UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ANTONIO EDUARDO MOREIRA

PLATAFORMA PARA COMERCIALIZAÇÃO DE AVES CAIPIRAS COM RASTREABILIDADE

GUARAPUAVA

2022

ANTONIO EDUARDO MOREIRA

PLATAFORMA PARA COMERCIALIZAÇÃO DE AVES CAIPIRAS COM RASTREABILIDADE

Platform for Commercialization of Free-range Poultrys with Traceability

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Sistemas para Internet do Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Andres Jessé Porfirio

GUARAPUAVA

2022

ANTONIO EDUARDO MOREIRA

PLATAFORMA PARA COMERCIALIZAÇÃO DE AVES CAIPIRAS COM RASTREABILIDADE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnológo em Sistemas para Internet do Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná (UTFPR).

Data da aprovação: 13/junho/2022

Prof. Andres Jessé Porfirio
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Diego Marczal
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Dênis Lucas Silva
Mestre
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

GUARAPUAVA
2022

ANTONIO EDUARDO MOREIRA

PLATAFORMA PARA COMERCIALIZAÇÃO DE AVES CAIPIRAS COM RASTREABILIDADE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnológo em Sistemas para Internet da Universidade Tecnológica Federal do Paraná (UTFPR).

Data da aprovação: 13/junho/2022

Prof. Andres Jessé Porfirio
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Diego Marczal
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Dênis Lucas Silva
Mestre
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

GUARAPUAVA

2022

Dedico este trabalho a Universidade Tecnológica Federal do Paraná - Campus Guarapuava e aos professores e servidores da instituição.

AGRADECIMENTOS

Agradeço primeiramente a minha família por me incentivar a fazer faculdade e sempre me apoiarem.

Agradeço aos professores da Universidade Tecnológica Federal do Paraná, em especial o meu orientador Prof. Dr. Andres Jessé Porfirio, que contribuiu para realização desse trabalho, além de outros professores.

Ei, irmão, nunca se esqueça: na guarda, guerreiro, levanta a cabeça. Levanta a cabeça truta, onde estiver seja lá como for. Tenha fé porque até no lixão nasce flor. (MCS, Racionais, 2002).

RESUMO

O agronegócio é um dos setores mais fortes do Brasil, o país é o maior exportador de carne de frango do mundo, o setor sempre teve grande presença nos números do PIB do país. A avicultura industrial é um dos setores do agronegócio que possui diversos softwares e tecnologias de apoio a diferentes processos, porém, quando se trata da avicultura caipira, que costuma ser gerida por grupos familiares, existem algumas deficiências quanto à informatização. As aves caipiras, se comparadas às industriais, possuem vantagens de sabor de carne e valor de mercado, um macho da raça Índio Gigante, por exemplo, pode custar até cem mil reais. Devido ao valor elevado das aves caipiras, os criadores desses animais precisam de segurança e transparência na comercialização. Atualmente não existe uma plataforma dedicada à comercialização de aves caipiras. Sendo assim, as compras/vendas desses animais normalmente são feitas informalmente por meio de aplicativos de comunicação, esses aplicativos não garantem segurança aos usuários no momento da compra, oportunizando negociações fraudulentas. A plataforma apresentada neste trabalho tem recursos que permitem que os usuários possam registrar informações dos animais cadastrados, permitindo assim consultar a sua rastreabilidade. Em um *marketplace* (comércio online) convencional, quando um animal é vendido, perde-se o registro dele, enquanto que na plataforma apresentada, é armazenado todo histórico de criadores que foram donos do animal, e também é possível fazer registros de medidas e vacinação do animal a fim de passar mais segurança ao comprador.

Palavras-chave: aves caipiras; marketplace; indio gigante; rastreabilidade; gerenciamento.

ABSTRACT

Agribusiness is one of the strongest sectors of Brazil's economy, the country is the biggest chicken meat exporter in the world, the sector always had a big presence in the country's GDP. Industrial poultry farming is an agribusiness sector that has many software and technologies to help different processes, however, in the country poultry farming, which usually is managed by family groups, exists an informatization deficiency. The free-range birds, if compared to the industrial birds, have advantages of meat flavor and market value, a male of Índio Gigante breed, for example, it may even cost one hundred thousand reais. Due to high value, the breeders of these animals need security and transparency in the commercialization. Currently does not exist a platform dedicated to the commercialization of free-range birds. Therefore, the sales and purchases of these animals normally are done informally through communication applications, these applications do not give security to the users in the purchases, providing opportunities for fraudulent negotiations. The platform presented in this work has features that keep track of the animals registered on the platform. In a conventional marketplace, when an animal is sold, all his registers are lost, while in the proposed platform, the history of bird breeders is stored, and also is possible to make registers about the animal in order to give more security to the buyer.

Keywords: free-range birds; marketplace; indio gigante; traceability; management.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Interface de Aplicação Programada, do inglês *Application Programming Interface*
BFF	*Back-end* para *Front-end*, do inglês *Back-end for Front-end*
HTTP	Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol*
JSON	Objeto de Notção JavaScript, do inglês *JavaScript Object Notation*
JWT	JSON Web Token
MVP	Produto Viável Mínimo, do inglês *Minimum Viable Product*
NPM	Gerenciador de Pacote Node, do inglês *Node Package Manager*
PIB	Produto Interno Bruto
PR	Solicitação de Recebimento, do inglês *Pull Request*
RFC	Pedido de Comentários, do inglês *Request for Comments*
SSR	Renderização do Lado do Servidor, do inglês *Server Side Rendering*
VPN	Rede privada virtual, do inglês *Virtual Private Network*

SUMÁRIO

1 INTRODUÇÃO

1.1 DESCRIÇÃO GERAL

Em 2020, o agronegócio teve participação em 26,6% no Produto Interno Bruto (PIB) brasileiro, que equivale a quase 2 trilhões de reais. Mais especificamente, no setor agropecuário foi atingido recorde de crescimento do PIB gerado. Além do agropecuário, para indústria de rações e outros insumos do agronegócio também houve crescimento significativo (CNA, 2021).

De acordo com Rodrigues (2005), no início da primeira década do século, o Brasil já tinha grande potencial no ramo do agronegócio. Em meados de 2004, o agronegócio já representava mais de 30% do PIB nacional, dentre sua áreas, a produção e exportação de carnes de frango e boi sempre tiveram números expressivos, e ainda em 2004 o Brasil já era o maior exportador de carne bovina e carne de frango. Até os dias de hoje o Brasil se mantém como o maior exportador de carne de frango (EMBRAPA, 2021b) e de carne bovina (EMBRAPA, 2021a). Ainda de acordo com Rodrigues, a criação de tecnologias voltadas para o agronegócio agrega mais valor ao setor. O Brasil, além de ser um grande exportador do agronegócio, sempre teve um grande potencial tecnológico para o avanço da área. Uma vez que o país evolua tecnologicamente no setor, o país passa a ter mais potencial no mercado mundial, podendo até se tornar uma referência com novas tecnologias.

No caso das aves, existem diferentes raças para criação, algumas delas são mais comuns para criação industrial de grande escala, enquanto que outras costumam ser criadas em menor escala. As aves caipiras são produzidas em menor escala e possuem vantagens genéticas se comparadas às aves industriais, além disso, tem sabor considerado melhor. Dentre as raças de aves caipiras, existe a do Índio Gigante, que foi desenvolvida no Brasil e possui dupla aptidão, sendo elas: ornamental (ou ave de elite) e de corte. As aves caipiras destinadas a elite possuem alto valor agregado de mercado, com fêmeas custando de duzentos a dez mil reais e machos podendo atingir de quinhentos até cem mil reais. As aves de elite podem ser usadas como matriz ou reprodutor para criação de pintinhos, devido sua genética que permite que os animais comecem a reproduzir mais cedo que outras raças. Já no âmbito do corte, a raça do Índio Gigante, por apresentar desenvolvimento de carcaça avantajado, logo próximo dos três meses de vida já está apta para ser comercializada como frango de abate, isso porque sua genética favorece o ganho de peso em menos tempo em relação a outras raças de galinhas, tornando-o uma ótima opção para a agricultura familiar (DEUS, 2019).

Mesmo sendo uma raça com tantas oportunidades, a comunidade de criadores de Índio Gigante passa por dificuldades organizacionais. Dentre estas, a falta de algum mecanismo seguro na hora de comprar e vender o animal é um problema de vários criadores, oportunizando a ocorrência de negociações fraudulentas.

Diante do exposto é apresentada a plataforma CIG, que tem recursos para facilitar a busca por aves caipiras, e permitir que os usuários realizem o trabalho de gestão e acompanha-

mento dos animais. Desta forma, passando transparência e segurança na comercialização das aves Índio Gigante através de registros de medida, peso, fotos e vacinas do animal que podem ser feitas na plataforma. Além disso, quando um criador compra uma ave de outro criador na plataforma, todo o histórico de atualizações da ave é mantido e disponível para visualização dos criadores.

Até o momento, não existe uma plataforma específica para esse fim, logo, a CIG é a pioneira do mercado, e abre espaço para que criadores de aves caipiras comercializem as aves por meio de um canal de comunicação direto entre os criadores. Ressalta-se que existem plataformas como OLX (2021) e MercadoLivre (2021) que fornecem soluções para venda de produtos online, porém são plataformas para soluções mais genéricas que não abordam diretamente algumas necessidades do comércio de aves, como por exemplo, manter a rastreabilidade dos animais.

Ademais, as plataformas atualmente existentes, por não serem específicas para o contexto citado, pecam no sentido de não oferecerem recursos que inspirem segurança e confiabilidade nos criadores, tais como: a manutenção de registros sobre os animais, histórico de criadores que foram proprietários da ave, e permitir que as informações de um animal sejam atualizadas, mantendo um histórico de fotos, medidas e vacinas dos animais. Diante disso, este trabalho apresenta uma plataforma específica para o comércio de aves caipiras como, por exemplo, a raça do Índio Gigante. Com isso, pretende-se instrumentar os criadores aumentando a autonomia para expôr o seu trabalho de maneira digital, passando segurança aos compradores sem intermediadores.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver uma plataforma para comercialização de aves caipiras com rastreabilidade.

1.2.2 Objetivos específicos

- Criar uma aplicação capaz de registrar e gerenciar usuários;
- Criar uma aplicação capaz de registrar e gerenciar criatórios, aves e registros de aves;
- Criar uma aplicação capaz de registrar e gerenciar anúncios, perguntas de anúncios e respostas de perguntas;
- Criar uma aplicação capaz de registrar compras e visualizar vendas;
- Criar um sistema web com interface que permita que o usuário gerencie as informações do criatório, aves, anúncios e responda as perguntas dos anúncios;

- Criar um sistema web com interface que permita que o usuário compre aves e navegue pelos perfis dos criadores.

1.3 ORGANIZAÇÃO DO TRABALHO

O trabalho é organizado conforme segue: O Capítulo 2 apresenta os trabalhos correlatos; Na sequência, o Capítulo 3 mostra o referencial teórico; A metodologia é exposta no Capítulo 4; É apresentado o desenvolvimento no Capítulo 5; No Capítulo 6 são mostrados os resultados; Por fim, no Capítulo 7 são apresentadas as considerações finais.

2 TRABALHOS CORRELATOS

Este capítulo apresenta trabalhos relacionados ao sistema proposto, partindo da relação entre tecnologia e agronegócio, seguida da exposição de sistemas similares.

2.1 TECNOLOGIA E AGRONEGÓCIO

O trabalho de Vieira, Baccili e Delfino (2011) evidencia a importância da tecnologia aliada ao agronegócio. A tecnologia vem adentrando cada vez mais nos mais diferentes setores do mercado, a criação de plataformas digitais para nichos específicos pode servir de alavanca para economia do setor. Os autores mencionam que existem diversos softwares usados no ramo do agronegócio, que predominantemente é ocupado por produtores rurais e cooperativas agropecuárias, mas nenhum deles voltado para comércio online nichado para aves caipiras. De acordo com os autores, em 2008 já existiam mais de 100 empresas de software trabalhando dedicadamente para soluções tecnológicas do ramo do agronegócio, a presença da tecnologia da informação está cada vez mais presente no dia a dia dos criadores, e com a Internet ficou mais fácil quebrar barreiras culturais e proporcionar mais proximidade entre o produtor rural e a tecnologia. Dado o atual cenário de globalização, um alto índice de informação é disponibilizado à população, tornando difícil a filtragem de conteúdos de origem que realmente têm procedência, com isso, observa-se novamente a deficiência de busca de informação para o setor.

2.2 SISTEMAS SIMILARES

O MFRural (2004) é um portal voltado para o Agronegócio Brasileiro. Os usuários podem comprar e vender produtos, a proposta é ser como um mercado físico, onde é permitida a venda desde aeronaves/caminhões até animais. No portal existe uma categorização desses produtos a fim de servir como filtro para os usuários. As aves caipiras podem ser anunciadas no portal, porém, são tratadas como os demais produtos, não existe um tratamento específico.

O AnimalsForSale (2017) é um portal para comercialização de animais, sejam filhotes ou adultos. Todos animais são categorizados no portal, os usuários podem filtrar essas categorias na busca. Um ponto relevante do portal é a possibilidade de verificar se os criadores dos animais são criatórios registrados. Os criadores registrados têm uma página para mais informações do criatório que contem todos animais em oferta, porém não existe um sistema de acompanhamento de registros do animal e nenhum tipo de histórico na plataforma.

Existem várias plataformas voltadas ao comércio online que visam instrumentalizar os profissionais autônomos para expor seus produtos de maneira digital na Internet. Uma destas plataformas é a OLX (2021), que proporciona um ambiente de loja virtual, onde usuários podem anunciar produtos de diversos segmentos, categorizá-los, e outros usuários podem buscar para

comprar, podendo também fazer a utilização de filtros para busca. A plataforma trata-se de uma solução genérica para qualquer setor, não oferecendo recursos para o contexto das aves caipiras. Semelhante, o MercadoLivre (2021) é uma das empresas pioneiras do mercado, e também permite a criação de anúncios de produtos para venda. Os usuários vendedores têm acesso a um painel para gerenciamento da sua loja online, que agrega uma área para exposição de produtos, áreas destinadas ao esclarecimento de dúvidas dos compradores, e chat.

Considerando o exposto, até o momento da escrita deste trabalho não foi encontrado nenhum sistema que atenda às especificidades da comercialização de aves caipiras, tais como a manutenção do histórico dos animais (rastreabilidade) com gerência de detalhes dos criatórios. Com isso, ressalta-se a necessidade do desenvolvimento de um sistema específico para este contexto.

3 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico que fundamenta o presente trabalho. Inicialmente, é detalhada a metodologia ágil Kanban, em seguida é exposta a arquitetura modular adotada, e, por fim, é apresentado o Atomic Design.

3.1 METODOLOGIA ÁGIL KANBAN

Metodologias ágeis são formadas por um conjunto de regras que visam facilitar o processo de desenvolvimento utilizando padrões organizacionais. No desenvolvimento de software, a gestão das atividades é facilitada com o uso de metodologias ágeis de desenvolvimento. São exemplos: Scrum, Kanban, RUP e XP (SANTANA, 2017). Dentre elas, o presente projeto apoia-se no Kanban.

O termo “Kanban” vem do japonês, que significa “Cartão Visual”. Kanban é uma metodologia ágil que não possui iterações durante o desenvolvimento do projeto, não existe um tempo fechado para que as tarefas sejam realizadas, a entrega é sempre contínua durante todo período de desenvolvimento. A ideia do Kanban é ter uma lista de tarefas, chamada *backlog*, ordenada por priorização com base no valor gerado, assim, a equipe desenvolve as tarefas de acordo com a demanda (GENARI; FERRARI, 2016). O Kanban é recomendado para situações que podem ter muitas alterações ao longo do processo de desenvolvimento, não existem ritos e papéis de cada membro do time definidos, deixando mais aberto para necessidade de cada projeto.

O Kanban permite a criação de um quadro para auxiliar na visualização das tarefas que estão sendo realizadas. Exemplo de estrutura do quadro na Figura 1.

Figura 1 – Exemplo de Quadro Kanban.
Fonte: o autor.

No exemplo citado, a primeira coluna do quadro é destinada para o *backlog*, que são as tarefas que ainda não podem ser desenvolvidas por algum tipo de bloqueio, como dependências de outras tarefas ou falta de maturidade da ideia. A coluna “selecionada para desenvolvimento” é destinada para a lista de tarefas que podem ser realizadas, e previamente o desenvolvimento já foi refinado com a equipe. As demais colunas servem para fazer o acompanhamento do status da tarefa. O Kanban não define padrões para as colunas, logo a distribuição do quadro pode ser feita de acordo com a necessidade do projeto (BOEG, 2010).

3.2 ARQUITETURA MODULAR

Existem diferentes arquiteturas para o desenvolvimento de sistemas Web, este projeto fundamenta-se no modelo cliente-servidor (BERSON, 1996). Aplicações cliente-servidor podem ser construídas de forma modular, compostas por aplicações apartadas que se comunicam entre si. Isso permite a divisão de uma grande aplicação em pequenos sub-produtos, facilitando a entrega, manutenção e evolução contínua do sistema. A principal motivação desta arquitetura é permitir que cada módulo consiga operar praticamente de forma isolada, com um mínimo de acoplamento aos demais (LOUDON, 2018).

Uma arquitetura formada por componentes modulares tem aplicações rodando em *back-end* (lado servidor), responsáveis por fazer o armazenamento e gerenciamento de dados, e aplicações *front-end* (lado cliente), que se comunicam com as aplicações servidor para consumir e exibir esses dados diretamente no hardware do usuário. Uma aplicação *front-end* pode se comunicar com uma aplicação *back-end* através de diferentes protocolos, como por exemplo, o Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol* (HTTP), que permite a transferência de dados hipertexto (RIZO; SANTO, 2020).

As aplicações do *back-end* podem ser: aplicações modulares, trata-se de um aplicação com seu próprio contexto, mas acoplada a outras aplicações da arquitetura; microsserviços, aplicação que opera totalmente independente sem acoplação a outro serviço.

Nesse contexto, existem diferentes tecnologias para o desenvolvimento de aplicações *front-end* e *back-end*. A linguagem de programação JavaScript, muito usada no desenvolvimento de aplicações na plataforma Web, é uma linguagem nativamente interpretada pelos navegadores, e fracamente tipada. Originalmente projetada para uso em *front-end*, o JavaScript possui ferramentas que possibilitam o seu uso em *back-end*, atuando como uma solução flexível para ambos os cenários. Embora seja uma linguagem popular e versátil, o JavaScript possui limitações em relação à geração de documentação causada, principalmente, pela ausência de tipos. Diante disso, foi desenvolvido o TypeScript, uma linguagem de código aberto de autoria da Microsoft baseado em JavaScript. A construção de aplicações *front-end* pode ser apoiada por bibliotecas e *frameworks*, como por exemplo: React, Vue e Angular (MICROSOFT, 2012). Considerando aplicações *back-end*, também é comum o uso de diferentes linguagens e *frameworks*, por exemplo: NodeJS, Laravel, Ruby on Rails, entre outros.

No contexto *front-end*, o presente projeto fundamenta-se no React, uma biblioteca JavaScript mantida pelo Facebook[1] voltada para a criação de interfaces de usuário. O React trabalha com componentes de forma modular, facilitando a criação da arquitetura de aplicações e favorecendo a expansão dos projetos. Já no contexto *back-end*, destaca-se o NodeJS, um software que tem um mecanismo baseado na *engine* V8 do Chrome, capaz de rodar código JavaScript no servidor (NODEJS, 2009).

3.3 ATOMIC DESIGN

Existem padrões de desenvolvimento que visam facilitar a criação da interface do software. O Atomic Design é uma metodologia de desenvolvimento para criação de interfaces, um dos princípios desta metodologia é que a prototipação do software seja feita a partir de componentes. Em outras metodologias de design, é comum que sejam desenvolvidas telas para prototipação do software, mas no Atomic Design, a ideia é ter componentes que possam ser utilizados. Um componente pode ser qualquer elemento da interface, botões e campos de entrada de texto são exemplos de componentes minimalistas e reutilizáveis. A partir destes componentes minimalistas, podem ser criados componentes maiores, com mais complexidade e que podem tratar de um contexto específico do software, como por exemplo, um formulário de cadastro ou uma tabela para exibição de dados. De acordo com Frost (2016), na metodologia Atomic Design, esses componentes são categorizados como:

- **Átomos**, elementos mínimos que podem estar na interface;
- **Moléculas**, formadas por Átomos, que representam áreas do software;
- **Organismos**, agrupamento das Moléculas que norteiam algum fluxo para o usuário;
- **Templates**, esqueleto da diagramação dos Organismos, Moléculas e Átomos na Página (estrutura da Página);
- **Pages**, feitos a partir de Templates, é o esqueleto já populado com os Organismos, Moléculas e Átomos;

Existe uma hierarquia entre esses componentes, e essa hierarquia auxilia a evitar duplicidade de código no desenvolvimento da interface. Considerando o processo de desenvolvimento da aplicação Web proposta neste trabalho, os Átomos são os componentes primitivos (como botões), as Moléculas representam componentes compostos (como uma botão com imagem), os Organismos representam agrupamentos de componentes de um mesmo contexto (como um barra de botões), os Templates são representados pelos protótipos de tela, e, por fim, as Páginas são os elementos já implementados no ReactJS. A Figura 2 apresenta um exemplo de categorização de componentes no Atomic Design.

[1] https://www.facebook.com/

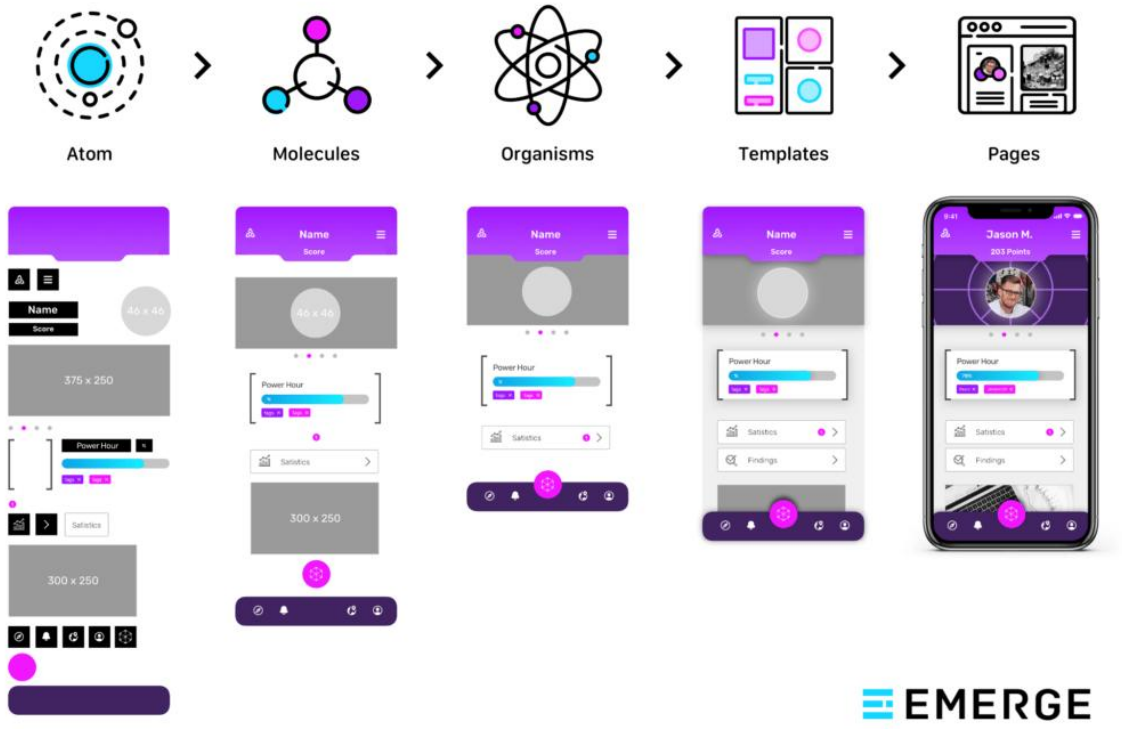

Figura 2 – Exemplo de Categorização de Componentes no Atomic Design.
Fonte: Emerge Interactive[a].

[a] https://www.emergeinteractive.com/

3.4 JSON WEB TOKEN

No desenvolvimento de software existem diferentes formas de fazer o controle de autenticação e autorização do usuário. O JSON Web Token (JWT), criado pela auth0 (2022a), é um dos métodos mais utilizados nas aplicações modernas.

Segundo o Pedido de Comentários, do inglês *Request for Comments* (RFC) oficial do JWT, o JWT permite que seja criado *tokens* que contém informações criptografadas. Para gerar um *token* JWT é necessário uma chave de assinatura, que depois do JWT gerado, pode ser utilizada para fazer a verificação do mesmo. Um *token* JWT pode armazenar informações no formato Objeto de Notção JavaScript, do inglês *JavaScript Object Notation* (JSON) e pode ser aberto sem a chave de assinatura, logo, não é interessante armazenar informações sensíveis num *token* JWT. Vale ressaltar que para fazer a verificação do JWT é necessário a chave de assinatura, pois é ela quem é usada para verificar se o *token* foi criado a partir dela, mas para apenas ler as informações do token, não é necessário. O JWT utiliza do algorítimo *HMAC* para fazer a assinatura e verificação do *token*. Por padrão, o algorítimo utilizado para fazer a criptografia do JWT é o *HS256*, mas é possível utilizar de alguma das outras opções disponíveis pela ferramenta. Um *token* JWT depois de aberto, nada é mais do que uma estrutura JSON que contém as informações que foram inseridas no *token*. Esse JSON é dividido entre a seção do

cabeçalho, que contém qual foi o algorítimo utilizado para criptografar o *token* e o corpo, que armazena as informações que foram salvas no *token*. (AUTH0, 2022b).

Na Figura 3 é possível ver um exemplo de um *token* JWT. No lado esquerdo da imagem, é exibido o *token* criptografado, e ao lado direito é esse mesmo *token* aberto, sendo mostrado as informações que contém nele. Nota-se que na parte de baixo da lateral direita, é possível ver a chave de assinatura que foi utilizado para gerar o *token*.

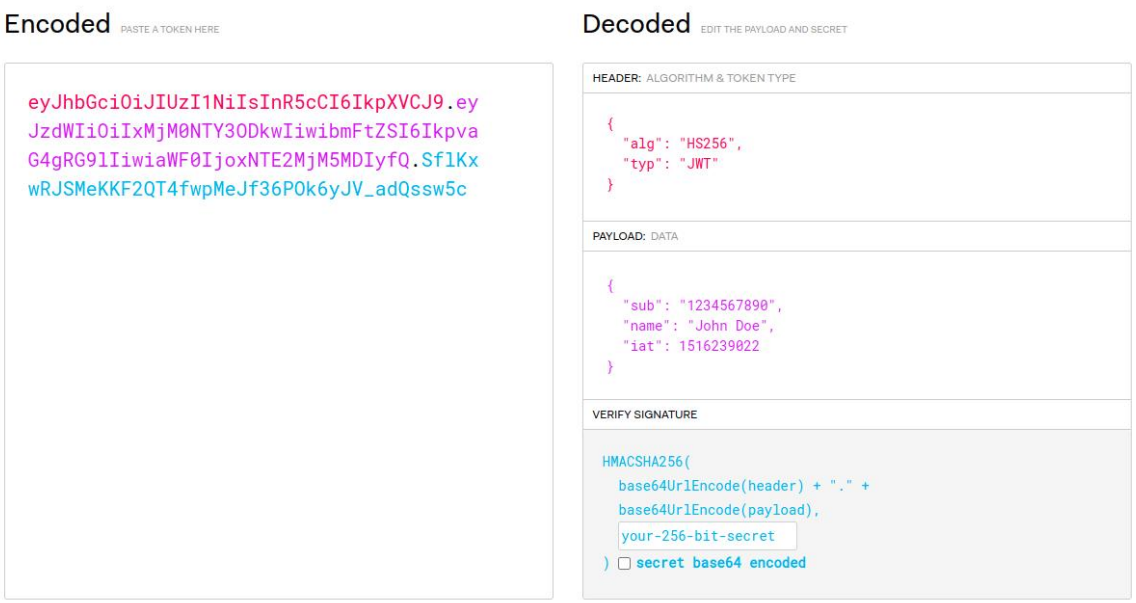

Figura 3 – Exemplo de *token* JWT
Fonte: auth0[a].

[a] https://auth0.com/pt

Na prática, um software que faz a autenticação e autorização via *token* JWT, gera o *token* do usuário no momento do login (autenticação), e após isso, esse *token* é utilizado para fazer a autorização desse usuário dentro das funcionalidades do software.

4 METODOLOGIA DE DESENVOLVIMENTO

Esse capítulo apresenta os procedimentos metodológicos que serão usados no desenvolvimento do projeto. Na seção 4.1 será apresentado o modelo de desenvolvimento que será aplicado.

4.1 MODELO DE DESENVOLVIMENTO

O sistema foi desenvolvido no modelo de Produto Viável Mínimo, do inglês *Minimum Viable Product* (MVP), que visa entregar o maior valor no menor espaço de tempo. De acordo com Guimarães e Silva (2021), o MVP é uma abordagem comprovada para a criação de produtos ou serviços, um produto mínimo é aquele que tem a solução mais rápida para resolver determinado problema a fim de validar o produto e começar o processo de aprendizagem de forma ágil. Ainda segundo os autores, a solução de um MVP não necessariamente precisa ser o caso ótimo em um primeiro momento, podendo ser evoluída com o tempo. Sendo assim, um produto ou serviço desenvolvido no modelo MVP passará por atualizações e mutações contínuas a fim de aperfeiçoar o produto desenvolvido.

Com isso, relacionando-se o MVP com o presente trabalho, a plataforma foi construída em etapas, as primeiras versões de desenvolvimento continham apenas recursos essenciais para o funcionamento, como o cadastro de aves e anúncios, e ao longo do desenvolvimento foram sendo construídos as demais funcionalidades da plataforma, como por exemplo a busca e compra de anúncios que foi um dos últimos recursos a ser implementado.

Foi utilizado do Github Actions para realizar o processo de integração e entrega contínua nos repositórios criados das aplicações que compõem a plataforma. Todos repositórios possuem configurados um processo de testes e varredura de código toda vez que uma nova Solicitação de Recebimento, do inglês *Pull Request* (PR) é feita. A integração com o Github Actions permitiu realizar a implantação da aplicação em um ambiente de testes diretamente a partir do repositório, para isso foi criado um ramo secundário nos repositórios que corresponde ao ambiente de testes.

Diante dos objetivos específicos, o seguinte *backlog* foi realizado durante o desenvolvimento da plataforma:

- Gestão de plantel e anúncios (*Backoffice*):
  - Criar uma Interface de Aplicação Programada, do inglês *Application Programming Interface* (API) para registrar e gerenciar usuários:
    - Configuração e criação da aplicação servidor;
    - Recurso de criação de usuário;
    - Recurso de autenticação de usuário;

    - Recurso de edição de informações de usuário;
    - Recurso para obter as informações de usuário;
- Criar uma API para registrar e gerenciar criatórios;
    - Configuração e criação da aplicação servidor;
    - Recurso de criação de criatório;
    - Recurso de edição de informações de criatório;
    - Recurso para obter as informações do criatório;
- Criar uma API para registrar e gerenciar aves;
    - Recurso de criação de ave;
    - Recurso de edição de informações de ave;
    - Recurso para obter as todas aves de um criatório;
    - Recurso para obter as informações de uma ave em específico;
    - Recurso para transferir ave de criatório;
- Criar uma API para registrar uma informação no histórico da ave;
    - Recurso de criação de um registro no histórico da ave;
    - Recurso de para obter o histórico de uma ave em específico;
- Criar uma API para interações no *Backoffice*:
    - Configuração e criação da aplicação servidor;
    - Recurso de edição de informações de criatório;
    - Recurso de criação de ave;
    - Recurso de edição de ave;
    - Recurso de transferência de ave;
    - Recurso de criação de um registro no histórico da ave;
    - Recurso de criação de um anúncio de ave;
    - Recurso de edição de informações de um anúncio de ave;
    - Recurso de criação de uma resposta em uma pergunta;
    - Recurso de cancelamento de uma negociação;
    - Recurso de confirmação de uma negociação;
    - Recurso de finalização de uma negociação;
- Criar uma API para exibição de conteúdo no *Backoffice*:
    - Recurso para obter informações da página de criatório;
    - Recurso para obter informações de uma ave;
    - Recurso para obter todas aves;
    - Recurso para obter todas vendas;

        - Recurso para obter todas compras;
    - Criar uma aplicação *front-end* para o *Backoffice*:
        - Configuração e criação da aplicação cliente;
        - Criar página principal;
        - Criar página para editar senha;
        - Criar página para editar informações do criatório;
        - Criar página para cadastro de ave;
        - Criar página para transferência de ave;
        - Criar página para cadastro de registro de ave;
        - Criar página para cadastro de anúncio;
        - Criar página para visualizar as informações do criatório;
        - Criar página para visualizar aves;
        - Criar página para visualizar vendas;
        - Criar página para visualizar compras;
- Plataforma para compras (*Marketplace*):
    - Criar uma API para registrar e gerenciar anúncios:
        - Configuração e criação da aplicação servidor;
        - Recurso de criação de comerciante;
        - Recurso de criação de anúncio;
        - Recurso de inativação de anúncio;
        - Recurso para obter todos anúncios de um comerciante;
        - Recurso de edição de informações de um anúncio;
        - Recurso para criação de uma pergunta no anúncio;
        - Recurso de criação de uma resposta em uma pergunta de anúncio;
    - Criar uma API para negociações:
        - Configuração e criação da aplicação servidor;
        - Recurso de criação de uma proposta;
        - Recurso de criação de evento na proposta;
        - Recurso para obter todas negociações (compras ou vendas) de um comerciante;
    - Criar uma API para interações no *Marketplace*:
        - Configuração e criação da aplicação servidor;
        - Recurso de criação de uma pergunta no anúncio;

    - Recurso de criação de uma proposta;
- Criar uma API para exibição de conteúdo no *Marketplace*:
    - Recurso para obter os anúncios;
    - Recurso para obter informações da página de criatório;
    - Recurso para obter informações de uma ave anunciada;
- Criar uma aplicação *front-end* para o *Marketplace*:
    - Configuração e criação da aplicação cliente;
    - Criar página para login;
    - Criar página para cadastro de usuário/criatório;
    - Criar página para visualizar os principais anúncios;
    - Criar página para busca de anúncios;
    - Criar página para visualizar um criatório;
    - Criar página para visualizar uma ave anunciada.

5 DESENVOLVIMENTO

Nesse capítulo serão apresentados os resultados dos processos e etapas realizados durante o desenvolvimento do projeto, tais como a organização do desenvolvimento e a arquitetura cliente-servidor, que atende todas necessidades do projeto.

A plataforma possui uma arquitetura modular, como descrito na seção 3.3, que permite uma fácil expansão e manutenção do projeto. A arquitetura do *back-end* é formada por oito aplicações servidores, que se comunicam entre si para funcionamento da plataforma, e dois bancos de dados. Enquanto que o *front-end* conta com seis aplicações clientes, onde todas elas fazem uso de um pacote de componentes da interface da plataforma. Arquitetura da plataforma é exemplificada na Figura 4.

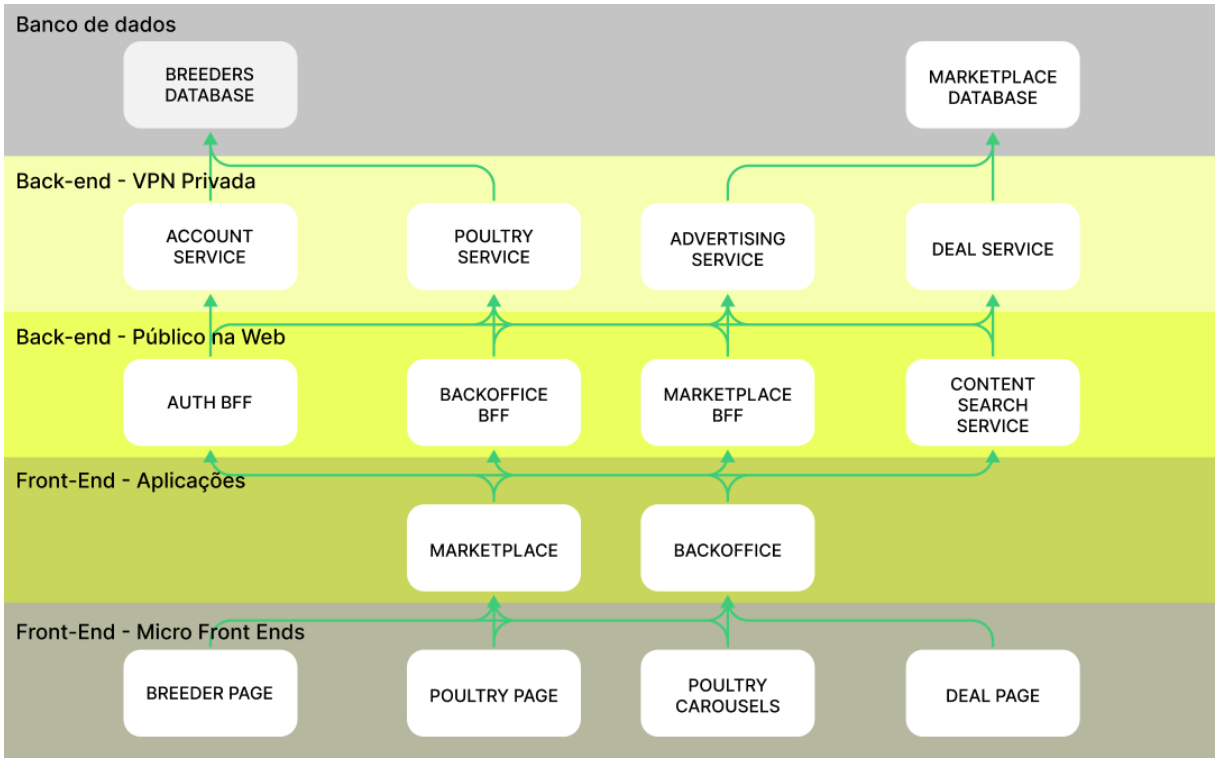

Figura 4 – Diagrama da Arquitetura.
Fonte: o autor.

5.1 ARQUITETURA *BACK-END*

O *back-end* da plataforma foi feito com NodeJS e Typescript, a arquitetura é formada por oito aplicações, dentre elas, quatro microsserviços e quatro aplicações modulares. Essas aplicações são APIs que se comunicam entre si através de requisições HTTP. Cada aplicação tem uma função dentro da arquitetura, que será apresentada na sequência, e todas funcionando juntas compõem a plataforma.

Para a criação das aplicações, foi elaborado um *template* de projeto[1]. Esse *template* é uma aplicação NodeJS, que contém um servidor *back-end* configurado com Express e Typescript. Esse servidor foi usado como base para as aplicações da plataforma, facilitando o desenvolvimento, além de padronizar toda estrutura e código das diferentes aplicações.

O *template* tem dois *pipelines* configurados, um de testes e outro de *lint*. Todas aplicações, por tratarem-se de APIs, possuem documentações para apresentar todas rotas HTTP, essas documentações podem ser acessadas através de uma rota específica na aplicação para visualização. Pode-se observar um exemplo de documentação na Figura 5.

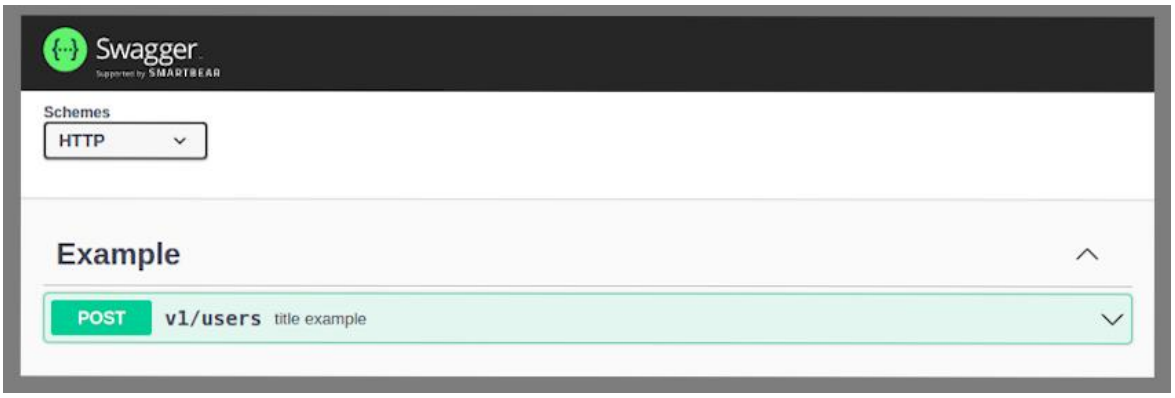

Figura 5 – Exemplo de Documentação de API.
Fonte: o autor.

Também, para facilitar a criação de novas aplicações na plataforma, foi elaborada uma biblioteca NodeJS, que executa códigos JavaScript a partir da linha de comando[2]. Essa biblioteca pode ser facilmente instalada através do Gerenciador de Pacote Node, do inglês *Node Package Manager* (NPM), e a partir disso, os comandos fornecidos pela biblioteca podem criar novos projetos utilizando o *template* anteriormente citado. A biblioteca contém um script que clona o projeto do *template* e altera as informações da aplicação para o desejado. Um exemplo de criação de aplicação pode ser visto na Figura 7.

A fim de facilitar a futura expansão do projeto, a arquitetura foi construída desde o início de forma modular. Esta arquitetura apresenta uma segmentação entre *Backoffice* e *Marketplace*, que são dois subprodutos da plataforma. *Backoffice* é todo o contexto da plataforma que trata das ferramentas de venda. Gerenciamento de aves, histórico, anúncios e vendas, são exemplos de cenários que estão dentro do contexto do *Backoffice*. Enquanto que o contexto do *Marketplace* concentra-se na exibição de anúncios e experiência de compra na plataforma.

Esses dois subprodutos da plataforma são tratados apartadamente na arquitetura, visando que, se no futuro algum outro nicho do agronegócio passe a ser atendido pela plataforma, não haja problemas com essa nova integração. Então, de um lado, têm-se o módulo de aves

[1] https://github.com/edumoreira1506/cig-service-template
[2] https://github.com/edumoreira1506/create-cig-project

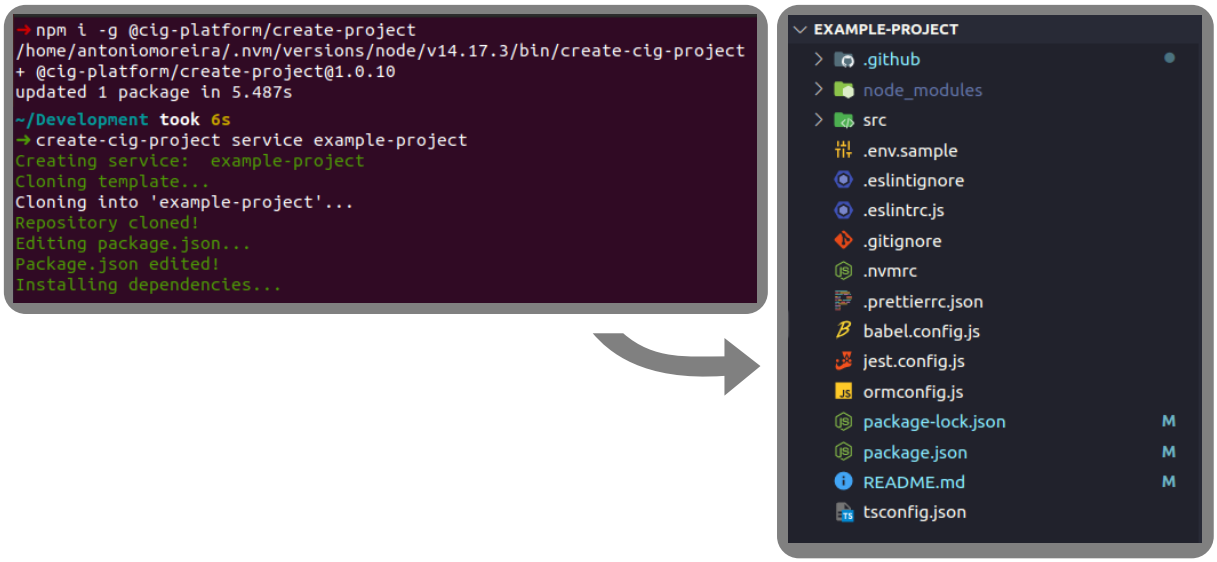

Figura 6 – Exemplo de Criação de Projeto via Linha de Comando.
Fonte: o autor.

caipiras, que trata todo gerenciamento de criatórios, aves, e registros de aves. Por outro lado, o módulo do *Marketplace* trata dos anúncios e compras na plataforma.

5.1.1 Microsserviços

Um microsserviço é uma aplicação que trata de um contexto específico da plataforma e consegue operar de forma totalmente isolada das demais aplicações da plataforma. Os microsserviços são aplicações que funcionam como APIs que se comunicam entre si através de requisições HTTP. Alguns dos microsserviços estão rodando dentro de uma Rede privada virtual, do inglês *Virtual Private Network* (VPN), cujas rotas não tem autenticação e podem ser acessadas apenas dentro da própria VPN.

O primeiro microsserviço construído foi o *Account Service*, responsável por todo contexto de usuário na plataforma. A API deste microsserviço possui rotas para criação e edição de usuários. O *Poultry Service* trata do contexto dos criatórios, aves, e registros de aves. Um criatório é vinculado a um usuário, uma ave é vinculada a um criatório, e um registro de ave é vinculado a uma ave. Logo, esse microsserviço trata de todas particularidades do nicho de ave caipira, se num futuro a plataforma for expandida para um novo nicho, um novo microsserviço poderá ser criado para atender as necessidades, podendo facilmente ser acoplado ao *Account Service*, que apenas se relaciona com as demais aplicações e tem apenas conhecimento do contexto de usuário. Esses dois microsserviços estão conectados a um mesmo banco de dados, não possuem autenticação/autorização em suas rotas, rodam dentro da VPN, e pode-se dizer que ambos tratam-se do *back-end* do gerenciamento do plantel dos criadores, isto é: gerenciamento de informações do usuário, informações do criatório, aves, e registros de aves.

Estes microsserviços não tem conhecimento do *Marketplace*, são o lado da plataforma que lida apenas com o gerenciamento das aves e as demais informações citadas.

O *Advertising Service* e o *Deal Service* são os microsserviços que tratam o contexto do *Marketplace*. Assim como os anteriormente citados, esses também não possuem autenticação/autorização em suas rotas e rodam dentro da VPN. No momento que um criatório registra uma ave na plataforma, essa ave não está instantaneamente disponível para compra no *Marketplace*, para que a ave passe a ser exibida no *Marketplace*, o usuário precisa criar um anúncio para essa ave. Então, uma ave é uma entidade diferente do anúncio, e por isso, são tratados separadamente na arquitetura da plataforma. Na prática, quando um usuário registra uma ave na plataforma, ele interage com o *Poultry Service*, porém quando esse usuário cria um anúncio para essa ave, as informações do anúncio são gerenciadas pelo *Advertising Service*. E se num futuro, novos tipos de animais ou produtos passarem a ser comercializados no *Marketplace*, esse contexto de anúncios está abstraído em apenas um microsserviço, que pode facilmente ser acoplado a outras aplicações, pois trata de anúncios genéricos no *Marketplace*.

Toda parte de compra, venda e acompanhamento de compra é encapsulada no microsserviço *Deal Service*. Esse microsserviço se conecta ao mesmo banco de dados do *Advertising Service*. O serviço faz todas validações necessárias no momento da compra e também disponibliza através da API, a exibição das negociações realizadas por determinado criatório.

5.1.2 Aplicações modulares

Aplicação modular é uma aplicação *back-end* que tem seu contexto de atuação, porém, diferente dos microsserviços, não necessariamente a aplicação opera de forma totalmente independente, ela pode ser acoplada a outros elementos da arquitetura. Um *Back-end* para *Front-end*, do inglês *Back-end for Front-end* (BFF) é uma aplicação *back-end* que é construída para servir de API para uma aplicação *front-end*. Alguns dos microsserviços anteriormente citados, são as aplicações que contém as regras de negócios da plataforma. As aplicações *front-end* não podem se comunicar diretamente com os microsserviços que rodam dentro da VPN. Então, nenhum BFF roda dentro da VPN e serve de ponte entre a aplicação *front-end* e um microsserviço que roda dentro da VPN. Como os microsserviços que rodam dentro da VPN não possuem autenticação em suas rotas, é no BFF que é feito essa autenticação. Uma rota de BFF que possui autenticação consulta o *Auth Service* por baixo dos panos para fazer a autenticação do usuário. O BFF é o responsável por receber todos dados e manusear entre as aplicações *back-end*, como por exemplo, quando um usuário realiza a compra de uma ave, o seguinte fluxo acontece: o *Auth Service* é consultado para fazer a autenticação do usuário; o *Poultry Service* é consultado para verificar as informações do criatório e ave; o *Advertising Service* é consultado para verificar as informações do anúncio; e por fim, o *Deal Service* é consultado para registrar a compra. A aplicação BFF é a responsável por fazer essas interações com os microsserviços e manter o bom funcionamento da plataforma.

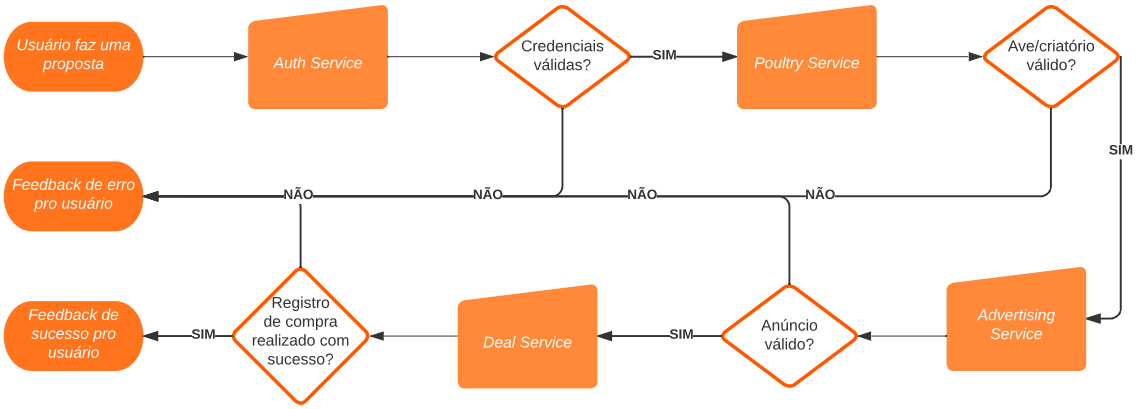

Figura 7 – Diagrama exemplificando interações entre os microsserviços no fluxo de início de negociação. Fonte: o autor.

O *Marketplace BFF* é a aplicação que possui uma API que permite fazer uma proposta em um anúncio e registro de pergunta em anúncio. Já o *Backoffice BFF* possui uma API para a aplicação *front-end* do *Backoffice*. O *Backoffice* é a área da plataforma que o usuário pode cadastrar as aves, registros de aves e anúncios. O BFF tem rotas que permite: visualização de negociações (compras/vendas), visualização de aves, visualização de informações do criatório, edição de informações do criatório, responder perguntas em anúncio, e criar anúncio.

O *Auth Service* é a aplicação responsável pela autenticação e autorização do usuário. Esse é o único serviço responsável por assinar e verificar os *tokens* JWT, todas as requisições que precisam de autorização, vão acabar consumindo esse serviço por conta disso. O interessante de ter concentrado em apenas um lugar a forma de autenticação/autorização da plataforma, é que se caso um dia seja necessário alterar o tipo de *token* ou até mesmo a estratégia de autenticação/autorização, tudo isso já está isolado nesse serviço e todas demais aplicações da plataforma já comunicam-se com ele. Esse serviço não roda dentro da VPN, pois é utilizado pelas aplicações *front-end* para autenticação, cadastro e edição de usuário e também é utilizado pelas aplicações *back-end* para verificar os *tokens* enviados nas requisições. O usuário é uma entidade que existe tanto no contexto do *Marketplace* quanto no *Backoffice*, dessa forma, esse contexto fica isolado nesse serviço e pode ser consumido por qualquer aplicação de ambos os lados.

O *Content Search Service* é a aplicação responsável pela listagem de aves, criatórios e anúncios na plataforma. Esse serviço não roda dentro da VPN, e é utilizado pelas aplicações *front-end* para visualização de conteúdo. No *Marketplace* o usuário pode fazer a busca de anúncios de aves, e como dito anteriormente, as aves e anúncios são entidades diferentes, portanto são tratadas em aplicações diferentes, logo esse serviço em questão é o responsável por fazer essa relação entre os anúncios, aves e criatórios e enviar essas informações para o *front-end*.

5.1.3 Banco de dados

O banco de dados utilizado é o PostgreSQL, na Figura 8 é possível ver a modelagem. As tabelas do lado direito da figura representam o banco de dados que os microsserviços do lado do *Marketplace* (*Advertising Service* e *Deal Service*) consomem. Enquanto que do lado esquerdo da figura são as tabelas que estão no banco de dados que os microsserviços do *Backoffice* (*Account Service* e *Poultry Service*) consomem.

5.2 ARQUITETURA *FRONT-END*

O *front-end* da plataforma é formado por seis aplicações e cinco bibliotecas. Seguindo o mesmo contexto modular anteriormente citado, existem duas aplicações principais no *front-end*, uma delas trata de toda exibição de conteúdo do *Marketplace*. Enquanto que a outra aplicação contém todo o painel administrativo do criador, onde é possível registrar aves, anúncios, registros de aves e editar informações do criatório. As outras quatro aplicações, são pequenas aplicações onde cada uma delas contém apenas uma página da plataforma. Dentre as bibliotecas, quatro delas são utilizadas para comunicação com as aplicações *back-end* que são públicas na internet, e a outra é a biblioteca de componentes baseados no Atomic Design.

5.2.1 Comunicação entre *front-end* e *back-end*

A comunicação entre as aplicações *front-end* e *back-end* é feita por meio de requisições HTTP. Foram criadas quatro bibliotecas NPM para fazer a comunicação entre as aplicações *front-end* e *back-end*, cada uma delas serve para fazer a comunicação com uma aplicação específica: *Marketplace BFF*, *Backoffice BFF*, *Content Search Service* e *Auth Service*.

A criação de uma biblioteca, além de apartar o contexto das interações com as aplicações *back-end* do restante da aplicação, também reduz a repetição de código na plataforma. Se novas aplicações, sejam *front-end* ou *back-end*, precisarem se comunicar com alguma das aplicações, podem facilmente fazer uso de uma das bibliotecas. As bibliotecas servem como um cliente de API, que tem métodos para executar todas requisições para as aplicações *back-end*.

5.2.2 Componentes

Uma biblioteca de componentes foi criada com React e Typescript, os componentes foram construídos seguindo os padrões do Atomic Design. A biblioteca contém uma lista de componentes agnósticos usados pelas aplicações *front-end* para a construção da interface da plataforma. Como trata-se de uma biblioteca apartada das aplicações *front-end*, a mesma pode

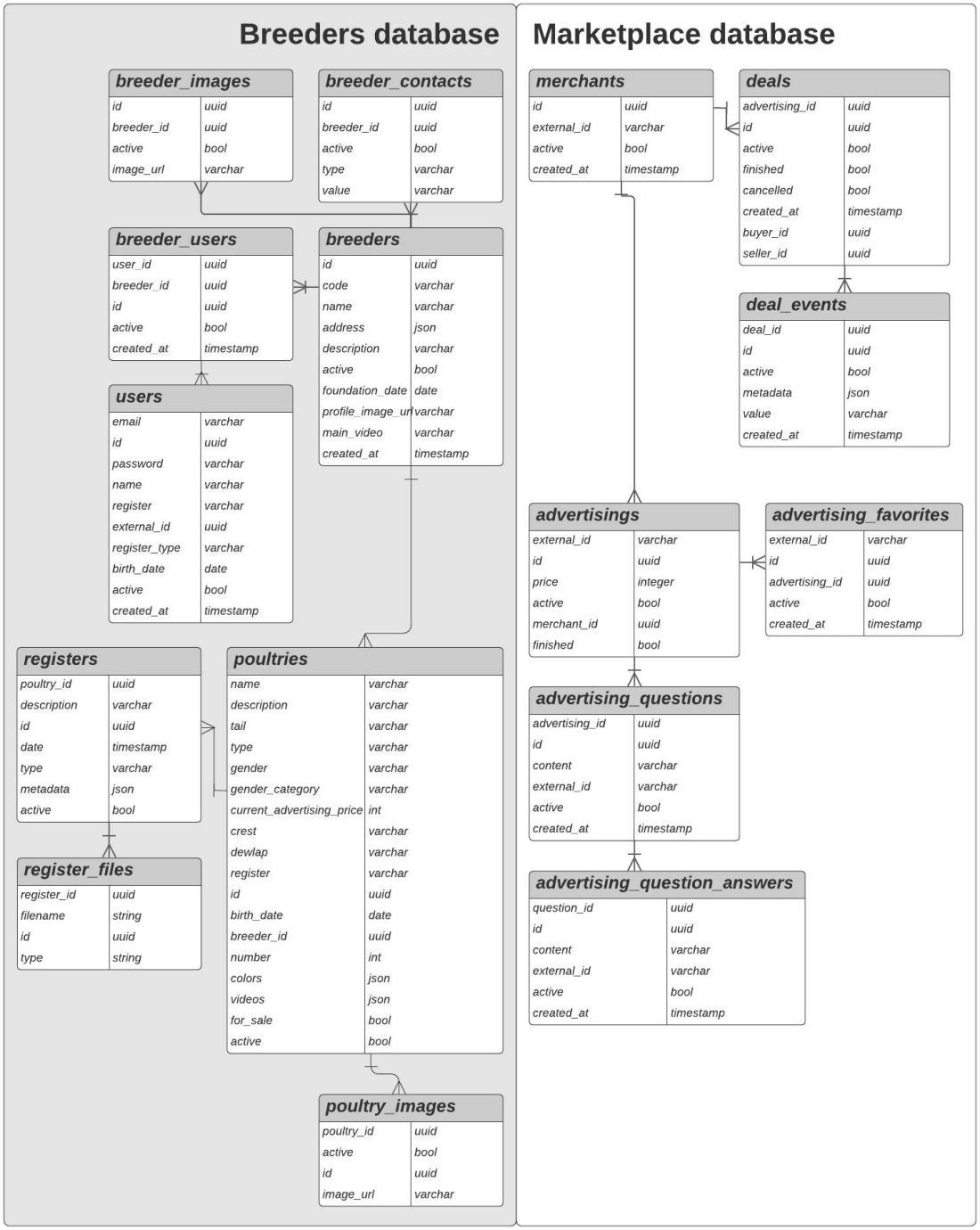

Figura 8 – Modelagem do Banco de Dados.
Fonte: o autor.

ser usada por qualquer aplicação *front-end*. Na figura 9 é possível ver todos componentes que foram criados.

Olhando para o Atomic Design na prática, observa-se que na lista de componentes átomos, tem o componente *LinkButton*. Esse componente pode ser visualizado na Figura 10. Trata-

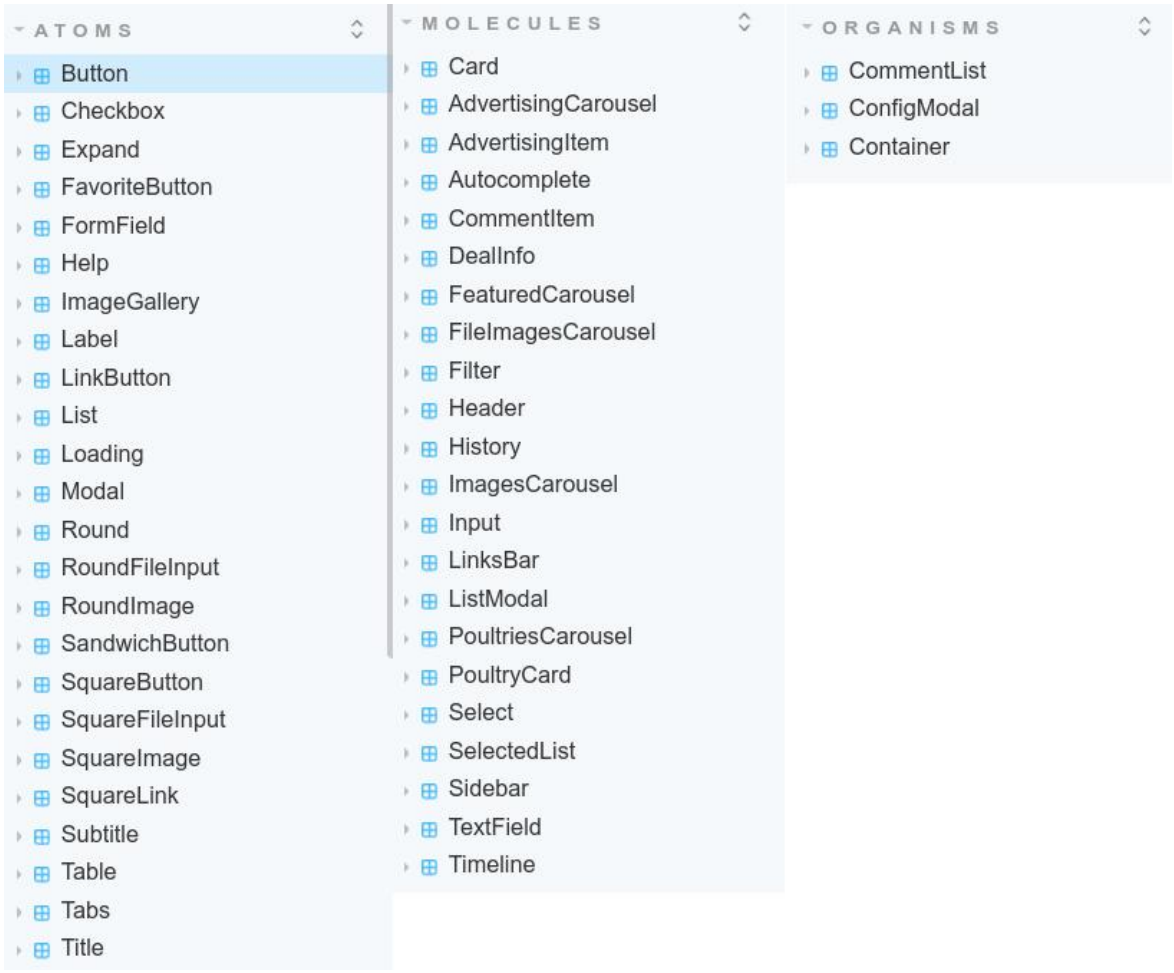

Figura 9 – Componentes criados na biblioteca.
Fonte: o autor.

se de um componente de botão que apenas recebe as propriedades de título, e um *callback* para quando for clicado, portanto foi tratado como átomo.

Na Figura 11 é exibido um componente um pouco mais complexo, o *Sidebar*, a grosso modo, é uma lista de itens, esse componente faz o uso dentro dele de outro componente, que é o átomo *LinkButton* anteriormente citado. Como trata-se de um componente que tem mais de um átomo dentro dele, foi tratado como molécula.

Por fim, na Figura 12 é exibido um componente ainda mais complexo que o anterior, o *Container*. Esse é um componente que é usado pelas aplicações para inserir o layout fixo do menu lateral e superior em todas as telas da plataforma. Dentro desse componente é usado outros componentes moléculas e átomos da biblioteca, sendo assim, trata-se de um organismo.

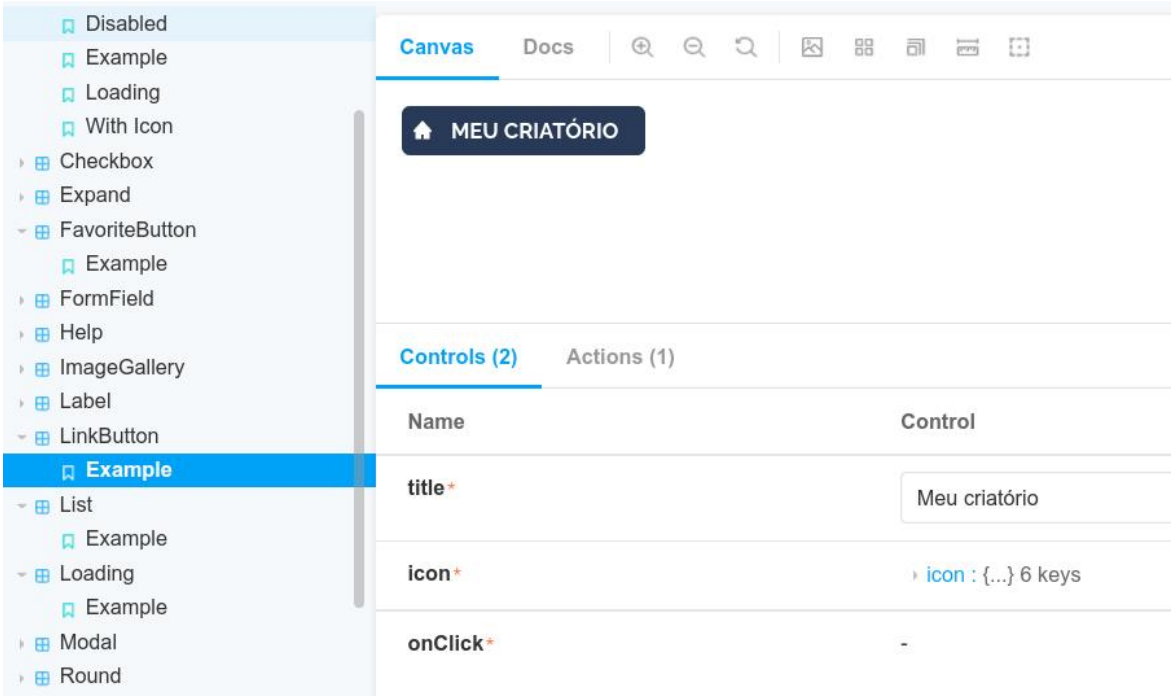

Figura 10 – Componente átomo *LinkButton*.
Fonte: o autor.

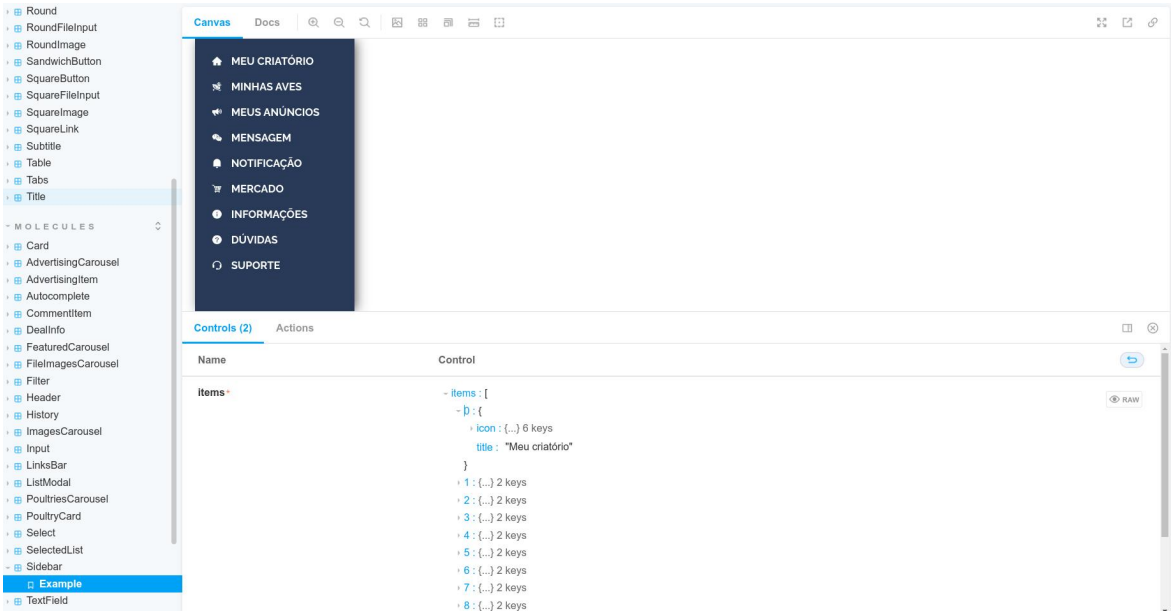

Figura 11 – Componente molécula *Sidebar*.
Fonte: o autor.

5.2.3 Aplicações

A aplicação *front-end* do *Marketplace* foi feita com React, NextJS e Typescript. A aplicação utiliza de uma biblioteca para comunicação com o *Marketplace BFF*. Essa aplicação contem as página de: login, cadastro de usuário/criatório, exibição e busca de anúncios.

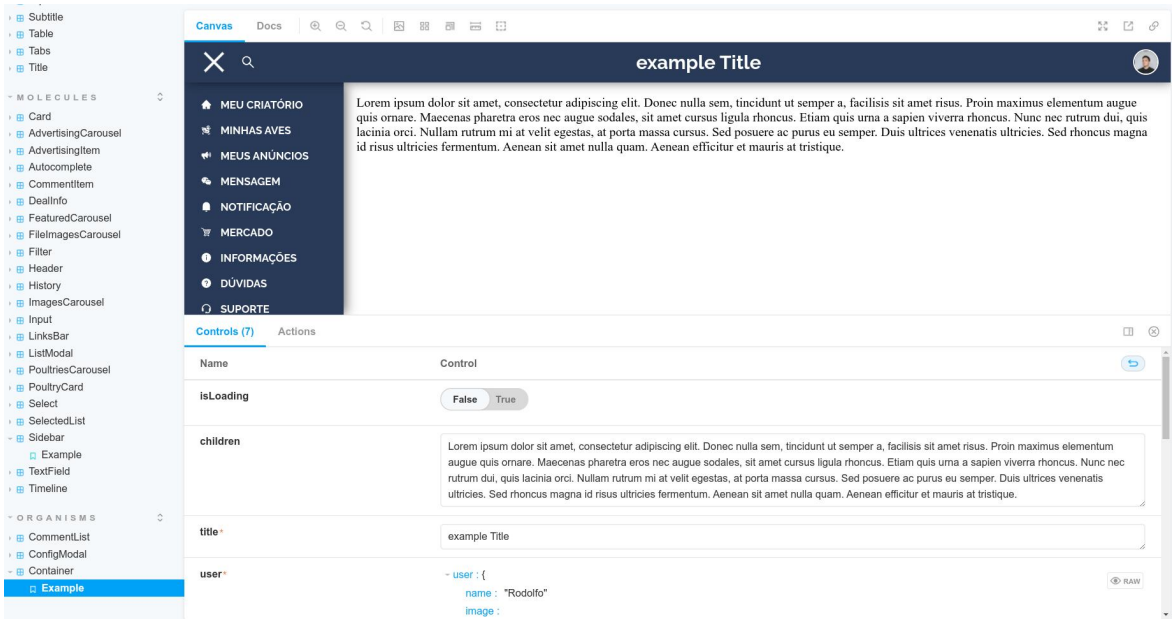

Figura 12 – Componentes organismo *Container*.
Fonte: o autor.

O NextJS, criado pela Vercel (2021), permite que uma aplicação React realize Renderização do Lado do Servidor, do inglês *Server Side Rendering* (SSR). Com o SSR, as páginas da aplicação podem ser ranqueadas nos buscadores, e isso faz com que a plataforma seja mais acessível para novos usuários. Como o *Marketplace* é baseado em anúncios, é interessante que esses anúncios sejam ranqueados nos buscadores a fim de facilitar a busca dos criadores e animais comercializados.

A aplicação *front-end* do *Backoffice* foi feita com React e Typescript, e também utiliza de uma biblioteca para comunicar-se com o *back-end*, porém, desta vez com o *Backoffice BFF*. Essa aplicação contem todas as páginas do painel administrativo do criatório. Por tratar-se de uma área administrativa da plataforma, onde apenas usuários autenticados têm acesso, não há a necessidade de ranquear essas páginas nos buscadores, portanto não foi usado o NextJS para essa aplicação.

5.2.4 Micro-aplicações

Na arquitetura *front-end* da plataforma, existem as aplicações principais, que foram citadas anteriormente, e também existem as micro-aplicações, que são pequenas aplicações *front-end* que podem ser usadas dentro de outra aplicação *front-end*. Uma micro-aplicação *front-end* também pode ser chamada de *micro front-end* (SATHYAN *et al.*, 2017).

Os princípios do *micro front-end* são os mesmos que os dos microsserviços do *back-end*: quebrar uma grande aplicação em pequenos pedaços que possam trabalhar de forma independente uns dos outros. Trabalhando com *micro front-ends* é possível fazer a publicação

de cada uma das aplicações separadamente, não é necessário fazer a publicação do projeto inteiro para atualizar apenas uma pequena parte do *front-end*.

Uma aplicação *micro front-end* pode ser usada por mais de uma aplicação *front-end*, isso evita a duplicidade de código e faz com que ambas aplicações sempre estejam utilizando da mesma versão do *micro front-end*, diferentemente de uma biblioteca, que é necessário fazer a atualização manualmente toda vez que é gerada uma versão nova da mesma. No caso de aplicações *micro front-end*, por tratarem-se de aplicações, sempre que houver uma nova publicação, irá refletir em todas aplicações *front-end* que estejam usando da aplicação *micro front-end*.

Foram construídas aplicações *micro front-ends* que encapsulam páginas da plataforma. Como citado anteriormente, são duas aplicações principais no *front-end*, dentre essas duas aplicações, existem algumas páginas que são exibidas em ambos cenários, como por exemplo a página da ave/anúncio, no *Backoffice* é possível ter uma pré-visualização da página de uma ave anunciada no *Marketplace*. Tendo uma aplicação *micro front-end* é possível que ambas aplicações (*Backoffice* e *Marketplace*) usem do mesmo código através do *micro front-end*.

A aplicação *Poultry Carousels* trata-se de um *micro front-end* que é capaz de comunicar-se com o microsserviço *Content Search Service* através de uma biblioteca. Para que o *micro front-end* seja usado dentro de outra aplicação, é necessário informar o *id* de um criatório, e a partir disso, a aplicação consegue obter do *Content Search Service* todas as aves registradas naquele criatório separadas por sexagem da ave. Na figura 13 é possível ver um exemplo de visualização do *micro front-end*. Esse *micro front-end* é usado no *Backoffice* para o usuário visualizar todas aves que tem cadastradas no criatório e na página de pré-visualização do criatório, enquanto que no *Marketplace* é utilizado na página do criatório. No exemplo da imagem é possível visualizar dois ícones no canto superior direito de cada ave, um ícone representa a ação de “editar” e o outro de “visualizar”. O ícone de “editar” só é mostrado na página de visualização de aves do criatório do *Backoffice*, nos demais cenários, apenas o botão de “visualizar” é exibido.

O *Breeder Page* é o *micro front-end* que contem a página do criatório, assim como o *Poultry Carousels*, somente é necessário ser informado um *id* de criatório para que a aplicação consulte o *Content Search Service* através de uma biblioteca e exiba os dados na interface. Esse *micro front-end* faz uso do *Poultry Carousels* para exibição de aves do criatório. Esse *micro front-end* é usado pelo *Backoffice* na página de pré-visualização de criatório e no *Marketplace* é usado na página do criatório. A figura 13 contém um exemplo do *Breeder Page* sendo executado.

O *Poultry Page* é o *micro front-end* responsável pela página da ave, o *micro front-end* requer que seja informado um *id* de ave e um *id* de criatório, a partir disso, assim como as aplicações *micro front-end* anteriormente citadas, o *Content Search Service* é consultado para exibir os dados na interface. O *micro front-end* em questão é usado no *Backoffice* e no *Marketplace* nas páginas de visualização de ave. Na figura 13 é possível visualizar um exemplo da aplicação *Poultry Page* executando.

Por fim, o *Deal Page* é a aplicação *micro front-end* que contem a página de negociação. Uma negociação sempre envolve um comprador e um vendedor, logo, para que a aplicação seja executada corretamente, é necessário informar o *id* de uma negociação e também se o usuário que está visualizando a negociação é o comprador ou o vendedor. Esse *micro front-end* é usado apenas no *Backoffice*, porém, em dois cenários diferentes, um deles é a visualização de compra, e o outro é a visualização de venda. Na figura 13 é possível visualizar o exemplo de uma página de negociação finalizada.

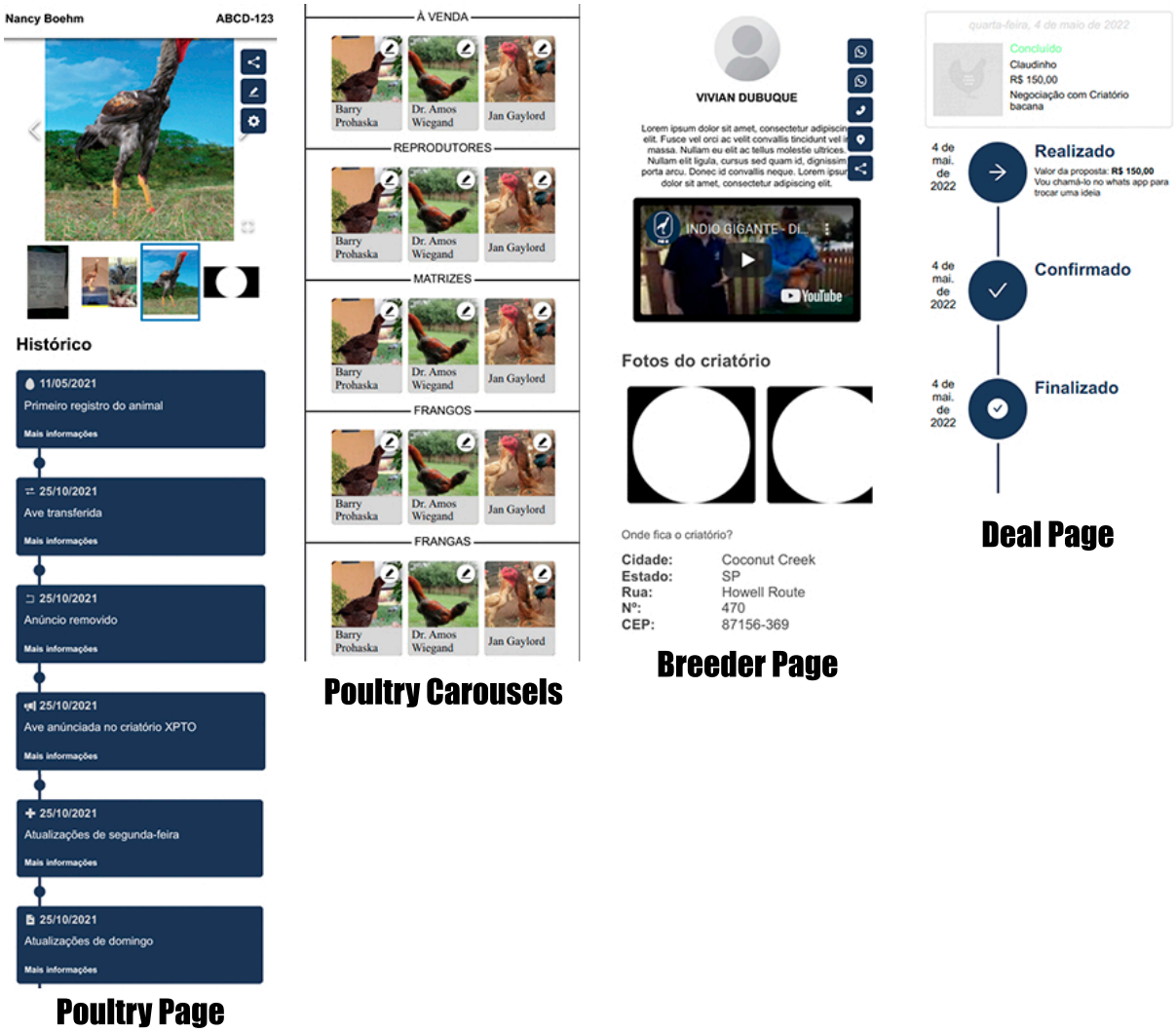

Figura 13 – Exemplo de exibição das aplicações *micro front-end*.
Fonte: o autor.

6 ANÁLISE E DISCUSSÃO DE RESULTADOS

Nesse capítulo serão mostrados os recursos que foram implementados na plataforma e o detalhamento das regras de negócios de cada um dos recursos da plataforma.

6.1 USUÁRIO

A entidade de usuário é a base da plataforma, pois todas demais entidades dependem indiretamente de que um usuário exista para que elas possam existir também. Na prática, o usuário é a pessoa que está utilizando da plataforma.

6.1.1 Cadastro de usuário

O cadastro de novos usuários é feito através da página de cadastro do *Marketplace*. O cadastro é dividido em duas partes, a primeira incluindo as informações do usuário, enquanto que a segunda contém as informações do criatório. Na Figura 14 são mostrados os formulários de cadastro.

Na primeira parte do cadastro o usuário pode optar por continuar com *Facebook* ou *Gmail*, ao optar por uma dessas opções o usuário deverá permitir acesso aos dados e depois os campos de e-mail e nome serão preenchidos automaticamente com base nas informações da rede social escolhida. O formulário de cadastro do criatório precisa ser preenchido manualmente sempre. O código do criatório precisa ter 4 letras maiúsculas.

Para um usuário ser cadastrado é necessário que ele envie e-mail e senha para que seja possível fazer o login depois na plataforma, porém, quando o cadastro é feito com rede social, não é armazenada uma senha no banco de dados, mas é usado um identificador da rede social escolhida para que seja usado depois para fazer o login. Na tabela de usuários existe o campo *register type*, que justamente serve para identificar o tipo de cadastro do usuário: *default*, para cadastros padrões com e-mail e senha; *facebook*, para cadastros realizados utilizando o facebook; *email*, para cadastros feitos através do gmail.

Na prática, quando o usuário termina de preencher o formulário de cadastro e todas informações estão válidas, será cadastrado um usuário, criatório, usuário do criatório e também um comerciante. O usuário contém as informações do usuário, o criatório as informações do criatório, o usuário do criatório é o vínculo feito entre o usuário e o criatório e o comerciante é usado posteriormente para fazer operações dentro do *Marketplace*. A rota de cadastro fica no *Auth Service*, e pode ser acessada utilizando da biblioteca de comunicação com esse microsserviço.

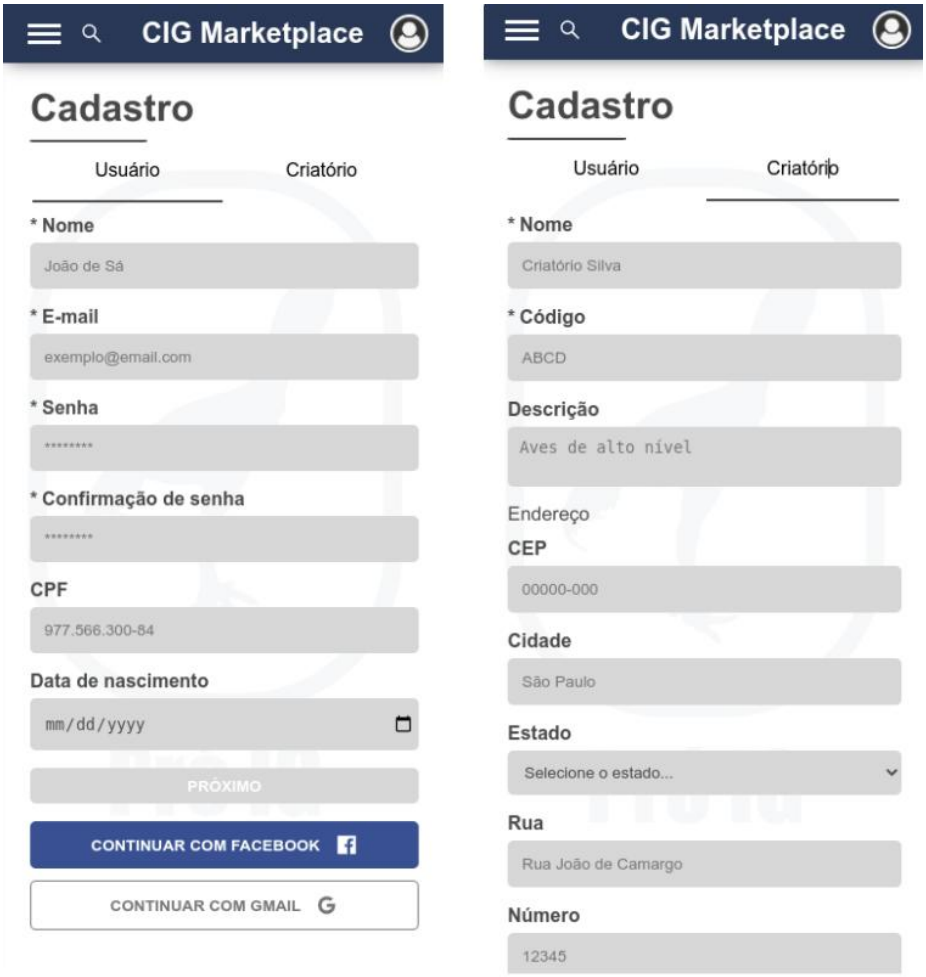

Figura 14 – Página de cadastro do *Marketplace*.
Fonte: o autor.

6.1.2 Autenticação

A autenticação de usuário é feita na página de login do *Marketplace*. A autenticação pode ser feita com e-mail e senha normalmente, ou se o usuário for cadastrado por alguma rede social, também é possível fazer login usando a rede social. Na figura 15 é possível visualizar a página de login do *Marketplace*.

Quando o usuário clica para fazer o login na plataforma, é disparado uma requisição pro *back-end* que valida as credenciais enviadas. No caso de sucesso, é enviado um *token* JWT como resposta que contém as informações cadastradas vinculadas aquele usuário/criatório. A Figura 16 apresenta um exemplo de *token* gerado, onde é possível observar a presença de dados públicos do usuário, como id de usuário, lista de criatórios, id de comerciante, entre outros.

Todos os recursos que podem ser utilizados pelo usuário no *Marketplace* e no *Backoffice* tratam-se de recursos “privados”, isso é, necessita que seja um usuário autenticado para realizar as operações. No caso da autenticação realizada com sucesso, o *token* gerado é aramazenado

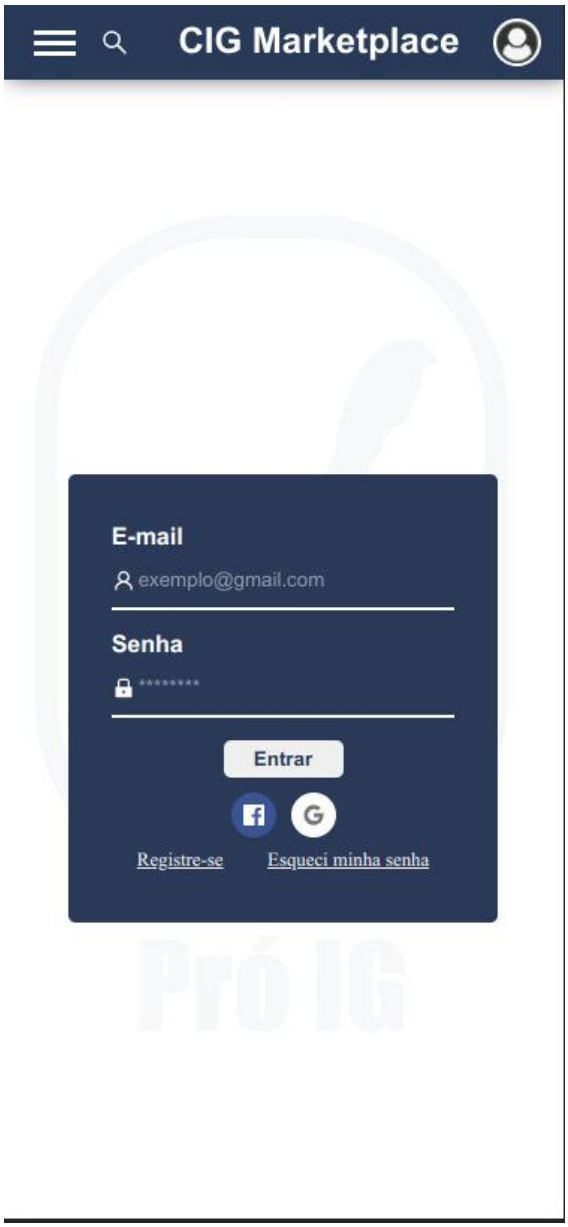

Figura 15 – Página de login do *Marketplace*.
Fonte: o autor.

no *front-end* para ser utilizado nas próximas requisições aos recursos “privados” da plataforma. Na prática, o token é enviado no cabeçalho de *Authorization* para as requisições enviadas ao *back-end*. Para que o *token* não expire, a cada iteração do usuário com a plataforma, é usado o recurso de atualização de token.

No caso de um usuário que tenha o *register type* como *default*, é enviado o e-mail e senha para que o *back-end* faça a validação, enquanto que cadastros feitos com redes sociais, não é enviado um parâmetro de senha ao *back-end*, porém é enviado o *externalId* que é usado no *back-end* para autenticação.

As rotas de autenticação e atualização de *token* estão no *Auth Service* e podem ser acessadas utilizando da biblioteca de comunicação com o microsserviço em questão. No caso

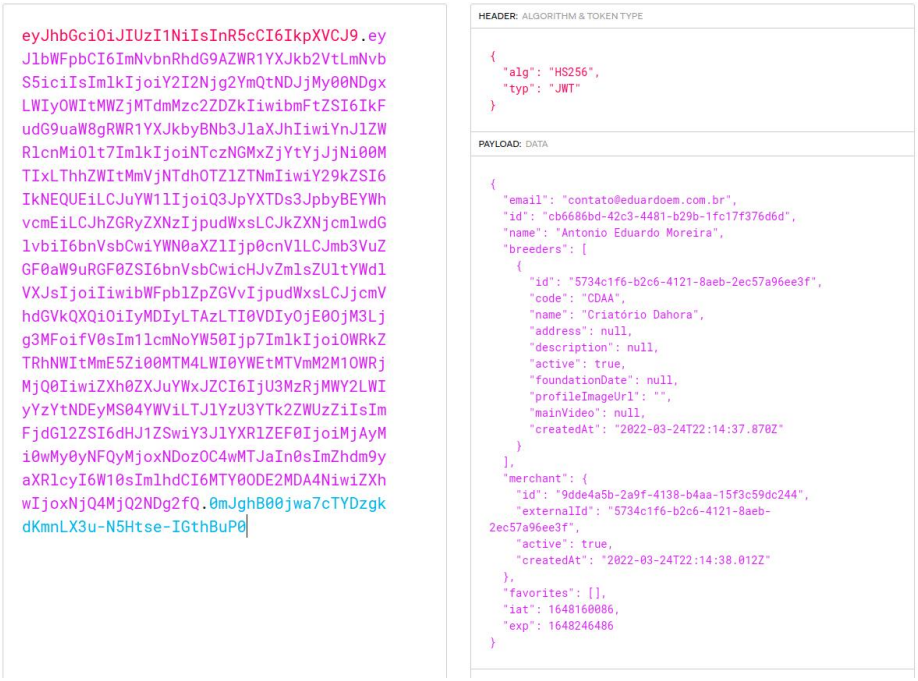

Figura 16 – Token exemplo.
Fonte: o autor.

de quando as credenciais não são válidas, o *Auth Service* informa o erro ao cliente, enviando no corpo da resposta o motivo, além de retornar o código de status HTTP 400.

Todas aplicações *back-end* tem um padrão para envio de erros e mensagens de sucesso. Quando alguma aplicação *back-end* responde com um erro, esse erro é mostrado ao usuário, na Figura 18 é possível ver um exemplo de erro mostrado ao usuário.

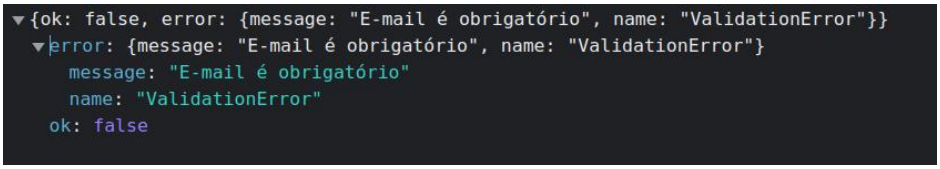

Figura 17 – Exemplo de formato de resposta de erro.
Fonte: o autor.

Uma vez que as credenciais sejam válidas, será enviado no corpo da resposta o *token* gerado, esse *token* é armazenado no estado da aplicação *Marketplace* e após isso, o usuário é redirecionado a página principal do *Backoffice*, onde também é salvo o *token* no estado local dessa outra aplicação.

6.1.3 Edição de senha

Os usuários registrados na plataforma, uma vez que estejam autenticados, na área do *Backoffice* é possível fazer a edição de senha. A página de edição de senha pode ser acessada

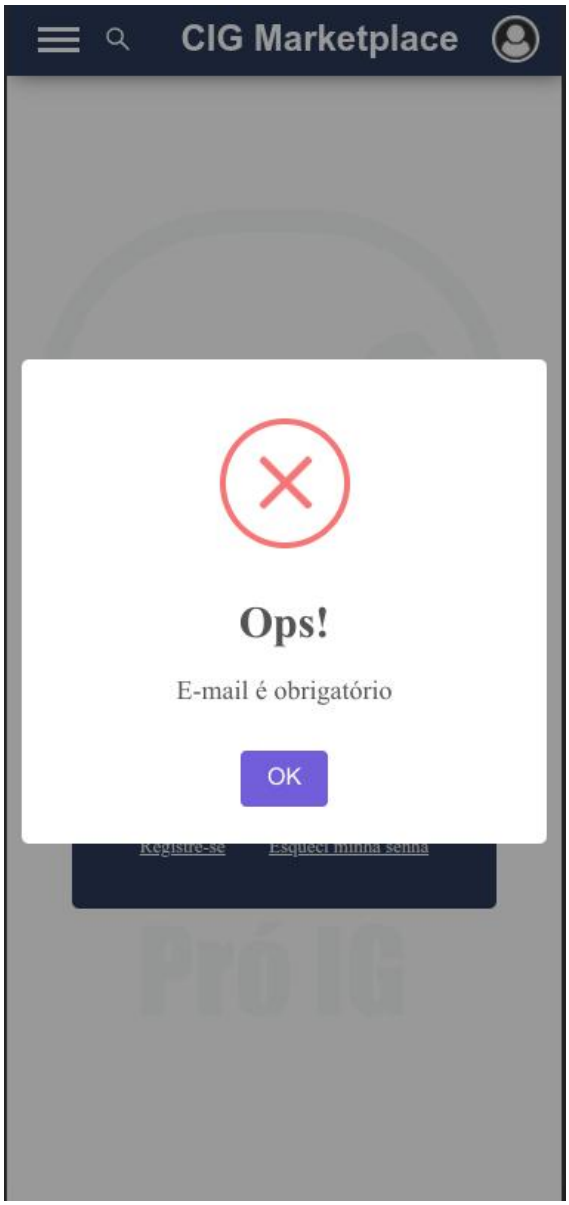

Figura 18 – Exemplo de erro mostrado em tela ao usuário.
Fonte: o autor.

ao clicar no ícone de usuário no canto superior direito, e depois o clique do usuário no item “Editar senha” redireciona-o à pagina de edição de senha.

Usuários que foram registrados utilizando de alguma das redes sociais disponíveis não tem acesso a página de edição de senha, pois esses usuários não tem uma senha cadastrada na plataforma. O recurso do *back-end* que permite que o usuário faça a edição de senha é acessado pelo *front-end* através do *Auth Service*.

6.1.4 Recuperação de senha

Os usuários cadastrados na plataforma utilizando do método padrão de e-mail e senha podem solicitar que seja enviado um e-mail informando a senha, em caso da senha ser esquecida pelo usuário. O fluxo de “Esqueci minha senha” pode ser acessado na página de login, basta o usuário informar o e-mail cadastrado. Se o e-mail informado for de um usuário válido, é enviado um e-mail para o usuário informando a senha registrada na plataforma.

O recurso de recuperação de senha fica no *Auth Service* e pode ser acessado pelo *Marketplace* utilizando da biblioteca de comunicação com a aplicação.

6.2 CRIATÓRIO

O criatório é a entidade da plataforma que armazena as informações referente a um criatório de aves. Como citado na seção anterior, para que um usuário faça o cadastro na plataforma, é necessário que ele forneça as informações de usuário e também do criatório.

6.2.1 Gestão de criatório

As informações do criatório podem ser alteradas no *Backoffice*, uma vez que o usuário esteja autenticado na plataforma é possível que o *front-end* acesse o recurso no *Backoffice BFF* para alteração de dados do criatório. Um criatório pode ser alterado somente por usuários que tenham vínculo aquele criatório. A página de edição de criatório pode ser acessada pelo menu lateral esquerdo.

O código do criatório é a única informação do criatório que não pode ser editada depois que é feito o cadastro. A página de edição de criatório é dividida em três partes: a primeira parte que contém as informações básicas do criatório, o campo de “Vídeo de apresentação” é um campo de texto que deve ser um link de um vídeo do *youtube* ou alguma outra plataforma de vídeo; a segunda parte que diz respeito ao endereço do criatório, vale ressaltar que nesse formulário de endereço, quando o usuário digita um CEP válido no campo de entrada de CEP, é feita uma consulta na API da Google para consultar as coordenadas geográficas do CEP, se caso a API da Google envie uma resposta com coordenadas geográficas válidas, um mapa é exibido com um marcador localizado nas coordenadas recebidas, o usuário pode ajustar a localização desse marcador se necessário; a terceira parte que contém as informações de contato do criatório, no momento, um contato de criatório pode ser do tipo “Telefone” ou “WhatsApp”. Na figura 19 é possível visualizar a página de edição do criatório.

As informações do criatório são exibidas na página do criatório do *Marketplace*, durante a edição é possível pré-visualizar as alterações realizadas, clicando no botão “Pré-visualizar alterações”. Na lateral superior direita são exibidos alguns ícones referente às informações do

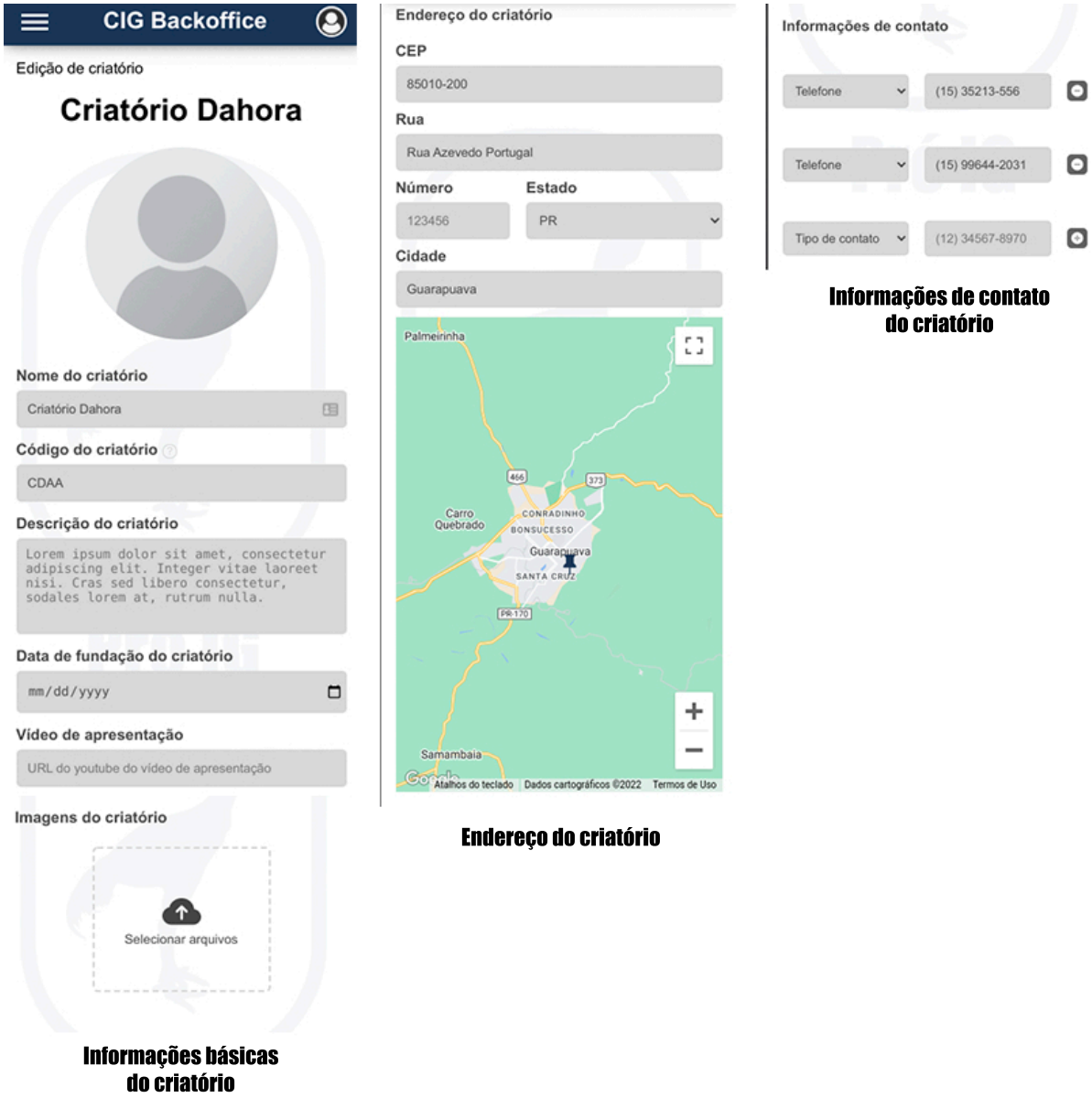

Figura 19 – Página edição de criatório.
Fonte: o autor.

criatório, os ícones de *WhatsApp* são os contatos do tipo *WhatsApp* que foram cadastrados naquele criatório, o clique num ícone desse tipo redireciona o usuário para mandar uma mensagem no *WhatsApp* para o número informado. Os ícones de telefone são referentes aos contatos do tipo telefone, o clique em um desses ícones abre o aplicativo de ligações padrão do dispositivo com o número discado. O clique no ícone de localização leva o usuário até a área do endereço, se o criatório não tiver um endereço cadastrado, o ícone não é exibido. Por fim, o ícone de compartilhamento tem a ação de copiar o link da página do criatório no *Marketplace* para área de transferência.

6.2.2 Página do criatório

Uma página do criatório pode ser acessada por qualquer usuário da internet, trata-se de uma página pública que mostra as informações básicas de um criatório e as aves cadastradas. Todas informações que são gerenciadas na área do *Backoffice* de edição de criatório são mostradas na página do criatório.

O usuário pode visualizar na página do criatório todas as aves, anunciadas ou não, cadastradas. É possível visualizar e assistir o vídeo de apresentação, caso tenha cadastrado, e também as imagens do criatório.

O clique em alguma ave da listagem de aves leva o usuário até a página da ave clicada. As aves são listadas em cinco grupos: matrizes, reprodutores, frangos e frangas, que agrupam as aves com base na sexagem cadastrada, e o grupo “à venda”, que contém todas aves que estão anunciadas no *Marketplace* independente da sexagem. Esses grupos listam no máximo 30 aves por vez, isso é, se caso o criatório tenha mais que 30 aves à venda, serão exibidas somente as últimas 30 aves anunciadas no grupo “à venda”, porém, foi implementado um sistema de paginação, que se o usuário paginar o componente de listagem de aves até o final, mais 30 aves serão exibidas após as 30 primeiras.

6.3 AVES

As aves são as grandes estrelas da plataforma, elas são vinculadas a criatórios e podem ser visualizadas no *Marketplace*. Lembrando que a entidade ave não é a mesma entidade de anúncio, então um criatório pode ter várias aves cadastradas, porém, nenhuma delas anunciada, dessa forma, o criatório pode usar a plataforma como sistema de gerenciamento de histórico de aves.

6.3.1 Cadastro de ave

A área de cadastro de ave fica no *Backoffice*, um usuário autenticado pode acessar a página de cadastro de ave, clicando na opção do menu lateral esquerdo “Meu plantel”, que redirecionará o usuário para a listagem de aves do criatório, e depois clicando no botão “Nova ave”, o usuário será redirecionado para a página de cadastro de ave.

O cadastro de ave é separado em quatro seções. A primeira seção contém todas as informações básicas da ave: raça, que no momento só tem disponível a opção “Índio Gigante”; gênero, pode ser “Macho” ou “Fêmea”; sexagem, pode ser “Frango” ou “Reprodutor” para machos e “Franga” ou “Matriz” para fêmeas; tipo de crista, que pode ser “Bola” ou “Ervilha”; tipo de barbela, que pode ser “Dupla”, “Ausência total” ou “Única”; tipo de rabo, que pode ser “Alto”, “Médio” ou “Baixo”; nome, que é um campo de texto livre; descrição, que pode ser um breve

texto que o criatório queira complementar sobre informações da ave; número de anilha interna, é o número que é usado pelo criatório para controle interno do criatório; data de nascimento, é a data de nascimento da ave, a única regra em relação a esse campo é que se a ave for um frango ou franga, ela obrigatoriamente precisa ter menos de 11 meses de idade, caso contrário é informado um erro ao usuário; medida, medida em centímetros do comprimento atual da ave; peso, peso em quilogramas atual da ave. Na Figura 20 é possível visualizar a seção de informações básicas do formulário de cadastro de ave.

A segunda seção do cadastro de ave contém as informações de coloração do animal (Figura 20), são três definições de cores que podem ser feitas, são definições de cor de: plumagem, canelas e olhos. As cores disponíveis para seleção são: preto e branco.

A terceira e quarta seção compõem a área de mídias do animal, são respectivamente, a área de vídeos e imagens, como ilustrado na Figura 20. Os vídeos são campos de texto que o usuário pode inserir links do *Youtube* ou outra plataforma de vídeo, que será exibido na página da ave, as opções de vídeo são: apresentação, animal andando e medição do animal. As imagens podem ser no formato PNG, JPEG ou JPG, e também serão mostradas na página da ave. A primeira imagem cadastrada na ave, será a imagem principal exibida nos anúncios.

Todas as informações podem ser alteradas através da página de edição de ave, disponível no *Backoffice*.

6.3.2 Gestão de aves

No *Backoffice* o usuário pode visualizar todas aves cadastradas no criatório, clicando na opção do menu lateral esquerdo “Meu plantel”. Na página de listagem de aves as aves são distribuídas no mesmo padrão da página do criatório, conforme mostra a Figura 21.

Todas aves tem o ícone de “Editar” na lateral superior direita, o clique nesse botão leva o usuário até a página de edição da ave.

Na página de edição de ave, quando a ave atinge 11 meses de idade e está cadastrada como frango ou franga, é informado ao usuário que aquela ave atingiu a idade mínima para ser um reprodutor ou matriz, pergunta-se ao usuário se ele deseja fazer essa alteração da sexagem da ave. Se o usuário confirmar a alteração, a ave será alterada para a sexagem reprodutor ou matriz, dependendo do seu gênero.

Na página de listagem de aves, o clique em alguma ave fora da área do ícone de edição, redireciona o usuário até a página de visualização da ave (Figura 22). A página de visualização do *Backoffice* tem a mesma aparência da página que será exibida no *Marketplace*.

O clique no o ícone superior direito com símbolo de compartilhamento, copia o link da página da ave no *Marketplace* para área de transferência. O clique no segundo ícone, que é o da engrenagem, abre um diálogo com quatro opções que podem ser selecionadas pelo usuário. O clique na opção “Editar ave” leva o usuário até a página de edição de ave, as demais opções serão explicadas na sequência.

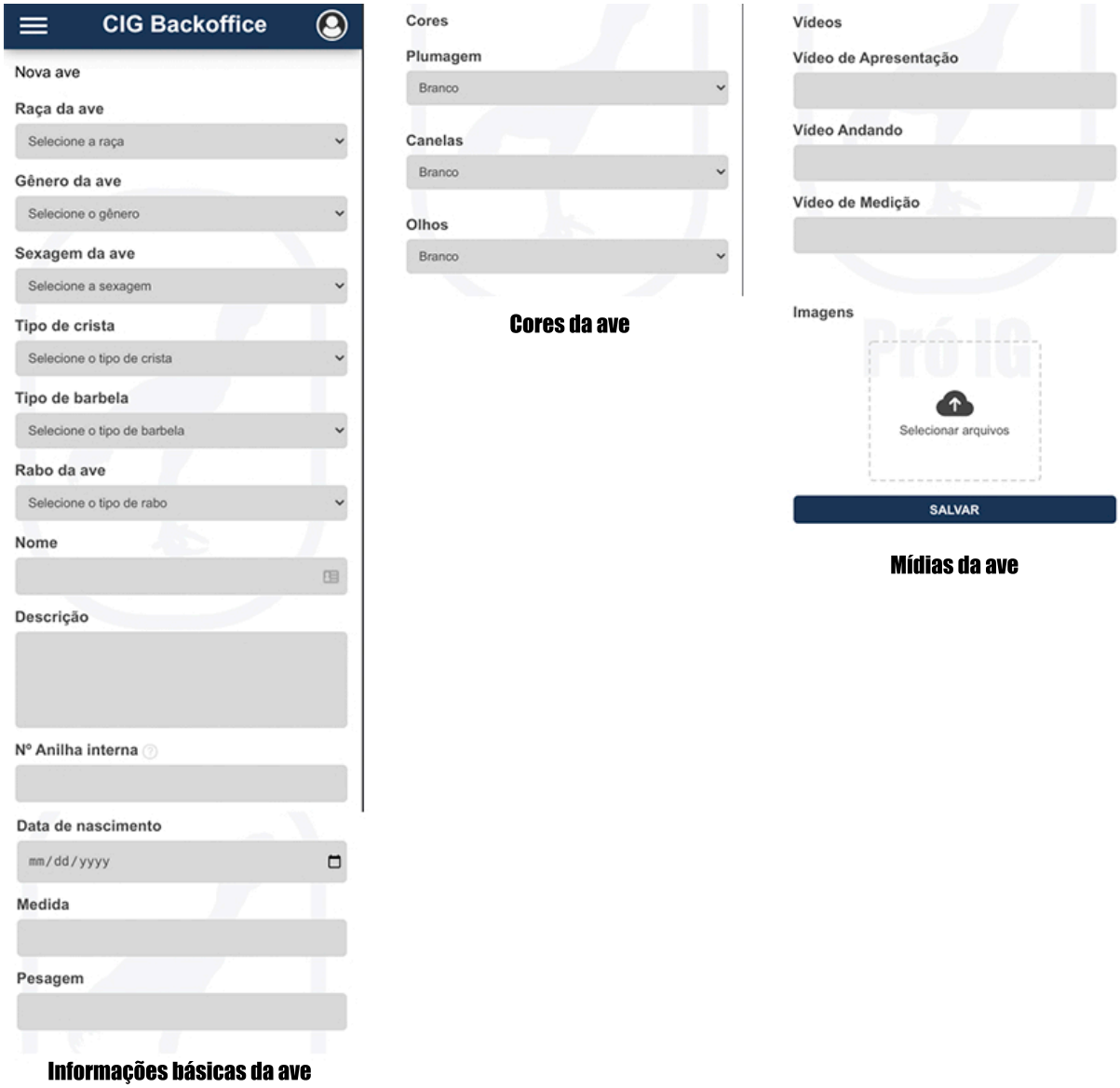

Figura 20 – Página de cadastro de ave.
Fonte: o autor.

6.3.3 Histórico da ave

O grande diferencial da plataforma é a questão da rastreabilidade do animal, para isso foi implementado o recurso de histórico da ave. Uma vez que a ave esteja cadastrada na plataforma, é possível que o criatório registre informações no histórico dessa ave, e mesmo que no futuro essa ave passe a ser de outro criatório, todas essas informações vinculadas ao histórico serão mantidas.

Na página de visualização de ave do *Backoffice*, clicando no ícone de engrenagem no canto superior direito é aberto um diálogo que contém a opção “Atualizar informações da ave”, o clique nessa opção redireciona o usuário à página de cadastro de registro de ave. O usuário pode fazer três tipos de registros na ave: imagens, que funciona como uma espécie de publicação de rede social, o usuário pode informar um texto e também vincular imagens aquela

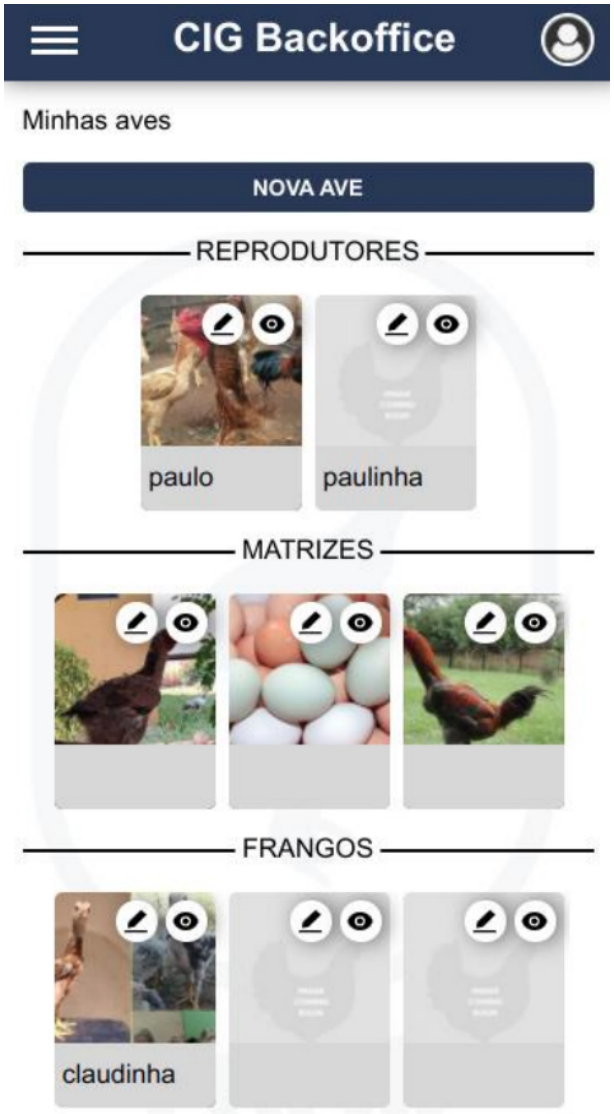

Figura 21 – Página de visualização de aves.
Fonte: o autor.

publicação (Figura 23); medição e pesagem, que é o cadastro que contém as informações de peso e medida do animal em uma determinada data, na página de cadastro de registro de medição e pesagem da ave, o usuário pode visualizar todos os registros realizados anteriormente em forma de tabela (Figura 23), vale ressaltar que a informação de medida e pesagem que o usuário informa no cadastro de ave, na prática é um registro que será feito no histórico da ave; vacinação, que é o registro de uma vacina realizada na ave em um determinada data, assim como medição e pesagem, também é possível visualizar todas vacinas registradas em forma de tabela na página de registro de vacina (Figura 23). Os campos de descrição dos registros de vacina e medição/pesagem podem ser visualizados ao expandir um item na tabela.

O histórico de registros da ave é exibido ao usuário em formato de linha do tempo na página de visualização da ave do *Backoffice* e *Marketplace*.

Figura 22 – Página de visualização de ave.
Fonte: o autor.

No caso dos registros de vacina e medição/pesagem, essas informações também são mostradas abaixo da área do histórico, porém no formato de tabela, que é o mesmo formato mostrado na página de cadastro de registro.

6.3.4 Transferência de ave

Na plataforma é possível fazer anúncios das aves cadastradas, permitindo que os usuários negociem as aves. Mesmo a plataforma tendo o espaço para negociação no *Marketplace*, também é possível que a ave seja negociada por alguma forma externa da plataforma, e para isso existe o recurso de “Transferência de ave”.

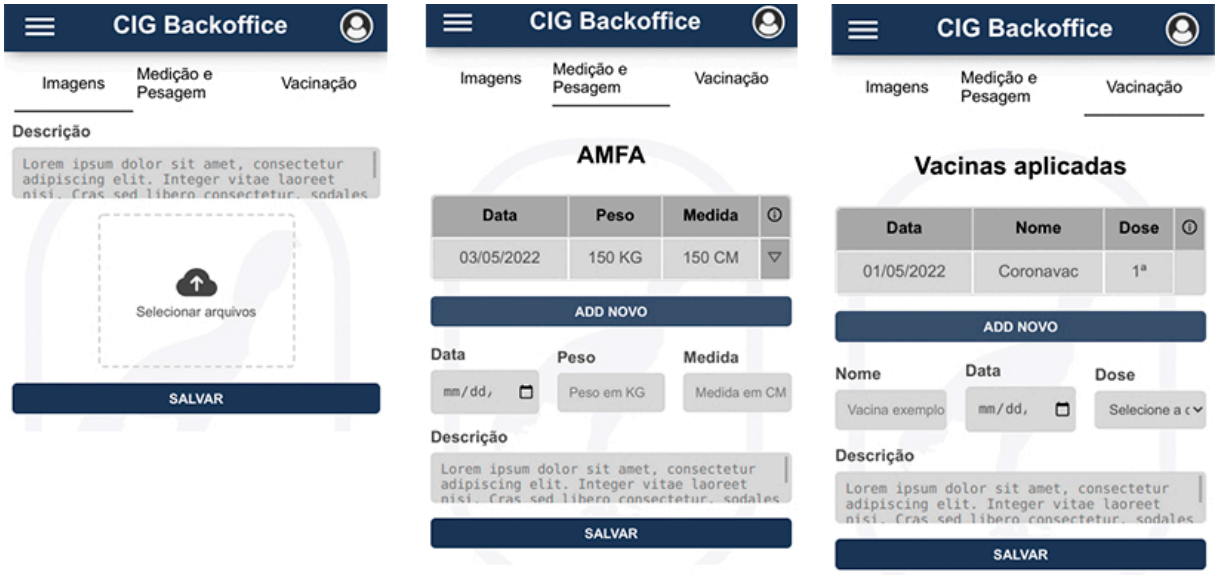

Figura 23 – Páginas de cadastro de registro de ave.
Fonte: o autor.

Na página de visualização de ave, clicando no ícone de configurações no menu superior direito e selecionando a opção “Transferir para outro criatório”, é aberto o diálogo de transferência de ave, ilustrado na Figura 24.

Para que uma ave seja transferida de um criatório para outro, o usuário pode digitar no campo de entrada o nome do criatório de destino da ave. Conforme o texto digitado pelo usuário, é buscado no *Content Search Service* os criatórios que contém aquele texto no nome e os resultados são exibidos ao usuário em forma de lista (Figura 25).

O usuário deve selecionar uma opção da lista e selecionar a caixa de confirmação de transferência para que o botão de “Confirmar transferência” seja habilitado.

O único impedimento de uma transferência de ave, é se a ave estiver anunciada e tenha negociações confirmadas em andamento, nesse caso é necessário que a negociação seja cancelada para que a ave possa ser transferida. Quando uma ave é transferida, se caso tiver um anúncio ativo, ele será removido, e se caso tiver propostas não confirmadas e nem canceladas ao anúncio, todas serão canceladas automaticamente. Os recursos de negociação e anúncio serão explicados com mais detalhes posteriormente.

6.3.5 Página da ave

Uma página de ave, assim como a página do criatório, é pública na internet e pode ser acessada por qualquer usuário. A página da ave pode ser acessada no *Marketplace* através de um link direto ou para aves anunciadas, é possível buscá-las pela interface do *Marketplace*.

Na página da ave são mostradas todas imagens, descrição e demais informações básicas. Na área de informações básicas, é mostrado a medição e pesagem do animal, essa informação é sempre pega do último registro de medição e pesagem feito no histórico animal.

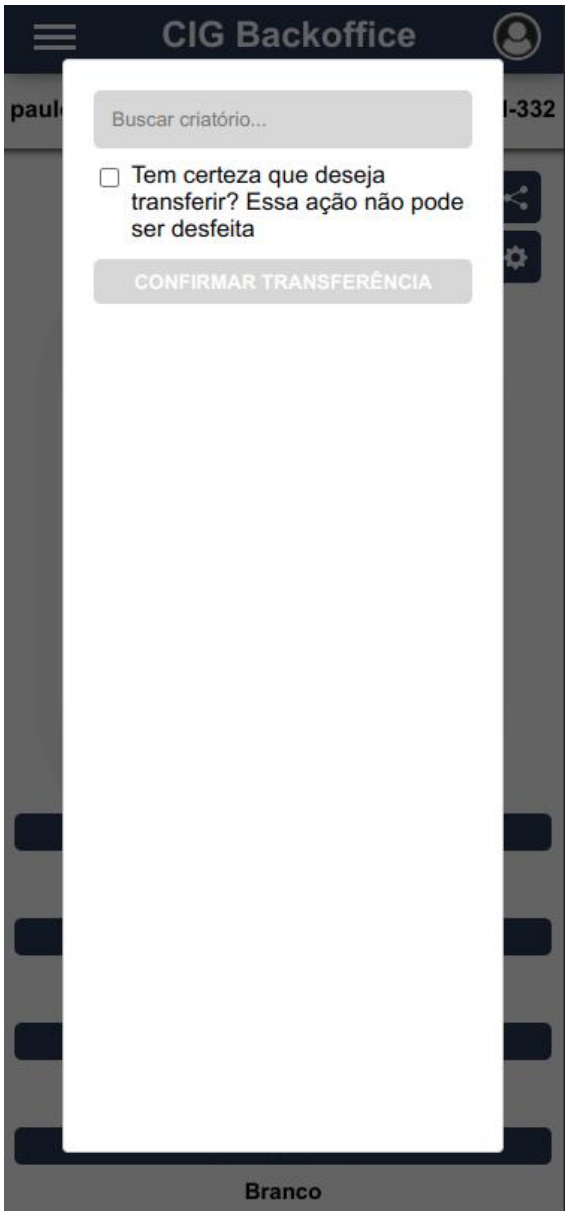

Figura 24 – Página de visualização de ave com diálogo de transferência de criatório aberto.
Fonte: o autor.

É possível visualizar todo histórico de registros em visualização de linha do tempo, e também os registros de vacina e medição/pesagem podem ser visualizados separadamente dos demais registros em forma de tabela, assim como no *Backoffice*. No caso de uma ave anunciada, é mostrado o valor do anúncio fixamente no rodapé da página, e se caso o usuário esteja autenticado na plataforma, é exibido o botão de “Fazer proposta” que inicia o processo de negociação (Figura 26).

Se caso o criatório que fez o anúncio da ave, tenha cadastrado pelo menos uma forma de contato do tipo “WhatsApp”, é exibido o botão de “Mensagem” junto ao botão de “Comprar”.

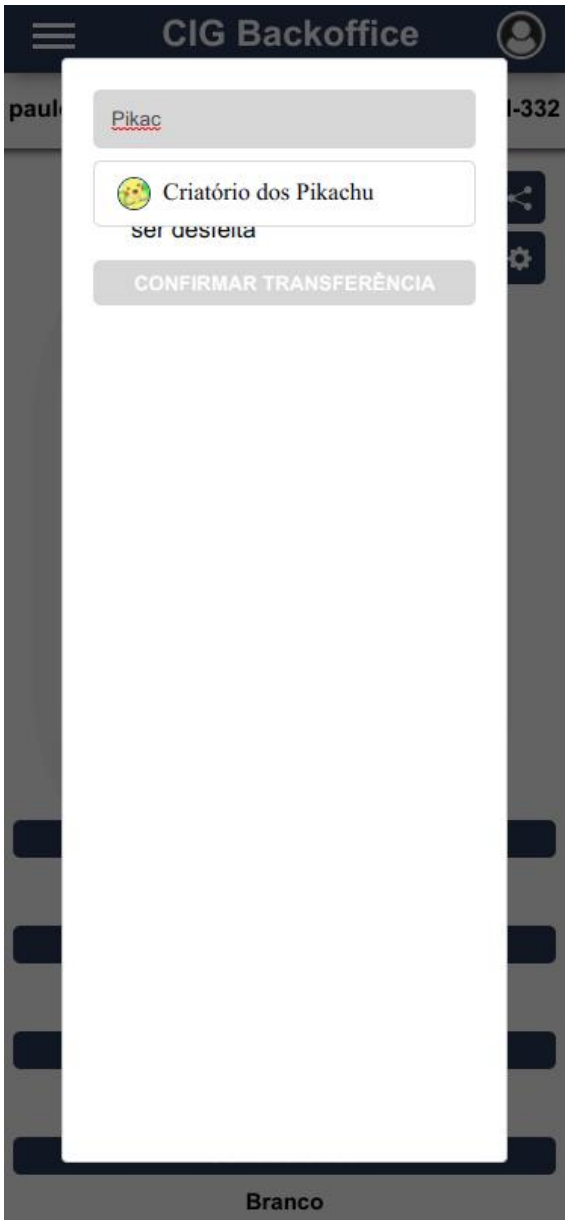

Figura 25 – Área de transferência de criatório com listagem de criatórios exibida.
Fonte: o autor.

Para aves anunciadas, é possível que usuários autenticados façam perguntas no anuncio, o recurso de perguntas será explicado com mais detalhes na sequência.

6.4 ANÚNCIOS

Os anúncios são as entidades principais do lado da plataforma do *Marketplace*. Como já citado, uma ave que acabou de ser cadastrada não está anunciada, para isso é necessário que seja criado um anuncio daquela ave.

Figura 26 – Página da ave do *Marketplace* com foco no botão "Fazer proposta".
Fonte: o autor.

6.4.1 Cadastro de anúncio

Uma ave pode ser anunciada uma vez que não tenha outro anúncio ativo vinculado a ela. Quando um anúncio de ave é feito, a ave passa a aparecer na sessão "À venda" da página do criatório e também aparece como anúncio no *Marketplace* da plataforma.

Na página da ave do *Backoffice*, ao clicar no botão de configurações no canto superior direito e depois em "Anunciar ave", inicia-se o fluxo de criação de anúncio de ave. O usuário precisa informar o preço que será anunciado a ave, e após isso, o anúncio será vinculado à ave. Quando um anuncio é realizado, é feito um registro no histórico que pode ser visualizado na área de histórico na página da ave.

Um anúncio pode ter o valor alterado contanto que não tenha nenhuma proposta confirmada feita a ele, a mesma validação se aplica para o caso do anúncio ser removido. Quando um anúncio de ave é removido, é feito um registro no histórico da ave.

6.4.2 Perguntas em anúncios

Na página da ave no *Marketplace*, quando existe um anúncio vinculado à ave, é exibido uma seção de perguntas, que exibe as perguntas realizadas por criatórios, possivelmente interessados em comprar, e as respostas do criatório anunciante às perguntas realizadas (Figura 27).

Uma vez que a ave é anunciada, ela pode passar a receber perguntas no anúncio, os usuários podem fazer perguntas no anúncio através do *Marketplace*, basta apenas estar logado que é possível fazer uma pergunta.

O criatório anunciante pode visualizar e responder as perguntas realizadas no anúncio pela página da ave no *Backoffice*. As perguntas podem ajudar que os usuários tirem suas dúvidas antes de fazer uma proposta no anúncio.

6.4.3 Página principal de anúncios

O *Marketplace* tem uma página principal que contém os anúncios disponíveis na plataforma. Esses anúncios são divididos entre quatro grupos, cada grupo tem no máximo 30 itens.

Os anúncios são separados entre os grupos de acordo com a sexagem da ave. O clique no botão “ver tudo” leva à página de busca de anúncios com o filtro de sexagem escolhido. O clique no anúncio leva à página do anúncio da ave. O clique na imagem do criatório leva à página do criatório. O anúncio mostra a imagem principal, preço, medida e data de nascimento da ave.

6.4.4 Página de busca de anúncios

A página de busca de anúncios é a página que o usuário pode realizar pesquisas de aves de acordo com o que esteja buscando. O usuário pode selecionar uma opção de ordenação e também é possível utilizar de filtros para deixar a busca mais específica (Figura 28). O usuário também pode clicar no ícone da lupa no cabeçalho e digitar num campo de entrada o texto desejado para realizar a busca.

Os anúncios são listados com base nas opções de filtro/ordenação selecionados. O clique no anúncio leva a página do anúncio. O clique na imagem do criatório leva a página do criatório. O anúncio mostra a imagem principal, preço, medida e data de nascimento da ave. São listados no máximo 30 anúncios, quando o usuário chega até o final da página, devem ser carregados mais 30 anúncios até que não tenham mais anúncios disponíveis a serem listados.

Os anúncios favoritados são listados para o usuário ao selecionar o filtro de favoritos. O filtro de favoritos é mostrado ao usuário somente se ele estiver autenticado.

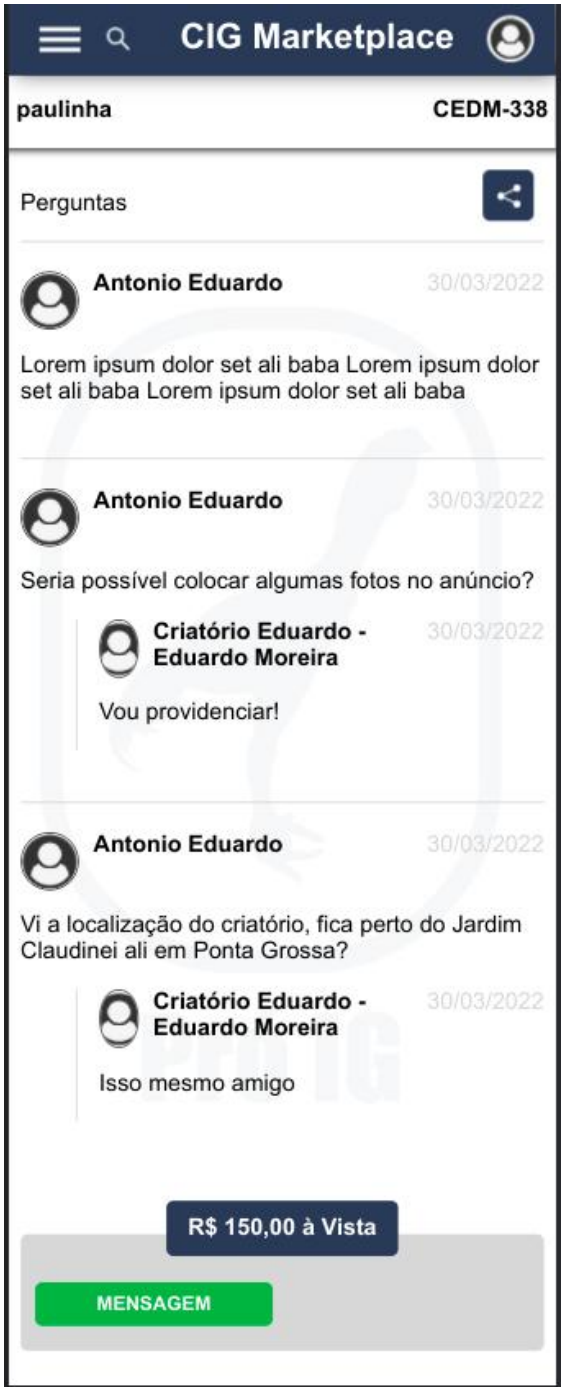

Figura 27 – Página de ave anunciada com foco na área de perguntas.

Fonte: o autor.

6.4.5 Favoritar anúncio

Os anúncios de aves cadastrados na plataforma podem ser favoritados pelos usuários a fim de facilitar a busca do anúncio posteriormente. O ícone de favoritar/desfavoritar anúncio é exibido a usuários autenticados na página principal e na página de busca.

Figura 28 – Páginas de mecânicas de busca de anúncios.
Fonte: o autor.

Como citado na seção anterior, na página de busca de anúncios é possível de utilizar de um filtro que lista somente anúncios favoritados pelo usuário.

6.5 NEGOCIAÇÕES

A entidade de negociação representa o registro de uma negociação que esteja sendo feita entre um criatório comprador e um criatório vendedor. As aves anunciadas na plataforma podem receber negociações de criatórios a qualquer momento que o anúncio esteja ativo.

6.5.1 Criando uma negociação

O usuário pode iniciar uma negociação fazendo uma proposta ao anúncio através da página da ave no *Marketplace*. O clique no botão “Fazer proposta” abre um diálogo perguntando o valor da proposta. Após o valor da proposta ser informado, o usuário pode complementar num campo livre de texto alguma informação a mais em relação a proposta. A única regra para que uma negociação possa ser iniciada na plataforma, é que não exista outra negociação confirmada em andamento.

No caso de uma negociação cadastrada com sucesso, o usuário é redirecionado ao *Backoffice* para a página da negociação que ele acabou de iniciar. O usuário pode acessar essa página de negociação, acessado no menu lateral esquerdo direito a opção “Compras”, que levará o usuário até a página de listagem de negociações que o criatório é o comprador.

O ciclo de vida de uma negociação é orientado a eventos. Uma vez que uma negociação é iniciada, essa negociação pode ter eventos registrados vinculados a ela, que servem para fazer o acompanhamento do ciclo de vida da negociação. Quando a negociação acaba de ser criada, é feito um registro de um evento de “Criação de negociação”. Nessa etapa da negociação o usuário comprador pode somente cancelar a proposta ou aguardar um retorno do criatório vendedor. Uma negociação pode ser cancelada a qualquer momento pelo criador vendedor ou comprador.

6.5.2 Confirmação de negociação

Quando uma ave anunciada recebe uma proposta, o usuário do criatório vendedor pode visualizar a proposta na página de listagem de vendas (Figura 29), clicando na opção do menu lateral esquerdo “Vendas”.

Uma negociação que acabou de ter a proposta realizada pode ser aceita ou negada pelo criatório vendedor na página de visualização de negociação de venda. Se a proposta for confirmada pelo criatório vendedor, será registrado um evento de “Confirmação de negociação”. Quando a negociação é confirmada, significa que o criatório vendedor aceitou a proposta do criatório comprador, e também, para ambos cenários, passa a ser mostrado o botão de “Chamar no WhatsApp” na página de visualização de negociação para criatórios que tenham informação de contato do tipo “WhatsApp” cadastradas.

O usuário do criatório vendedor, quando confirma uma negociação, pode somente fazer o cancelamento da venda ou aguardar a finalização da negociação por parte do comprador. Na prática, seria nesse momento que os criadores teriam que fazer o contato para combinar a forma de pagamento e transporte do animal, que não são responsabilidades da plataforma. A negociação fica no estado de confirmada enquanto não for finalizada ou cancelada.

6.5.3 Finalização de negociação

Uma ave anunciada que contém uma negociação confirmada em andamento permanece vinculada ao criatório vendedor até que seja informado que a negociação foi finalizada por parte do comprador. Quando a negociação é confirmada, na página de visualização de negociação do usuário comprador, é mostrado o botão de finalizar negociação.

Quando o usuário do criatório comprador finaliza a negociação, a ave é transferida de um criatório a outro, o anúncio é removido do *Markteplace*, e é feito o registro de um evento de “Finalização de negociação”. A partir disso a ave passa a estar disponível na listagem de aves do criatório comprador, e é possível visualizar no histórico da ave a transferência de criatório. A página de visualização de negociação passa a não ter mais botões para interação do usuário, e serve somente para visualização do histórico da negociação.

Figura 29 – Página de listagem de negociações não iniciadas pelo usuário.
Fonte: o autor.

6.5.4 Cancelamento de negociação

Uma negociação pode ser cancelada em qualquer momento do ciclo de vida da negociação, isso é possível, pois pode haver um desacordo entre os criadores ou até mesmo surgir um imprevisto como falecimento da ave durante a negociação. Por esse motivo, enquanto uma negociação ainda não foi finalizada ou cancelada, é possível cancelar a negociação por ambas as partes no *Backoffice*. É possível preencher o motivo de cancelamento no momento de cancelar a negociação.

Esse processo de cancelamento também é usado no caso de uma proposta não ser aceita pelo criatório vendedor. Quando uma proposta é cancelada, é feito um registro de evento

de “Cancelamento de negociação” e é exibido um botão de “Refazer proposta” para o usuário que iniciou a negociação na página de visualização de negociação (Figura 30).

Figura 30 – Fluxo de cancelamento de proposta.
Fonte: o autor.

7 CONCLUSÃO

Como principal resultado pretendido com este trabalho têm-se a criação de uma plataforma de comercialização de aves caipiras com rastreabilidade do animal. Diante disso, foi desenvolvida uma plataforma que permite aos usuários gerenciar e anunciar aves de maneira digital. Foi apresentado um *Marketplace* totalmente dedicado para o nicho de aves caipiras que visa facilitar a busca e a comercialização desses animais, além de fornecer aos criadores mais segurança e transparência.

Foi apresentada uma metodologia baseada em Kanban, onde foram definidas tarefas agrupadas por contexto para o desenvolvimento da plataforma. A metodologia foi seguida e mostrou-se ideal para o projeto, dado que a plataforma foi sendo criada aos poucos. Além disso, o Atomic Design, que foi o padrão de construção de componentes escolhido para o desenvolvimento, mostrou-se bastante útil para a questão da padronização e hierarquia entre os componentes. É comum durante o desenvolvimento de um projeto, que o desenvolvedor acabe "se perdendo" nos padrões utilizados na construção da interface, o Atomic Design ajudou muito nessas definições, além é claro, de evitar repetição de código. Como o projeto teve várias aplicações *front-end*, todas elas fizeram uso dos componentes átomos/moléculas/organismos criados.

Os contextos foram separados entre *Marketplace* e *Backoffice*, que juntas compõem a arquitetura da plataforma apresentada. Além disso, foi apresentada a arquitetura e toda estrutura da plataforma, composta por seis microsserviços, dois serviços *Back-end For Front-end* (BFF), quatro pacotes para comunicação entre *back-end* e *front-end*, seis aplicações cliente e uma biblioteca de componentes.

Os trabalhos correlatos citados no Capítulo 2 não possuem recursos que permitem realizar a rastreabilidade do animal, dessa forma, acabam gerando dificuldades organizacionais na comunidade de criadores. O presente trabalho implementou recursos para criação de histórico do animal, para que os criadores possam fazer registros frequentes de vacinação, medição, pesagem e fotos, assim dando mais visibilidade do histórico do animal, e no momento que um criador for comprar um animal, ele tem acesso de todas essas informações detalhadas. O trabalho também pode ser considerado um avanço tecnológico para a comunidade de criadores de aves caipiras, já que agora os criadores podem utilizar da plataforma para expor seus criatórios e animais de maneira digital, tornando-se mais acessíveis através dos buscadores.

A plataforma foi totalmente construída pensando no nicho de aves caipiras, logo, se for bem adotada pelos criadores, pode se tornar até mesmo um parâmetro de confiabilidade na comunidade de criadores. Por fim, os trabalhos citados anteriormente não possuem recurso de árvore genealógica do animal, o que é uma informação bastante utilizada pelos criadores, a plataforma apresentada no trabalho também não possui esse recurso.

Diante do exposto, são elencadas algumas possibilidades para trabalhos futuros, como a publicação do projeto, divulgação da plataforma em redes sociais para atrair usuários, cole-

tar *feedback* afim de fazer possíveis melhorias e ajustes na plataforma e a implementação do recurso de árvore genealógica do animal.

REFERÊNCIAS

ANIMALSFORSALE. **AnimalsForSale**. 2017. Disponível em: https://www.animalsforsale.com.br/. Acesso em: 11 de julho de 2021.

AUTH0. **JWT**. 2022. Disponível em: https://jwt.io. Acesso em: 19 de março de 2022.

AUTH0. **JWT**. 2022. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 19 de março de 2022.

BERSON, A. **Client/server architecture**. [*S.l.*]: McGraw-Hill, Inc., 1996.

BOEG, J. Kanban em 10 passos. **Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt. C4Media**, 2010.

CNA. **PIB do Agronegócio alcança participação de 26,6% no PIB brasileiro em 2020**. 2021. Disponível em: https://www.cnabrasil.org.br/boletins/pib-do-agronegocio-alcanca-participacao-de-26-6-no-pib-brasileiro-em-2020. Acesso em: 29 de junho de 2021.

DEUS, H. G. d. Taxas de eclosão e fertilização em galinhas caipiras fertilizadas pelos métodos de monta natural e de inseminação artificial. Universidade Federal de Uberlândia, 2019.

EMBRAPA. **Brasil é o quarto maior produtor de grãos e o maior exportador de carne bovina do mundo, diz estudo**. 2021. Disponível em: https://bit.ly/3BHLGDt. Acesso em: 01 de julho de 2021.

EMBRAPA. **Embrapa Suínos e Aves**. 2021. Disponível em: https://www.embrapa.br/suinos-e-aves/cias/estatisticas/frangos/mundo. Acesso em: 01 de julho de 2021.

FROST, B. **Atomic design**. [*S.l.*]: Brad Frost Pittsburgh, 2016.

GENARI, J. O. S.; FERRARI, F. C. Times de alto desempenho no contexto das metodologias scrum e kanban. **Revista TIS**, v. 4, n. 3, 2016.

GUIMARÃES, A. C.; SILVA, S. E. P. da. Comparação entre os métodos de 'processo de desenvolvimento de produto'e 'produto mínimo viável'. **Gestão da Produção em Foco Volume 47**, p. 54, 2021.

LOUDON, K. Desenvolvimento de grandes aplicações web. **Revista Telfract**, v. 1, n. 1, 2018.

MERCADOLIVRE. **Mercado Livre**. 2021. Disponível em: https://www.mercadolivre.com.br/. Acesso em: 01 de julho de 2021.

MFRURAL. **MFRural**. 2004. Disponível em: https://www.mfrural.com.br/. Acesso em: 10 de julho de 2021.

MICROSOFT. **Typescript**. 2012. Disponível em: https://www.typescriptlang.org/. Acesso em: 06 de julho de 2021.

NODEJS. 2009. Disponível em: https://nodejs.org/en/. Acesso em: 06 de julho de 2021.

OLX. **OLX**. 2021. Disponível em: https://www.olx.com.br/. Acesso em: 01 de julho de 2021.

RIZO, V. H. M.; SANTO, F. do E. Migração de partes de uma aplicação desktop para o formato de api rest: estudo de caso. **Revista Interface Tecnológica**, v. 17, n. 1, p. 118–128, 2020.

RODRIGUES, R. Terra, gente e tecnologia impulsionam crescimento do agronegócio brasileiro. **Revista USP**, n. 64, p. 50–57, 2005.

SANTANA, K. O. d. Metodologias ágeis para desenvolvimento de software: um exemplo real de uso da metodologia scrum. Universidade Federal Fluminense, 2017.

SATHYAN, S. *et al.* Soft-switched interleaved dc/dc converter as front-end of multi-inverter structure for micro grid applications. **IEEE Transactions on Power Electronics**, IEEE, v. 33, n. 9, p. 7645–7655, 2017.

VERCEL. **NextJS**. 2021. Disponível em: https://nextjs.org/. Acesso em: 25 de agosto de 2021.

VIEIRA, F. C.; BACCILI, V. C. L.; DELFINO, S. R. Aplicabilidade da tecnologia da informação no agronegócio. **RETEC-Revista de Tecnologias**, v. 4, n. 1, 2011.