UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ANTONIO EDUARDO MOREIRA

CIG: PLATAFORMA PARA COMERCIALIZAÇÃO DE AVES CAIPIRAS

MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

GUARAPUAVA
2021

ANTONIO EDUARDO MOREIRA

CIG: PLATAFORMA PARA COMERCIALIZAÇÃO DE AVES CAIPIRAS

Monografia de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Sistemas para Internet – TSI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Andres Jessé Porfirio

GUARAPUAVA
2021

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 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

MOREIRA, Antonio Eduardo. CIG: Plataforma para Comercialização de Aves Caipiras. 2021. 20 f. Monografia de Trabalho de Conclusão de Curso de graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2021.

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. Além disso, tais aplicativos originalmente não foram criados para esse fim, então existem algumas limitações, como por exemplo, em um *marketplace* (comércio online) convencional, quando um animal é vendido, perde-se o registro dele, enquanto que na plataforma proposta, será armazenado todo histórico de criadores que foram donos do animal, e também será possível fazer registros sobre o animal afim de passar mais segurança ao comprador.

Palavras-chave: Aves Caipiras. Marketplace. Índio Gigante.

ABSTRACT

MOREIRA, Antonio Eduardo. CIG: Platform for Commercialization of Free-range Birds. 2021. 20 f. Monografia de Trabalho de Conclusão de Curso de graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2021.

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, besides that, 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. Besides that, these applications weren't created to these market niche, so exists some limitations, such as, in a conventional marketplace, when an animal is sold, all the registers from the animal is lost, while in the proposed platform, the history of bird breeders will be stored, and also will be possible to make registers about the animal in order to give more security to the buyer.

Keywords: Free-range Birds. Marketplace. Índio Gigante.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 DESCRIÇÃO GERAL

Em 2020, o agronegócio teve participação em 26,6% no 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. 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 visa facilitar a busca por aves caipiras, que são criadas em criatórios bem avaliados no mercado e que desenvolvem o trabalho de gestão e acompanhamento dos animais com assiduidade. Desta forma, passando

transparência e segurança na comercialização das aves Índio Gigante através de registros de anilha das aves e atualizações de vacinação que serão feitas na plataforma. Além disso, quando um criador comprar uma ave de outro criador na plataforma, todo o histórico de atualizações da ave será mantido e disponível para visualização dos criadores.

Até o momento, não existe uma plataforma específica para esse fim, logo, a CIG visa ser a pioneira do mercado, abrindo 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.

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 também informações atualizadas, fotos e vídeos dos animais. Diante disso, justifica-se a necessidade de criação de 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 OBJETIVO E ATIVIDADES

1.2.1 Objetivo geral

Desenvolver uma plataforma para comercialização de aves caipiras.

1.2.2 Atividades

- Criar uma API[1] para registrar e gerenciar usuários;
- Criar uma API para registrar e gerenciar criatórios;
- Criar uma API para registrar e gerenciar aves;
- Criar uma API para registrar e gerenciar registros de aves;
- Criar uma API para registrar e gerenciar anúncios;
- Criar uma API para criadores visualizarem suas vendas;
- Criar uma API para criadores fazerem perguntas e respostas sobre os anúncios de criatórios;
- Criar uma API para criadores fazerem e visualizarem compras de aves;
- Criar uma API para criadores avaliarem o criatório depois de uma compra;
- Criar um sistema web com interface que permita que o usuário gerencie as informações do criatório, aves, anúncios e responda as perguntas dos anúncios;

[1] Interface de Programação de Aplicações, do inglês *Application Programming Interface*.

- 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 a revisão de literatura; A metodologia é exposta no Capítulo 3; É apresentado os resultados preliminares no Capítulo 4; Por fim, no Capítulo 5 são apresentadas as considerações finais.

2 REVISÃO DE LITERATURA

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

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, e 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.

2.1 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 afim 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.

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, categorizar esses produtos, e outros usuários podem buscar produtos 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.

2.2 REFERENCIAL TEÓRICO

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. 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 continua 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.

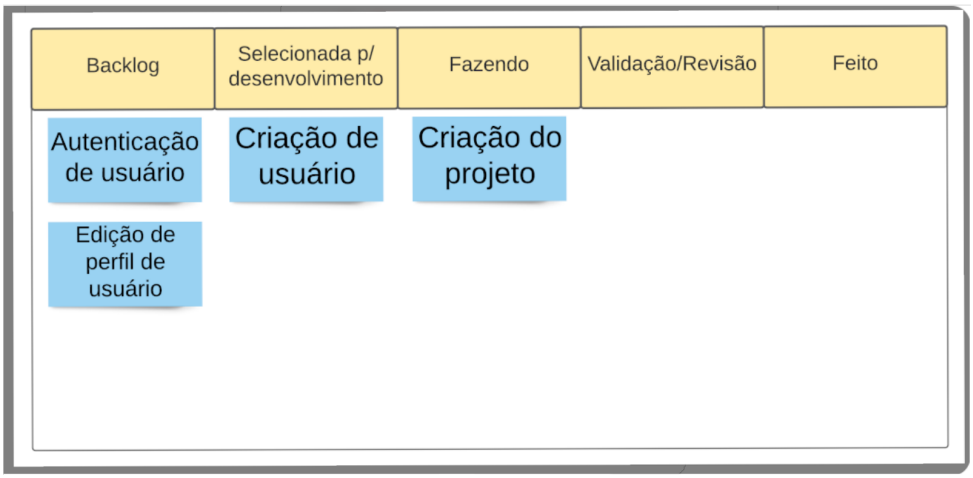

Figura 1 – Exemplo de Quadro Kanban.
Fonte: o autor.

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. 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).

Além das metodologias ágeis, 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 HTTP, que permite a transferência de dados hipertexto (RIZO; SANTO, 2020).

Neste 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).

Ainda no *front-end*, 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

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

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.

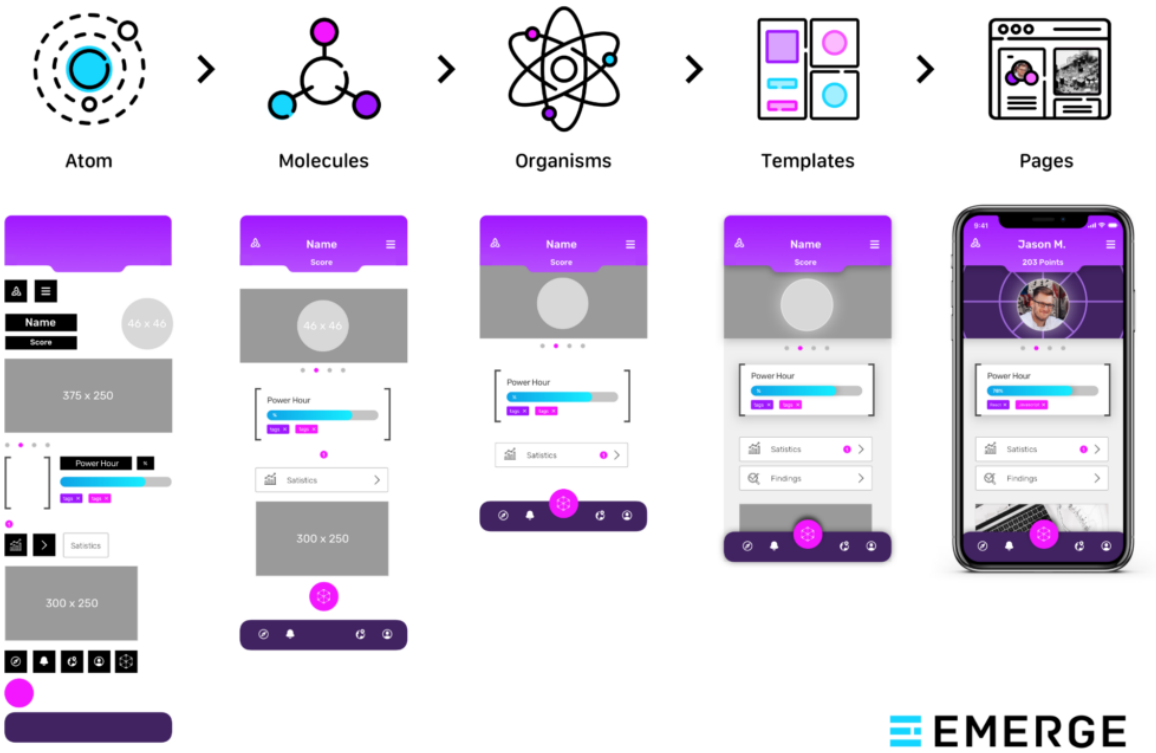

Figura 2 – Exemplo de Categorização de Componentes no Atomic Design.
Fonte: Emerge Interactive[a]

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

3 METODOLOGIA DE DESENVOLVIMENTO

Esse capítulo apresenta os procedimentos metodológicos que serão usados no desenvolvimento do projeto. Na seção 3.1 será apresentado o modelo de desenvolvimento que será aplicado.

3.1 MODELO DE DESENVOLVIMENTO

O sistema será desenvolvido no modelo MVP[1], que visa entregar o maior valor no menor espaço de tempo. A ideia é fazer a primeira versão da plataforma, como um experimento que contenha as funcionalidades essenciais para o funcionamento do sistema, afim de confirmar a capacidade de crescimento do produto. O MVP da plataforma deve permitir que um criador realize a compra de aves e acompanhe seus anúncios, com o decorrer do desenvolvimento da plataforma, as demais funcionalidades vão sendo acrescentadas separadamente (RIZO; SANTO, 2020).

Diante dos objetivos específicos[2], a princípio foi gerado o seguinte *backlog*:

- Gestão de plantel e anúncios (*Backoffice*):
  - Criar uma API para registrar e gerenciar usuários:
    - Configuração e criação da aplicação servidor;
    - Criar rota HTTP para criação de usuário;
    - Criar rota HTTP para autenticação do usuário;
    - Criar rota HTTP para editar as informações do usuário;
    - Criar rota HTTP para inativar conta do usuário;
    - Criar rota HTTP para obter as informações do usuário;
  - Criar uma API para registrar e gerenciar criatórios;
    - Configuração e criação da aplicação servidor;
    - Criar rota HTTP para criação de criatório;
    - Criar rota HTTP para editar as informações do criatório;
    - Criar rota HTTP para inativar conta do criatório;
    - Criar rota HTTP para obter as informações do criatório;
  - Criar uma API para registrar e gerenciar aves;
    - Criar rota HTTP para criação de ave;
    - Criar rota HTTP para editar as informações da ave;
    - Criar rota HTTP para inativar a ave;
    - Criar rota HTTP para obter as todas aves de um criatório;
    - Criar rota HTTP para obter as informações de uma ave em específico;
  - Criar uma API para registrar uma informação no histórico da ave;

[1] Produto Viável Mínimo, do inglês *Minimum Viable Product*.
[2] Considerando o escopo da proposta, foram definidos os dois primeiros.

        * Criar rota HTTP para criação de um registro no histórico da ave;
        * Criar rota HTTP para obter o histórico de uma ave em específico;
        * Criar rota HTTP para editar o registro no histórico da ave;
        * Criar rota HTTP para obter um registro do histórico em específico;
    - Criar uma API para interações no *Backoffice*:
        * Configuração e criação da aplicação servidor;
        * Criar rota HTTP para autenticação de usuário;
        * Criar rota HTTP para editar informações do criatório;
        * Criar rota HTTP para criação de ave;
        * Criar rota HTTP para edição de ave;
        * Criar rota HTTP para inativar ave;
        * Criar rota HTTP para criação de um registro no histórico da ave;
        * Criar rota HTTP para editar um registro no histórico da ave;
        * Criar rota HTTP para criação de um anúncio de ave;
        * Criar rota HTTP para editar informações de um anúncio de ave;
        * Criar rota HTTP para criação de uma resposta em uma pergunta;
    - Criar uma API para exibição de conteúdo no *Backoffice*:
        * Criar rota HTTP para obter informações da página de criatório;
        * Criar rota HTTP para obter informações de uma ave;
        * Criar rota HTTP para obter todas aves;
        * Criar rota HTTP para obter os anúncios;
        * Criar rota HTTP para obter informações de um anúncio;
        * Criar rota HTTP para obter todos anúncios;
        * Criar rota HTTP para obter todas perguntas de um anúncio;
        * Criar rota HTTP para obter todas vendas;
    - 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 usuário;
        * Criar página para editar informações do criatório;
        * Criar página para cadastro 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 anúncios;
- Plataforma para compras (*Marketplace*):
    - Criar uma API para registrar e gerenciar anúncios:

    - Configuração e criação da aplicação servidor;
    - Criar rota HTTP para criação de comerciante;
    - Criar rota HTTP para criação de anúncio;
    - Criar rota HTTP para inativar anúncio;
    - Criar rota HTTP para obter todos anúncios de um comerciante;
    - Criar rota HTTP para editar informações de um anúncio;
    - Criar rota HTTP para criação de uma pergunta no anúncio;
    - Criar rota HTTP para criação de uma resposta em uma pergunta de anúncio;
- Criar uma API para compras e vendas:
    - Configuração e criação da aplicação servidor;
    - Criar rota HTTP para criação de uma compra;
    - Criar rota HTTP para edição de status da compra;
    - Criar rota HTTP para obter todas compras de um comerciante;
    - Criar rota HTTP para obter todas vendas de um comerciante;
    - Criar rota HTTP para criação de avaliação de uma compra realizada;
- Criar uma API para interações no *Marketplace*:
    - Configuração e criação da aplicação servidor;
    - Criar rota HTTP para criação de usuário/criatório;
    - Criar rota HTTP para autenticação de usuário;
    - Criar rota HTTP para criação de uma pergunta no anúncio;
    - Criar rota HTTP para criação de uma compra;
- Criar uma API para exibição de conteúdo no *Marketplace*:
    - Criar rota HTTP para obter os anúncios;
    - Criar rota HTTP para obter informações da página de criatório;
    - Criar rota HTTP para obter informações de um anúncio;
    - Criar rota HTTP para obter informações de uma ave;
- 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 visualizar um criatório;
    - Criar página para visualizar uma ave;
    - Criar página para visualizar um anúncio.

4 RESULTADOS PRELIMINARES

Nesse capítulo serão apresentados os resultados preliminares no desenvolvimento do projeto, tais como a organização do desenvolvimento e a arquitetura cliente-servidor, que atende todas necessidades do projeto. A plataforma terá uma arquitetura modular, como descrito no capítulo 2.2, que permite uma fácil expansão e manutenção do projeto.

A arquitetura do *back-end* será formada por seis aplicações servidores, que se comunicarão entre si para funcionamento da plataforma, e dois bancos de dados. Enquanto que o *front-end* terá duas aplicações cliente, dois pacotes para comunicação com as aplicações servidores e um pacote para os componentes da interface da plataforma. Arquitetura da plataforma é exemplificada na Figura 3.

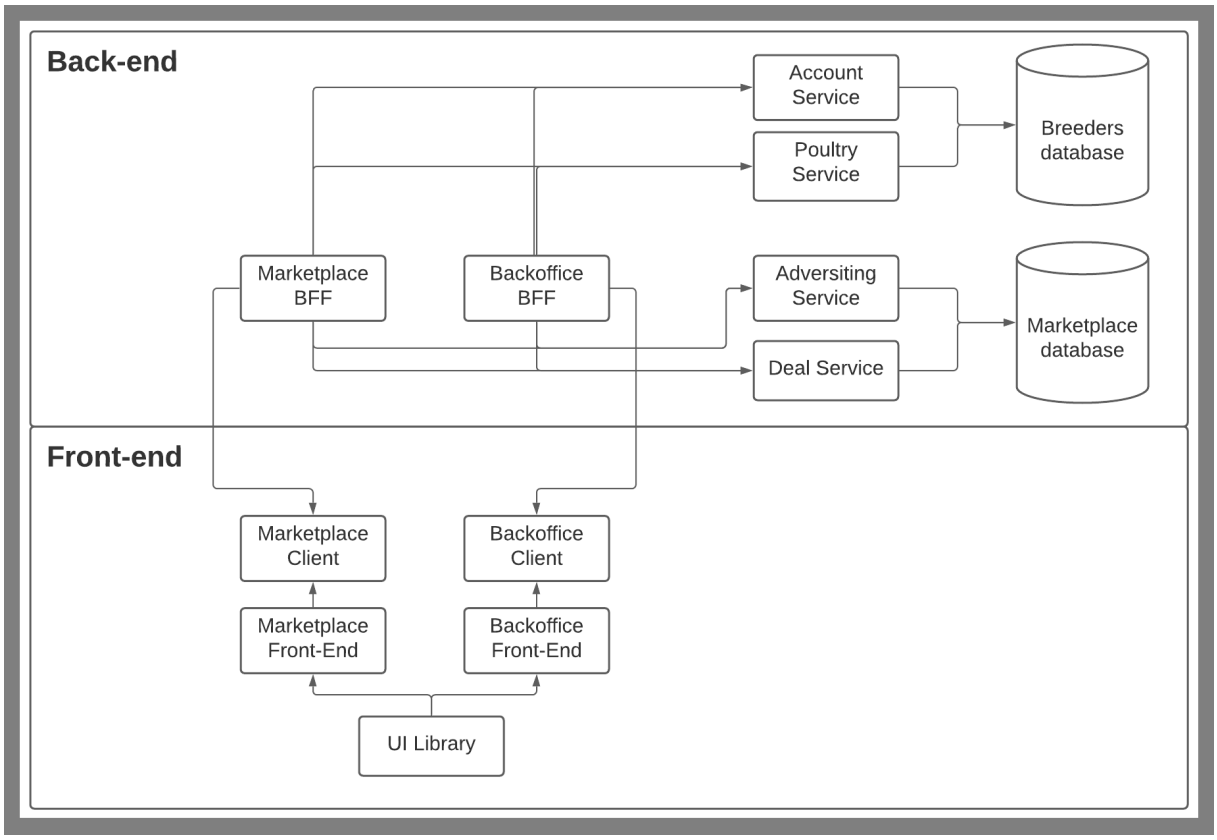

Figura 3 – Diagrama da Arquitetura.
Fonte: o autor.

4.1 ARQUITETURA *BACK-END*

O *back-end* da plataforma será feito com NodeJS e Typescript, a arquitetura será formada por seis aplicações, dentre elas, dois BFF[1]s e quatro micro-serviços. Essas aplicações

[1] *Back-end* para *Front-end*, do inglês *Back-end For Front-end*.

serão APIs que se comunicarão entre si através de requisições HTTP. Cada aplicação terá uma função dentro da arquitetura, que será apresentado na sequência, e todas funcionando juntas compõem a plataforma.

Para a criação das aplicações, foi elaborado um *template* de projeto (MOREIRA, 2021b). Esse template é uma aplicação NodeJS, que contém um servidor *back-end* configurado com Express e Typescript. Esse servidor pode ser 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, terão documentações para apresentar todas rotas HTTP, essas documentações terão uma rota específica na aplicação para visualização. Pode-se observar um exemplo de documentação na Figura 4.

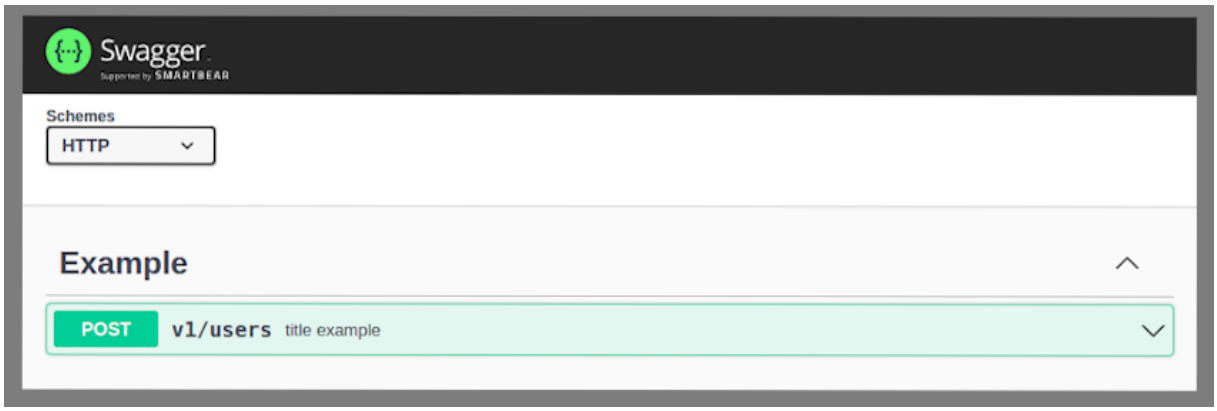

Figura 4 – Exemplo de Documentação de API.
Fonte: o autor.

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 (MOREIRA, 2021a). Essa biblioteca pode ser facilmente instalada como pacote NPM (Node Package Manager), 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 instalação da biblioteca e criação de aplicação pode ser visto na Figura 5.

Afim de permitir uma fácil expansão do projeto no futuro, a arquitetura está sendo 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 serão tratados apartadamente na arquitetura, visando que, se no futuro algum outro nicho do agronegócio passe a ser atendido pela plataforma,

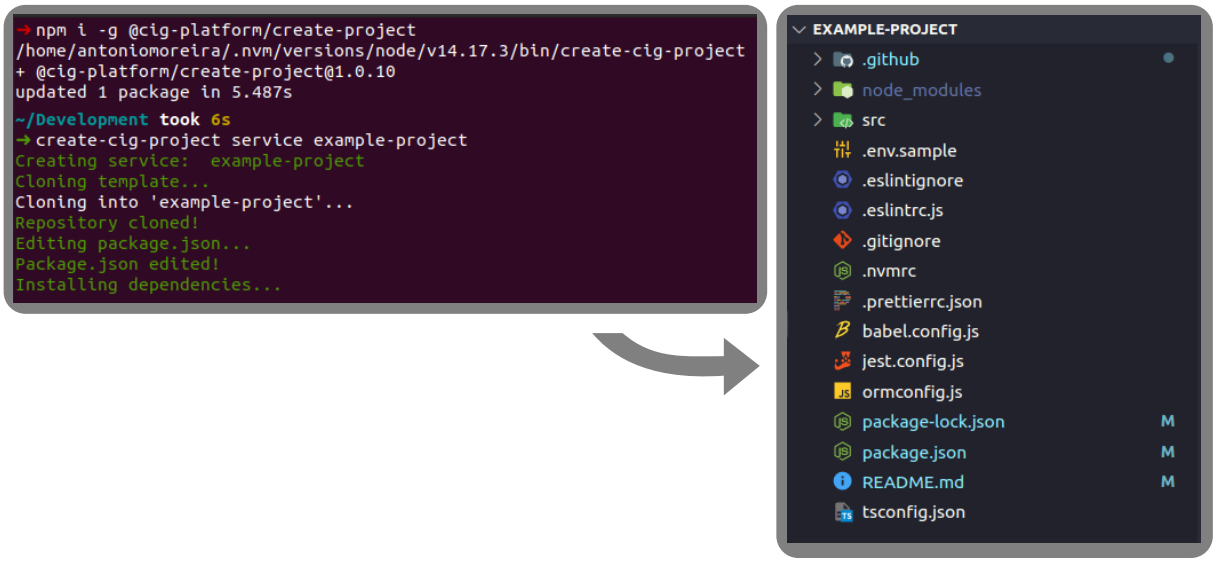

Figura 5 – Exemplo de Criação de Projeto via Linha de Comando.
Fonte: o autor.

não haja problemas com essa nova integração. Então, de um lado, têm-se o módulo de aves caipiras, que tratará todo gerenciamento de criatórios, aves, e registros de aves. Por outro lado, o módulo do *marketplace* tratará dos anúncios e compras na plataforma.

4.1.1 Micro-serviços

Um micro-serviço é uma aplicação que tratará de um contexto específico da plataforma. Os micro-serviços serão aplicações que funcionaram como APIs que se comunicarão entre si através de requisições HTTP. Os micro-serviços vão estar dentro de uma VPN[2], cujas rotas não terão autenticação e poderão ser acessadas apenas dentro da própria VPN.

O primeiro micro-serviço a ser construído será o *Account Service*, responsável por todo contexto de usuário na plataforma. A API deste micro-serviço terá rotas para criação, edição e autenticação de usuários. O *Poultry Service* irá tratar do contexto dos criatórios, aves, e registros de aves. Um criatório será vinculado a um usuário, uma ave será vinculada a um criatório, e um registro de ave será vinculado a uma ave. Logo, esse micro-serviço vai tratar de todas particularidades do nicho de ave caipira, se num futuro a plataforma for expandida para um novo nicho, um novo micro-serviço 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 micro-serviços estarão conectados a um mesmo banco de dados, 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 micro-serviços não devem ter conhecimento do *marketplace*, eles são o lado da plataforma que devem apenas lidar com o gerenciamento das aves e as demais informações citadas.

[2] Rede privada virtual, do inglês *Virtual Private Network*

O *Adversiting Service* e o *Deal Service* serão micro-serviços que irão tratar o contexto do *marketplace*. No momento que um criatório registrar uma ave na plataforma, essa ave não estará instantaneamente disponível para compra no *marketplace*, para que a ave passe a ser exibida no *marketplace*, o criatório deverá 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 registrar uma ave na plataforma, ele irá interagir com o *Poultry Service*, porém quando esse usuário for criar um anúncio para essa ave, todas informações do anúncio como preço e comentários serão gerenciados pelo *Adversiting 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 micro-serviço, que pode facilmente ser acoplado a outras aplicações, pois tratará de anúncios genéricos no *marketplace*.

Toda parte de compra, venda e acompanhamento de compra será encapsulado no micro-serviço *Deal Service*. Esse micro-serviço se conectará ao mesmo banco de dados do *Adversiting Service*. O serviço fará todas validações necessárias no momento da compra e também disponibilizará através da API, a exibição das vendas realizadas por determinado criatório ou usuário.

4.1.2 BFFs

BFF é uma aplicação *back-end* que é construída para servir de API para uma aplicação *front-end*. Os micro-serviços anteriormente citados, são as aplicações que vão conter as regras de negócios da plataforma. As aplicações *front-end* não podem se comunicar diretamente com os micro-serviços, por não estarem dentro da VPN. Então, diferente dos micro-serviços, os BFFs não vão rodar dentro da VPN. Um BFF servirá de ponte entre a aplicação *front-end* e os micro-serviços. O BFF será o responsável por receber todos dados e manusear entre as aplicações *back-end*, como por exemplo, quando um usuário realizar a compra de uma ave, todos os micro-serviços serão consultados: o *Account Service* será consultado para fazer a autenticação do usuário; o *Poultry Service* será consultado para verificar as informações do criatório e ave; o *Adversiting Service* será consultado para verificar as informações do anúncio; e por fim, o *Deal Service* será consultado para registrar a compra. Uma aplicação BFF será a responsável por fazer essas interações com os micro-serviços e manter o bom funcionamento da plataforma.

O *Marketplace BFF* será a aplicação que servirá todos dados a aplicação *front-end* do *marketplace*. A API dessa aplicação terá autenticação em todas as rotas, e deverá disponibilizar rotas que permitam: visualização de compras realizadas, visualização de anúncios, visualização de criatório, compra de anúncio, fazer pergunta em anúncio.

O *Backoffice BFF* fornecerá uma API para a aplicação *front-end* do *backoffice*. O *backoffice* será a área da plataforma que o usuário poderá cadastrar as aves, registros de aves e anúncios. O BFF deve ter rotas que permitam: visualização de anúncios, visualização de aves, visualização de informações do criatório, edição de informações do criatório, responder

perguntas em anúncio, e criar anúncio.

4.1.3 Banco de dados

O banco de dados utilizado será o PostgreSQL. Na Figura 6 é possível ver a modelagem inicial do banco de dados.

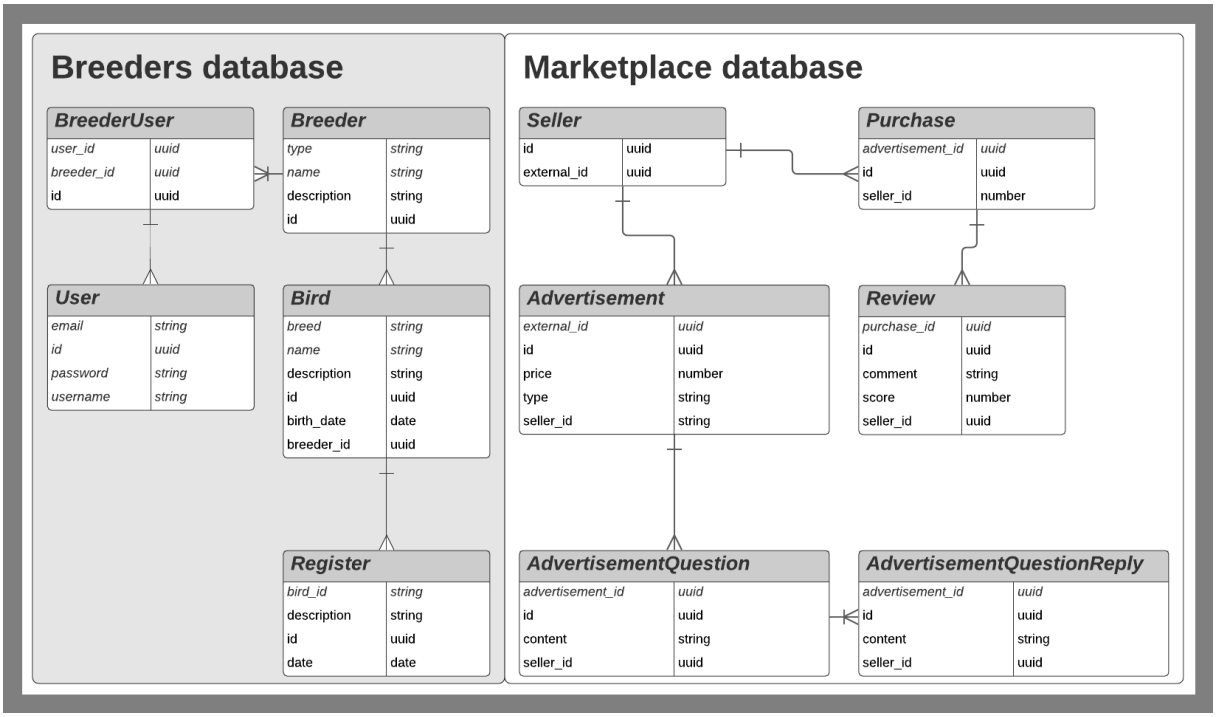

Figura 6 – Prototipação do Banco de Dados.
Fonte: o autor.

As tabelas do lado direito da figura representam o banco de dados que os micro-serviços *marketplace* (*Adversiting Service* e *Deal Service*) irão consumir. Enquanto que do lado esquerdo da figura são as tabelas que estarão no banco de dados que os micro-serviços *backoffice* (*Account Service* e *Poultry Service*) irão consumir.

4.2 ARQUITETURA *FRONT-END*

O *front-end* da plataforma será formado por duas aplicações e três bibliotecas. Seguindo o mesmo contexto modular anteriormente citado, uma das aplicações *front-end* irá tratar de toda exibição de conteúdo do *marketplace*. Enquanto que outra aplicação conterá todo o painel administrativo do criador, onde será possível registrar aves, anúncios, registros de aves e editar informações do criatório e perfil. Dentre as bibliotecas, duas delas serão utilizadas para comunicação com os BFFs, e a outra será uma biblioteca de componentes baseados no Atomic Design.

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

A comunicação entre as aplicações *front-end* e *back-end* será feita através de requisições HTTP. As APIs dos BFFs têm como objetivo facilitar que as aplicações *front-end* consumam os dados do *back-end* sem a necessidade de acesso direto aos seus micro-serviços. Serão criadas duas bibliotecas NPM para fazer a comunicação entre as aplicações *front-end* e *back-end*: Uma delas servirá para fazer interações com o *Marketplace BFF*, enquanto que a outra permitirá fazer interações com o *Backoffice BFF*.

A criação de uma biblioteca, além de apartar o contexto das interações com os BFFs 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 algum BFF, podem facilmente fazer uso de uma das bibliotecas.

A estrutura das bibliotecas devem servir como um cliente de API, que terá métodos para executar todas requisições para os BFFs.

4.2.2 Componentes

Será criada uma biblioteca de componentes com React e Typescript, essa componentização será feita seguindo os princípios do Atomic Design. A biblioteca terá uma lista de componentes agnósticos que servirão para a construção da interface da plataforma. Como trata-se de uma biblioteca apartada das aplicações *front-end*, a mesma pode ser usada tanto pela aplicação *front-end* do *Backoffice* quanto do *Marketplace*.

4.2.3 Aplicações

A aplicação *front-end* do *Marketplace* será feita com React, NextJS e Typescript. A aplicação irá utilizar de uma biblioteca para comunicação com o *Marketplace BFF*, que por sua vez, disponibilizará os dados a serem exibidos. Essa aplicação conterá as página de: login, cadastro de usuário/criatório, exibição de anúncios, compra, histórico de compras.

A aplicação *front-end* do *Backoffice* será feita com React e Typescript, e também utilizará de uma biblioteca para comunicar-se com o *back-end*, porém, desta vez com o *Backoffice BFF*. Essa aplicação conterá todas as páginas do painel administrativo do criatório.

5 CONCLUSÃO

O resultado pretendido com este trabalho é facilitar a busca por aves caipiras através de uma plataforma *marketplace*. Também pretende-se gerar mais segurança e transparência aos criadores que comercializam aves de elite a partir do histórico da ave que ficará armazenado na plataforma, evitando que criadores caiam em negociações fraudulentas.

Foi apresentada uma metodologia baseada em Kanban, onde foram definidas tarefas agrupadas por contexto para o desenvolvimento da plataforma. Os contextos foram separados entre *Marketplace* e *Backoffice*, que juntas compõem a arquitetura da plataforma apresentada.

Como resultados preliminares, foi apresentada a arquitetura do sistema, composta por quatro micro-serviços, dois serviços *Back-end For Front-end* (BFF), dois pacotes para comunicação entre back-end e front-end, duas aplicações cliente e uma biblioteca de componentes.

A próxima etapa no desenvolvimento do projeto é a criação dos *mockups* dos componentes que irão compor a biblioteca de componentes. Após isso, será dado o início ao desenvolvimento dos micro-serviços e *BFF's* da plataforma.

Referências

ANIMALSFORSALE. **AnimalsForSale**. 2017. Disponível em: <https://www.animalsforsale.com.br/>. Acesso em: 11 de julho de 2021. Citado na página 4.

BERSON, A. **Client/server architecture**. [S.l.]: McGraw-Hill, Inc., 1996. Citado na página 6.

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. Citado na página 6.

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. Citado na página 1.

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. Citado na página 1.

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. Citado na página 1.

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. Citado na página 1.

FROST, B. **Atomic design**. [S.l.]: Brad Frost Pittsburgh, 2016. Citado na página 7.

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. Citado na página 5.

LOUDON, K. Desenvolvimento de grandes aplicações web. **Revista Telfract**, v. 1, n. 1, 2018. Citado na página 6.

MERCADOLIVRE. **Mercado Livre**. 2021. Disponível em: <https://www.mercadolivre.com.br/>. Acesso em: 01 de julho de 2021. Citado 2 vezes nas páginas 2 e 5.

MFRURAL. **MFRural**. 2004. Disponível em: <https://www.mfrural.com.br/>. Acesso em: 10 de julho de 2021. Citado na página 4.

MICROSOFT. **Typescript**. 2012. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 06 de julho de 2021. Citado na página 6.

MOREIRA, A. E. **Projeto CLI**. 2021. Disponível em: <https://github.com/edumoreira1506/create-cig-project>. Acesso em: 08 de agosto de 2021. Citado na página 13.

MOREIRA, A. E. **Template de Projeto**. 2021. Disponível em: <https://github.com/edumoreira1506/cig-service-template>. Acesso em: 08 de agosto de 2021. Citado na página 13.

NODEJS. 2009. Disponível em: <https://nodejs.org/en/>. Acesso em: 06 de julho de 2021. Citado na página 6.

OLX. **OLX**. 2021. Disponível em: <https://www.olx.com.br/>. Acesso em: 01 de julho de 2021. Citado 2 vezes nas páginas 2 e 4.

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. Citado 2 vezes nas páginas 6 e 9.

RODRIGUES, R. Terra, gente e tecnologia impulsionam crescimento do agronegócio brasileiro. **Revista USP**, n. 64, p. 50–57, 2005. Citado na página 1.

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. Citado na página 4.