagem de consulta (MQL - MongoDB Query Language) é construído em JSON e *JavaScript* (MONGODB, 2022).

- *Express.js*: é um *framework JavaScript*, que é executado dentro de um servidor *Node.js*. O Express autodenomina-se um "*framework web* rápido, sem opinião e minimalista para *Node.js*", e isso é exatamente o que é. Através deste *framework* é possível efetuar o roteamento de suas URLs sem muito esforço. A partir de requisições do seu *front-end*, você pode se conectar a funções *Express.js* que alimentam seu aplicativo. Essas funções, por sua vez, usam os *drivers* de conexão do *MongoDB* presentes no *Node.js*, para acessar e atualizar seu documentos em sua base dados (JASRAJ, 2021).

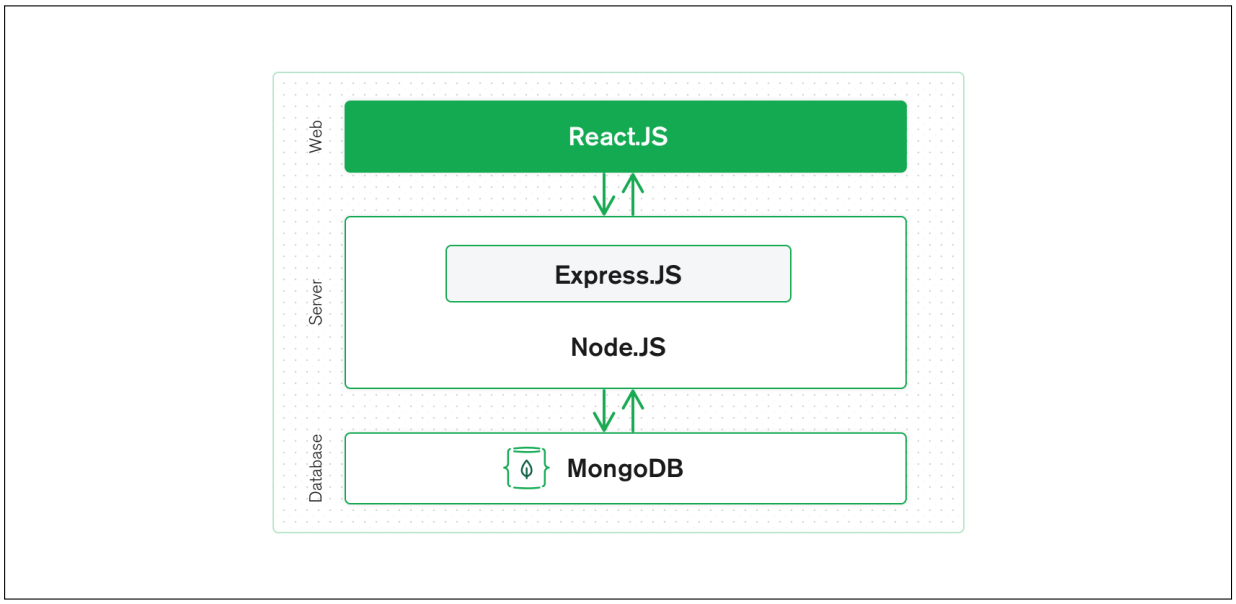
- *React.js*: é uma biblioteca *JavaScript* para a criação de interfaces de usuário complexas por meio de componentes simples, que podem ser conectados aos dados do seu servidor *back-end* e depois são renderizados como HTML. A principal vantagem do *React.js* é a possibilidade da criação de SPAs, que permite que a cada alteração da página, ao invés de recarregar toda a página ou redirecionar o usuário para uma página nova, apenas o conteúdo principal é atualizado de forma assíncrona, mantendo toda a estrutura da página estática (JASRAJ, 2021).

- *Node.js*: é definido como um ambiente de execução *JavaScript* do lado do servidor sem a necessidade de um *browser*. As principais vantagens de se utilizar este ambiente são:

- Flexibilidade: O NPM é o gerenciador de pacotes do *Node.js* e também é o maior repositório de *softwares* do mundo. Isso faz do *Node.js* uma plataforma com potencial para ser utilizada em qualquer situação.
- Leveza: Criar um ambiente *Node.js* e subir uma aplicação é uma tarefa que não exige muitos recursos computacionais em comparação com outras tecnologias mais tradicionais. Se utilizado em conjunto com ferramentas como o *Docker*, o ganho na velocidade de *deploy* e replicação de máquinas pode ser muito significativo e em ambientes escaláveis isso significa menos custo e mais eficiência.
- Alta compatibilidade: Conta com suporte das principais empresas de produtos e serviços de *cloud* do mercado, como a *AWS, Google Cloud* e *Microsoft Azure* que oferecem na maioria de seus produtos suporte nativo ao *Node.JS*.

Na Figura 2.10 podemos visualizar uma representação de como ocorre a divisão das camadas do MERN e como elas se comunicam.

Figura 2.10 – Representação do funcionamento da MERN *stack*



Fonte: MongoDB

Os benefícios de utilizar a MERN *stack* incluem a velocidade e a eficiência do desenvolvimento, a flexibilidade do código, a escalabilidade do aplicativo e a compatibilidade com a maioria dos sistemas operacionais. Também é compatível com a maioria das bibliotecas e *frameworks JavaScript*, o que a torna uma opção ideal para a criação de aplicativos *web* modernos. Além disso, é necessário conhecer apenas uma linguagem de programação (*JavaScript*) e a estrutura do documento JSON para entender todo o sistema.

2.3.4 *Tailwind* CSS

Tailwind CSS é um *framework* que oferece a possibilidade de você criar *layouts* usando uma estrutura de CSS pronta. Isso permite que você otimize o tempo de criação de uma interface sem precisar fazer tudo manualmente. O *Tailwind* CSS verifica todos os seus arquivos referentes as *views* do projetos, capturando os nomes das classes e gerando os estilos correspondentes e, em seguida, gravando-os em um arquivo CSS estático (TAILWINDCSS, 2022).

O *Tailwind* CSS permite um maior controle na estilização das páginas e também consegue trabalhar com diversos *plugins* e ferramentas para facilitar o desenvolvimento, além de possuir uma grande biblioteca com diversos componentes criados por outros desenvolvedores que podem ser adicionados ao projeto conforme a demanda. Quando necessário, o *Tailwind* CSS permite que suas classes sejam customizadas ou que novas classes sejam adicionadas de forma fácil.

3 ANÁLISE DE MERCADO

No cenário atual existem diversas plataformas de *marketplace* que possibilitam a compra e venda de bovinos. Neste capítulo serão apresentadas 3 ferramentas utilizadas por pecuaristas para efetuar a negociação dos bovinos, destacando as vantagens e desvantagens de cada uma delas.

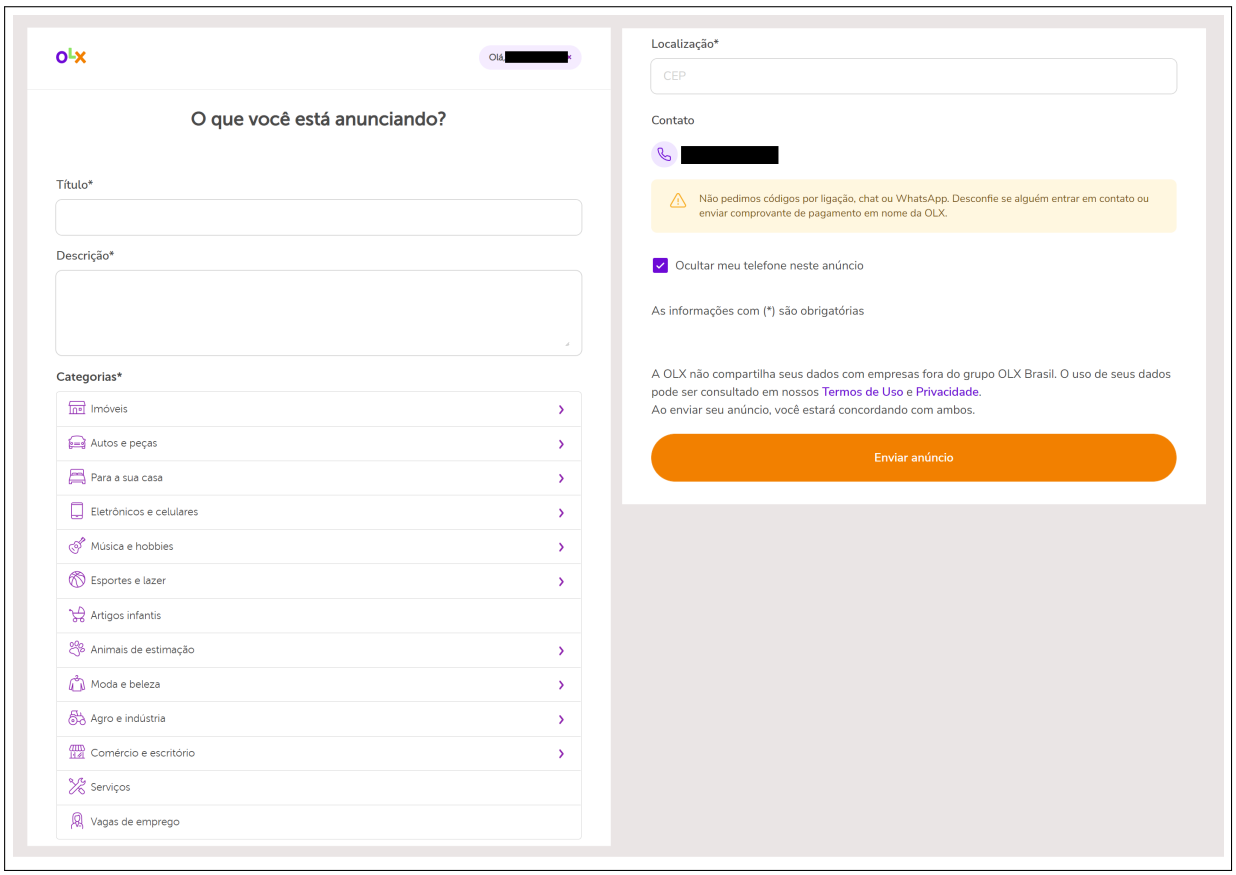
3.1 OLX

A OLX é um dos maiores sites de compra e venda do mundo no segmento de *marketplace*, que conforme descrito pela empresa tem como principal objetivo “conectar pessoas para que elas possam comprar e vender de forma rápida e fácil". A empresa chegou no Brasil em 2010, como resultado da parceria entre o grupo sul-africano Naspers e o norueguês Schibsted (CANALTECH, 2021).

A plataforma tem um funcionamento bem intuitivo, facilitando o cadastro de produtos ou de serviços na rede. Para divulgar um item o vendedor deve se cadastrar no site, incluir o item que deseja vender e esperar que os interessados entrem em contato. No momento de efetuar um anúncio a plataforma disponibiliza uma interface que permite ao vendedor adicionar informações sobre o produto (título, descrição, preço, fotos, localização, categoria, etc) (EUGÊNIO, 2021).

Na Figura 3.1 pode-se visualizar a estrutura do formulário para anunciar na plataforma da OLX.

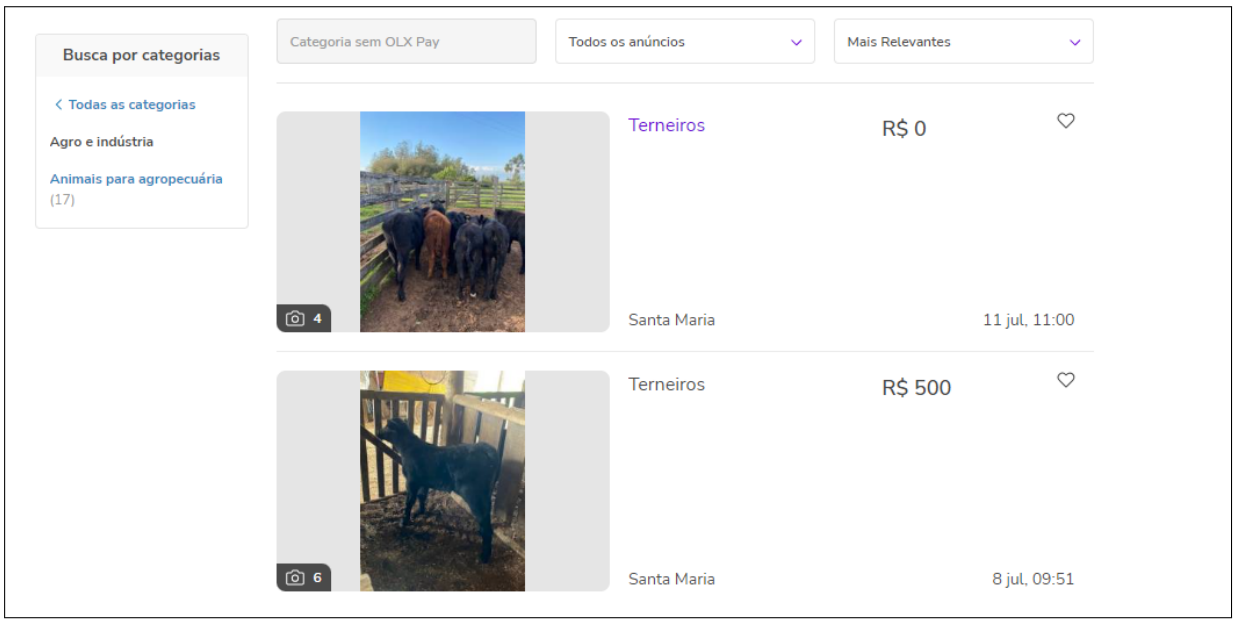
Figura 3.1 – Formulário para anunciar na plataforma da OLX



Fonte: OLX Brasil

Para efetuar a pesquisa por algum produto a plataforma da OLX permite filtrar os anúncios (por categoria, localidade, preço, relevância, etc), agilizando o processo de busca. Os produtos buscados são apresentados em forma de *cards* com algumas informações prévias para uma melhor visualização do usuário. Na Figura 3.2 pode-se observar uma busca com filtros por cabeças de gado na região de Santa Maria.

Figura 3.2 – Interface de busca com filtro da plataforma da OLX



Fonte: OLX Brasil

Desde o final do ano de 2021 a plataforma da OLX implantou uma funcionalidade de pagamento denominada *OLX Pay*, que permite que o comprador efetue a transação da compra por meio da plataforma, possibilitando uma maior segurança para ambos os usuários envolvidos na negociação. O vendedor só irá receber o valor da compra após a confirmação do recebimento por parte do comprador (OLX, 2021).

Depois de uma breve analise da plataforma, pode-se destacar as seguintes vantagens e desvantagens relacionadas a venda de bovinos:

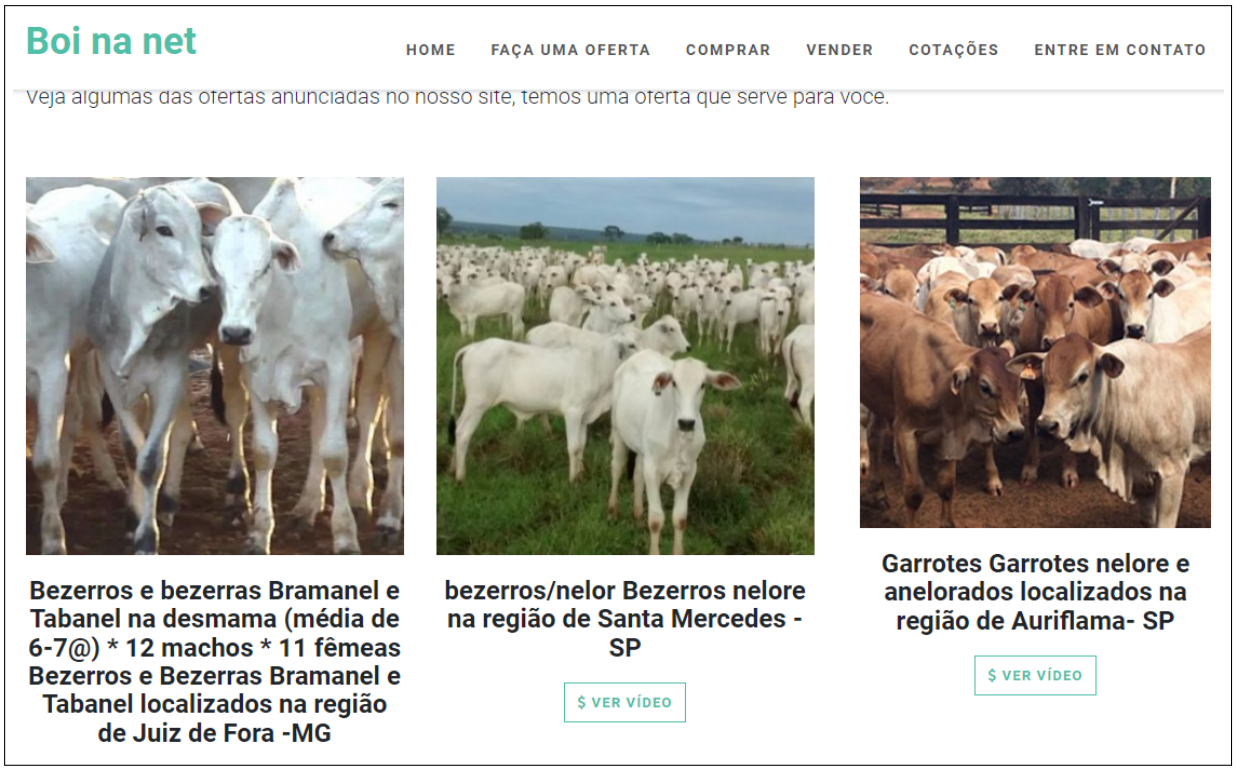
- Vantagens
  - Permite a inserção de fotos e da descrição nos anúncios.
  - Possui um chat interno para interação entre comprador e vendedor.
  - Possibilidade de pagamento por meio da plataforma.
  - Busca com filtros.
- Desvantagens
  - Plataforma generalista, atua em diversos nichos.
  - Formulários para anúncios muito simples, possuem poucos campos de informações.
  - Despadronização dos anúncios, pois a descrição acaba se sobressaindo perante as outras informações.

3.2 BOI NA NET

Boi na net trata-se de uma plataforma que comercializa exclusivamente bovinos. Criada no ano de 2015, permite que os pecuaristas anunciem e comprem gado de forma rápida e fácil.

Na Figura 3.3 pode-se visualizar como funciona a interface de anúncios da plataforma. Os anúncios são exibidos em forma de *cards*, contendo uma breve descrição do produto juntamente com uma foto e um botão para visualização de um video dos animais.

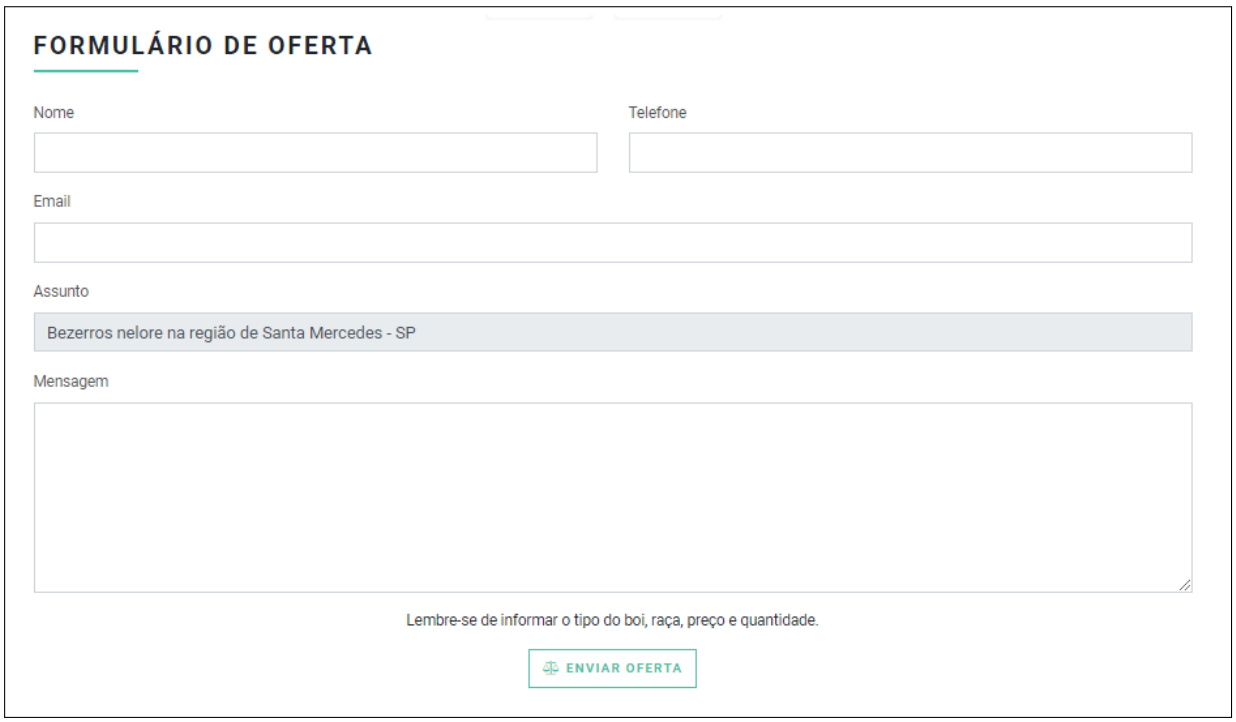
Figura 3.3 – Interface de anúncios da plataforma do Boi na net



Fonte: Boi na net

Toda vez que o usuário desejar anunciar ou efetuar a compra de algum bovino pela plataforma ele deve preencher o formulário apresentado na Figura 3.4, envia-lo e aguardar um retorno por parte dos administradores. O mesmo formulário utilizado para enviar uma oferta é utilizado para enviar um anúncio para a plataforma.

Figura 3.4 – Formulário para enviar oferta na plataforma do Boi na net



Fonte: Boi na net

Como a plataforma não exibe o contato dos vendedores e não permite a interação entre usuários, acaba apenas acontecendo uma intermediação entre os negociantes por parte dos administradores. Depois de interagir com a plataforma conseguiu-se detectar as seguintes vantagens e desvantagens da plataforma:

- Vantagens
  - Permite efetuar a descrição detalhada do anúncio.
  - Grande demanda de clientes.
  - Confiabilidade no mercado de bovinos.
- Desvantagens
  - Anúncios despadronizados.
  - Não permite a interação entre os usuários.
  - Não permite o cadastro de usuários.
  - Não apresenta classificação dos vendedores.
  - Não apresenta dados dos vendedores e compradores.
  - Não permite *feedback* dos compradores nos anúncios.
  - Formulários de cadastro muito vagos, com poucas informações.

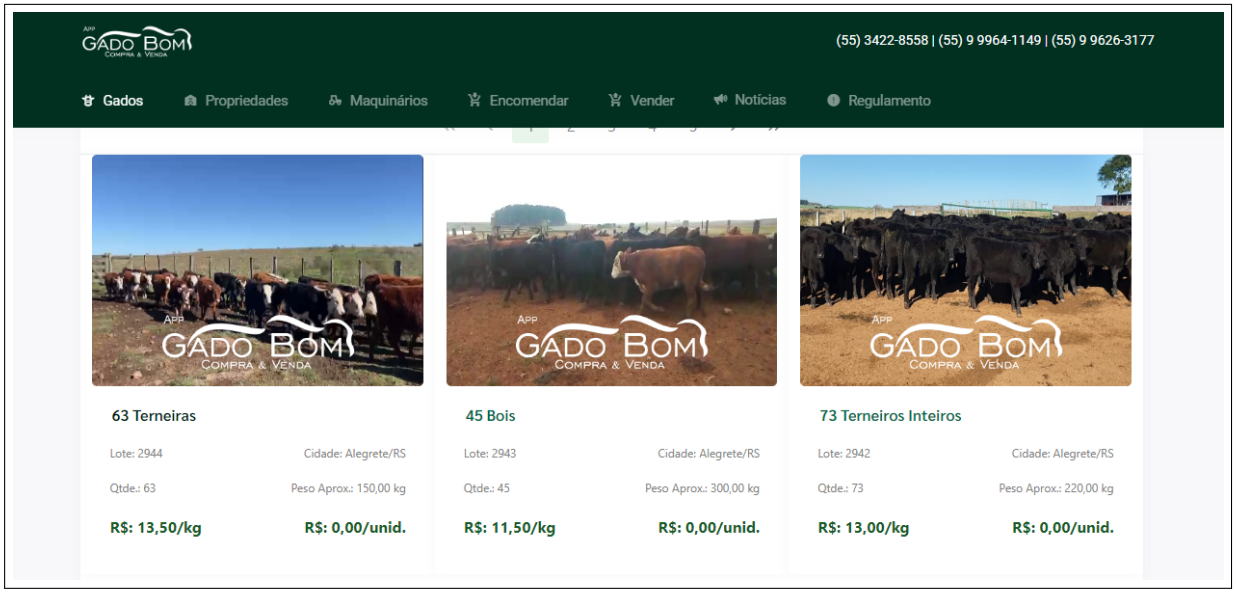
- Não permite pagamento por meio da plataforma.
- Não possibilita a filtragem dos anúncios e nem possui campo de busca.

3.3 APP GADO BOM

App gado bom é uma plataforma para compra e venda de bovinos, propriedades rurais e maquinários, possui uma interface agradável, bem construída e de fácil manuseio. Os anúncios são filtrados de acordo com uma das três categorias listadas anteriormente.

Os anúncios possuem informações detalhadas referentes aos bovinos anunciados (peso, preço, fotos dos animais, raça, descrição, localidade, etc), assim como pode-se visualizar na Figura 3.5.

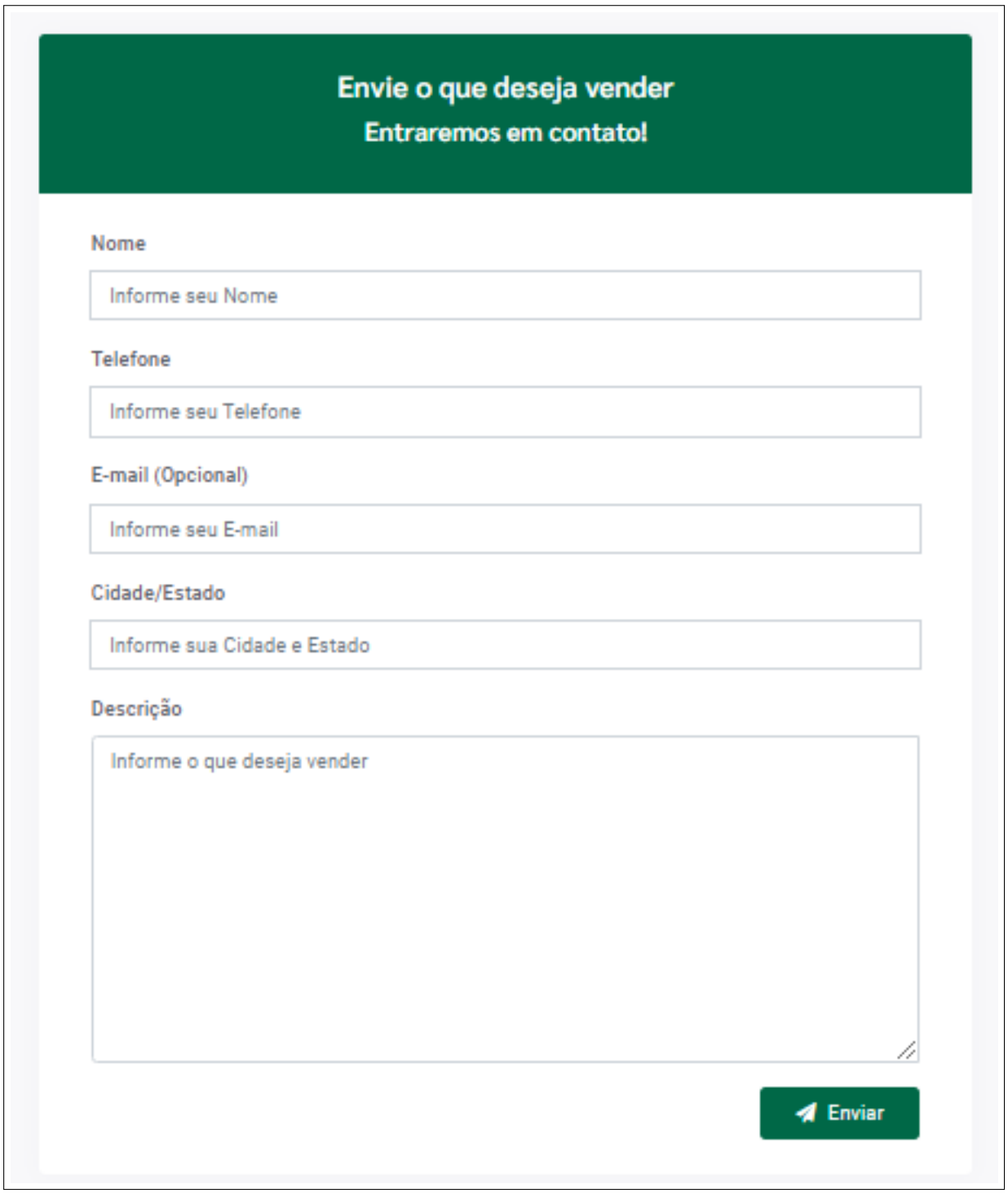
Figura 3.5 – Interface de anúncios da plataforma do App gado bom



Fonte: App Gado Bom

Para enviar uma proposta o usuário pode optar por duas opções: utilizar a função de envio de oferta pelo whatsapp ou por meio de um formulário com os seus dados. Quando desejar anunciar gado, o usuário deverá enviar o formulário presente na Figura 3.6 para um contato inicial com a plataforma e deverá aguardar um retorno solicitando as informações mais detalhadas do produto.

Figura 3.6 – Formulário para anunciar na plataforma do App gado bom



Fonte: App Gado Bom

Finalizando a analise da plataforma do App Gado Bom, consegue-se destacar as seguintes vantagens e desvantagens:

- Vantagens
  - Anúncios padronizados.

  - Filtragem parcial dos anúncios.
  - Detalhamento nas informações.
  - Integração com a API do *whatsapp*

- Desvantagens
  - Não permite o cadastro do usuário.
  - Não permite a interação direta entre os negociantes.
  - Lacuna de tempo para anunciar um produto.
  - Formulários de cadastro de anúncios muito simples e pouco detalhistas.
  - Não permite pagamento por meio da plataforma.

3.4 COMPARATIVO ENTRE AS PLATAFORMAS RELACIONADAS

Após uma descrição de cada plataforma nas subseções anteriores, pode-se efetuar uma comparação entre elas, com a finalidade de absorver os pontos fortes e os pontos fracos de cada uma para auxiliar na elaboração da plataforma do trabalho.

Por meio da Tabela 3.1 consegue-se perceber uma grande desvantagem das plataformas voltadas exclusivamente para a negociação de bovinos em relação a OLX que atua em diversos nichos. Essa grande desvantagem sucede-se principalmente pela falta de autonomia do usuário quando faz o uso das plataformas do App gado bom e do Boi na net.

Na plataforma da OLX o usuário possui um perfil e consegue manipular seu anúncios e compras conforme desejar, enquanto nas outras plataformas existe uma restrição em que os anúncios só podem ser manipulados pelos administradores da plataforma. Essa falta de liberdade acaba por tornar ruim a experiência do usuário em fazer o uso da plataforma.

Outro ponto negativo das plataformas que atuam diretamente na venda de bovinos e que ao invés de aproximar os negociantes estas acabam executando um papel de intermediários entre as partes, não permitindo um contato direto entre os compradores e vendedores. Esse contato direto entre os negociantes é o principal fator que configura o modelo de negócios C2C.

A OLX por outro lado consegue cumprir o principal objetivo do C2C e possibilita que a negociação entre os usuários ocorra completamente pela plataforma sem a necessidade de intermediários, possibilitando a fácil interação entre as partes envolvidas. Esse contato direto entre os negociantes acaba por permitir que o comprador consiga suprir todas as suas dúvidas em relação a um produto antes de efetuar a compra.

Tabela 3.1 – Comparativo entre as plataformas relacionadas ao trabalho

	App gado bom	Boi na net	OLX
Atua exclusivamente com bovinos	**Não**	**Sim**	**Não**
Permite criar, editar e excluir perfil	**Não**	**Não**	**Sim**
Permite criar, editar e excluir um anúncio	**Não**	**Não**	**Sim**
Permite a classificação de um anúncio	**Não**	**Não**	**Não**
Permite favoritar um anúncio	**Não**	**Não**	**Sim**
Permite denunciar um anúncio	**Não**	**Não**	**Sim**
Possui chat para conversação	**Não**	**Não**	**Sim**
Possui campo de busca por anúncios	**Não**	**Não**	**Sim**
Possui filtros de busca	**Não**	**Não**	**Sim**
Possui formulários para cadastro de anúncios detalhados	**Não**	**Não**	**Não**
Possui anúncios padronizados	**Sim**	**Não**	**Não**
Apresenta os dados dos vendedores e compradores	**Não**	**Não**	**Sim**
Possui sistema de pagamento	**Não**	**Não**	**Sim**

Fonte: Autor do trabalho

A maior desvantagem da OLX em relação as plataformas do App gado bom e Boi na net é diversificação dos seus anúncios. Essa exploração de diversos nichos por parte da OLX apresenta uma baixa adesão por parte dos pecuaristas e consequentemente acaba tornando-se menos confiável perante os compradores.

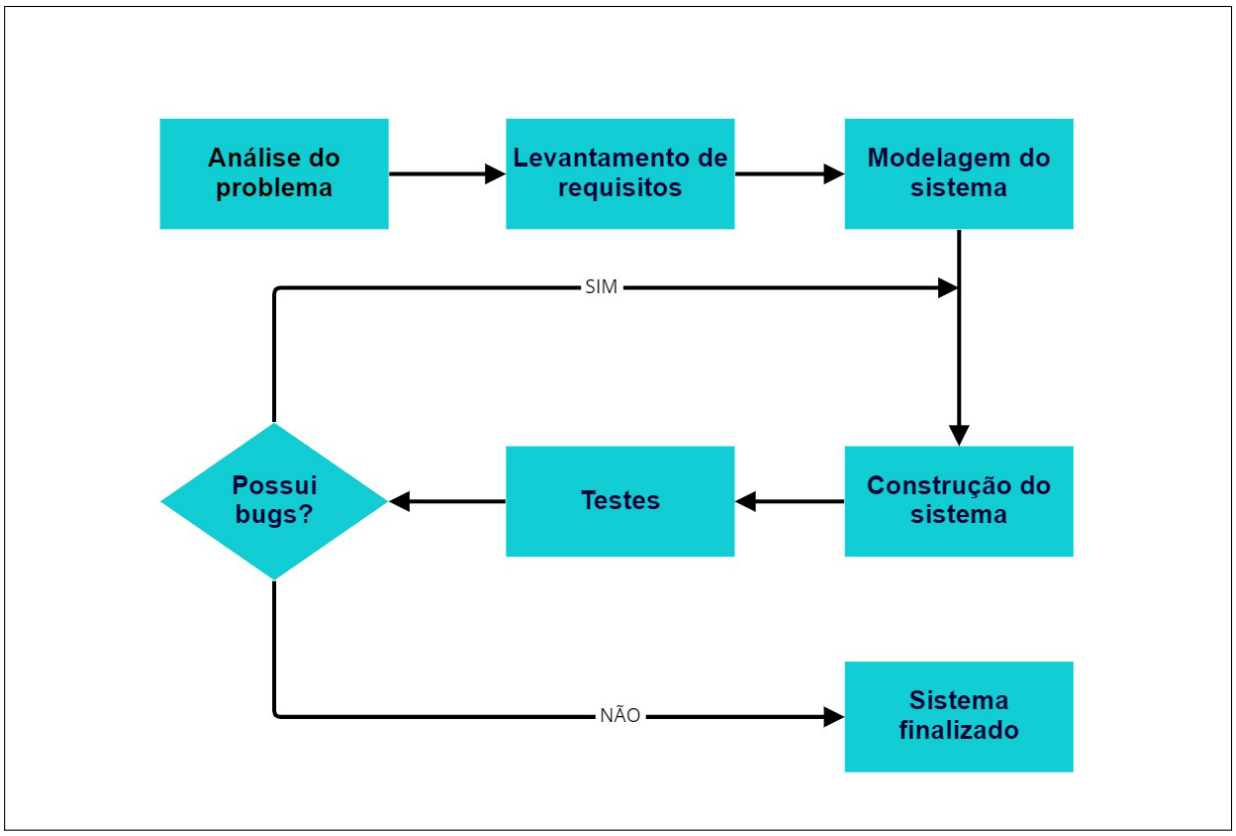
Toda essa análise permitiu efetuar a criação de uma plataforma para suprir os principais problemas do processo, explorando os pontos fortes e corrigindo os pontos fracos presentes nas principais plataformas utilizadas pelo setor atualmente.

4 METODOLOGIA E DESENVOLVIMENTO

Quando se trabalha na elaboração de um produto ou sistema, é importante seguir um roteiro com uma série de passos previsíveis. Esse roteiro denominado de “processo de *software*” possibilita a obtenção de resultados de alta qualidade e dentro de um prazo estabelecido (PRESSMAN, 2011).

Como pode-se visualizar na Figura 4.1 foi utilizado uma abordagem do fluxo de processo iterativo, o qual permite que uma ou mais atividades do fluxo se repitam.

Figura 4.1 – Fluxograma das etapas de desenvolvimento



Fonte: Autor do trabalho

Também foram utilizados para a construção do fluxograma alguns conceitos do modelo cascata, o qual sugere uma abordagem sequencial e sistemática, começando com a análise do problema e levantamento de requisitos afim de entender as necessidades por parte do cliente, avançando pela parte de planejamento, modelagem e construção da ferramenta, e terminando no suporte e incremento contínuo do *software* concluído. A seguir serão abordadas cada etapa do processo de *software*, explicando detalhadamente como cada uma foi executada para a construção do sistema.

4.1 ANÁLISE DO PROBLEMA

Com a pandemia de *Covid-19*, muitas empresas e comerciantes precisaram adaptar seus negócios para que não fechassem as portas, grande parte dos empreendedores optaram por digitalizar seus negócios por meio do *e-commerce*. Enquanto empresas de grande porte, em sua maioria optaram por vender em plataformas próprias, as pequenas mostraram preferência por plataformas do tipo C2C ou *marketplaces*.

A preferência por *marketplaces*, por parte dos pequenos e médios empreendedores, sucedeu-se por diversos fatores que facilitaram a entrada deste público no mercado virtual. Conforme descrito por (PAGAR.ME, 2021a), a principais vantagens de se comercializar em *marketplaces* são:

- Alta visibilidade: grande parte das plataformas já são consolidadas, e recebem diversos acessos diariamente. Ao vender em um *marketplace*, o empreendedor acaba se beneficiando da grande quantidade de clientes da plataforma, ganhando uma maior visibilidade para seu negócio em um curto período de tempo . Com essa grande demanda de clientes da plataforma, o vendedor acaba por reduzir os custos relacionados a divulgação, *marketing* e SEO (Search Engine Optimization), ou seja, otimização para ranqueamento nos buscadores como o Google.

- Mais oportunidades de vendas: com a visibilidade e confiabilidade que os *marketplaces* proporcionam, as vendas acabam se expandindo para um público diverso em todo o território nacional, aumentando o número de vendas e maximizando os lucros.

- Menor custo inicial: O *marketplace* oferece uma plataforma pronta para vendas, com toda a infraestrutura necessária de tecnologia, segurança e meios de pagamento, assim como estratégias de *marketing* consolidadas. Tudo isso fica disponível apenas se pagando uma taxa referente ao total de vendas, um valor muito menor do que se construir uma loja virtual do zero e investir em *marketing*.

- Público diversificado: o principal diferencial do *marketplace* é o público diversificado. Isso significa que, ao disponibilizar as mercadorias o vendedor pode atrair novos consumidores, inclusive aqueles que não estavam procurando diretamente pela sua loja. Essa diversidade de consumidores, possibilita os vendedores inovarem e aumentarem sua gama de produtos de acordo com as novas demandas, isso acaba contribuindo com o crescimento do negócio.

Buscando utilizar-se das vantagens descritas anteriormente para auxiliar os empreendedores de Santa Maria, foi detectado uma necessidade de implantar uma plataforma de vendas *online* utilizando o modelo de negócios C2C ou *marketplace* no setor pecuário da região. Durante uma breve pesquisa para verificar como ocorre a negociação de gado atualmente, foi observado a utilização de diversos meios de divulgação para os anúncios.

Os principais meios utilizados pelos vendedores e compradores são as redes sociais (WhatsApp, Facebook, Instagram) e algumas plataformas de *marketplace* como a OLX, Boi na net e App gado bom. Essa diversidade de meios para divulgação de anúncios acaba por desencadear diversos problemas no processo de negociação de gado, os quais são explicados abaixo:

- Criação de anúncio repetida: toda vez que um comerciante desejar vender um produto será necessário repetir o processo de criação de anúncio em cada plataforma, pois uma plataforma se distingue da outra e na maioria das vezes não permite a importação de anúncios pré-definidos.
- Descentralização: o uso de diversos meios de divulgação acaba "espalhando" os negociantes em diversas plataformas, o que acaba por dificultar a venda e a busca por um produto, como por exemplo: um vendedor anuncia um produto no Facebook, enquanto o comprador que possui o perfil de compra está buscando por um produto semelhante na OLX.
- Cadastros múltiplos: a descentralização do negociantes em várias plataformas acaba por obrigar o anunciante e o comprador a possuírem cadastro em cada plataforma que forem utilizar.
- Plataformas de venda generalistas: a maioria dos *marketplaces* utilizados pelos vendedores permite a venda de produtos de diversos setores, consequentemente os formulários para criação de anúncios são genéricos, o que limita a inserção de informações mais especificas referentes aos bovinos, como por exemplo a adição da raça, peso e tipo do animal.
- Despadronização dos anúncios: plataformas como *Facebook* e *Instagram* proíbem a venda de animais em seus *marketplaces*, e na maioria das vezes os usuários acabam burlando essa proibição e acabam anunciando os bovinos por meio de grupos e publicações em suas linhas do tempo por meio de anúncios despadronizados e na maioria das vezes com poucas informações.
- Dificuldade na comunicação entre as partes: na maior parte das plataformas é necessário informar números de telefone e/ou e-mail para comunicação entre os negociantes, como consequência isso pode acabar poluindo o *chat* das ferramentas usadas no dia a dia pelos usuários. Um exemplo são anúncios no *Facebook* que na maioria das vezes os vendedores informam seus telefones para contato direto ou fazem o uso do *chat* presente na plataforma, porém as conversas acabam se misturando com mensagens de assuntos pessoais do usuário.

Após detectados e analisados os problemas no processo de compra e venda de bovinos, foi executada a etapa de levantamento de requisitos, a qual será abordada na

seção a seguir.

4.2 LEVANTAMENTO DE REQUISITOS

Projetar e construir um programa de computador elegante que resolva o problema errado não atende às necessidades de ninguém. É por isso que é importante entender o que o cliente quer antes de começar a projetar e construir um sistema (PRESSMAN, 2011).

Descreve-se também que durante esta etapa é definido o que é necessário para a criação do sistema, onde os requisitos básicos são refinados e modificados de acordo com as necessidades do cliente. À medida que os problemas são definidos, ocorre uma avaliação para decidir quais as prioridades, o que é essencial e quando é necessário manter uma funcionalidade (PRESSMAN, 2011).

A engenharia de requisitos fornece o que é apropriado para um melhor entendimento sobre os desejos do cliente, analisando as necessidades, avaliando a viabilidade, negociando uma solução razoável, especificando a solução sem ambiguidades, validando a especificação e gerenciando as necessidades à medida em que se concebe o sistema (PRESSMAN, 2011).

Para o levantamento de requisitos foram realizadas 5 conversas informais com criadores de gado da região, com a finalidade de obter uma visão mais técnica sobre o assunto e entender melhor as necessidades do público alvo da plataforma a ser criada.

Do público entrevistado 2 pessoas trabalham somente com gado de leite, 2 trabalham somente com gado de corte e 1 pessoa trabalha tanto com gado de corte quanto com gado de leite. Em relação ao tamanho da propriedade, 3 produtores possuem propriedades de pequeno porte (até 50 ha), 1 produtor possui propriedade de médio porte (de 50 ha à 500 ha) e 1 produtor possui propriedade de grande porte (acima de 500 ha).

Quando questionados sobre o uso de ferramentas para auxiliar no anúncio e na venda de gado, todos os 5 entrevistados informaram utilizar apenas de redes sociais (Whatsapp e Facebook) para efetuar o contato com os compradores, enquanto apenas 1 deles utiliza o *Whatsapp* para anunciar seu rebanho. Para encontrar compradores os entrevistados utilizam-se de opções simples, como por exemplo: buscam indicações de compradores com outros produtores de gado, possuem um comprador direto, efetuam a venda diretamente para frigoríficos e também fazem o uso dos comissionados (intermediários entre o vendedor e o comprador).

Ao perguntar para os produtores se eles teriam interesse em fazer o uso da plataforma para anunciar suas cabeças de gado, todos os 5 entrevistados se mostraram interessados em como funcionaria a ferramenta e demonstraram desejo em usa-la. Pensando na construção da plataforma foi perguntado aos produtores quais funcionalidades seriam

importantes serem implementadas para que pudessem suprir as necessidades do público alvo, e as principais respostas foram: anúncios devem conter a raça, o peso, imagens e vídeos dos animais, diferenciação entre gado de leite e de corte, confiabilidade nos anúncios, dados referentes as vacinações e ao calendário sanitário do rebanho, informações detalhadas do vendedor, busca com filtros, entre outros. Todas as perguntas e respostas utilizadas para o levantamento de requisitos podem ser encontradas no Anexo A;

A partir das informações obtidas através dos pecuaristas, foi feito um refinamento para definir quais os principais requisitos que a ferramenta deverá possuir.

Os requisitos podem ser divididos em 2 tipos: Funcionais - são todos os problemas e necessidades que devem ser atendidos e resolvidos pelo sistema por meio de funções e serviços; Não-funcionais - estão relacionados à forma em como o sistema tornará realidade o que está sendo proposto (CUNHA, 2022).

Abaixo segue uma lista com os requisitos a serem implementados:

- **Funcionais**
  - Efetuar *login*
  - Efetuar *logout*
  - Cadastrar, editar e deletar usuário próprio
  - Cadastrar, editar e deletar anúncio
  - Buscar anúncio
  - Filtrar busca
  - Visualizar anúncios
  - Favoritar anúncios
  - Classificar anúncios
  - Denunciar anúncios
  - Iniciar conversa via chat
- **Não-funcionais**
  - O sistema será *web*, precisando apenas de conexão com a *internet* e um dispositivo para acessar
  - Interface limpa e de fácil manuseio
  - O sistema deverá agilizar o processo de negociação de bovinos, por permitir a centralização do público do setor pecuário da região.

Depois de definidos os requisitos necessários para a aplicação foi realizada a etapa da modelagem do sistema, afim de permitir uma melhor visualização de como será a estrutura do projeto e como será o comportamento de cada funcionalidade.

4.3 MODELAGEM DO SISTEMA

Durante esta seção serão abordados assuntos referentes as modelagens do sistema, como a criação do diagrama de casos de uso, diagrama de classes, e a modelagem da base de dados.

4.3.1 Diagrama de casos de uso

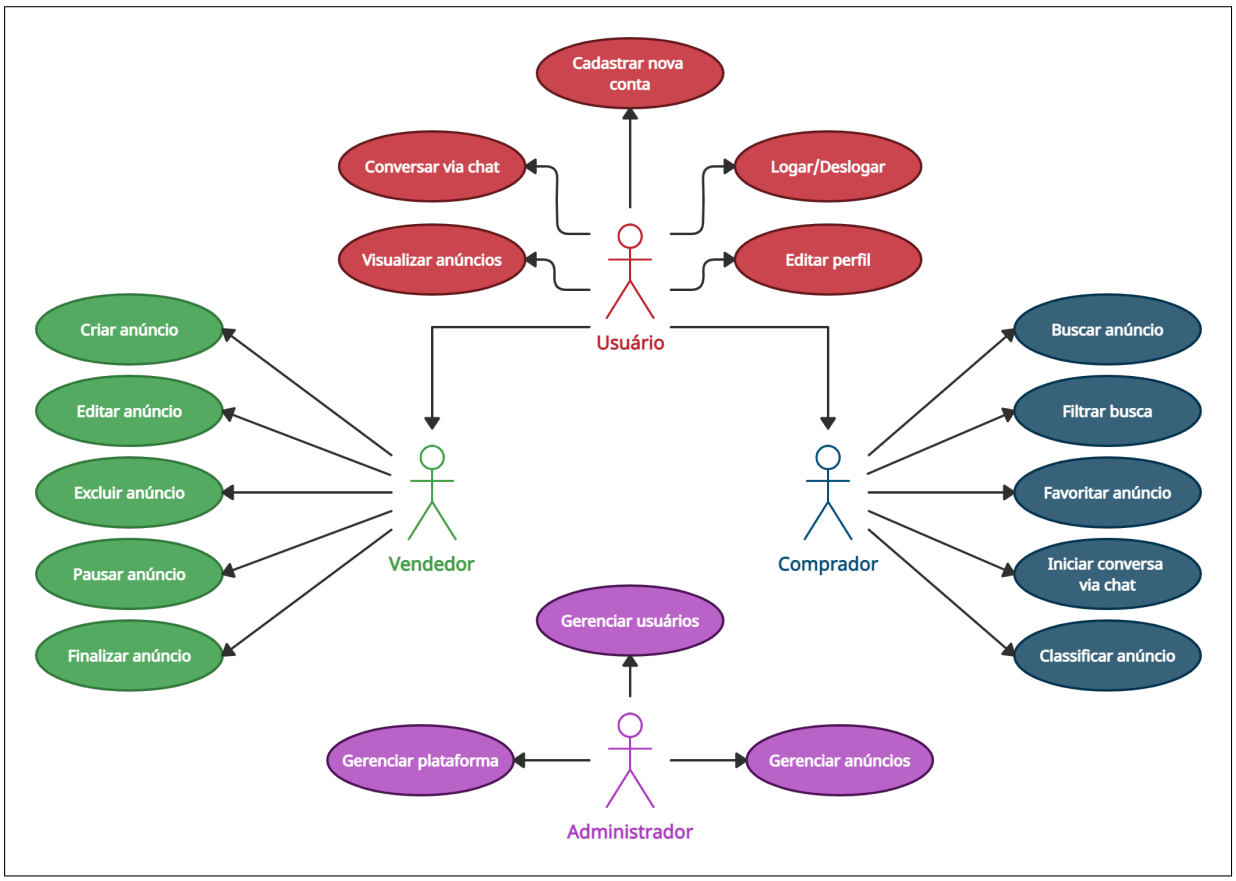
Primeiramente foi modelado o diagrama de casos de uso para visualizar quais os principais atores da plataforma, e consequentemente definir e limitar as ações de cada um deles. Como abstraído na Figura 4.2 foram definidos 4 atores principais, o comprador, o vendedor, o usuário global e o administrador.

Ainda analisando a Figura 4.2, pode-se visualizar que todas as funcionalidades do sistema poderão ser usadas apenas com 1 cadastro de usuário, porém as funcionalidades do sistema serão liberadas de acordo com a ação que se deseja executar no momento. Quando o usuário desejar vender gado será possível criar, editar, excluir, finalizar e pausar um anúncio; quando desejar comprar gado poderá buscar anúncios com o auxilio de filtros, favoritar anúncios, iniciar uma conversa com um vendedor via chat e classificar um anúncio.

Também podemos visualizar na Figura 4.2 que independente do tipo de ação (comprar ou vender) que o usuário desejar efetuar no sistema ele poderá executar as seguintes funcionalidades: cadastrar uma nova conta, conversar via chat, visualizar todos os anúncios, logar/deslogar e editar seu perfil.

Como administrador o usuário poderá gerenciar os usuários (adicionar, editar, bloquear e excluir), os anúncios (visualizar, excluir e editar) e poderá gerenciar algumas configurações da plataforma.

Figura 4.2 – Diagrama de casos de uso do sistema



Fonte: Autor do trabalho

4.3.2 Modelagem da base de dados

Após a construção do diagrama de casos de uso, modelou-se a base de dados NoSQL orientado a documentos utilizando o formato de dados JSON (JavaScript Object Notation), o qual permite uma construção mais flexível, podendo adicionar novas informações ao documento conforme a demanda da plataforma.

A abordagem do NoSQL não sugere uma modelagem especifica para a abstração da base dados assim como é feito nos bancos de dados relacionais, porém neste trabalho foi construído um modelagem afim de documentar todas as informações salvas na base de dados para um melhor entendimento.

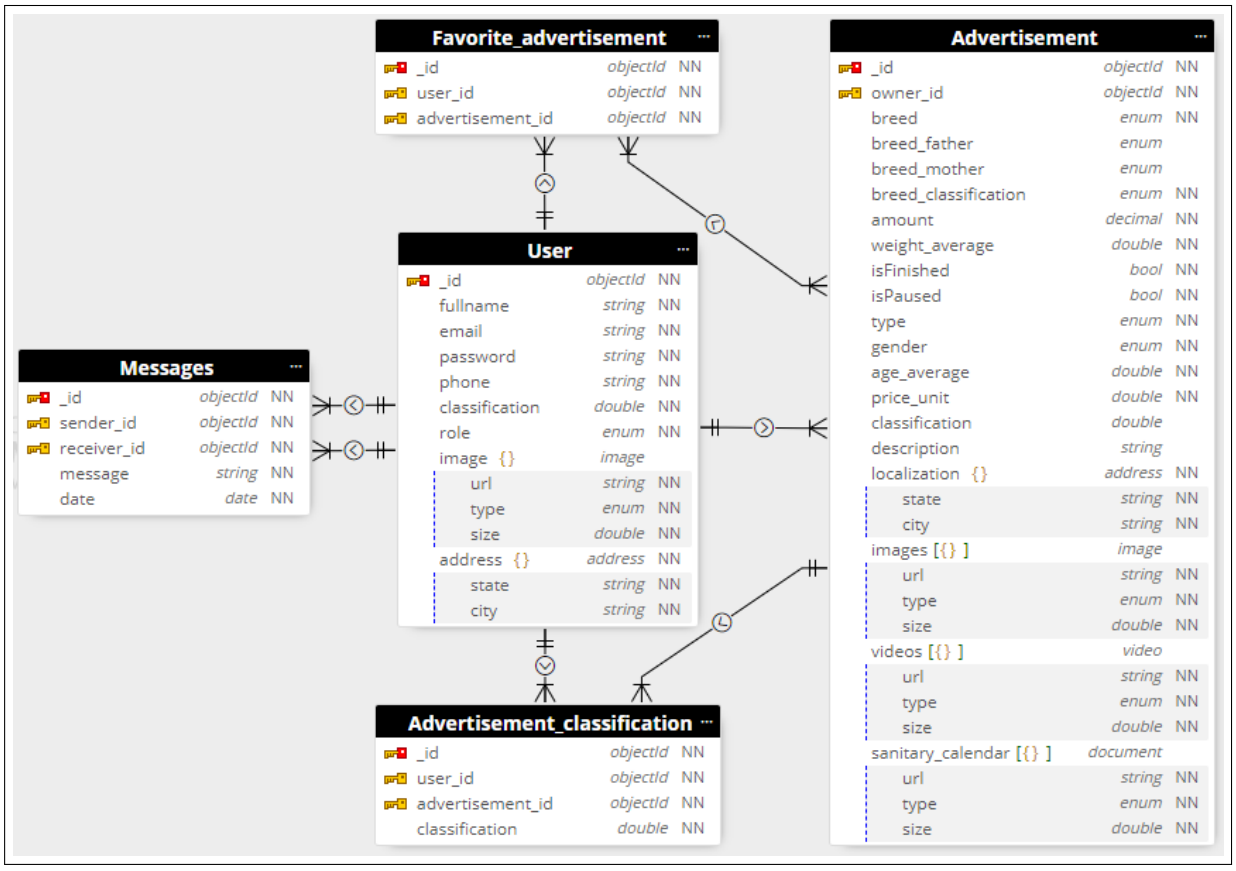
Como pode-se visualizar na Figura 4.3 foram modeladas 5 coleções. A coleção de “User”, que será responsável por armazenar os documentos referentes aos cadastros dos usuários da plataforma, que poderão ser de dois tipos (USER e ADMIN). Para armazenar os documentos pertencentes aos anúncios foi criada uma coleção nomeada de “Advertisement” onde serão salvas informações sobre os anúncios de bovinos como raça, peso,

idade, quantidade, gênero, preço, classificação, e também será salvo uma referência para o usuário que criar o anúncio.

Ainda na Figura 4.3 pode-se observar a abstração de uma coleção nomeada de “Favorite_advertisement”, que ficará responsável por salvar uma referência do usuário e outra referência do anúncio favoritado. Também foi criado uma coleção com o nome de “Messages”, à qual ficará responsável por armazenar as mensagem trocadas entre os usuários da plataforma.

Criou-se também uma coleção denominada “Advertisement_classification”, que receberá os documentos pertencentes as classificações de cada anúncio, dentro desta coleção será salvo um campo com o identificador do anúncio e um identificador do usuário que classificou o anúncio, juntamente com o valor da classificação que irá variar de 0 à 5.

Figura 4.3 – Modelo abstrato da base de dados NoSQL

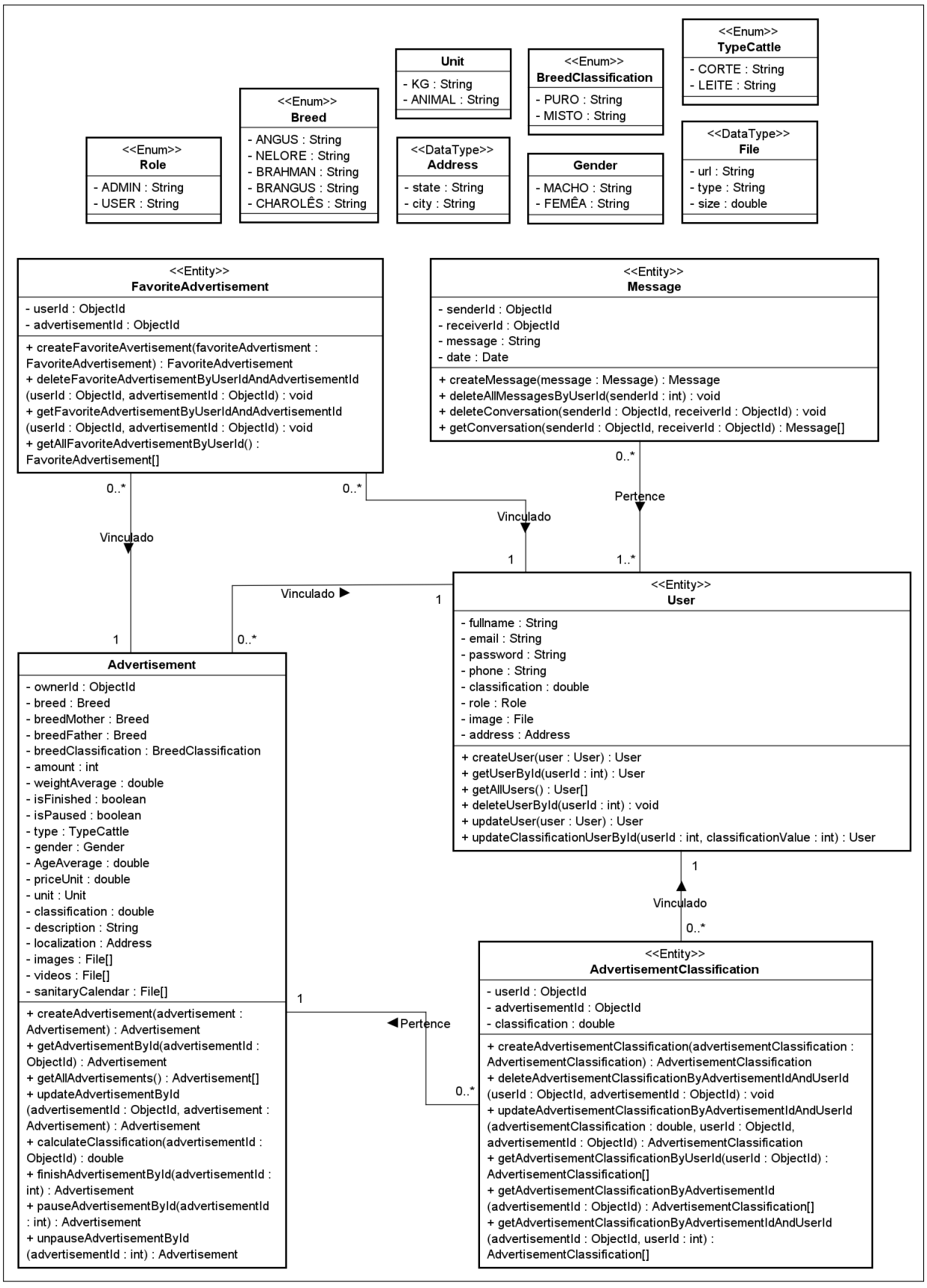


Fonte: Autor do trabalho

4.3.3 Diagrama de classes

Depois de modelada a base de dados foi feito a criação do diagrama de classes da aplicação, que pode ser visualizado na Figura 4.4.

Figura 4.4 – Diagrama de classes do sistema



Fonte: Autor do trabalho

Com a criação deste diagrama pode-se abstrair cada objeto da aplicação e seus respectivos métodos. Cada objeto modelado irá se tornar um *model* na aplicação e, consequentemente cada método irá se transformar em uma função distinta do sistema. Outro ponto importante da abstração das classes é que pode-se representar as possíveis relações entre os objetos.

Como pode ser observado a aplicação possuirá 5 *models* principais que ficarão responsáveis por persistir e recuperar as informações da base de dados. Cada uma das funções abstraídas efetuara uma regra de negócio do sistema e irá definir as ações que poderão ser executadas pelo usuário.

4.4 CLASSIFICAÇÃO DE ANÚNCIOS

Com o objetivo de proporcionar uma maior confiabilidade para a plataforma, foi elaborado um sistema de classificação para anúncios e usuários, funcionalidade está que é um ponto fraco das plataformas relacionadas com o trabalho que foram citadas no Capítulo 3.

A classificações dos anúncios serão calculadas a partir dos valores salvos na coleção de “Advertisment_classification” presente na base de dados abstraída na Figura 4.3. Cada documento da coleção irá conter uma referência para o anúncio classificado, uma referência para o usuário que o classificou e o valor da classificação que pode variar de 1 à 5. A importância da referência do usuário estar salva no documento e para que o anúncio seja classificado apenas uma vez por cada usuário.

Toda vez que um anúncio receber uma classificação, será criado o documento na base de dados referente a está ação e em seguida será feito o cálculo para atualizar a classificação do anúncio. Para efetuar o cálculo serão recuperados todos os documentos que possuírem o mesmo “advertisment_id” na coleção “Advertisement_classification”. E será calculada a média das classificações para gerar a nova classificação do anúncio.

Para calcular a classificação de um usuário serão utilizados os documentos referentes ao anúncios que o usuário possui na plataforma. Quando a classificação de um anúncio ser atualizada, serão recuperados todos os anúncios pertencentes ao mesmo “owner_id” na coleção “Advertisment” e a partir do somatório das classificações será calculado a média, atualizando assim a classificação do usuário.

Tanto a classificação de um anúncio, quanto a classificação de um usuário poderão ser visualizadas dentro da plataforma de forma pública proporcionando assim uma maior segurança na hora que o usuário desejar efetuar uma negociação.

4.5 FERRAMENTAS UTILIZADAS

Depois de modelado o sistema, foi feita a escolha das tecnologias (linguagem de programação, padrão de projeto, *frameworks*, banco de dados, entre outras) necessárias para a criação da plataforma.

Inicialmente foi definido que o sistema fará o uso do padrão de projetos MVC, dividido em 3 camadas distintas: *front-end*, *back-end* e banco de dados. As tecnologias escolhidas para a construção do sistema fazem referência a *stack* MERN, que é composta pelas seguintes tecnologias: MongoDB, Express.js, React.js e Node.js.

O *front-end* será responsável por permitir a interação dos usuários com a plataforma por meio de interfaces gráficas compostas por formulários, botões, imagens, textos, links, etc. Para a construção desta camada foi utilizado o *React.js*, que permite a criação de componentes e páginas JSX (extensão de sintaxe para *JavaScript*), possibilitando a criação de elementos HTML juntamente com funcionalidades do *JavaScript*. A vantagem de utilizar esta biblioteca e a facilidade com que é possível consumir e enviar dados para o *back-end*.

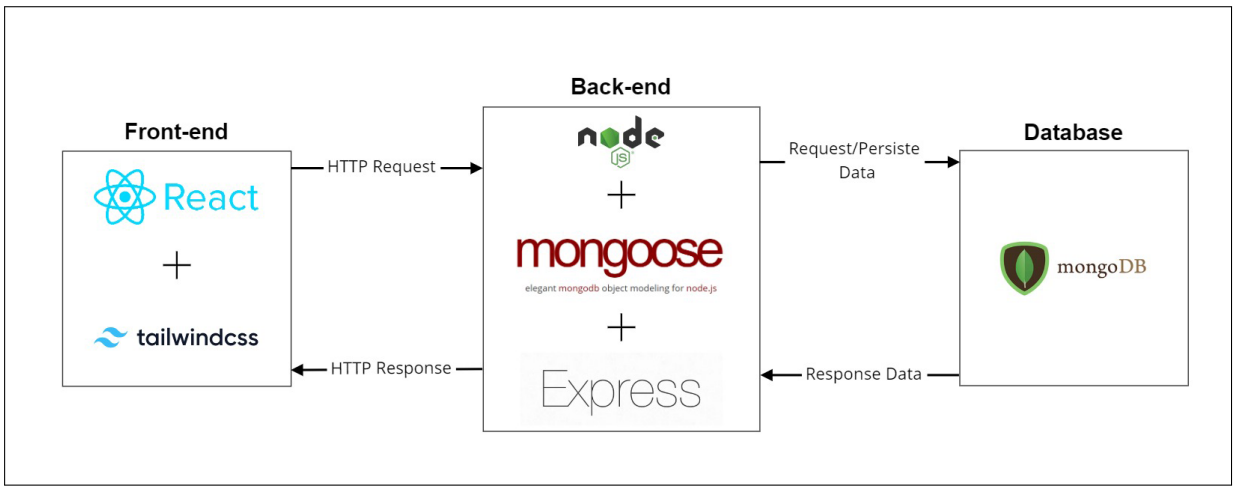
Para fazer a estilização das páginas geradas com o *React.js*, afim de proporcionar uma melhor usabilidade e clareza para o usuário foi utilizado o *Tailwind* CSS, que possibilita inserção de estilos de forma rápida e fácil nos componentes gráficos das interfaces criadas. As vantagens de se utilizar esse *framework* na aplicação é que existem diversas classes prontas que auxiliam no momento de tornar a página responsiva e também porque é totalmente flexível à customizações.

O *back-end* ficará responsável por efetuar a conexão do *front-end* com a base de dados. Para implementar está camada foi decido utilizar *Node.js*, pois possibilita a execução de código *JavaScript* do lado do servidor. Também foram utilizados 2 pacotes principais do NPM: o *Express.js* que será utilizado para a criação de rotas; e o *mongoose* que fará a criação dos *models* e a conexão com a base de dados.

Em relação a base de dados foi decidido por fazer o uso do *MongoDB*, que é um banco de dados NoSQL orientado a documentos. A principal vantagem de se trabalhar com este banco de dados atrelado ao *Node.js* é a existência do *mongoose* que possibilita a criação dos *models* e a conexão com a base de dados de forma rápida e prática.

Na Figura 4.5 consegue-se visualizar como serão utilizas as tecnologias principais da plataforma e como serão feitas as comunicações entre elas.

Figura 4.5 – Ferramentas utilizadas na implementação



Fonte: Autor do trabalho

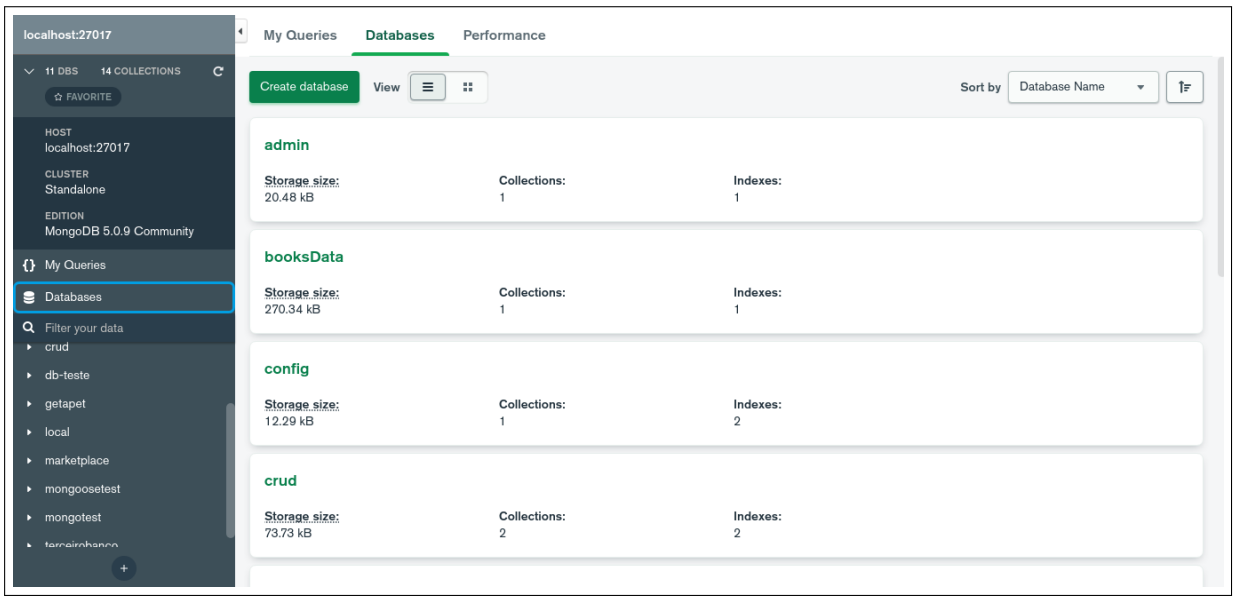
4.6 DESENVOLVIMENTO DO SISTEMA

Nesta seção serão descritas todas as etapas percorridas para a criação da plataforma, desde a criação da base de dados até a finalização do projeto.

4.6.1 Criação da base de dados no MongoDB

A primeira etapa da construção da plataforma foi a criação da base de dados no MongoDB para poder efetuar a integração com o *back-end*. Para fazer a criação utilizou-se a interface gráfica *MongoDB Compass*, a qual permite que a criação do *database* seja feita apenas com um clique em um botão, como podemos visualizar na Figura 4.6. Toda a estruturação da base de dados, como a criação das coleções e do documentos serão definidas no *back-end*, sem a necessidade de definir a estrutura dentro da base de dados.

Figura 4.6 – Interface do MongoDB Compass para criar a base de dados



Fonte: Autor do trabalho

4.6.2 Criação do *back-end*

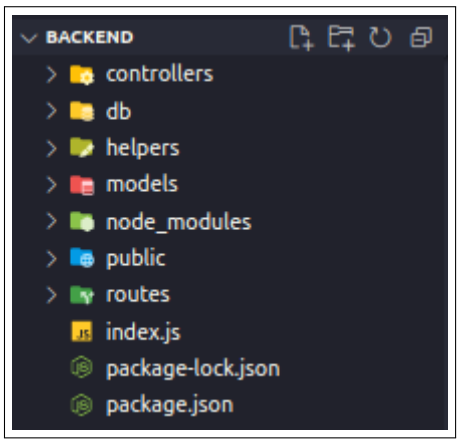
Depois de criada a base de dados, foi feita a construção do *back-end*, onde foi construído um servidor com estilo arquitetural de API, permitindo a criação de *endpoints* para facilitar a interação com o *front-end*. A seguir serão descritas a principias etapas da criação do *back-end*.

4.6.2.1 Inicialização e organização do sistema

O primeiro passo para a criação do *back-end* foi a inicialização e configuração do projeto utilizando o NPM. Para efetuar a inicialização foi executado o comando “npm init -y” dentro da pasta do projeto, a execução deste comando fez a criação do arquivo “package.json”, que é responsável por armazenar todas as informações do projeto, como os *scripts de execução*, as dependências instaladas e suas versões, entre outras configurações.

Depois de feita a inicialização do arquivo de configurações do projeto, foi feito a criação das pastas e arquivos necessários para implementação utilizando a estrutura MVC conforme a Figura 4.7.

Figura 4.7 – Estrutura de pastas e arquivos do *back-end*



Fonte: Autor do trabalho

4.6.2.2 Conexão com o MongoDB

Depois de estruturadas as pastas e os arquivos do *back-end* foi definida a conexão do servidor *Node.js* com o banco de dados através do *mongoose*, para possibilitar a criação dos *models* do sistema. No Código 4.1 pode-se observar como é feita a conexão com a base de dados utilizando o *mongoose*. Na linha 1 é feita a importação da biblioteca para que ela possa ser usada no documento. Na linha 2 é declarada a URI local da base de dados do sistema. Entre as linhas 3 e 10 foi criado uma função assíncrona para efetuar a conexão com o *MongoDB* utilizando o método "*connect*" do *mongoose* e também foram tratados os possíveis erros de conexão. Na linha 11 foi feito a chamada para a função de conexão com a base de dados e na linha 12 foi feito a exportação da conexão com o banco.

Código 4.1: Conexão com o banco de dados usando *mongoose*

```
1  const mongoose = require('mongoose')
2  const uri = "mongodb://localhost:27017/marketplace"
3  const run = async () => {
4      try{
5          await mongoose.connect(uri)
6          console.log('Sucess connecting to MongoDB')
7      }catch(e){
8          console.error('Error connecting to MongoDB')
9      }
10 }
11 run()
12 module.exports = mongoose
```

Fonte: Autor do trabalho

4.6.2.3 Implementação dos models

Com a conexão estabelecida entre o *back-end* e o banco de dados, construiu-se os *models* da aplicação. Utilizando o *mongoose* consegue-se definir os atributos, os tipos e outras configurações do *model* com poucas linhas de código. No Código 4.2 pode-se observar como exemplo a criação do *model* de usuário, onde todos os campos criados no *mongoose* serão mapeados para base de dados de acordo com as configurações setadas.

Na linha 1 está sendo importado o método de conexão criado na subseção 4.6.2.2, para possibilitar que as ações do *model* possam ser persistidas na base de dados. Na linha 2 está sendo declarado o método *Schema* pertencente ao *mongoose* e é responsável por fazer a criação do esquema dentro do *MongoDB*. Entre as linhas 3 e 25 estão sendo definidos os atributos e suas restrições.

Código 4.2: *Model* de usuário

```
1  const mongoose = require('../db/conn')
2  const {Schema} = mongoose;
3  const User = mongoose.model('User',
4      new Schema({
5          fullname: { type: String, required: true},
6          email: { type: String, required: true},
7          password:{ type: String, required: true},
8          phone:{ type: String,  required: true},
9          classification: { type: Number, required: true},
10         role: { type: String, enum: ['ADMIN', 'USER']},
11         image: {
```

```
12              type: {
13                  url:{ type: String, required: true },
14                  type: { type: String, enum: ['JPEG', 'PNG', '
                        JPG'], required: true },
15                  size: { type: Number, required: true }
16              }
17          },
18          address: {
19              type: {
20                  state: { type: String, required: true },
21                  city: { type: String, required: true}
22              }
23          }
24      })
25  )
```

Fonte: Autor do trabalho

4.6.2.4 Implementação dos middlewares

Com os *models* definidos, foram implementados os *middlewares* da aplicação. Esses middlewares serão responsáveis por processar as requisições do usuário para a API. Para este sistema foram criados 2 *middlewares*: um para verificar se um usuário esta autenticado na plataforma e o outro para efetuar o *upload* de arquivos através dos formulários.

O *middleware* responsável pela autenticação efetua uma busca pelo token de autorização no cabeçalho da requisição HTTP, se encontrar-lo será feita uma validação para permitir ou não o acesso do usuário ao sistema, para a implementação deste *middleware* foi utilizada a biblioteca “jsonwebtoken” do *Node.js*, que possibilita a autenticação de usuários através de *tokens* que salvam informações referentes aos usuários de forma criptografada.

O *middleware* criado para o *upload* de arquivos quando utilizado irá efetuar o salvamento dos arquivos em um diretório especificado nas configurações do projeto. Para a criação deste *middleware* foi utilizada uma biblioteca do *Node.js* denominada de “multer”, que permite fazer o *upload* de documentos, imagens, vídeos, entre outros.

4.6.2.5 Implementação das rotas

Com as implementações dos middleweres e dos *models* finalizadas, criou-se as rotas de acesso para os serviços da API. Para a criação desta funcionalidade foi utilizado

um método presente no *Express.js*. O método “Router()” permitiu a criação dos *endpoints* de forma rápida e prática utilizando-se dos *middlewares* e das configurações necessárias para o funcionamento da plataforma.

Código 4.3: *Endpoints* de usuário

```
1  const express = require('express');
2  const router = express.Router();
3  const UserController = require('../controllers/UserController')
4  const verifyToken = require('../helpers/verifyToken')
5  const {imageUpload} = require('../helpers/imageUpload')
6
7  router.post('/register', imageUpload.single("image"),
       UserController.register)
8  router.post('/login', UserController.login)
9  router.get('/checkuser', UserController.checkUser)
10 router.get('/:id', UserController.getUserById)
11 router.get('/', verifyToken, UserController.getAllUsers)
12 router.delete('/:id', verifyToken, UserController.
       deleteUserById)
13 router.patch('/edit/:id', verifyToken, imageUpload.single("
       image"), UserController.updateUserById)
14 router.patch('/edit/classification/:id', UserController.
       updateClassificationById)
15
16 module.exports = router;
```

Fonte: Autor do trabalho

No Código 4.3 pode-se visualizar como exemplo a criação as rotas para as ações do usuário. Entre as linhas 1 e 5 estão sendo importados os arquivos do *Express.js*, do *controller* de usuário e os *middlewares* para verificação de *token* e para *upload* de imagem.

A partir da linha 7 até a linha 14, estão sendo definidas as rotas referente aos usuários que ficarão expostas para o *front-end*. Para a criação das rotas são definidos os HTTPVerbs (GET, PUT, DELETE, PATCH, UPDATE, POST), os *middlewares* que serão executados antes das requisições serem processadas e também é definido qual *controller* fará a execução da requisição.

5 ESTUDOS DE CASO

Neste capítulo serão apresentados os 3 principais estudos de casos da plataforma, detalhando como o sistema se comporta para a execução de cada um deles. Também serão apresentadas as principias telas do sistema para um melhor entendimento sobre cada funcionalidade.

5.1 CADASTRO DE USUÁRIO

Para efetuar o cadastro de um novo usuário, o pecuarista deverá selecionar a opção de cadastro na barra superior da aplicação que irá redireciona-lo através da URL para a página de cadastro de usuário, que pode ser visualizada na Figura 5.1. A página de cadastro contém um formulário que deverá ser preenchido com os dados solicitados e deverá ser submetido através do botão “Cadastrar” no final da página.

Figura 5.1 – Interface para o cadastro de usuário no sistema

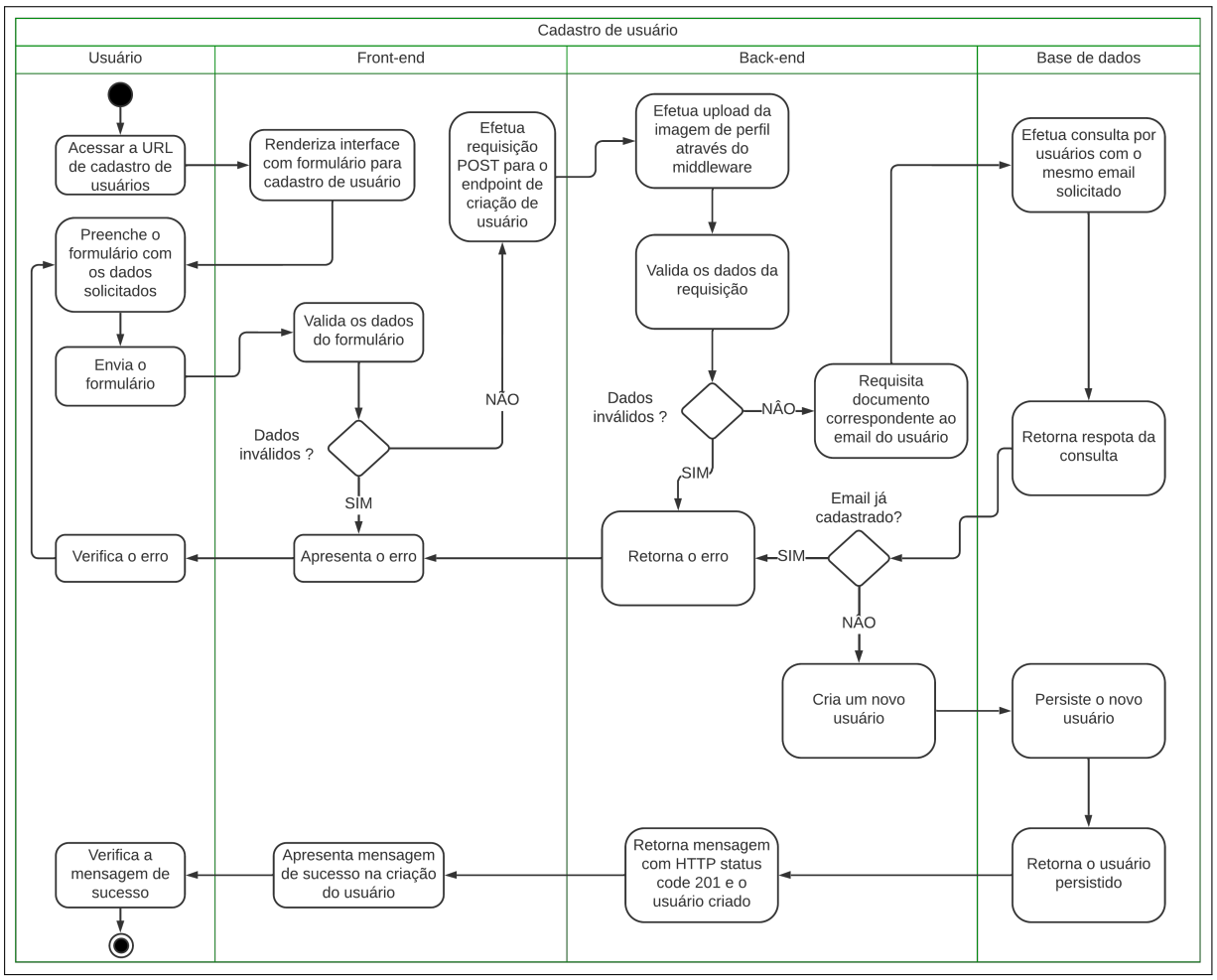
Fonte: Autor do trabalho

Depois de submetido o formulário, o *front-end* irá validar os dados recebidos e em caso de alguma inconsistência ou falta de informação irá apresentar os erros para o usuário. Se os dados estiverem preenchidos corretamente eles serão repassados para o *back-end* no corpo de uma requisição *HTTP POST* para o *endpoint* de registro de usuário criado no Capítulo 4. Quando o *back-end* receber a requisição ele irá chamar o *middleware* para efetuar o *upload* da imagem de perfil, caso ela existir, no local de armazenamento de mídias da aplicação.

Com o *upload* da imagem concluído, o *back-end* irá efetuar uma validação mais detalhada dos dados recebidos na requisição, caso estes possuam alguma inconsistência, serão retornados os erros para o *front-end* que serão apresentados para o usuário. Caso os dados estejam corretos será executada uma consulta na base de dados para verificar se o *e-mail* informado é único, caso já exista algum registro com este *e-mail* na base de dados, será retornado um erro para o *front-end* que será apresentado para o usuário.

Se os dados forem válidos e o *e-mail* informado não possuir referências na base de dados, será criado um objeto com o novo usuário e o mesmo será persistido como um novo documento na coleção de usuário da aplicação, retornando para o *front-end* uma resposta com *HTTP status code* 201 (*created*) com o usuário criado. E para concluir o *front-end* apresentará uma mensagem de sucesso no cadastro para o novo usuário da plataforma. No Figura 5.2 pode-se visualizar um diagrama de atividades que representa todos os passos descritos anteriormente.

Figura 5.2 – Diagrama de atividades para o cadastro de usuário no sistema



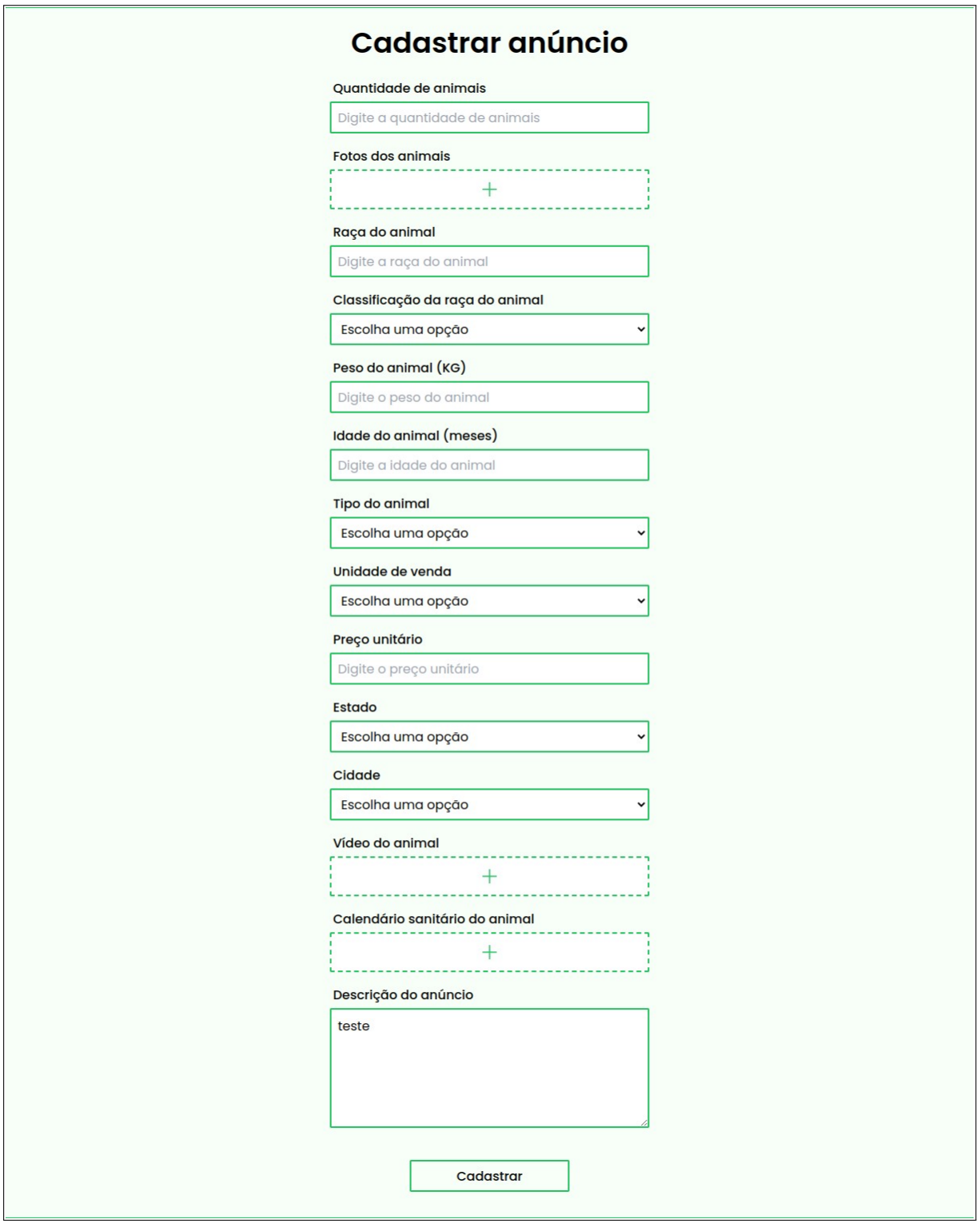
Fonte: Autor do trabalho

5.2 CADASTRO DE ANÚNCIO

Toda vez que o usuário desejar efetuar o cadastro de um anúncio ele deverá clicar no botão “Anunciar” presente na barra de navegação do sistema. Depois de clicado no botão será feita uma requisição *HTTP GET* para o *endpoint* no *back-end* responsável por verificar a existência de um *token* de autenticação e efetuar a validação. Caso o *token* não exista ou não seja válido, será retornada para o *front-end* e apresentada para o usuário uma mensagem de acesso negado para está funcionalidade.

Se o *token* existir e for válido, o usuário será redirecionado para a página de anúncio de bovinos que pode ser visualizada na Figura 5.3

Figura 5.3 – Interface para o cadastro de anúncio no sistema



Fonte: Autor do trabalho

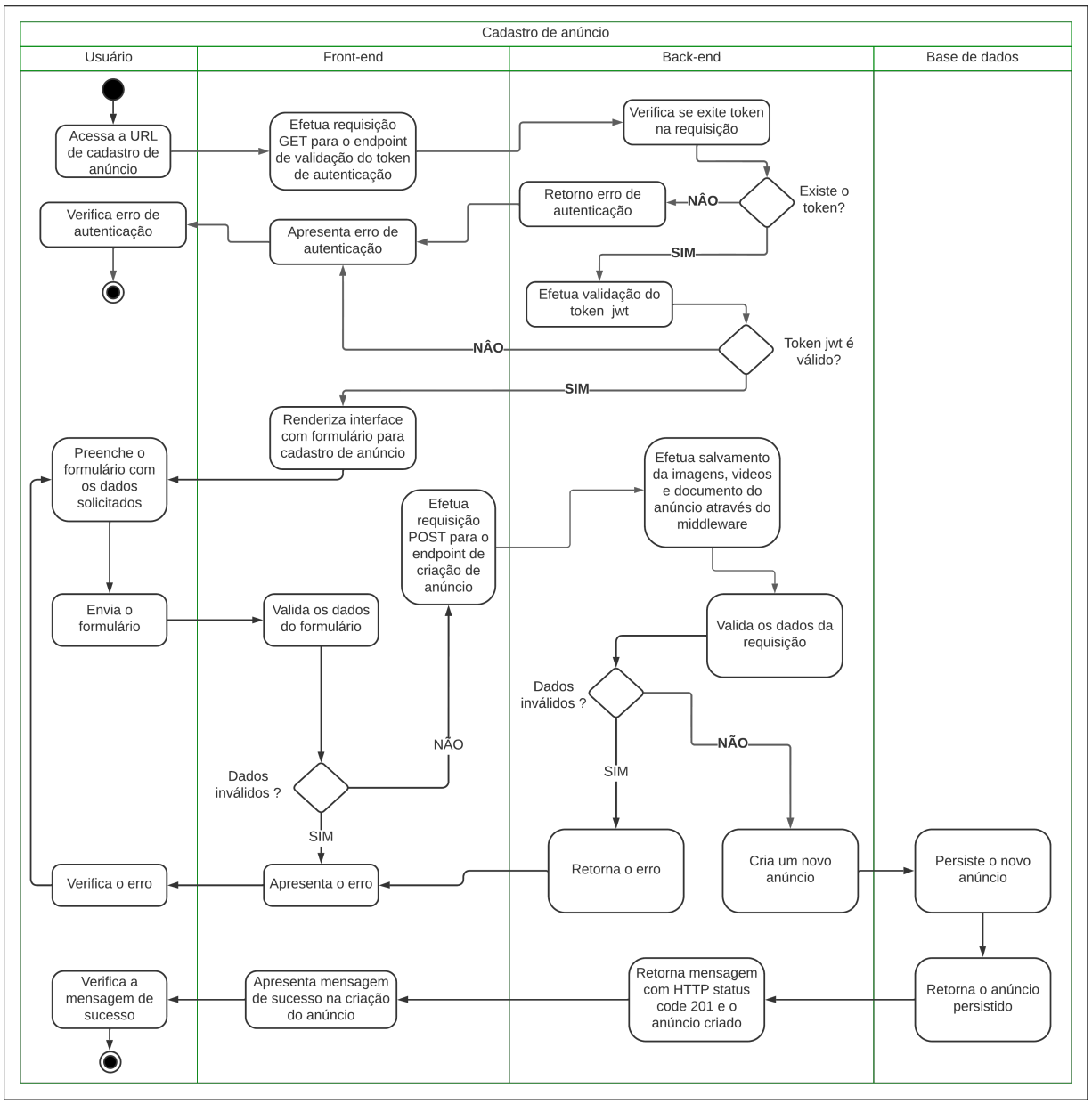
Depois que o usuário preencher todos os campos e clicar no botão cadastrar o *front-end* irá efetuar uma validação nos campos, e caso haja alguma inconsistência nos dados serão apresentados alguns erros para o usuário informando quais campos estão incorretos ou vazios.

Se os campos não possuírem nenhum erro será efetuada uma requisição do tipo *HTTP POST* para o *endpoint* de criação de anúncio. O *back-end* orá receber a requisição e efetuara o *upload* das imagens, vídeos e documentos, caso existirem.

Depois de arquivadas as mídias da requisição, o *controller* responsável por efetuar a criação de um anúncio irá validar os dados recebidos, se os dados possuírem alguma inconsistência será retornada uma mensagem de erro para o *front-end* que será apresentada para o usuário.

Se os dados não possuírem inconsistências, será feita a criação de um novo objeto do tipo anúncio a partir dos dados recebidos na requisição e será feita a persistência deste objeto na base de dados. Após a adição do novo anúncio na base de dados, será retornada uma reposta com *HTTP status code* 201 (*created*) e com o anúncio criado para o *front-end*, que apresentará uma mensagem de sucesso na criação do anúncio para o usuário. Na Figura 5.4 pode-se visualizar o diagrama de atividades referente aos passos descritos anteriormente.

Figura 5.4 – Diagrama de atividades para o cadastro de anúncio no sistema

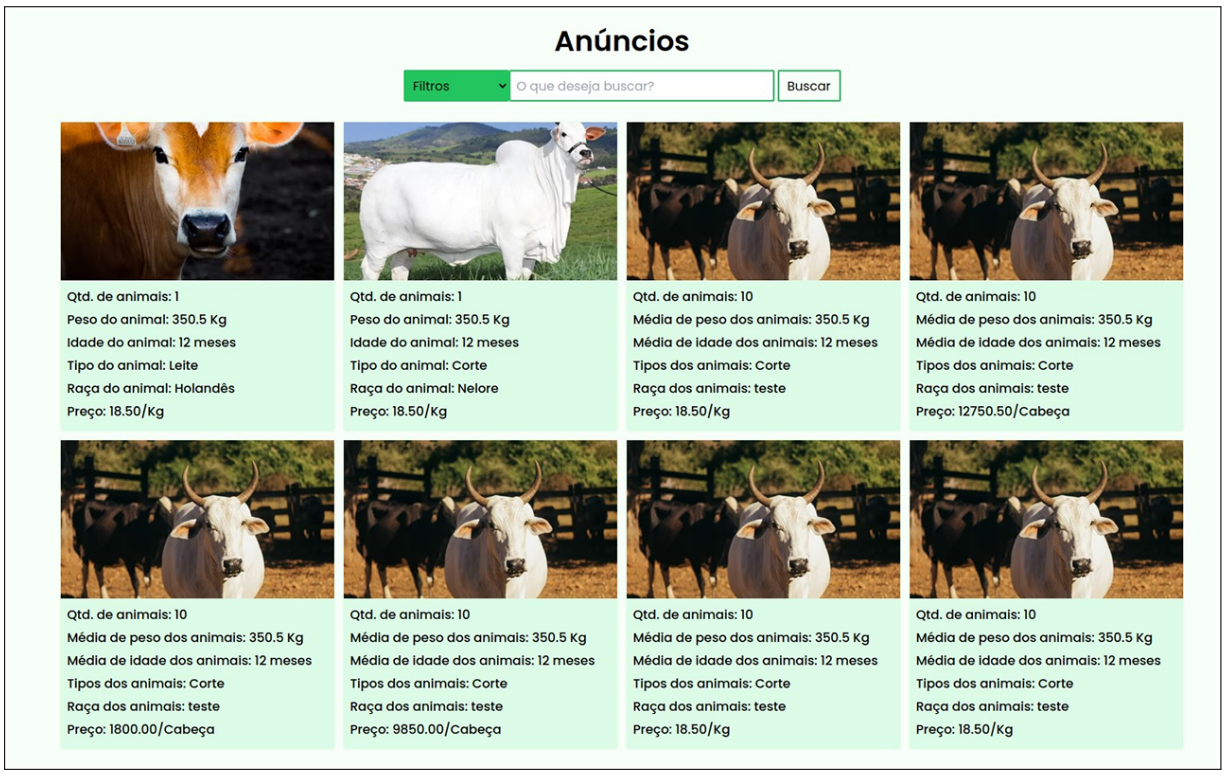


Fonte: Autor do trabalho

5.3 CLASSIFICAÇÃO DE ANÚNCIO

Toda vez que um usuário desejar efetuar a classificação de anúncio ele deverá inicialmente acessar a página inicial da plataforma, onde serão apresentados todos os anúncios cadastrados na base de dados conforme pode-se visualizar na Figura 5.5

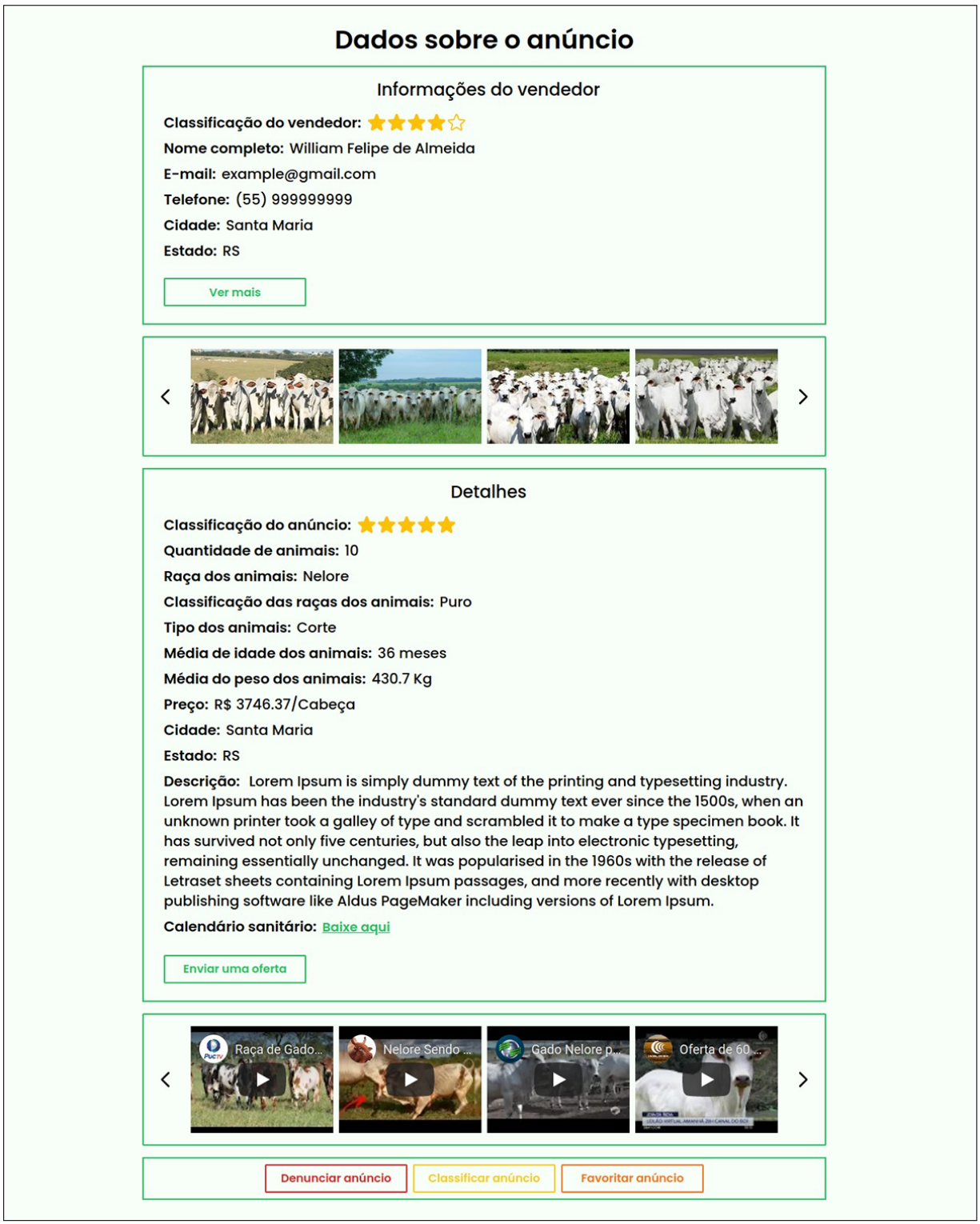
Figura 5.5 – Tela inicial do sistema



Fonte: Autor do trabalho

Na tela inicial o usuário deverá selecionar o anúncio que deseja classificar e clicar sobre ele. Está ação de clique irá redirecionar o usuário para a página com os todos os detalhes do anúncio, conforme pode-se observar na Figura 5.6. Na tela com os detalhes do anúncio o usuário poderá visualizar as informações do usuário, os detalhes do anúncio (fotos, descrição, vídeos, calendário sanitário, etc). Atrávés da tela de detalhes também será possível acessar o perfil do vendedor e enviar uma oferta.

Figura 5.6 – Tela de detalhes do anúncio



Fonte: Autor do trabalho

Para classificar o anúncio o usuário deverá clicar no botão classificar que encontra-se no final da página de detalhes. Após o clique o *front-end* irá efetuar uma requisição para o *back-end* para validar se o usuário está logado ou não na plataforma, caso o usuário não esteja logado ele será redirecionado para a página de *login* da aplicação. Se o usuário

logado for o “dono” do anúncio, o botão para a classificação será desativado.

Se o usuário estiver logado e o anúncio não pertencer a ele, será aberto um pop-up para que o usuário posso classificar o anúncio. Depois de classificado o anúncio, o *front-end* irá enviar uma requisição para o *back-end* através do *endpoint* de classificação.

O *controller* responsável por executar o *endpoint* de classificação, irá efetuar uma consulta a base de dados para verificar se o usuário já classificou o anúncio em algum momento, caso a base de dados retorne o documento correspondente a consulta, o mesmo só será atualizado. Se não retornar nenhum documento da base de dados referente a classificação, será efetuada a criação de um novo documento na base de dados referente a está ação.

Depois de criada a classificação do usuário para o anúncio, o *controller* efetuara a recuperação de toda as classificações destinadas ao anúncio e calculará a média de todas as classificações, atualizando a classificação geral.

Após a atualização da classificação geral do anúncio, o *controller* irá buscar por todos os anúncios do vendedor na base de dados e efetuará o calculo da média das classificações de todos os anúncios pertencentes a ele, atualizando assim a classificação geral do vendedor.

6 CONCLUSÃO

A plataforma desenvolvida neste trabalho conseguiu corrigir diversas falhas encontradas no processo de negociação de bovinos, desde a criação de um anúncio até a busca por um produto desejado.

Toda a construção do sistema foi feita utilizando *JavaScript*, suas bibliotecas e *frameworks*, algo que pode ser vantajoso para futuras implementações, pois o NPM possui diversas funcionalidades já implementadas que podem ser acopladas a plataforma facilmente.

Outras vantagens da construção do sistema é utilização do padrão de projetos MVC juntamente com os padrões e rotinas de uma API, que permitem que o *front-end* execute de forma independente do *backe-end*, possibilitando que manutenções executadas no *front-end*, não alterem o funcionamento das outras camadas do sistema. A criação do *back-end* utilizando API possibilita a integração com diversos clientes, desde uma aplicação *mobile*, até a criação de uma aplicação *desktop*.

Efetuando um comparativo com as plataformas relacionadas com o trabalho apresentadas no Capítulo 3, pode-se perceber que a plataforma criada conseguiu corrigir as principais deficiências e carências que as principais concorrentes do mercado possuem, desde a criação detalhada de um anúncio, até classificação de um usuário.

Na Tabela 6.1 consegue observar as principais vantagens da plataforma. A principal diferença da plataforma do trabalho para as concorrentes que atuam diretamente com o setor de bovinos, é a total liberdade que o usuário possuirá ao fazer seu o uso, podendo criar e editar suas informações e os seus anúncios conforme desejar.

Outro ponto forte da plataforma é a apresentação de classificações para usuários e anúncios. Todo processo de classificação é feito a partir dos *feedbacks* dos usuários, o que acaba proporcionando uma maior credibilidade na hora de anunciar na plataforma.

A principal desvantagem da plataforma em relação a sua concorrente (OLX) é a ausência de um sistema de pagamentos, algo que pode ser implementado futuramente na plataforma, permitindo que toda a negociação ocorra por meio dela.

Pode-se então concluir que o trabalho conseguiu cumprir com o principal objetivo de facilitar e agilizar o processo de negociação de bovinos, permitindo a centralização dos compradores e vendedores em um único lugar.

Tabela 6.1 – Comparativo com as plataformas concorrentes do mercado

	App gado bom	Boi na net	OLX	Plataforma do trabalho
Atua exclusivamente com bovinos	**Não**	**Sim**	**Não**	**Sim**
Permite criar, editar e excluir perfil	**Não**	**Não**	**Sim**	**Sim**
Permite criar, editar e excluir um anúncio	**Não**	**Não**	**Sim**	**Sim**
Permite a classificação de um anúncio	**Não**	**Não**	**Não**	**Sim**
Permite favoritar um anúncio	**Não**	**Não**	**Sim**	**Sim**
Permite denunciar um anúncio	**Não**	**Não**	**Sim**	**Sim**
Possui chat para conversação	**Não**	**Não**	**Sim**	**Sim**
Possui campo de busca por anúncios	**Não**	**Não**	**Sim**	**Sim**
Possui filtros de busca	**Não**	**Não**	**Sim**	**Sim**
Possui formulários para cadastro de anúncios detalhados	**Não**	**Não**	**Não**	**Sim**
Possui anúncios padronizados	**Sim**	**Não**	**Não**	**Sim**
Apresenta os dados dos vendedores e compradores	**Não**	**Não**	**Sim**	**Sim**
Possui sistema de pagamento	**Não**	**Não**	**Sim**	**Não**

Fonte: Autor do trabalho

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, P. **E-Commerce brasileiro cresce 75% no mês de maio segundo a Mastercard SpendingPulse**. Mastercard, 2020. Acesso em 12 jan. 2022. Disponível em: <https://www.mastercard.com/news/latin-america/pt-br/noticias/comunicados-de-imprensa/pr-pt/2020/junho/e-commerce-brasileiro-cresce-75-no-mes-de-maio-segundo-mastercard-spendingpulse/>.

BOLFE Édson L. et al. **Agricultura digital no Brasil**: tendências, desafios e oportunidades: resultados de pesquisa online. Campinas: Embrapa, 2020. 44 p. Disponível em: <https://www.embrapa.br/agropensa/produtos-agropensa>.

CANALTECH. **OLX Brasil**. Canaltech, 2021. Acesso em 4 jul. 2022. Disponível em: <https://canaltech.com.br/empresa/olx-brasil/>.

CHAUSSARD, C. **Por que tantas lojas querem migrar para marketplaces?** Flexy, 2019. Acesso em 12 jan. 2022. Disponível em: <https://blog.flexy.com.br/marketplaces-no-brasil/>.

CUNHA, F. **Requisitos funcionais e não funcionais**: O que são? Mestres da Web, 2022. Acesso em 22 mar. 2022. Disponível em: <https://mestresdaweb.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao/>.

DOGLIO, F. **REST API Development with Node.js: Manage and Understand the Full Capabilities of Successful REST Development**. 2nd ed. ed. Apress, 2018. ISBN 9781484237151; 1484237153; 9781484237144; 1484237145. Disponível em: <https://libgen.is/book/index.php?md5=3145F4A7CA209EB1556D9442D14E7645>.

EUGÊNIO, M. **Como funciona a OLX? Guia COMPLETO para criar anúncios**. dlojavirtual, 2021. Acesso em 4 jul. 2022. Disponível em: <https://www.dlojavirtual.com/tecnologia/plataformas/voce-sabe-como-funciona-o-olx-confira/>.

INDIO, C. **Pandemia fecha 39,4% das empresas paralisadas, diz IBGE**. Agencia Brasil, 2020. Acesso em 22 mar. 2022. Disponível em: <https://agenciabrasil.ebc.com.br/economia/noticia/2020-07/pandemia-fecha-394-das-empresas-paralisadas-diz-ibge>.

JASRAJ. **MERN Stack**. GeeksforGeeks, 2021. Acesso em 22 mar. 2022. Disponível em: <https://www.geeksforgeeks.org/mern-stack/>.

LIMA, V. **Quais São Os 10 Maiores Marketplaces Do Brasil Em 2022: Lista Definitiva**. Magis5, 2021. Acesso em 22 mar. 2022. Disponível em: <https://magis5.com.br/ranking-maiores-marketplaces-do-brasil/>.

MACORATTI, J. C. **Padrões de Projeto - Design Patterns**. Macoratti.net, 2004. Acesso em 22 mar. 2022. Disponível em: <https://www.macoratti.net/vb_pd1.htm>.

MARCIO. **Padrão MVC - Java Magazine**. DevMedia, 2011. Acesso em 22 mar. 2022. Disponível em: <https://www.devmedia.com.br/padrao-mvc-java-magazine/21995>.

MONGODB. **What is the MERN Stack?** MongoDB, 2022. Acesso em 22 mar. 2022. Disponível em: <https://www.mongodb.com/mern-stack>.

NIELSEN, E. . **Ebit | Nielsen Webshoppers 44**. Webshopers, 2021. Acesso em 22 mar. 2022. Disponível em: <https://eyagencia.com.br/wp-content/uploads/2021/09/Webshoppers_44-relatorio-2021-resultados-ecommerce-ebit.pdf>.

OLX, E. **OLX Pay: como funciona e o que é?** OLX dicas, 2021. Acesso em 4 jul. 2022. Disponível em: <https://dicas.olx.com.br/aquinaolx/olx-pay-como-funciona-o-que-e-tutorial/>.

PAGAR.ME. **Marketplace: tudo o que você precisa saber sobre vender nessa plataforma**: E-commerce. Pagar.me, 2021. Acesso em 22 mar. 2022. Disponível em: <https://pagar.me/blog/marketplace/>.

____. **O que é e-commerce: guia completo de como funciona e como criar um**: E-commerce. Pagar.me, 2021. Acesso em 22 mar. 2022. Disponível em: <https://pagar.me/blog/o-que-e-ecommerce/>.

PINHO, F. G. **Marketplaces crescem na pandemia e se especializam com vendas para nichos**. Folha de São Paulo, 2021. Acesso em 12 jan. 2022. Disponível em: <https://www1.folha.uol.com.br/mpme/2021/06/marketplaces-crescem-na-pandemia-e-se-especializam-com-vendas-para-nichos.shtml>.

PIRES, M. **As 10 principais plataformas de e-commerce B2B no Brasil**. B2list, 2021. Acesso em 22 mar. 2022. Disponível em: <https://b2list.com.br/principais-plataformas-ecommerce-b2b/>.

PRESSMAN, R. S. **Engenharia de Software**: Uma abordagem profissional. 7a edição. ed. AMGH Editora Ltda, 2011. ISBN 978-85-8055-044-3. Disponível em: <https://libgen.is/book/index.php?md5=54A4A07EB08DA31FC2B41A180861CD6B>.

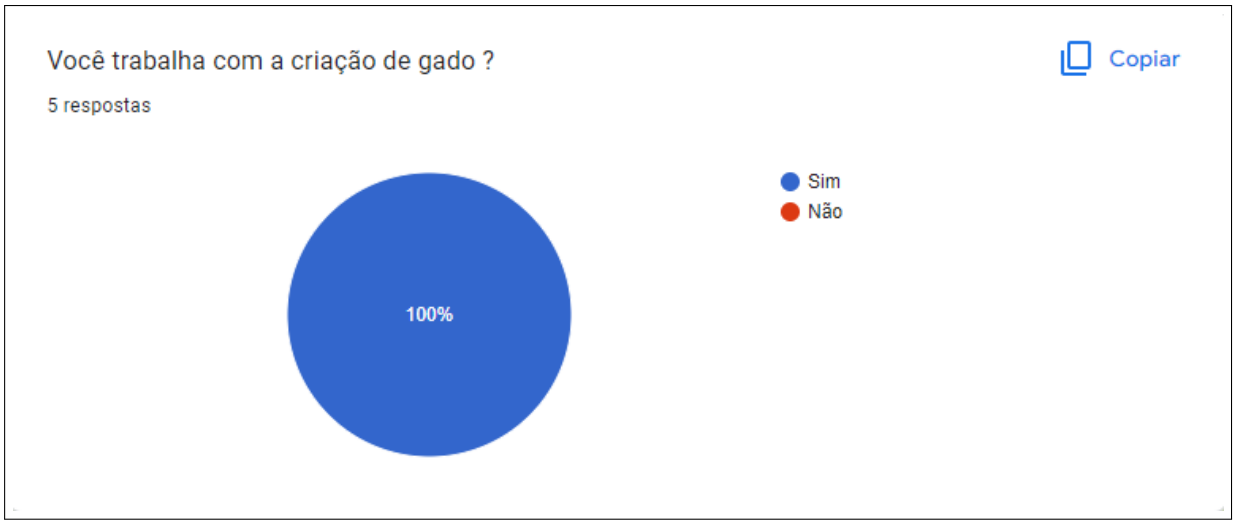
REDHAT. **API REST**. RED HAT, 2020. Acesso em 12 jan. 2022. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>.

TAILWINDCSS. **Get started with Tailwind CSS**. Tailwindcss, 2022. Acesso em 22 mar. 2022. Disponível em: <https://tailwindcss.com/docs/installation>.

ZUCHER, V. **O que é padrão MVC? Entenda arquitetura de softwares!** le wagon, 2020. Acesso em 22 mar. 2022. Disponível em: <https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc>.

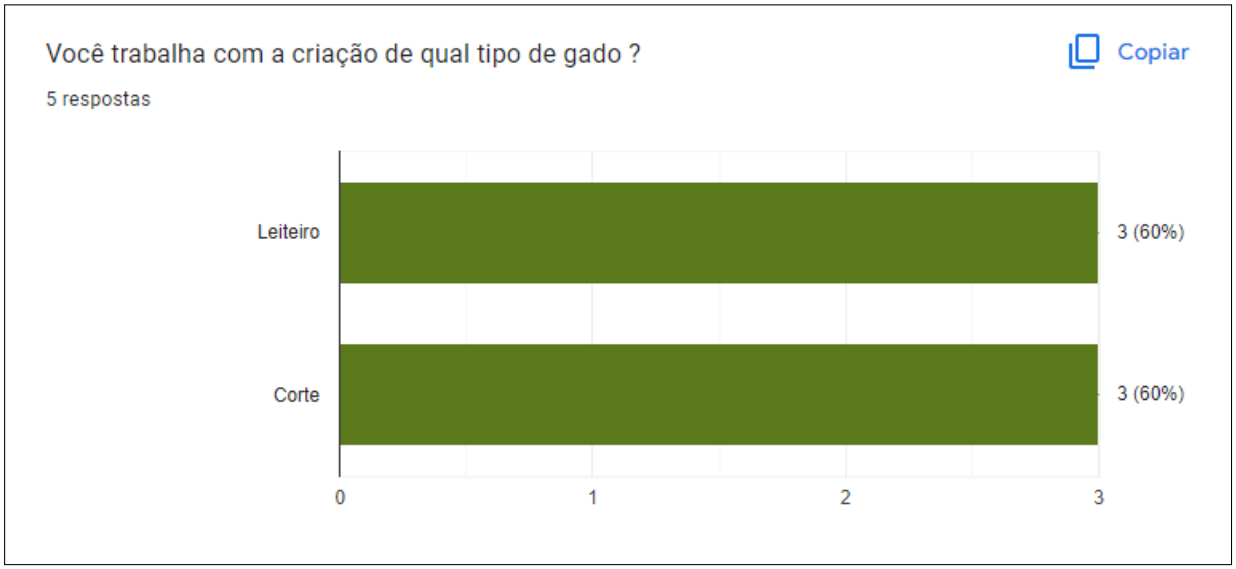
ANEXO A – FORMULÁRIO UTILIZADO DURANTE O LEVANTAMENTO DE REQUISITOS

Figura A.1 – Pergunta 1 do formulário



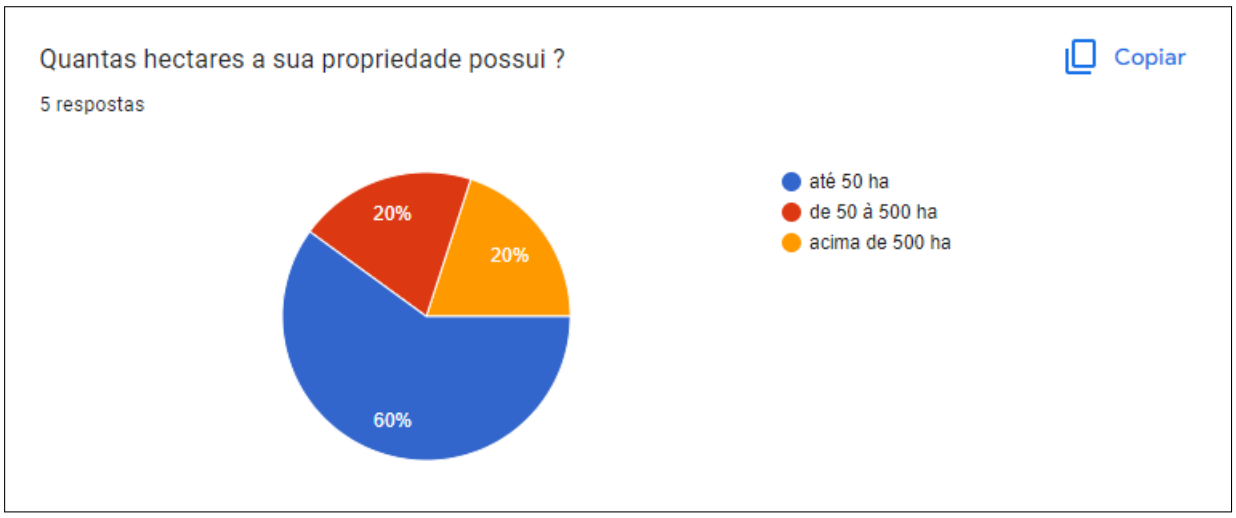
Fonte: Autor do trabalho

Figura A.2 – Pergunta 2 do formulário



Fonte: Autor do trabalho

Figura A.3 – Pergunta 3 do formulário

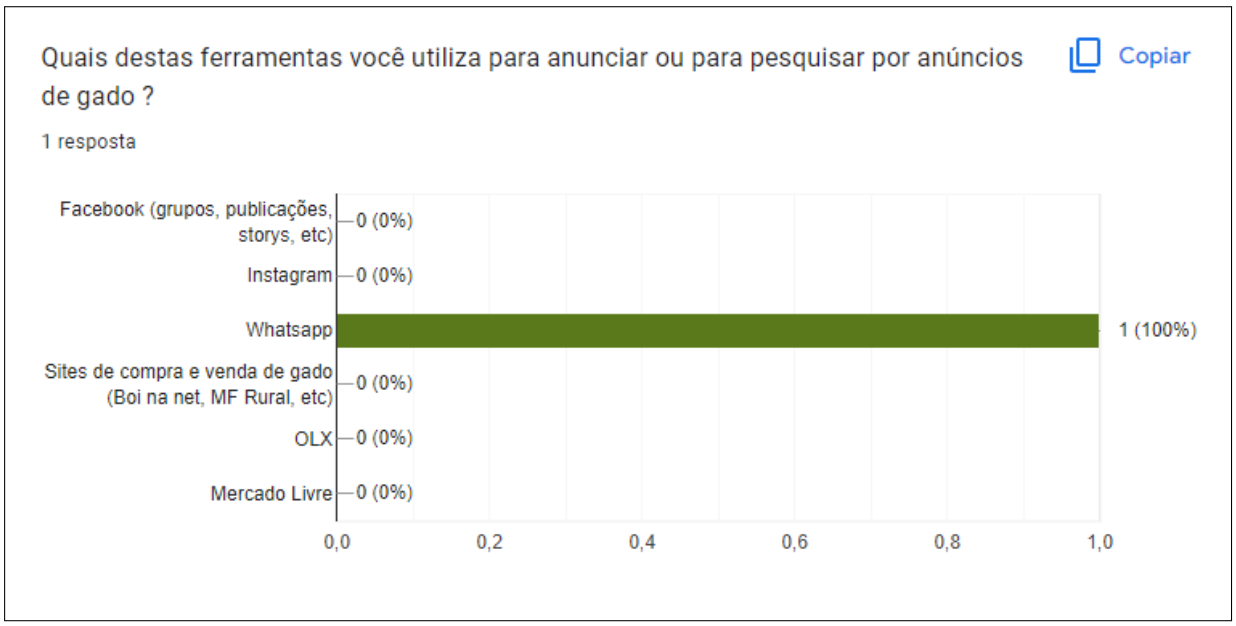


Fonte: Autor do trabalho

Figura A.4 – Pergunta 4 do formulário

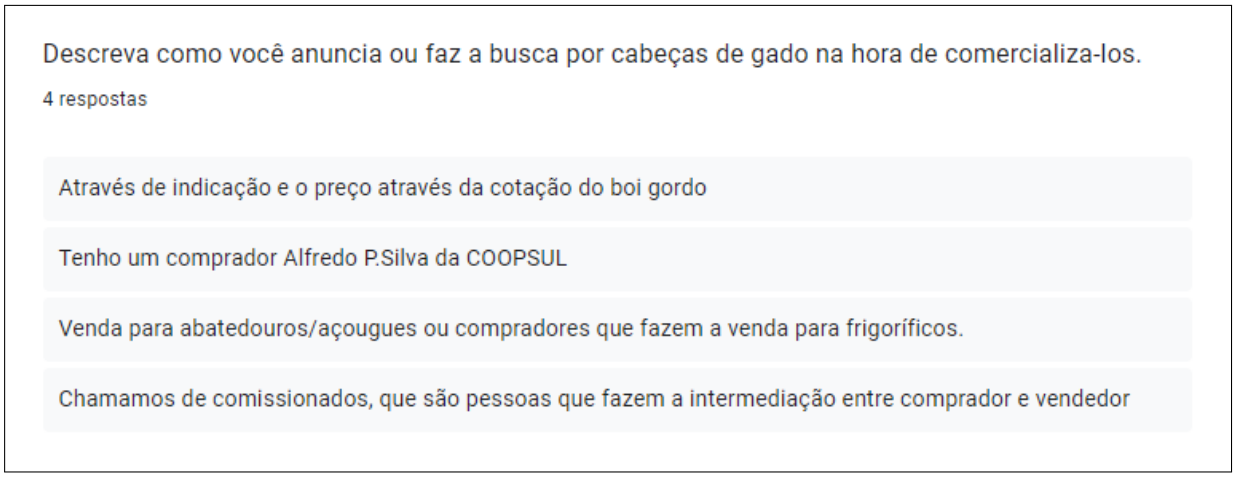
Fonte: Autor do trabalho

Figura A.5 – Pergunta 5 do formulário



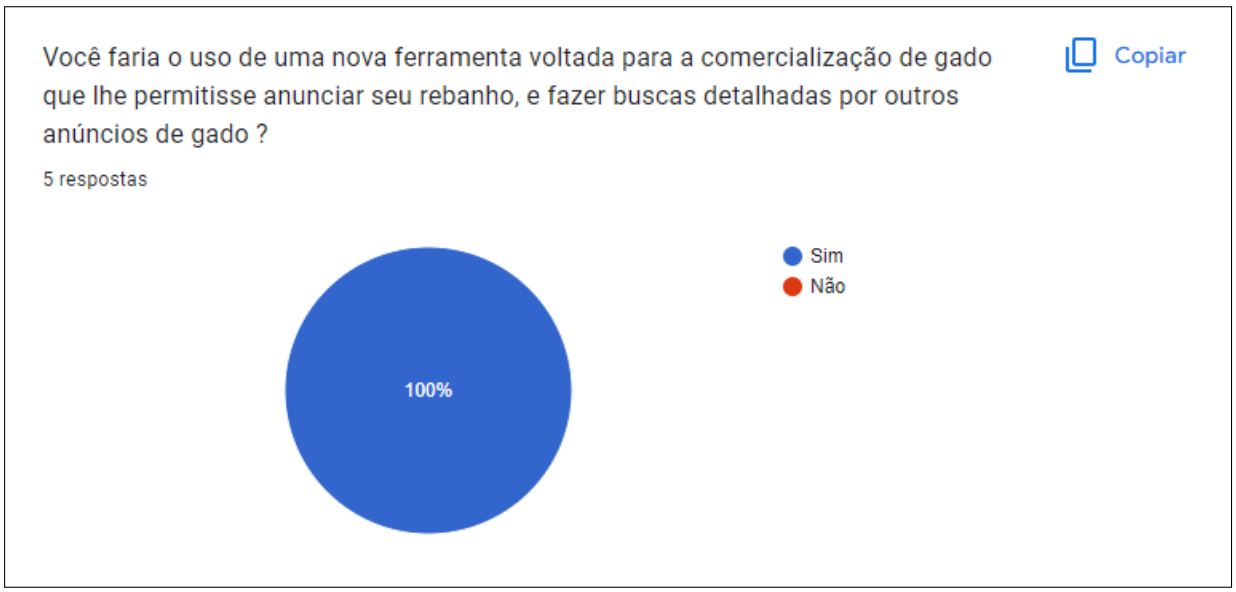
Fonte: Autor do trabalho

Figura A.6 – Pergunta 6 do formulário



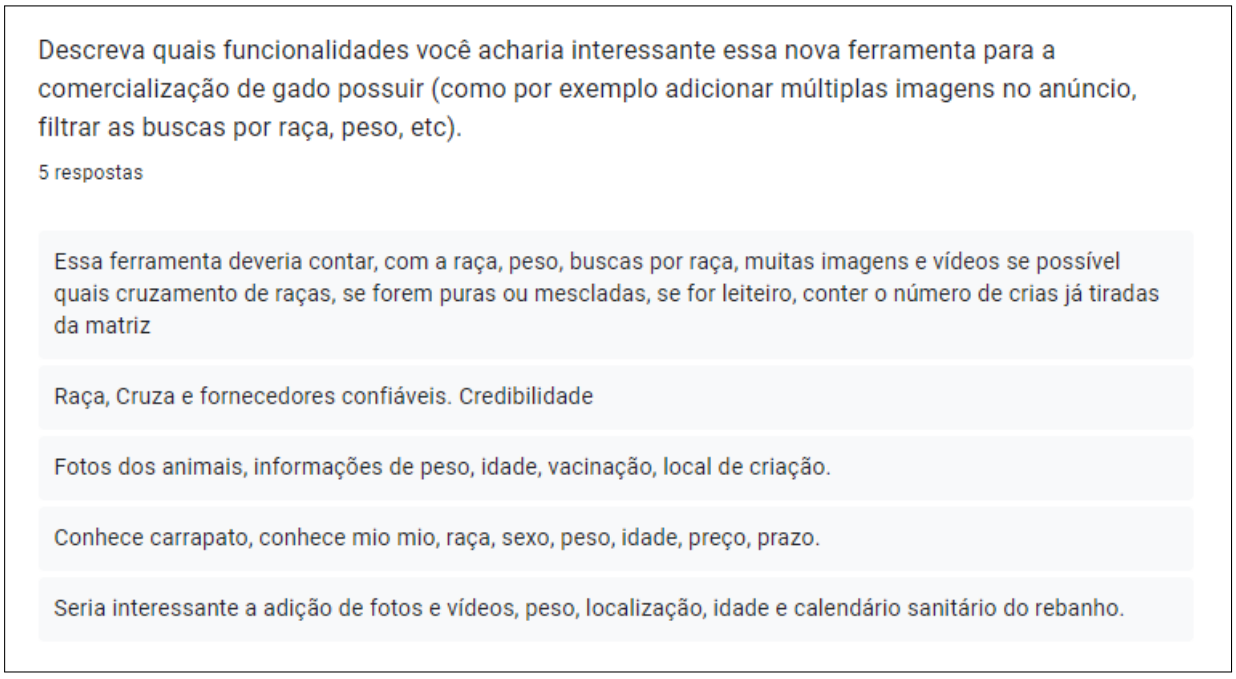
Fonte: Autor do trabalho

Figura A.7 – Pergunta 7 do formulário



Fonte: Autor do trabalho

Figura A.8 – Pergunta 8 do formulário



Fonte: Autor do trabalho