UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

Francisco das Chagas Teixeira Filho

Desenvolvimento de um *Backend* para Aplicação de Adoção de Animais para o Grupo Unidos pelas Patinhas - Pau dos Ferros/RN

PAU DOS FERROS - RN
2025

Francisco das Chagas Teixeira Filho

Desenvolvimento de um *Backend* para Aplicação de Adoção de Animais para o Grupo Unidos pelas Patinhas - Pau dos Ferros/RN

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel Interdisciplinar em Ciência e Tecnologia.

Orientador: Reudismam Rolim de Sousa, Prof. Dr.

PAU DOS FERROS - RN

2025

F478d

Filho, Francisco das Chagas Teixeira.
Desenvolvimento de um backend para aplicação de adoção de animais do grupo Unidos pelas Patinhas / Francisco das Chagas Teixeira Filho. - 2025.
50 f. : il.

Orientador: Reudismam Rolim de Sousa.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência e Tecnologia, 2025.

1. Backend. 2. API RESTful. I. Sousa, Reudismam Rolim de, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

Francisco das Chagas Teixeira Filho

Desenvolvimento de um *Backend* para Aplicação de Adoção de Animais para o Grupo Unidos pelas Patinhas - Pau dos Ferros/RN

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel Interdisciplinar em Ciência e Tecnologia.

Defendida em: **05/08/2025**.

BANCA EXAMINADORA

_____
Reudismam Rolim de Sousa, Prof. Dr. - (UFERSA)
Presidente

_____
Hortência Pessoa Rêgo Gomes, Prof. Me. - (UFERSA)
Membro Examinador

_____
Bruno Borges da Silva, Prof. Bel. - (UFERSA)
Membro Examinador

Dedico este trabalho a todos os professores e colegas do curso de Ciência e Tecnologia da Universidade Federal Rural do Semi-Árido (UFERSA), pelos quais nutro grande respeito e admiração. Sinto-me honrado por ter feito parte desta jornada ao lado de vocês.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por me conceder força, proteção e fé em todos os momentos, permitindo-me enfrentar os desafios com perseverança.

À minha mãe e aos meus irmãos pequenos, minha eterna gratidão por serem fonte diária de motivação e inspiração para que eu nunca deixe de buscar crescimento pessoal e profissional.

Ao meu pai, deixo um agradecimento especial por ser um exemplo constante de retidão, trabalho e caráter.

Aos meus irmãos Fernando e Tamires, mesmo distantes fisicamente, continuam sendo presença importante na minha vida. Agradeço pelo carinho e pelo vínculo que resiste ao tempo e à distância.

À minha avó, por todas as orações e palavras de fé que me fortaleceram ao longo da caminhada. Aos meus tios, tias e primos, agradeço pelo carinho e apoio ao longo desta jornada.

Aos meus amigos, de perto e de longe, que estiveram presentes nos momentos mais importantes.

A todos os professores que contribuíram para a minha formação, meu reconhecimento e apreço. Cada aula, cada conselho, cada partilha de conhecimento foi essencial para o meu crescimento acadêmico.

Por fim, agradeço de maneira especial ao meu professor orientador, Reudismam Rolim de Sousa, pelo apoio constante, pela paciência e pelas valiosas contribuições que foram fundamentais para a realização deste trabalho.

RESUMO

O crescente desafio do abandono de animais demanda ações eficazes por parte de Organizações Não Governamentais (ONGs), que frequentemente enfrentam limitações em seus processos de gestão. O grupo “Unidos pelas Patinhas”, de Pau dos Ferros - RN, possui uma aplicação móvel com *frontend* finalizado, porém, inoperante devido à ausência de um sistema de *backend* para gerenciar os dados e a lógica de negócio. Este trabalho aborda o desenvolvimento de um sistema de *backend* robusto e escalável para suprir esta lacuna. O objetivo principal é implementar uma API RESTful completa, seguindo o contrato definido pela aplicação cliente, para gerenciar o cadastro de animais, usuários, solicitações de adoção e doações. A metodologia adotada baseia-se em uma arquitetura em camadas (*Controller-Service-Repository*), garantindo a separação de responsabilidades e a manutenibilidade do código. As tecnologias empregadas incluem Node.js para o ambiente de execução, o *framework* Express.js, para a construção da API, Prisma como ORM para a interação com um banco de dados MySQL, e Jest para a implementação de testes unitários e de integração. Adicionalmente, a aplicação e seus serviços foram containerizados com Docker e Docker Compose, assegurando a consistência e a portabilidade do ambiente. Os resultados demonstram a entrega de uma API funcional, validada por uma suíte de testes abrangente e pronta para integração com o *frontend*, viabilizando a digitalização dos processos da ONG e potencializando seu impacto social.

Palavras-chave: desenvolvimento backend; node.js; api restful; adoção de animais; engenharia de software.

ABSTRACT

The growing challenge of animal abandonment demands effective actions from Non-Governmental Organizations (NGOs), which often face limitations in their management processes. The NGO “Unidos pelas Patinhas”, from Pau dos Ferros - RN, has a mobile application with a finished frontend, but it is inoperative due to the absence of a backend system to manage data and business logic. This final undergraduate project addresses the development of a robust and scalable backend system to fill this gap. The main objective is to implement a complete RESTful API, following the contract defined by the client application, to manage the registration of animals, users, adoption requests, and donations. The methodology adopted is based on a layered architecture (Controller-Service-Repository), ensuring the separation of concerns and code maintainability. The technologies employed include Node.js for the runtime environment, the Express.js framework for API construction, Prisma as the ORM for interaction with a MySQL database, and Jest for the implementation of unit and integration tests. Additionally, the application and its services were containerized with Docker and Docker Compose, ensuring environment consistency and portability. The results demonstrate the delivery of a functional API, validated by a comprehensive test suite and ready for integration with the frontend, enabling the digitalization of the NGO's processes and enhancing its social impact.

Keywords: backend development; node.js; restful api; animal adoption; software engineering.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Este projeto apresenta a criação da estrutura de *backend* para uma aplicação destinada à administração da adoção de pets do grupo "Unidos pelas Patinhas", na cidade de Pau dos Ferros/RN. O primeiro capítulo visa situar a investigação, iniciando com uma reflexão sobre o papel social da tecnologia e da universidade, para então delinear o problema específico, a justificativa do projeto e os objetivos que nortearam a implementação do artefato tecnológico em questão.

1.1 Tecnologia, universidade e o desafio social

Embora não exista a pretensão de enveredar pelos complexos caminhos da construção intelectual acerca da tecnologia e sua intersecção com a condição humana, faz-se necessário destacar alguns marcos filosóficos e sociais que despertaram o interesse e fundamentam a abordagem deste trabalho. Adota-se aqui o princípio humanista de que a tecnologia, em sua essência, não é um fim em si, mas uma ferramenta cujo valor se manifesta em sua capacidade de servir ao desenvolvimento humano e responder às questões mais elementares da sociedade.

Jonas (2006) retrata aspectos éticos em relação ao mundo tecnológico. É a partir dessa premissa que a universidade, como agente de transformação social, assume um papel central na formação de profissionais que, mesmo em áreas majoritariamente técnicas, compreendam sua responsabilidade na construção de um futuro mais justo e solidário.

A Universidade Federal Rural do Semi-Árido (UFERSA), por meio da matriz curricular do curso de Bacharelado Multidisciplinar em Ciência e Tecnologia, reflete esse compromisso ao propor uma formação que ultrapassa o cálculo e o código. A ementa do curso enfatiza a necessidade de "compreender a ciência e a tecnologia como processo histórico, social, cultural e epistemológico", capacitando o egresso a "atuar de forma multidisciplinar, considerando os aspectos políticos, econômicos, sociais, ambientais e culturais, com visão ética e humanística" (UFERSA, 2025). Este Trabalho de Conclusão de Curso é, portanto, um fruto dessa filosofia que alicerça a aplicação do conhecimento técnico adquirido para solucionar um problema social concreto e local.

1.2 O problema de pesquisa

Uma pesquisa recente encomendada pela Mars Petcare revelou uma realidade alarmante no Brasil. Mais de 30 milhões de cães e gatos vivem nas ruas ou em abrigos, uma crise que impacta diretamente a saúde pública e expõe a urgência de ações mais eficazes (Valor Econômico, 2024). Na linha de frente deste desafio, encontram-se Organizações Não Governamentais (ONGs), protetores independentes e grupos de voluntários que, com recursos limitados, dedicam-se ao resgate, cuidado e busca por lares para esses animais.

Em Pau dos Ferros/RN, essa luta é personificada pelo grupo "Unidos pelas Patinhas", um grupo de voluntários que atua desde 2015 na proteção de animais em vulnerabilidade. A própria UFERSA serve como um ponto de apoio e inspiração para o grupo, abrigando ações e feiras de arrecadação de fundos que são vitais para a continuidade do trabalho, como ilustram as figuras a seguir:

Figura 1 - Produtos da feirinha da ONG para arrecadação de fundos

Fonte: Arquivo pessoal (2025)

Figura 2 - Material de divulgação do trabalho voluntário do grupo

Fonte: Arquivo pessoal (2025)

Figura 3 - Espaço na UFERSA em Pau dos Ferros cedido para ações do grupo

Fonte: Arquivo pessoal (2025)

A gestão de processos em ONGs e grupos como a Unidos pelas Patinhas frequentemente ocorre por meio de métodos manuais, como planilhas ou registros descentralizados. Essa abordagem pode levar a erros, como duplicação de informações, perda de dados e dificuldade na atualização e recuperação de registros. No contexto da adoção de animais, a ausência de um sistema *backend* estruturado compromete a organização dos cadastros, a rastreabilidade dos processos de adoção e a segurança das informações armazenadas.

Além disso, a falta de uma infraestrutura tecnológica eficiente impacta diretamente a comunicação entre o grupo e os adotantes, dificultando a divulgação dos animais disponíveis e limitando o alcance das campanhas de adoção. O Unidos pelas Patinhas enfrenta desafios significativos na organização de seus dados e no gerenciamento das adoções, o que reduz a eficiência do processo e mantém elevada a população de animais sem um lar.

Embora o desenvolvimento de um aplicativo *mobile* tenha sido proposto como solução para facilitar a interação com adotantes, a ausência de um *backend* robusto pode limitar seu funcionamento e escalabilidade. Sem um sistema de gerenciamento centralizado, o grupo continua vulnerável a inconsistências nos dados e dificuldades na administração de campanhas e adoções.

Diante desse cenário, surge a seguinte questão: Como o desenvolvimento de um *backend* estruturado pode otimizar a gestão de adoção de animais no Unidos pelas Patinhas, garantindo segurança, escalabilidade e eficiência na administração dos dados?

Essa problemática reforça a necessidade de uma solução tecnológica confiável, capaz de centralizar informações, melhorar a comunicação com os adotantes e oferecer suporte eficiente às operações da grupo. O desenvolvimento do *backend* para a aplicação do Unidos pelas Patinhas visa preencher essa lacuna, garantindo um sistema que facilite a organização dos processos internos e contribua para o aumento das taxas de adoções.

A solução não será apenas um conjunto de códigos, mas a materialização de uma formação acadêmica que entende a tecnologia como uma força a serviço da comunidade.

1.3 Justificativa

A realização deste projeto justifica-se a partir de uma sinergia de relevâncias que se alinham à visão humanista proposta. Em primeiro lugar, a relevância social, que é mais evidente e urgente. A implementação de um sistema de gestão automatizado para a "Unidos pelas Patinhas" representa uma intervenção direta em um problema comunitário. Ao otimizar os processos de cadastro, acompanhamento e adoção, a tecnologia libera os voluntários de tarefas operacionais repetitivas, permitindo que o tempo e energia sejam dedicados ao cuidado dos animais, captação de recursos e ações de conscientização. Uma plataforma mais eficiente pode significar um aumento no número de adoções bem-sucedidas e, consequentemente, ampliar o impacto na sociedade local.

Em segundo lugar, a relevância acadêmica com a materialização da filosofia educacional da UFERSA. Uma vez que este trabalho serve como estudo de caso prático de como a ciência e a tecnologia, quando guiadas por uma consciência ética e social, podem gerar soluções sociais de grande impacto e alcance. Ele demonstra que a formação de um profissional da área tecnológica não se resume ao domínio de ferramentas e conhecimentos técnicos, mas à capacidade de integrar os diferentes saberes de forma crítica e criativa para responder a demandas humanas. A documentação do processo, desde a concepção da arquitetura até a validação do sistema, contribui com um conhecimento que pode inspirar outros estudantes e projetos de extensão universitária.

Por fim, a relevância tecnológica e gerencial que se manifesta na modernização dos processos. A transição de um modelo de gestão manual para uma plataforma de dados centralizada e segura, aumenta a eficiência e também abre portas para uma gestão mais estratégica. A capacidade de coletar e analisar dados sobre adoções, perfis de animais e engajamento de tutores pode fornecer informações valiosas para futuras campanhas e tomadas de decisão, assegurando a sustentabilidade e o crescimento do trabalho do grupo a médio e longo prazo.

1.4 Objetivos

A fim de direcionar a elaboração deste projeto e assegurar a apresentação de uma solução de alta qualidade, foram estabelecidos os objetivos a seguir, que estão organizados a partir de objetivo central que se desdobra em etapas detalhadas para alcançá-lo.

1.4.1 Objetivo Geral

Desenvolver um *backend* para a aplicação de adoção de animais do "Unidos pelas Patinhas" para otimizar a gestão dos processos internos e potencializar o impacto social da organização.

1.4.2 Objetivos Específicos

- Levantar os requisitos funcionais e não funcionais do *backend*, garantindo a segurança, escalabilidade e eficiência no gerenciamento dos dados da adoção.

- Projetar a arquitetura do *backend*, utilizando boas práticas e padrões técnicos para facilitar a manutenção e evolução do sistema.
- Implementar APIs para o gerenciamento de cadastros de animais, adotantes e processos de adoção, para futura integração com o *frontend* existente.
- Desenvolver mecanismos de autenticação e controle de acesso para garantir a segurança das informações.
- Testar e validar a implementação do *backend*, assegurando o correto funcionamento das funcionalidades propostas.

1.5 Estrutura do trabalho

O presente trabalho encontra-se organizado em seis capítulos. O Capítulo 1, que aqui se encerra, apresentou a contextualização filosófica e social do problema, a justificativa e os objetivos da pesquisa. O Capítulo 2 aprofunda-se na revisão de literatura, estabelecendo o embasamento teórico. O Capítulo 3 apresenta a proposta de Santos *et al*. (2022) para um aplicativo de apoio à adoção para a ONG Unidos pelas Patinhas. O Capítulo 4 detalha a metodologia de pesquisa e o levantamento de requisitos. O Capítulo 5 apresenta a proposta de *backend* da plataforma. O Capítulo 6 apresenta as considerações finais, sintetizando os resultados e sugerindo trabalhos futuros com foco em multidisciplinaridade e acesso aberto.

2 REVISÃO DE LITERATURA

A tecnologia tem desempenhado um papel fundamental na modernização e otimização de processos em diversas áreas, incluindo as atividades realizadas por Organizações Não Governamentais (ONGs). O uso de sistemas informatizados possibilita maior organização, eficiência e transparência na gestão de informações, especialmente, em instituições que atuam no resgate e adoção de animais. Segundo Santos *et al*. (2022), ferramentas tecnológicas facilitam o gerenciamento das demandas diárias dessas ONGs, permitindo maior controle sobre cadastros, campanhas e interações com possíveis adotantes.

O abandono de animais é um problema recorrente no Brasil, impactando diretamente a saúde pública e o bem-estar dos próprios animais. Conforme destacado por Santos *et al*. (2022), a adoção é uma solução eficaz, mas ainda enfrenta desafios, como a falta de divulgação adequada e a ausência de sistemas organizados para facilitar o processo.

Partindo dessa contextualização, este capítulo estabelece a fundação teórica sobre a qual este trabalho é construído. Aprofunda-se nos conceitos técnicos e metodológicos da Engenharia de *Software* indispensáveis para a construção de uma solução digital que seja não apenas funcional, mas também robusta, sustentável e confiável. O objetivo é demonstrar que a responsabilidade social do tecnólogo se materializa através da aplicação rigorosa de boas práticas de engenharia.

2.1 Tecnologia, sociedade e a *práxis* da engenharia de *software*

A tecnologia tem desempenhado um papel fundamental na modernização e otimização de processos em diversas áreas, incluindo as atividades realizadas por Organizações Não Governamentais (ONGs). O uso de sistemas informatizados possibilita maior organização, eficiência e transparência na gestão de informações, especialmente, em instituições que atuam no resgate e adoção de animais. Segundo Santos *et al*. (2022), ferramentas tecnológicas facilitam o gerenciamento das demandas diárias dessas ONGs, permitindo maior controle sobre cadastros, campanhas e interações com possíveis adotantes.

A partir do núcleo do pensamento humanista atual, é compreendido que a tecnologia deve apoiar o progresso humano, ao invés de ser o inverso. O filósofo Coeckelbergh (2020) ressalta a relevância de uma perspectiva ética em relação à tecnologia. Assim, a verdadeira avaliação do

progresso tecnológico não se limita apenas à sua sofisticação ou eficiência, mas se encontra na habilidade de resolver questões humanas e promover o bem comum.

Entretanto, a simples vontade de desenvolver uma tecnologia que traga benefícios sociais não é suficiente sem as habilidades técnicas necessárias. Um aplicativo destinado a apoiar uma ONG, por exemplo, pode gerar mais insatisfação do que apoio se for instável, inseguro ou difícil de manter. É exatamente aqui que a Engenharia de *Software* se destaca como a área que transforma o objetivo em ação prática. De acordo com Sommerville (2019), a Engenharia de *Software* abrange todas as teorias, métodos e ferramentas associados ao desenvolvimento profissional de software. Através de seus princípios, assegura-se que um sistema seja criado de maneira organizada, previsível e com um padrão de qualidade que o torne confiável para os usuários.

Portanto, a revisão dos conceitos a seguir vai além de um simples formalismo acadêmico; trata-se da descrição da estrutura técnica necessária para desenvolver uma solução digital que respeite seu propósito social.

2.2 A base dos sistemas modernos na arquitetura de software

A arquitetura de *software* desempenha um papel essencial na criação de sistemas computacionais, estabelecendo sua organização, os componentes principais, suas interações e as limitações tecnológicas que influenciam seu *design*. De acordo com Bass, Clements e Kazman (2012), a arquitetura de um sistema consiste em um conjunto de estruturas necessárias para compreender o sistema, envolvendo elementos de *software*, as conexões entre eles e as características de ambos. Um projeto arquitetônico adequado é o primeiro passo para assegurar que o sistema não apenas cumpra suas exigências funcionais (o que o sistema realiza), mas também atenda a seus atributos de qualidade.

Esses atributos, também conhecidos como requisitos não funcionais, são cruciais para o sucesso de um *software*. Para o sistema do "Unidos pelas Patinhas", os seguintes atributos são prioritários:

- **Manutenibilidade:** A facilidade com que o sistema pode ser corrigido, adaptado ou expandido no futuro. Uma arquitetura bem definida, com componentes desacoplados, é

fundamental para que novas funcionalidades possam ser adicionadas sem a necessidade de reescrever grandes partes do sistema.

- **Escalabilidade:** A capacidade do sistema de lidar com um aumento de carga de trabalho. Por exemplo, durante uma grande campanha de adoção que viralize nas redes sociais, o sistema deve ser capaz de suportar um pico de acessos sem falhar ou ficar lento.
- **Segurança:** A proteção dos dados contra acesso e modificação não autorizados. Isso é vital em um sistema que armazena informações pessoais de adotantes.
- **Desempenho:** A velocidade e a responsividade da aplicação. Um sistema lento pode frustrar os usuários e diminuir as chances de uma adoção ser concluída.

2.3 Estilos de arquitetura

Os estilos de arquitetura representam abordagens ou modelos reutilizáveis que oferecem soluções para problemas recorrentes no *design* de *software*. A seleção de um estilo tem um impacto direto sobre os atributos de qualidade do sistema.

- **Arquitetura em Camadas**: É o estilo aplicado neste projeto. Ele organiza a aplicação em camadas horizontais, em que cada camada tem uma função específica e se comunica apenas com as camadas vizinhas. A estrutura clássica, conforme descrito por Sommerville (2019), segmenta o sistema em uma camada de apresentação (*front-end*), uma camada de lógica de negócios (*back-end*) e uma camada de dados (banco de dados). A principal vantagem dessa abordagem é a separação de interesses, o que favorece o baixo acoplamento entre os componentes e torna a manutenção e evolução do sistema mais simples.
- **Arquitetura de Microsserviços**: Em contraste, a arquitetura de microsserviços estrutura uma aplicação como um conjunto de pequenos serviços independentes. Embora poderosa para sistemas de grande escala, sua complexidade de implantação e gerenciamento a tornaria excessiva para o escopo e as necessidades atuais da ONG, justificando a escolha da arquitetura em camadas como uma solução mais pragmática e adequada.

2.4 Comunicação e integração de sistemas: apis e o padrão rest

A comunicação entre a camada de apresentação (*front-end*) e a de negócios (*back-end*) em sistemas contemporâneos geralmente acontece via uma API (Interface de Programação de Aplicações). No contexto da *web*, uma API específica os métodos e formatos de dados que as aplicações podem empregar para interagir de maneira padrão e segura (Fielding, 2000).

O REST (Transferência de Estado Representacional) é um estilo de arquitetura destinado ao desenvolvimento de aplicações em rede. Uma API que segue as diretrizes do REST é chamada de RESTful. As principais diretrizes incluem: arquitetura cliente-servidor, comunicação sem estado (*stateless*), cacheabilidade e uma interface padronizada (Fielding, 2000). Essa interface padronizada é fundamental e se baseia no uso dos verbos do protocolo HTTP para manipular recursos:

- **GET:** Para recuperar dados (ex: listar animais).
- **POST:** Para criar um novo recurso (ex: cadastrar um novo animal).
- **PUT/PATCH:** Para atualizar um recurso existente.
- **DELETE:** Para remover um recurso.

Para a troca de dados, o formato JSON (JavaScript Object Notation) tornou-se o padrão, devido à sua leveza e facilidade de interpretação.

2.5 Ecossistema tecnológico para o desenvolvimento *back-end*

A escolha do ecossistema tecnológico representa uma decisão arquitetônica fundamental. Neste projeto, a seleção foi realizada com base na modernidade, solidez e na força de suas respectivas comunidades de desenvolvedores.

- **Node.js e o Paradigma Orientado a Eventos**: *Node.js* é um ambiente de execução para *JavaScript* do lado do servidor que se destaca por seu modelo de Entrada/Saída não-bloqueante, sendo ideal para desenvolver APIs rápidas e escaláveis (Nodejs, 2025). Sua escolha se justifica pela alta performance em operações de I/O, característica de APIs que dependem de muitas consultas ao banco de dados.
- **Express.js - Framework para a Construção da API**: O Express.js é um *framework web* minimalista e flexível para Node.js, amplamente reconhecido como padrão na criação de

APIs. Sua abordagem não-opinativa permitiu construir a arquitetura em camadas de forma granular, em contraste com frameworks mais robustos como o Django (Python), que, embora excelente, traria uma complexidade maior do que a necessária para o escopo do projeto (Expressjs, 2025).

- **Bancos de Dados Relacionais e o MySQL**: Para assegurar a integridade dos dados, este projeto adota o MySQL, um Sistema Gerenciador de Banco de Dados relacional renomado por sua robustez e consistência nas transações (MySQL, 2025).
- **Prisma: O ORM de Próxima Geração:** O Prisma foi escolhido como a ferramenta para conectar a aplicação ao banco de dados. Diferente de ORMs tradicionais como o Sequelize, o Prisma oferece segurança de tipos (*type safety*) e um sistema de migrações mais robusto, o que aumenta a produtividade e a confiabilidade do código da camada de dados (PRISMA, 2025).

2.6 Ferramentas e tecnologias adotadas

A construção de um sistema de software robusto depende não apenas de uma arquitetura bem definida, mas também da seleção criteriosa de ferramentas que suportem e otimizem o ciclo de desenvolvimento. A falta de uma justificativa clara para as escolhas tecnológicas pode indicar uma abordagem superficial, onde as ferramentas são usadas sem um entendimento profundo de suas vantagens e desvantagens no contexto específico do projeto. Esta seção, portanto, visa formalizar e justificar a adoção das principais tecnologias que compõem o ecossistema deste projeto, conectando suas características aos requisitos de qualidade definidos.

2.6.1 JSON Web Token Para Autenticação

JSON Web Token(JWT) é um padrão aberto(RFC 7519) que define uma maneira compacta e autossuficiente para transmitir informações de forma segura entre duas partes como um objeto JSON. Um JWT é composto por três partes: um cabeçalho(*Header*), uma carga útil(*Payload*) e uma assinatura(*Signature*).A assinatura digital garante a autenticidade e a integridade do token, assegurando que as informações não foram adulteradas.

A escolha do JWT para este projeto foi motivada por sua adequação ao padrão de arquitetura *RESTful* e aos requisitos não funcionais. Sua natureza *stateless* significa que o servidor não precisa armazenar informações de sessão para cada usuário, delegando a

responsabilidade de manter o estado para o cliente. Isso simplifica a arquitetura do servidor e melhora a escalabilidade, pois qualquer instância do servidor pode validar o token sem depender de um armazenamento de sessão compartilhado. Além disso, a segurança é garantida pela assinatura digital que impede a manipulação dos dados de permissão do usuário(como o perfil de Administrador) por agentes mal-intencionados.

2.6.2 Framework de Testes

Para garantir a confiabilidade da API, a abordagem adotada contempla testes automatizados utilizando o *framework Jest*. A realização de testes automáticos é fundamental na engenharia de software contemporânea, pois assegura que as novas modificações no código não introduzam regressões (falhas em funcionalidades pré-existentes) e confirma que o sistema opera de acordo com as expectativas. A escolha do *Jest* se deu por sua excelente integração com o ecossistema *Node.js* e por ser uma solução "tudo em um", que já inclui executor de testes e biblioteca de asserções, facilitando a criação de testes de integração para os *endpoints* da API (JEST, 2025).

2.6.3 *Docker* e *Docker Compose* para Conteinerização

Docker é uma plataforma aberta que automatiza a implantação, o escalonamento e a gestão de aplicações dentro de ambientes isolados e leves chamados containers. Um container empacota o código da aplicação e todas as suas dependências(bibliotecas, ferramentas de sistemas, etc), garantindo que ela funcione de maneira consistente em qualquer estrutura que suporte o Docker.

Docker Compose é uma ferramenta que estende o *Docker* para definir e executar aplicações multi-container. Utilizando um arquivo de configuração no formato YAML(acrônimo para *YAML Ain't Markup Language*), é possível orquestrar múltiplos serviços(como a aplicação *Node.js* e o banco de dados MySQL), suas redes e volumes com um único comando.

A adoção do *Docker* e *Docker Compose* foi uma decisão estratégica para garantir a portabilidade e a consistência do ambiente de desenvolvimento, testes e produção. Essa abordagem elimina problemas comuns relacionados à configuração de dependências, versões de bibliotecas e compatibilidade entre sistemas operacionais, já que todo o ambiente é encapsulado em contêineres padronizados. Com o Docker, é possível criar imagens imutáveis

que garantem que a aplicação será executada da mesma forma em qualquer máquina ou servidor, reduzindo falhas decorrentes de diferenças no ambiente.

3 PROPOSTA DE AMBIENTE *MOBILE* PARA A ONG UNIDOS PELAS PATINHAS

Neste capítulo é apresentada a proposta de Santos *et al*. (2022), que descreve uma proposta de uma aplicação *mobile* para apoio nas atividades do Unidos pelas Patinhas. Na Figura 4 pode ser vista a página inicial do aplicativo, em que pode ser visto o botão “Começar”, para iniciar a interação com o sistema.

Figura 4 - Página inicial do aplicativo para a Unidos pelas Patinhas

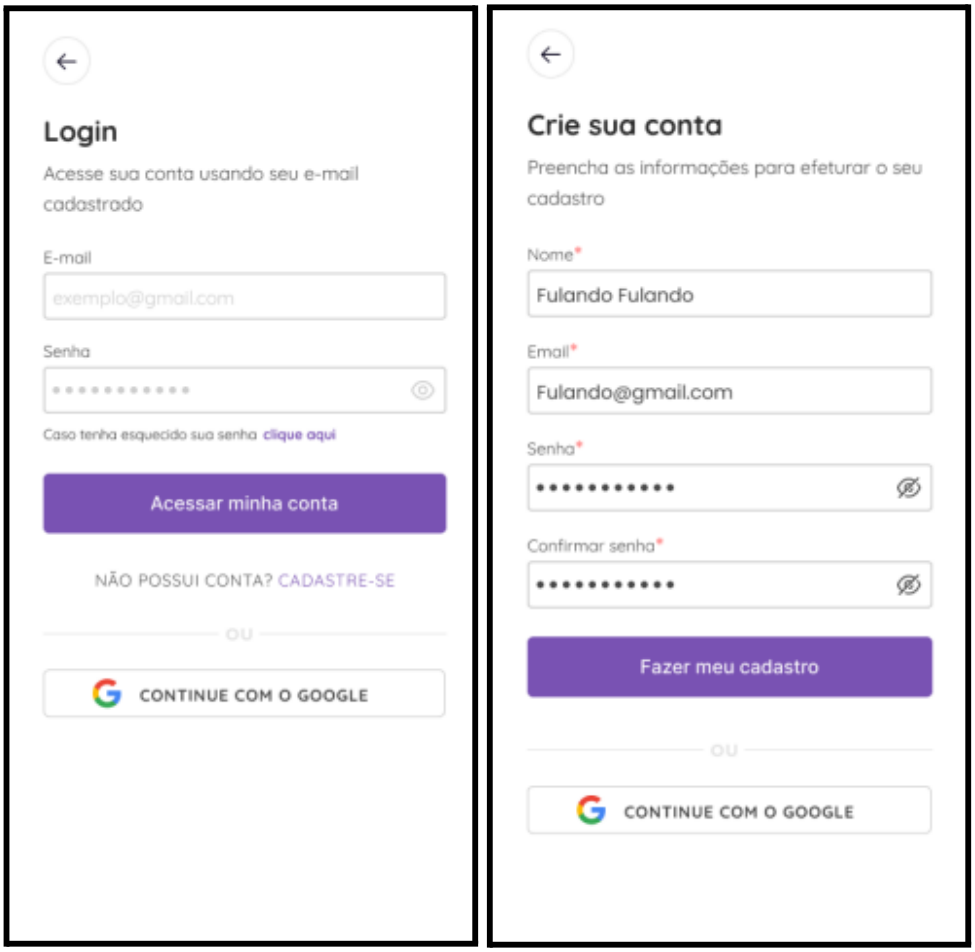
Fonte: Santos *et al*. (2022)

As telas da Figura 5 e Figura 6 permitem aos usuários realizarem *login* no sistema ou se cadastrar no sistema. Conforme pode ser visto na Figura 5a, para entrar no sistema o usuário precisa de suas informações de *login* (e-mail) e senha. Opcionalmente, o usuário pode utilizar a

sua conta cadastrada na plataforma da *Google* para realizar o cadastro. Caso o usuário não tenha uma conta, ele pode se cadastrar (Figura 5b), informando o seu nome, *e-mail* e senha (junto com a confirmação da senha). Após preencher os dados, o usuário clica no botão "Fazer meu cadastro". Alternativamente, o usuário pode realizar o cadastro utilizando a sua conta na plataforma da *Google*.

Figura 5 - Telas de *login* e de cadastro no aplicativo

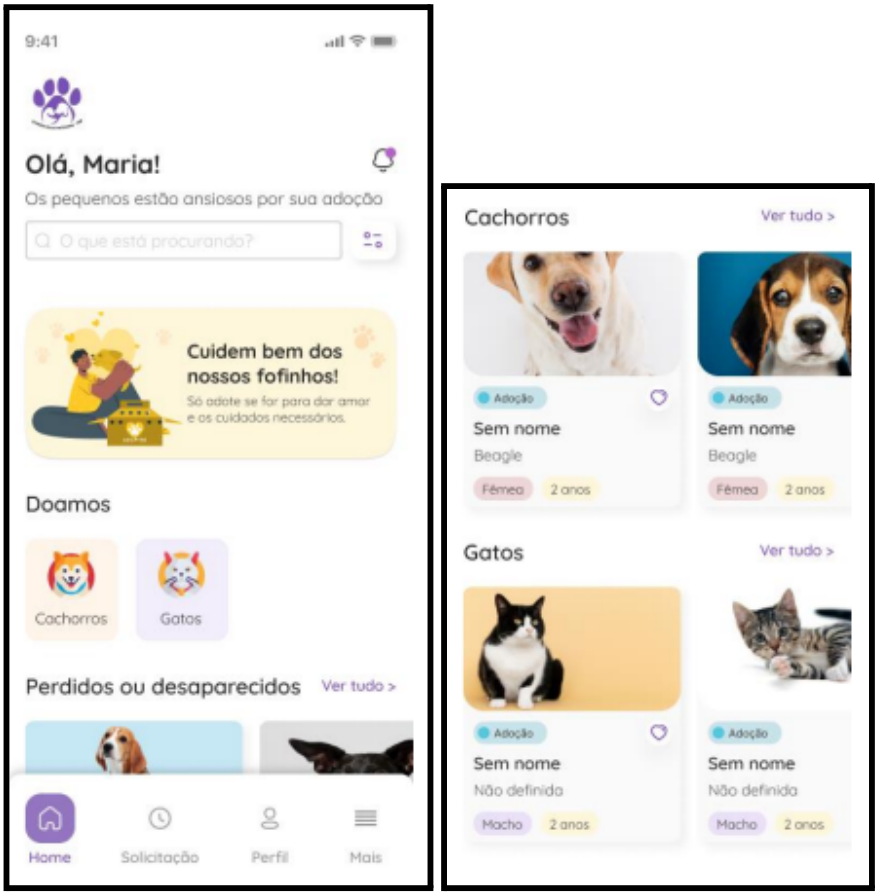
(a) tela de login (b) tela de cadastro



Fonte: Santos *et al*. (2022)

Na Figura 6 pode ser vista as diferentes partes da tela principal do aplicativo, apresentando informações sobre as doações, animais desaparecidos, cães e gatos e gatos disponíveis para a adoção.

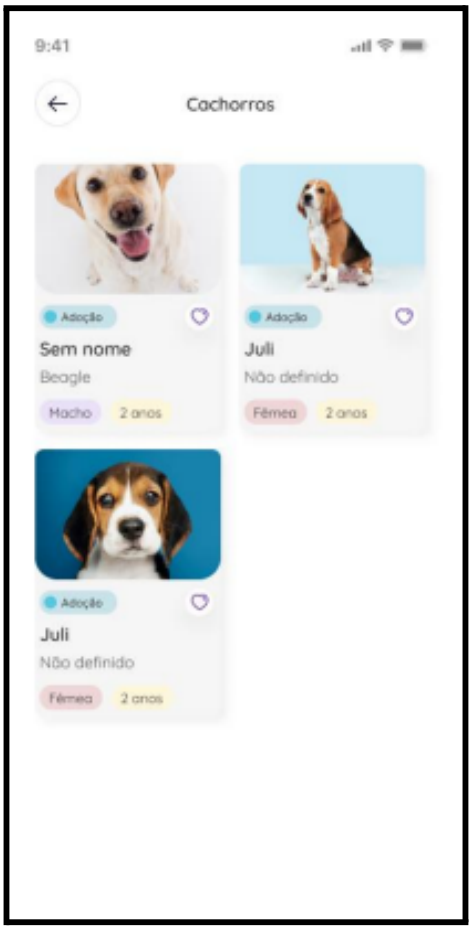
Figura 6 - *Design* de diferentes partes da tela principal do aplicativo



Fonte: Santos *et al*. (2022)

A tela da Figura 7 é uma continuação da página anterior, com informações de cachorros a serem doados.

Figura 7 - Continuação da tela principal do aplicativo



Fonte: Santos *et al*. (2022)

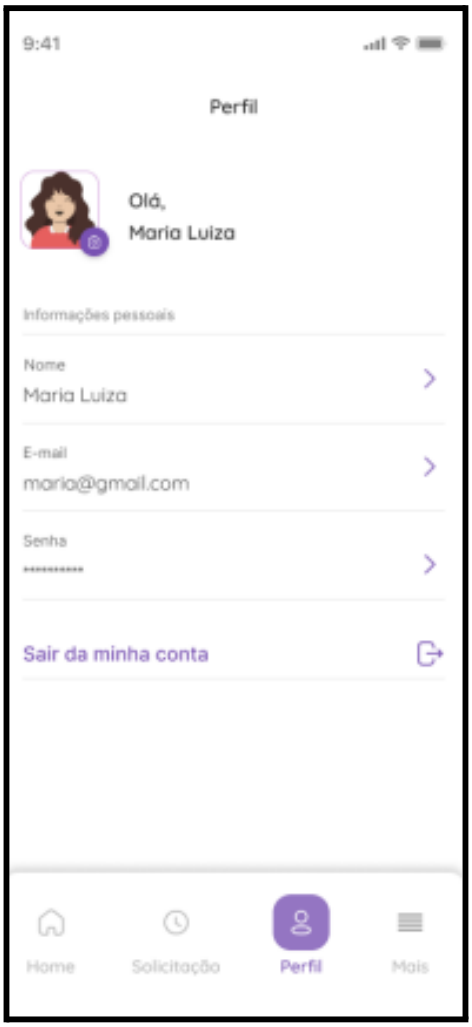
Por sua vez, a tela da Figura 8 é uma tela de perfil de um animal a ser doado. A tela apresenta uma foto de perfil do animal, imagens associadas ao animal, além de uma descrição e um botão com o número de contato do *WhatsApp*, para intermédio da doação.

Figura 8 - Perfil de um animal de estimação

Fonte: Santos *et al*. (2022)

De outro modo, a tela da Figura 9 é a tela de perfil do usuário. Ela apresenta uma foto do usuário, nome, *e-mail* e senha, possibilitando ao usuário editar essas informações.

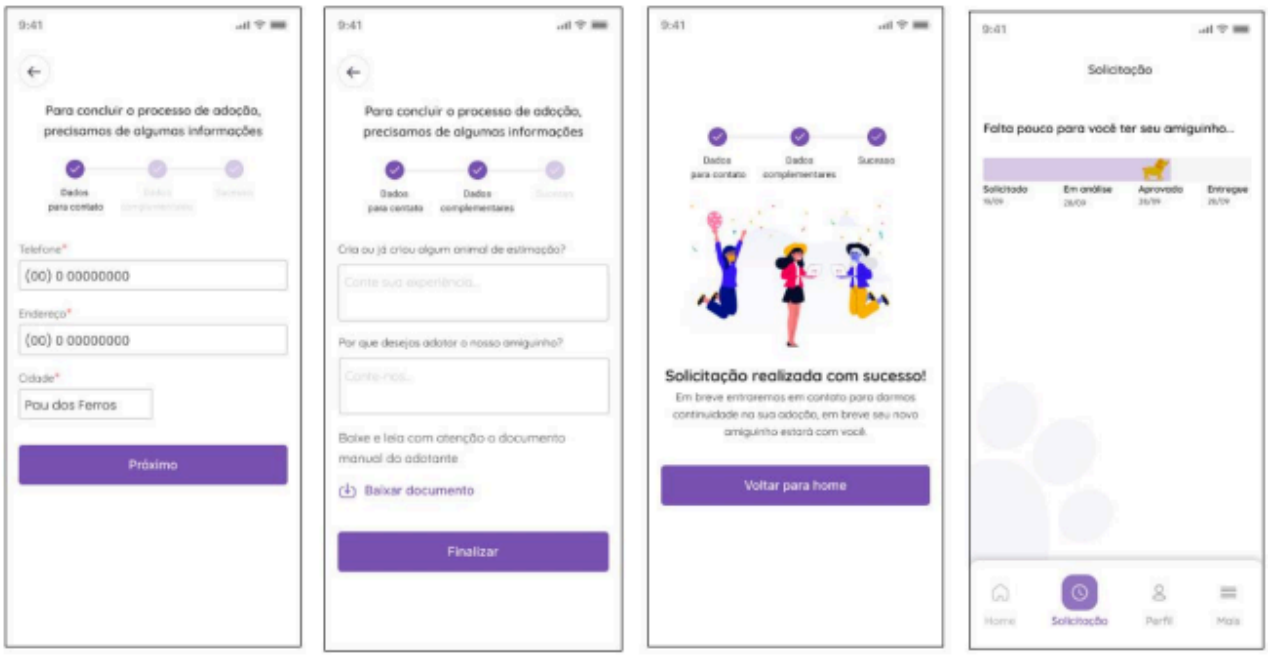
Figura 9 - Tela de perfil do usuário



Fonte: Santos *et al*. (2022)

Por fim, a tela da Figura 10 apresenta a sequência de passos que o usuário precisa realizar para realizar uma adoção. O interessado necessita informar suas informações de contato. Logo após, ele precisa informar se já criou algum animal antes, assim como os motivos para realizar a adoção. As próximas telas são referentes ao *status* do pedido (Solicitado, Em análise, Aprovado e Entregue).

Figura 10 - Sequência de etapas para se realizar uma adoção



Fonte: Santos *et al*. (2022)

O *backend* do ambiente proposto é voltado para apoio a adoção de animais. Uma vez que o código do *frontend* da aplicação para a Unidos pelas Patinhas não se encontrava disponível no momento da implementação do *backend*, a integração entre o *backend* desenvolvido e o *frontend* pode requerer etapas adicionais para a integração.

4 METODOLOGIA E PROJETO DA PESQUISA

Este capítulo apresenta a abordagem metodológica e as etapas da realização deste trabalho, da criação até a fase de avaliação. A metodologia escolhida foi crucial para assegurar que o desenvolvimento da aplicação ocorresse de maneira organizada, passível de replicação e conforme os objetivos da pesquisa, culminando em uma solução sólida e bem fundamentada.

4.1 Abordagem metodológica

Considerando-se a natureza do problema e os objetivos estabelecidos, este trabalho é classificado como uma pesquisa aplicada. Conforme Gil (2002, p 42.), a pesquisa aplicada "tem como meta gerar conhecimentos para aplicação prática, voltados à resolução de problemas específicos". Isso se aplica exatamente a este projeto, cuja finalidade é criar um artefato de *software* funcional para enfrentar os desafios operacionais específicos que "Unidos pelas Patinhas" encontra.

A abordagem de pesquisa que se ajusta de forma mais adequada a esse objetivo é a *Design Science Research* (DSR), ou Pesquisa em Ciência do *Design*. A DSR se destaca como uma metodologia relevante na Ciência da Computação e em Sistemas de Informação, concentrando-se na criação e avaliação de artefatos de Tecnologia da Informação (neste contexto, a arquitetura e a implementação do *backend*) com o intuito de solucionar problemas do mundo real, unindo rigor teórico com aplicabilidade prática.

4.2 Etapas do projeto

Para o desenvolvimento do *backend* da aplicação de adoção de animais da Unidos pelas Patinhas, foram realizadas as seguintes etapas:

1. **Revisão Bibliográfica:** Levantamento de referências sobre desenvolvimento de backends, boas práticas de arquitetura de software, segurança da informação e tecnologias adequadas para a implementação do sistema.
2. **Levantamento de Requisitos:** Identificação dos requisitos funcionais e não funcionais necessários para o *backend*, garantindo que o sistema atenda às necessidades da ONG e se integre corretamente ao *frontend* existente. Os resultados desta etapa são apresentados na seção 3.3.

3. **Desenvolvimento do *Backend*:** Implementação das APIs e funcionalidades do *backend* utilizando o ecossistema Node.js, garantindo escalabilidade, segurança e desempenho.
4. **Testes e Validação:** Realização de testes de integração para garantir o funcionamento correto do backend, cujos resultados são detalhados no Capítulo 6.

4.3 Levantamento de requisitos do sistema "Unidos Pelas Patinhas"

O levantamento de requisitos, resultado da primeira etapa da metodologia, é uma das fases mais críticas do processo de desenvolvimento (Sommerville, 2019). Com base na análise dos trabalhos de referência, os requisitos foram adaptados e detalhados para o escopo deste projeto de *back-end*.

4.3.1 Atores Do Sistema

Foram identificados dois atores principais que interagem com o sistema:

- **Administrador:** Representa os voluntários da ONG. Este ator possui permissões elevadas para gerenciar todo o ciclo de vida dos animais e das adoções.
- **Adotante:** Representa o usuário final que busca adotar um animal. Este ator pode visualizar os animais, buscar por características específicas e iniciar o processo de adoção.

4.3.2 Requisitos Funcionais

Os requisitos funcionais (RFs) definem os comportamentos específicos esperados do sistema. No Quadro 1, estão destacados os principais RFs que orientaram o desenvolvimento da API, conforme apresentado no Quadro 1.

Quadro 1 - Requisitos Funcionais

ID	Requisito	Descrição	Atores	Prioridade
RF001	Cadastrar Administrador	O sistema deve permitir o cadastro de usuários administradores, solicitando dados como nome completo, e-mail e senha.	Administrador	Essencial

RF002	Realizar *Login*	O sistema deve permitir que usuários (Administrador, Adotante) acessem a plataforma mediante e-mail e senha, retornando um token de autenticação.	Administrador, Adotante	Essencial
RF003	Cadastrar *Pets*	O sistema deve permitir que o Administrador cadastre novos animais para adoção, inserindo informações como nome, espécie, idade, sexo, porte, e fotos.	Administrador	Essencial
RF004	Atualizar *Pets*	O sistema deve permitir que o Administrador altere os dados de um *pet* já cadastrado.	Administrador	Importante
RF005	Listar *Pets*	O sistema deve permitir a exibição de uma lista com todos os *pets* disponíveis para adoção, exibindo seus principais dados.	Administrador, Adotante	Essencial
RF006	Excluir *Pets*	O sistema deve permitir que o Administrador remova o registro de um *pet* do sistema.	Administrador	Importante
RF007	Cadastrar Adotante	O sistema deve permitir o cadastro de usuários adotantes, solicitando dados como nome completo, e-mail, senha e telefone.	Adotante	Essencial
RF008	Buscar *Pets*	O sistema deve permitir a busca de *pets* por filtros como espécie, sexo, porte ou nome.	Adotante	Importante
RF009	Solicitar Adoção	O sistema deve permitir que um Adotante autenticado inicie uma solicitação de adoção para um *pet* disponível.	Adotante	Essencial
RF010	Listar Solicitações	O sistema deve permitir que o Administrador visualize uma lista de todas as solicitações de adoção recebidas.	Administrador	Essencial
RF011	Alterar *Status* da Solicitação	O sistema deve permitir que o Administrador altere o *status* de uma solicitação para "Pendente", "Aprovada" ou "Rejeitada".	Administrador	Essencial
RF012	Acompanhar Solicitação	O sistema deve permitir que o Adotante acompanhe o *status* de suas próprias solicitações de adoção.	Adotante	Importante

Fonte: Baseado em Santos *et al*. (2022)

4.3.3 Requisitos Não Funcionais

Os requisitos não funcionais (RNFs) definem os critérios de qualidade do sistema e são apresentados no Quadro 2.

Quadro 2 - Requisitos não funcionais (RNFs)

ID	**Requisito**	**Descrição**	**Categoria**
RNF001	Segurança	Todas as informações sensíveis, como senhas, devem ser criptografadas no banco de dados. A comunicação com a API deve ocorrer via HTTPS e o acesso a rotas protegidas deve ser controlado por tokens de autenticação (JWT) .	Segurança
RNF002	Desempenho	O tempo de resposta da API para requisições comuns (listagem, busca) deve ser, em média, inferior a 2 segundos em condições normais de uso	Desempenho
RNF003	Disponibilidade	O sistema deve operar de forma online, exigindo conexão com a internet para todas as funcionalidades. A infraestrutura de implantação deve visar alta disponibilidade .	Disponibilidade
RNF004	Escalabilidade	A arquitetura deve ser projetada para suportar um crescimento gradual no número de usuários, animais e solicitações sem degradação significativa do desempenho .	Escalabilidade
RNF005	Usabilidade (API)	A API deve ser bem documentada (via Swagger/OpenAPI), com endpoints claros e intuitivos, facilitando a integração com o aplicativo front-end e futuras manutenções.	Usabilidade

Fonte: Baseado em Santos *et al*. (2022)

5 PROPOSTA DO *BACKEND* DO SISTEMA

Neste capítulo, apresenta-se o *backend* do sistema para o "Unidos pelas Patinhas". O projeto descrito resulta diretamente da aplicação dos conceitos extraídos da revisão de literatura e da metodologia de pesquisa, visando atender aos requisitos identificados e assegurar os atributos de qualidade essenciais à aplicação, como manutenção, escalabilidade e segurança. A arquitetura será elucidada por meio do modelo de múltiplas visões sugerido por Bass, Clements e Kazman (2012), abrangendo as visões de Módulo, Componente & Conector e Alocação.

5.1 Visão geral da arquitetura em camadas

A estrutura do *backend* foi desenvolvida com a abordagem de arquitetura em camadas, que favorece a distinção de responsabilidades e o baixo acoplamento entre diversas partes do sistema. Dentro do cenário de uma API RESTful que utiliza Node.js e Express.js, essa separação é organizada em quatro camadas lógicas principais:

Camada de Rotas (*Routes*): Este componente atua como a interface da API. Sua função é estabelecer os *endpoints* (as URLs disponíveis), associar cada endpoint aos verbos HTTP (GET, POST, PUT, DELETE) e encaminhar as solicitações recebidas ao controlador adequado.
Camada de Controladores (*Controllers*): Serve como o elo entre as requisições recebidas e a lógica empresarial. É responsável por extrair as informações da requisição (corpo, parâmetros, cabeçalhos), acionar os serviços relacionados para realizar a ação requisitada, e finalmente, preparar e enviar a resposta HTTP de volta ao cliente.
Camada de Serviços (*Services*): Representa o cerne da aplicação, onde a lógica empresarial é implementada. Nesta camada, os serviços gerenciam as operações, aplicam as regras de negócio (como validações e cálculos) e coordenam a comunicação com a camada responsável pelo acesso a dados, a fim de manipular as informações corretamente.
Camada de Acesso a Dados (*Repositories/Models*): Esta é a camada mais interna, encarregada da interação com o banco de dados. Utilizando o ORM Prisma, essa camada facilita a complexidade das consultas SQL. Ela estabelece os modelos de dados (Models) e realiza as operações de Criação, Leitura, Atualização e Exclusão (CRUD) no banco de dados MySQL.

Essa arquitetura assegura que cada seção do sistema possua uma função única e claramente definida, promovendo alta coesão e baixo acoplamento, o que torna a testagem e manutenção do código mais simples

5.2 Perspectiva do módulo

A perspectiva do módulo ilustra a composição estática do código-fonte, demonstrando a maneira como o sistema se divide em unidades de implementação, como classes e módulos, além das interações entre essas partes.

As principais entidades (classes de modelo) que organizam o domínio da aplicação são:

Usuario: Engloba tanto os Administradores quanto os Adotantes. Inclui atributos fundamentais como id, nome, *e-mail*, senha (que será armazenada de forma criptografada) e perfil (um enumerador que define o tipo de usuário).

Animal: Representa os dados dos animais que estão disponíveis para adoção. Possui atributos como id, nome, espécie, idade, sexo, porte, descrição, *status* (disponível, adotado) e fotos. Estabelece uma relação de 1-para-N com a entidade Usuario, mostrando qual administrador fez o cadastro.

SolicitacaoAdocao: Refere-se ao processo de adoção. Contém atributos como *id*, *status* (pendente, aprovada, rejeitada, entregue) e data_solicitacao. Esta entidade cria uma conexão essencial entre Usuário (o adotante) e Animal, indicando qual usuário fez a solicitação de adoção de qual animal.

5.3 Perspectiva de componente e conector

A perspectiva de componente e conector representa o sistema durante sua operação, centrando-se nos componentes computacionais essenciais e nas suas interações.

Os principais elementos do *backend* incluem:

API Gateway (Rotas): Esse componente revela a interface pública da API. Ele recebe todas as requisições HTTP provenientes do cliente.

Componente de Autenticação: Tem a tarefa de gerenciar o registro e *logins* dos usuários, utilizando JWT para garantir a segurança dos *endpoints* que necessitam de autenticação.

Componente de Gestão de Animais: Envolve toda a lógica de negócio referente ao CRUD de animais.

Componente de Gestão de Adoções: Coordena o processo de criação e atualização das solicitações de adoção.

Prisma Client (Acesso a Dados): Este componente é responsável pela comunicação com o banco de dados, funcionando como a conexão entre os serviços de negócio e o SGBD MySQL.

Os conectores que possibilitam a interação entre esses componentes são principalmente chamadas de função/método no aplicativo Node.js e as requisições HTTP que ligam o cliente (aplicativo móvel) ao API *Gateway*.

5.4 Perspectiva de alocação

A perspectiva de alocação ilustra como os elementos de *software* são destinados à infraestrutura de *hardware* na qual o sistema operará, refletindo a configuração física da implementação. A arquitetura foi elaborada para suportar uma implementação em nuvem, assegurando escalabilidade e disponibilidade.

Os nós de *hardware* (sejam físicos ou virtuais) incluem:

Dispositivo do Cliente: Este representa os dispositivos móveis (*iOS/Android*) que terá a aplicação *frontend* instalada. Este nó estabelece comunicação com o servidor de aplicação através da rede (*internet*), utilizando o protocolo HTTPS para garantir segurança.

Servidor de Aplicação: Um servidor em um provedor de nuvem (como AWS, Google Cloud, etc.) onde a aplicação Node.js e o servidor web (por exemplo, Nginx) serão executados. Este nó é responsável por processar toda a lógica da API.

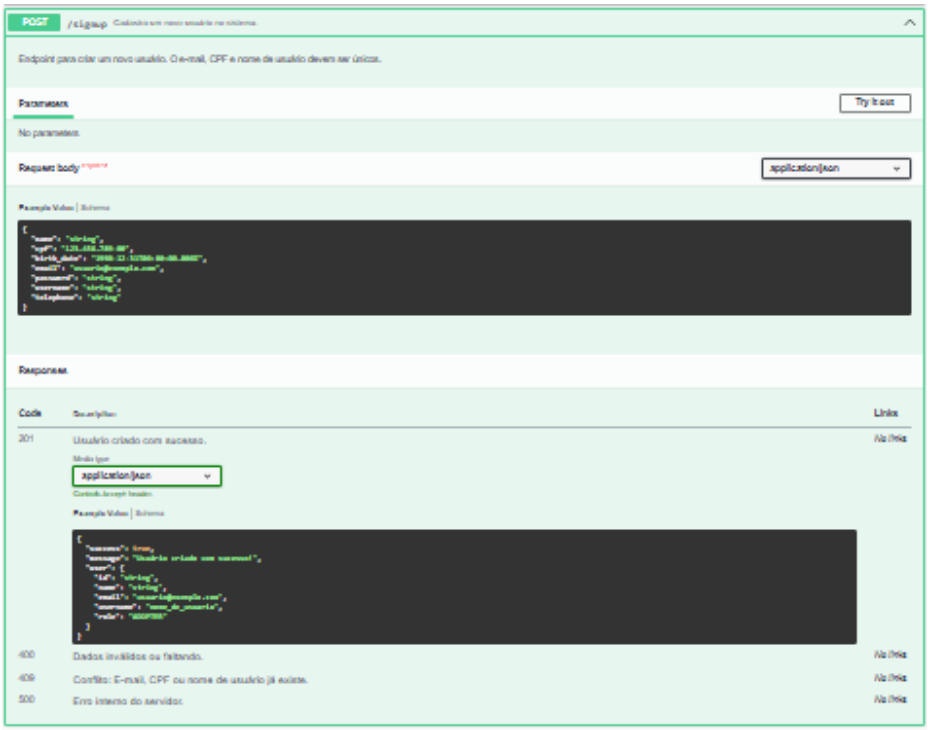
Servidor de Banco de Dados: Um serviço gerido de banco de dados (como AWS RDS ou um servidor que tenha o MySQL instalado) que mantém todos os dados da aplicação de maneira persistente. Ele se conecta ao Servidor de Aplicação, normalmente em uma rede privada, para aumentar a segurança.

5.5 Demonstração de *endpoints* do *backend*

Nesta seção são apresentados alguns *endpoints* do *backend* desenvolvido. Na Figura 11 é apresentado um *endpoint* para a realização de uma requisição POST de cadastro no *backend*. Na

parte “*Request body*”, é apresentado o padrão das informações a serem enviadas para o *backend*. Na parte inferior, pode-se ver que uma das possíveis respostas dados pelo *backend*, é a de código 201, que determina que o usuário foi criado com sucesso.

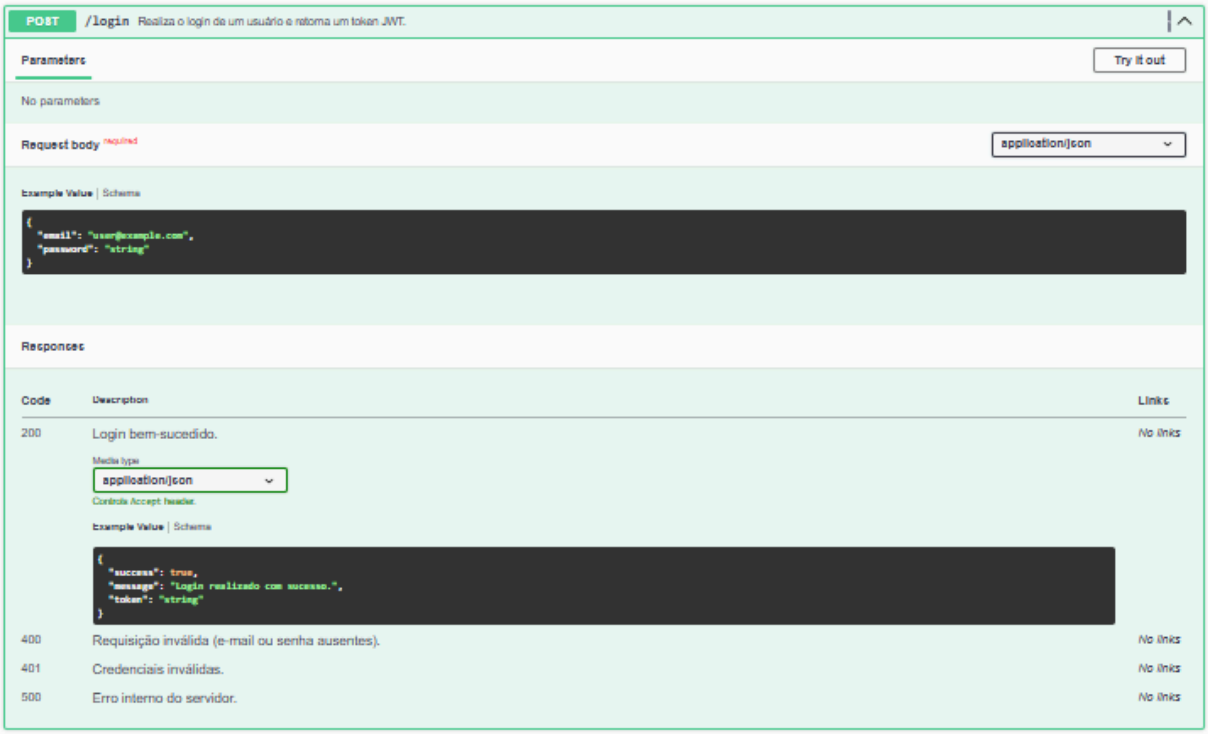
Figura 11 - *Endpoint* de uma solicitação *signup* de um usuário.



Fonte: Autoria própria (2025).

Por sua vez, na Figura 12, é apresentado um *endpoint* de uma requisição POST para se realizar um *login* na plataforma. Para isso, o padrão de informações enviados é formado pelo *email* do usuário e a senha na opção “Request body”. No exemplo, são mostradas as diferentes possibilidades de respostas. Por exemplo, a resposta de *login* bem sucedido é a de código 200.

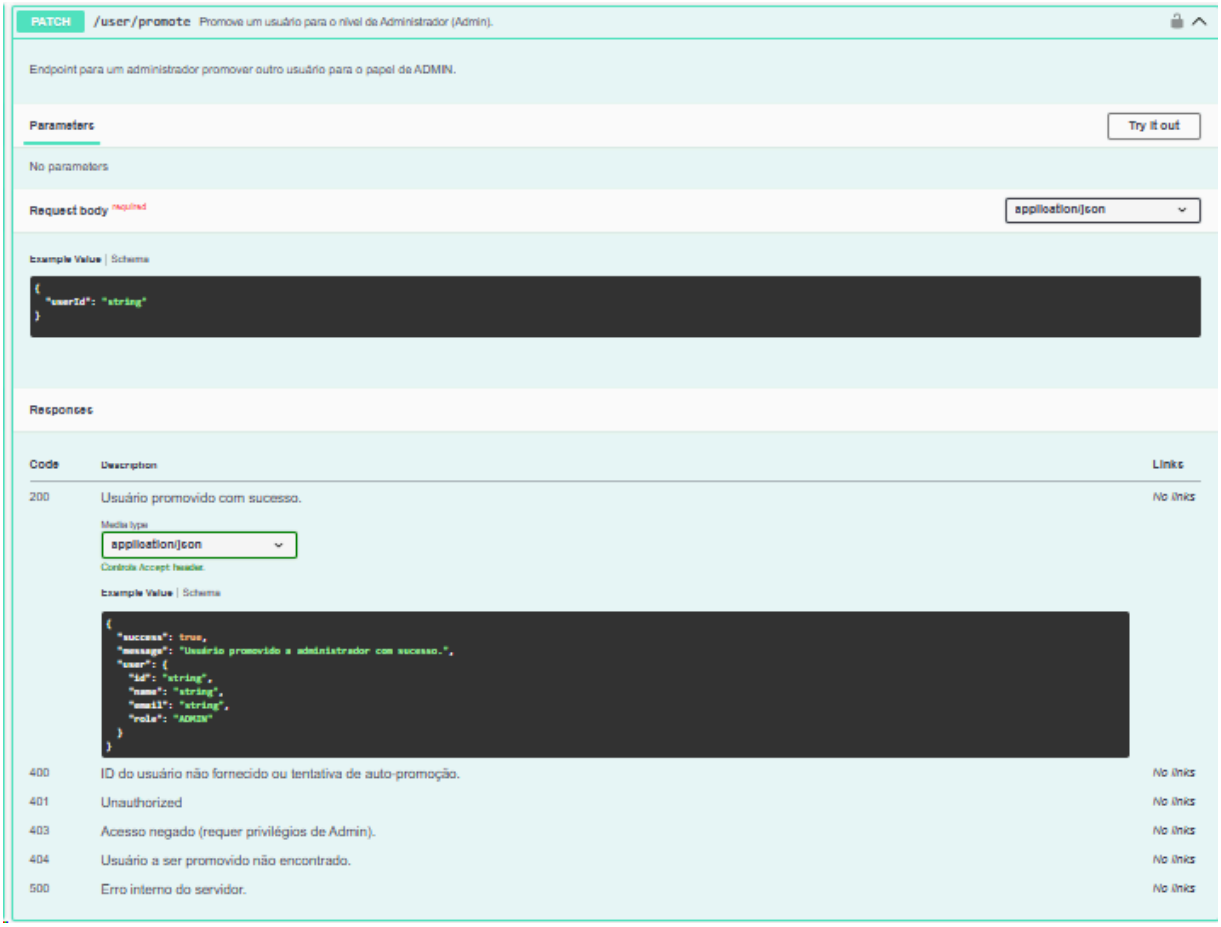
Figura 12 - *Endpoint* de uma solicitação de login



Fonte: Autoria própria (2025).

A Figura 13 apresenta um *endpoint* de acesso restrito a administradores, do tipo PATCH, utilizado para promover um usuário comum à função de administrador. Para a execução desta ação, o corpo da requisição deve conter o identificador(userId) do usuário a ser promovido. A confirmação da mudança de papel é dada pela resposta de *status* 200(OK).

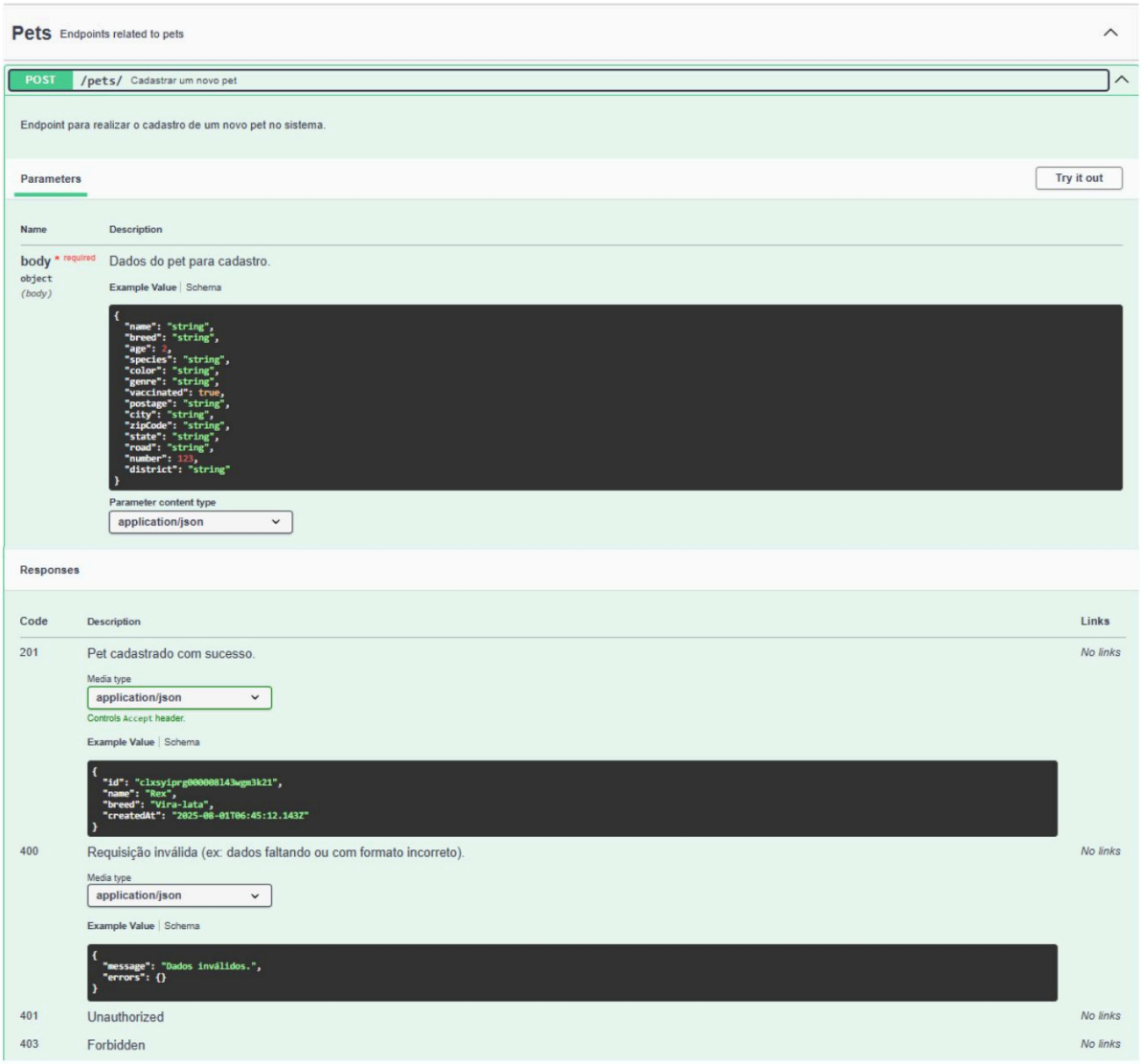
Figura 13 - *Endpoint* para promover um usuário a administrador



Fonte: Autoria própria (2025).

Para o cadastro de um novo *pet* na plataforma, utiliza-se o *endpoint* do tipo POST ilustrado na Figura 14, cujo acesso também é restrito a administradores. O corpo da requisição(Request Body) deve conter todos os dados pertinentes ao animal, como nome, espécie, idade e localização. O sucesso da operação é sinalizado pelo código de resposta 201(Created), indicando que o *pet* foi cadastrado no sistema.

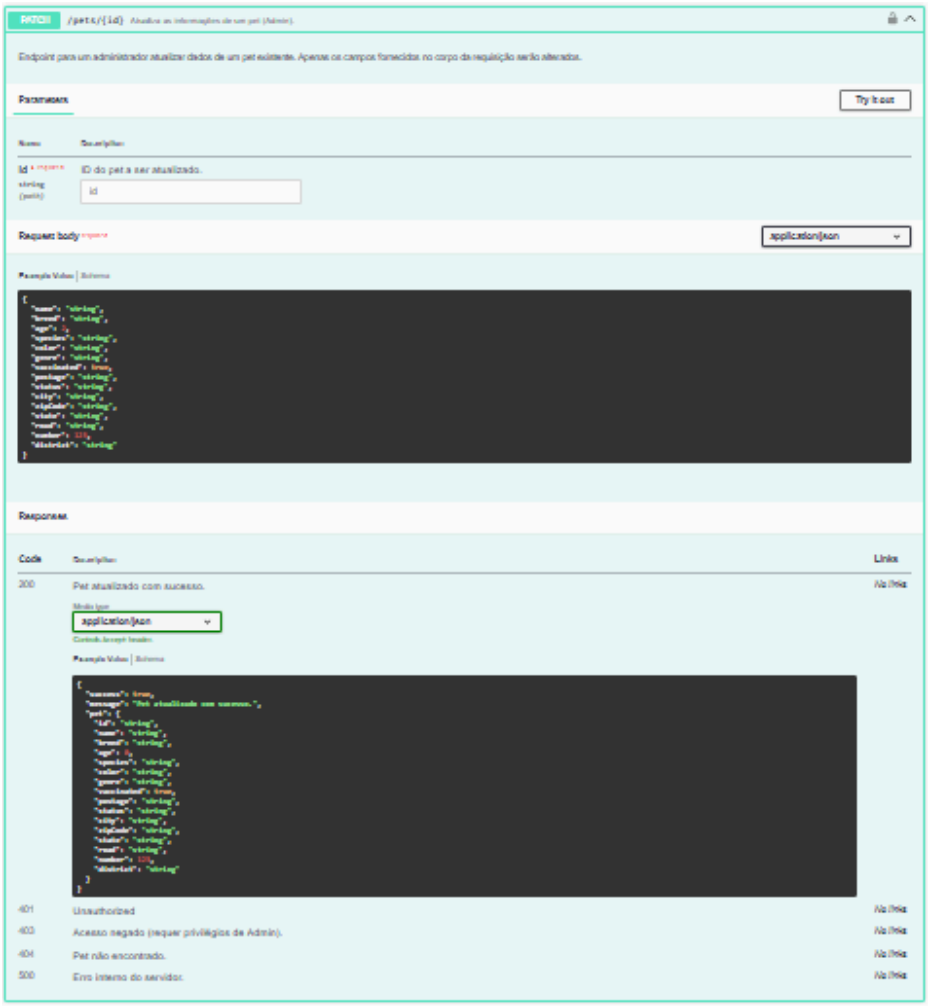
Figura 14 - *Endpoint* para cadastrar um novo pet



Fonte: Autoria própria (2025).

A Atualização dos dados de um *pet* já cadastrado é realizada através do *endpoint* do tipo PATCH, demonstrado na Figura 15. A requisição exige o identificador (petId) do pet na URL e envia no corpo apenas os campos que serão modificados, não sendo necessário reenviar todas as informações. A conclusão bem sucedida da atualização é indicada pelo *status* 200(OK).

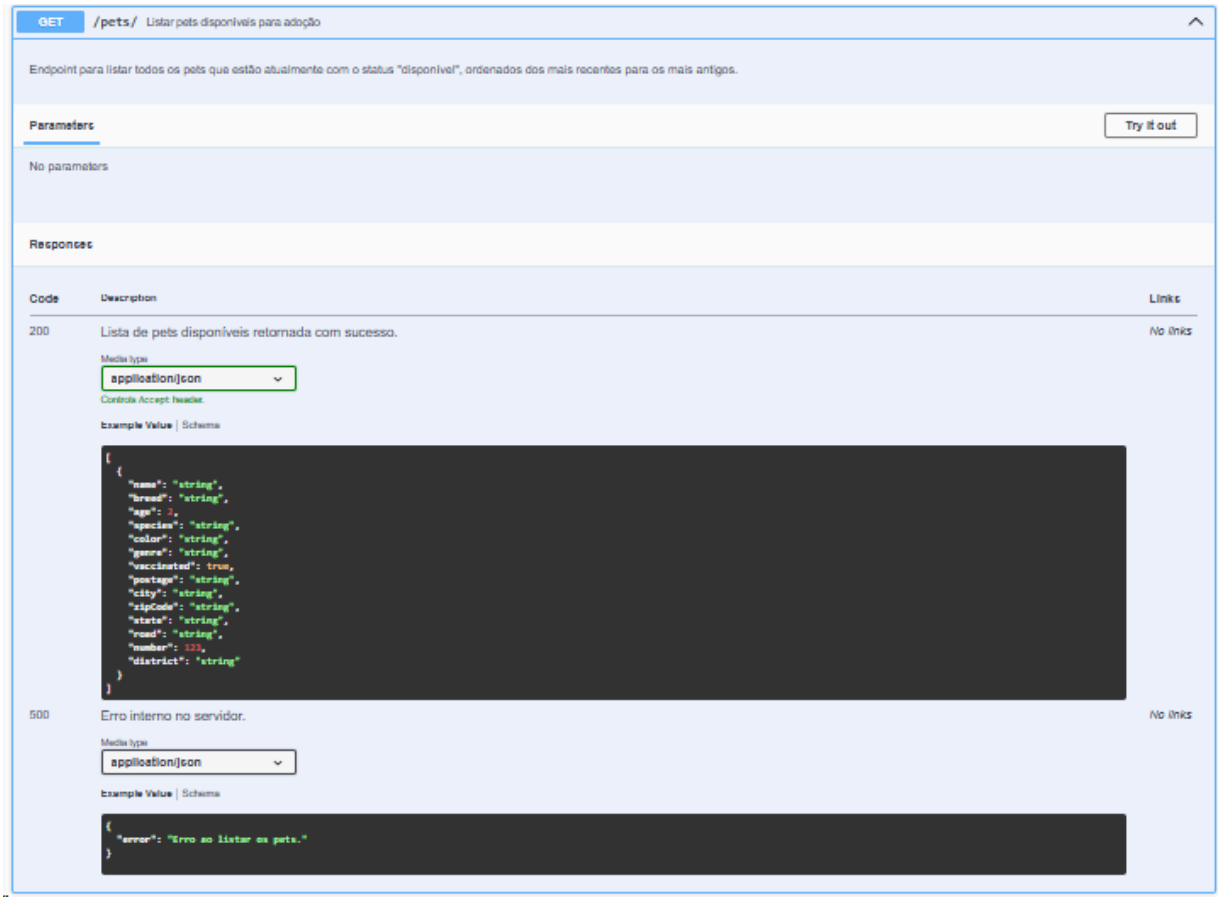
Figura 15 - *Endpoint* para um administrador atualizar os dados de um pet



Fonte: Autoria própria (2025).

Já na Figura 16 é apresentando um *endpoint* de uma requisição GET, para busca de informações no *backend*. Nesta requisição não há padrão de dados a serem enviados, uma vez que se trata de uma requisição de busca simples por informações. Na parte das respostas são apresentadas as diferentes possibilidades de resposta do servidor, a exemplo da resposta de código 200, que é de retorno bem sucedido do *backend* da listagem de *pets*.

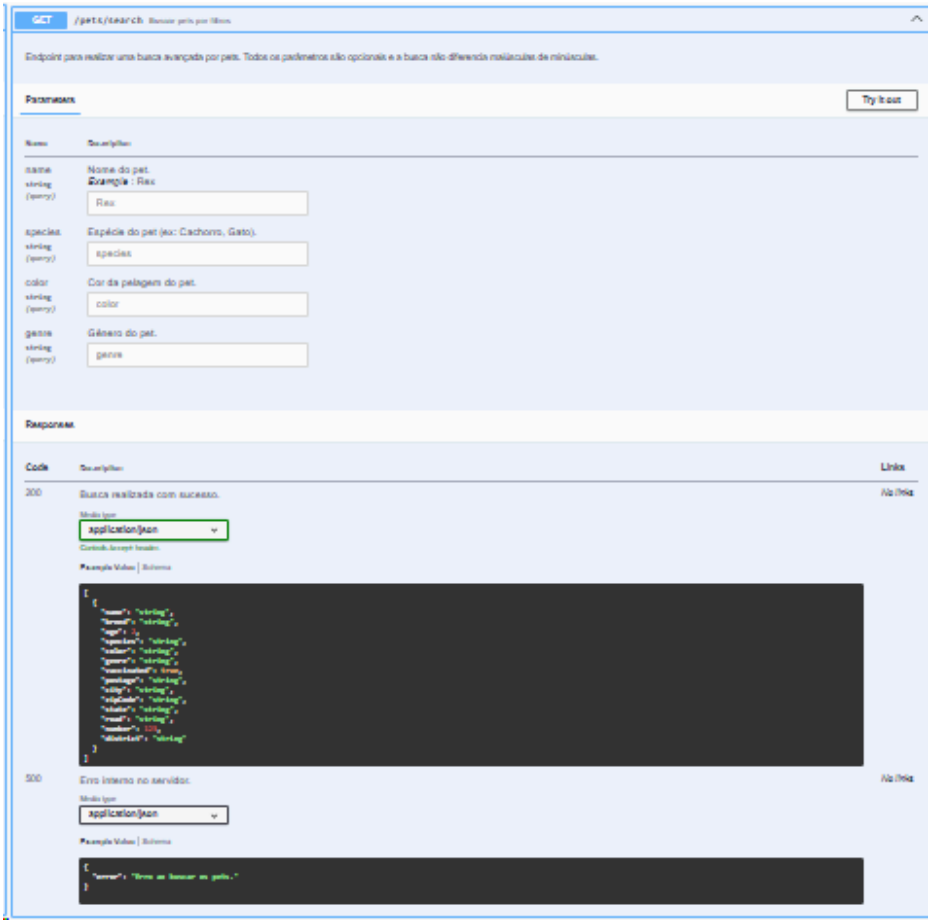
Figura 16 - *Endpoint* de uma solicitação para listagem dos pets disponíveis



Fonte: Autoria própria (2025)

No *endpoint* da Figura 17, é mostrada uma requisição GET para uma busca avançada de *pets*. A requisição avançada possibilita realizar uma busca pelo nome do *pet*, espécie, cor e gênero. Na parte inferior, são informadas as diferentes respostas possíveis, a exemplo da resposta de código 200 de busca bem sucedida.

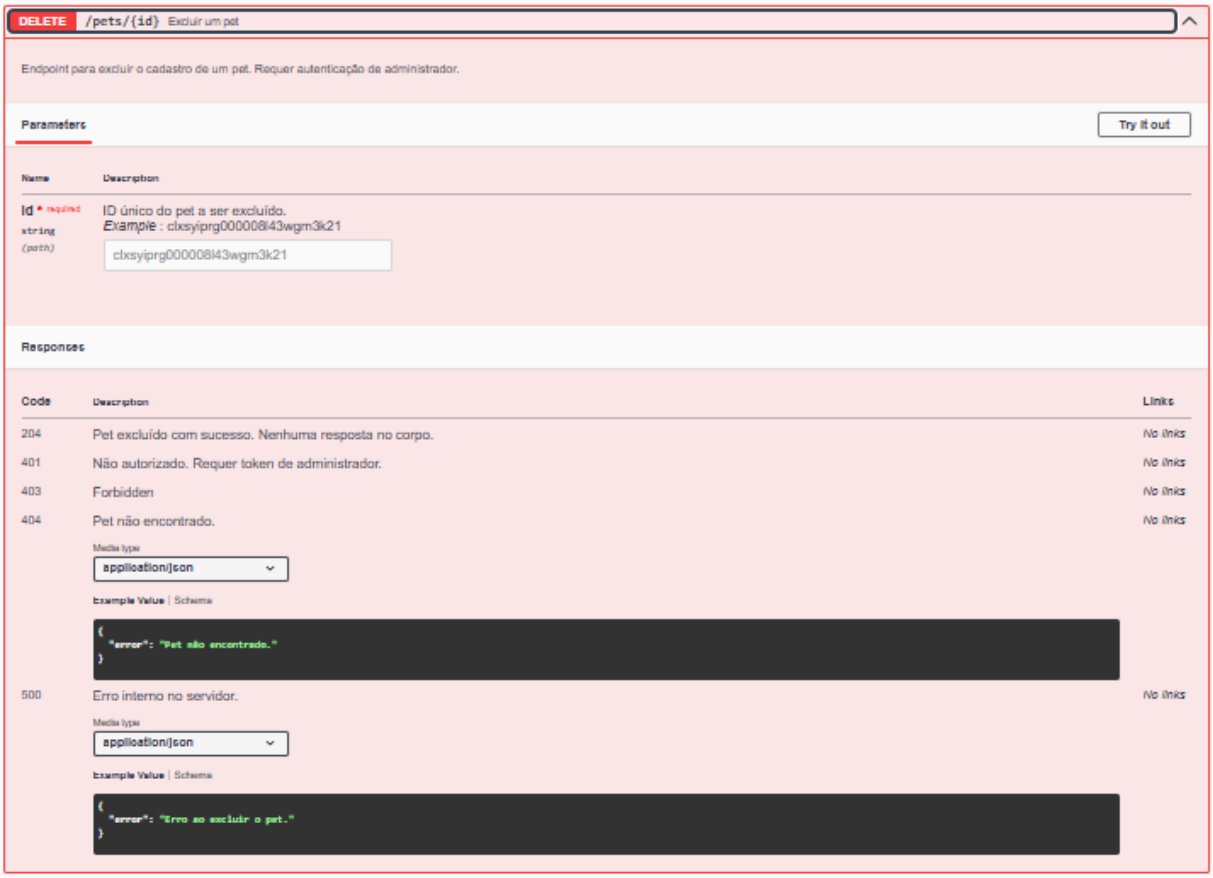
Figura 17 - *Endpoint* de uma busca avançada de pets.



Fonte: Autoria própria (2025).

Adicionalmente, na Figura 18, é exemplificado uma requisição do tipo *DELETE* para exclusão de um *pet*. O parâmetro informado para remoção é o identificador (id) do *pet*. Na resposta, diferentes respostas são possíveis, por exemplo, o retorno de código 204(No Content), para informar de uma remoção realizada com sucesso.

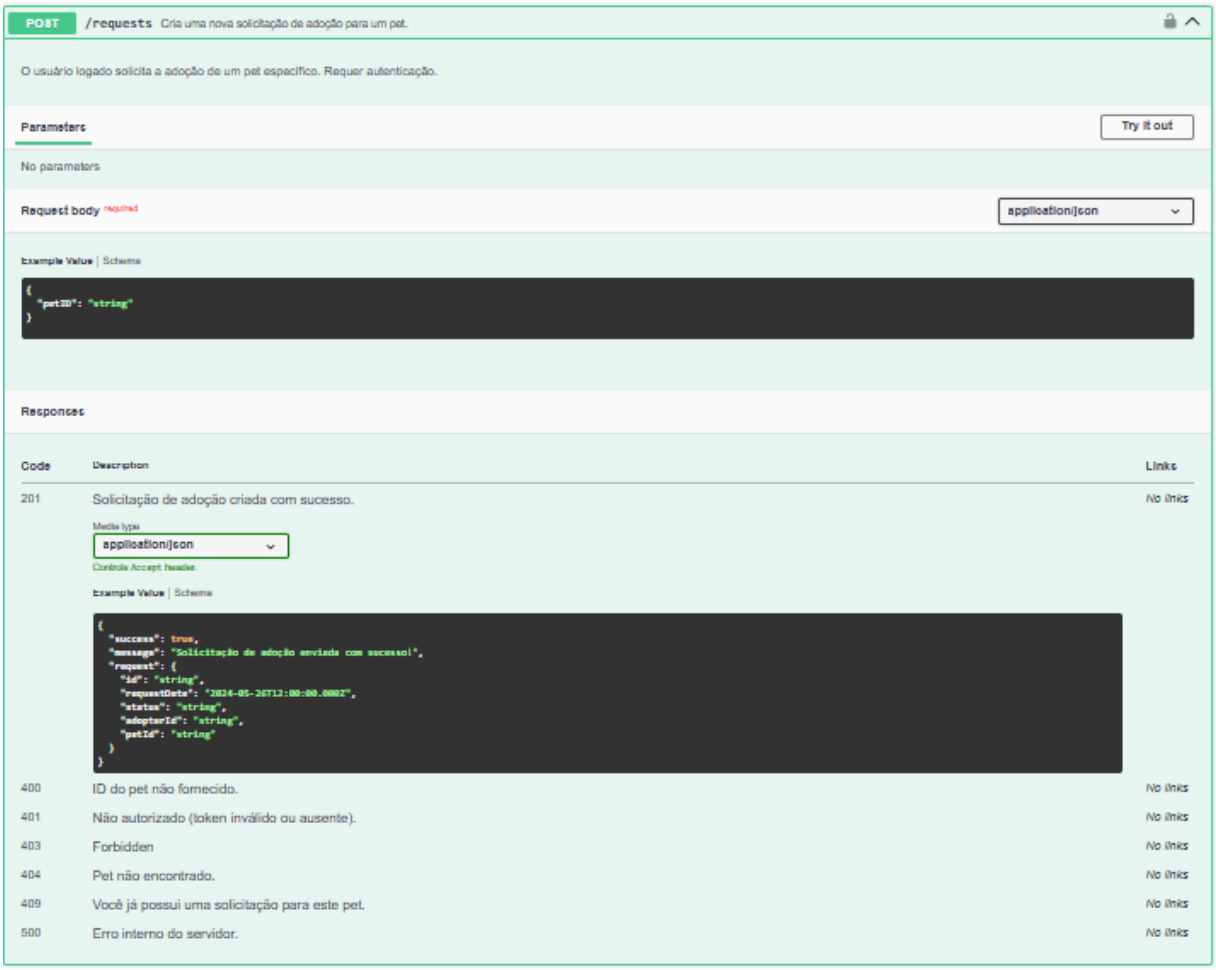
Figura 18 - *Endpoint* de uma remoção de um *pet*.



Fonte: Autoria própria (2025).

A interação central do usuário adotante com a plataforma é a manifestação de interesse em um animal, cujo POST está detalhado na Figura 19. Após a devida autenticação, o usuário formaliza seu pedido informando o *petId* do animal desejado. A plataforma, por sua vez, registra a solicitação e assegura ao usuário seu recebimento através do código de resposta 201.

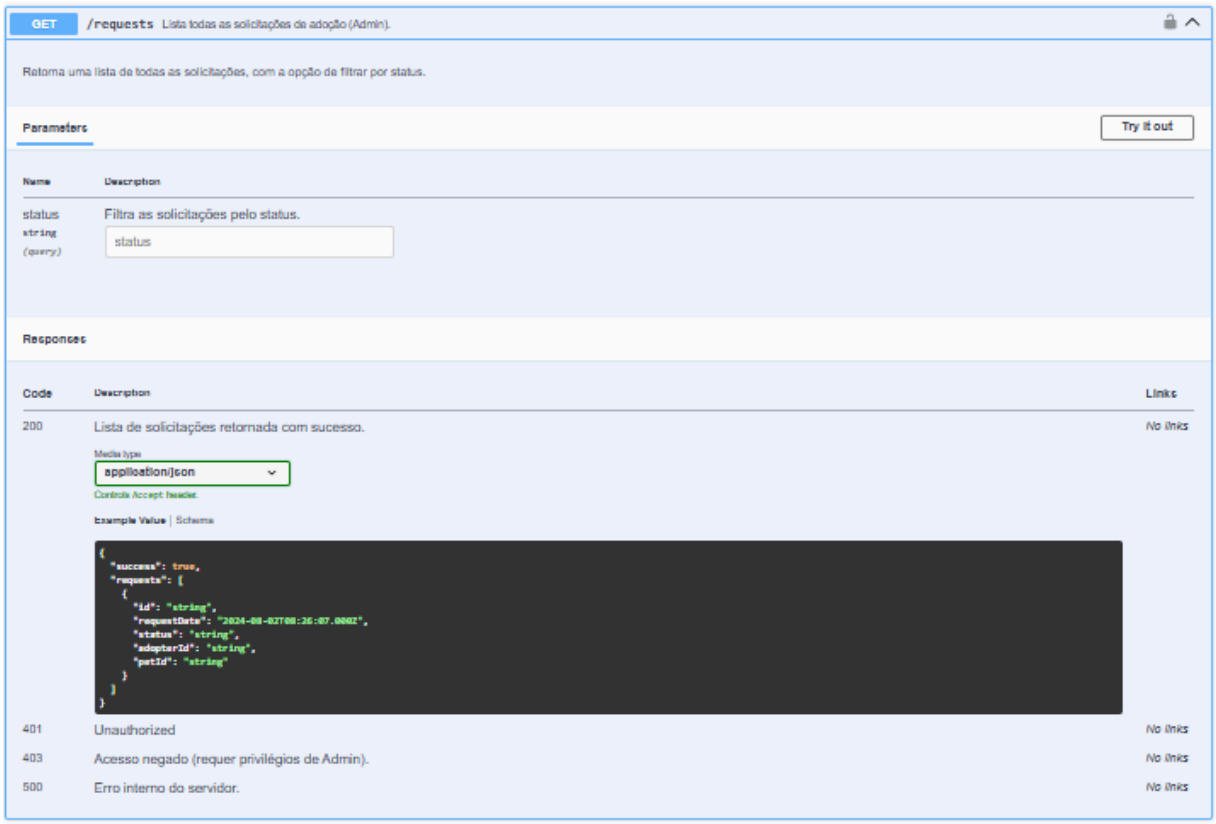
Figura 19 - *Endpoint* para criar uma solicitação de adoção



Fonte: Autoria própria (2025).

Para o efetivo gerenciamento e acompanhamento dos pedidos, os administradores dispõem do *endpoint* de listagem apresentado na Figura 20. Através de uma requisição GET, é possível obter o conjunto completo de solicitações, com a possibilidade de redefinir a busca por meio de query *parameters*, como o *status* do pedido. O retorno bem-sucedido entrega a lista de solicitações, fornecendo uma visão geral para a tomada de decisão.

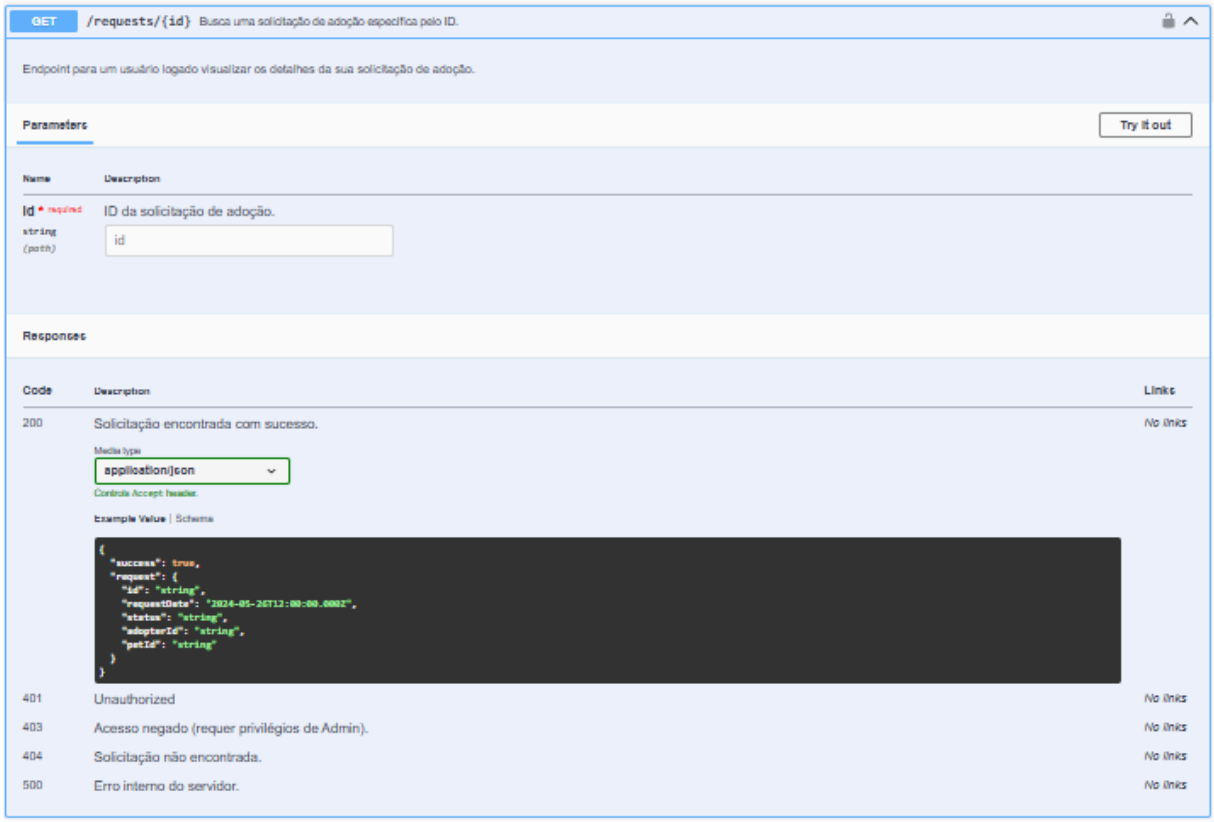
Figura 20 - *Endpoint* para um administrar visualizar todas as solicitações de adoção



Fonte: Autoria própria (2025).

A Figura 21, detalha o *endpoint* de consulta individual para o adotante acompanhar sua solicitação. Por meio de uma requisição GET, e informando o id da solicitação como parâmetro de rota na URL, o usuário solicitante obtém o status atual da solicitação no corpo do código de resposta 200.

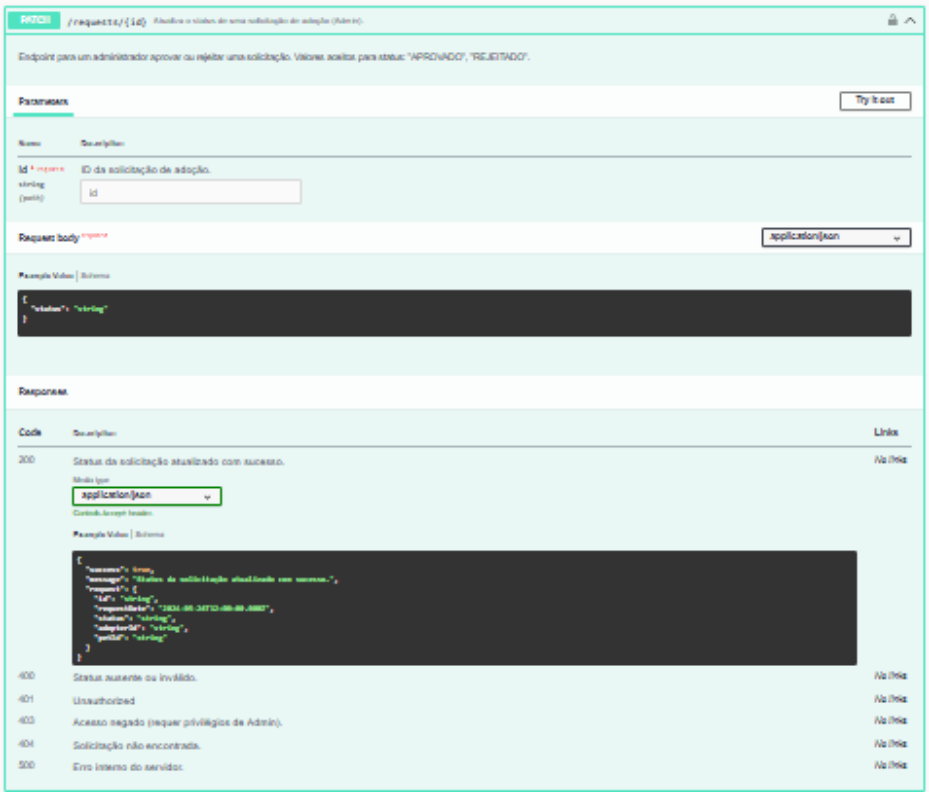
Figura 21 - *Endpoint* para visualizar uma adoção específica



Fonte: Autoria própria (2025).

Na figura 22, pode-se ver a operação de atualização, do tipo PATCH, permite ao administrador formalizar a adoção, alterando o status da solicitação. Para isso, o id da solicitação é passado pela URL e o novo status é enviado no corpo da requisição,

Figura 22 - *Endpoint* para atualizar uma adoção



Fonte: Autoria própria (2025).

6 CONSIDERAÇÕES FINAIS

O abandono de animais é uma realidade no Brasil e mundialmente. Para minimizar essa problemática, algumas instituições realizam trabalho de apoio aos animais abandonados e de combate ao abandono, a exemplo de ONGs, protetores independentes e grupos de voluntários. Em Pau dos Ferros, uma das instituições que vem trazendo resultados positivos é o Unidos pelas Patinhas. Como forma de apoio a suas operações, Santos *et al*. (2022) propôs uma aplicação *móvel* para apoio às operações associadas aos animais abandonados. No entanto, um aspecto importante da plataforma não havia sido implementado, referente ao *backend* do ambiente. Buscando minimizar essa lacuna, esse trabalho apresenta uma proposta de um *backend* para suporte a ambientes de controle de animais abandonados. Ao longo do trabalho foram apresentadas as principais características do ambiente, em termos de organização de módulos, componentes e conectores e implantação. Também foram apresentados alguns *endpoints* para exemplificar o funcionamento do *backend*.

O desenvolvimento do frontend e do backend ocorreu de forma independente, o que pode gerar divergências entre as especificações de cada módulo. Atualmente, a integração entre ambos encontra-se em fase de avaliação, sendo necessário determinar o estágio real do frontend para prosseguir com os testes. Apesar dessas questões, a arquitetura proposta demonstra potencial para oferecer uma solução sólida, extensível e alinhada às boas práticas da engenharia de software.

6.1 Trabalhos Futuros

A conclusão deste projeto não representa um ponto final, mas sim a fundação sobre a qual novas funcionalidades podem ser construídas. A evolução do sistema pode contemplar melhorias específicas. Entre elas, a implementação de notificações em tempo real, utilizando WebSockets com Socket.IO, para alertar administradores sobre novas solicitações de adoção sem a necessidade de consultas manuais.

Outra possibilidade é a criação de um módulo de relatórios e estatísticas para uso exclusivo da administração, contendo métricas como número de adoções por período, perfil dos animais mais procurados e taxa de conversão de solicitações. Esses dados forneceriam subsídios para decisões estratégicas e otimização de recursos.

A integração com redes sociais poderia automatizar a divulgação dos animais. Ao cadastrar um novo animal, o sistema geraria automaticamente um post com imagem, informações essenciais e link para o perfil no aplicativo, facilitando o trabalho dos voluntários e ampliando o alcance das campanhas de adoção.

Em síntese, o projeto consolidou um conjunto de práticas e soluções que reforçam a importância da combinação entre técnica e sensibilidade no desenvolvimento de sistemas voltados para demandas sociais. As propostas de evolução visam ampliar o impacto da aplicação, tornando-a ainda mais ágil, estratégica e integrada aos canais de comunicação utilizados pelo grupo Unidos pelas Patinhas.

REFERÊNCIAS

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. 3. ed. Boston, MA, USA: Addison-Wesley Professional, 2012.

COECKELBERGH, Mark. **AI ethics**. MIT press, 2020.

EXPRESSJS. **Express - Node.js web application framework**. Disponível em: <https://expressjs.com/>. Acesso em: 2 ago. 2025.

FIELDING, Roy Thomas. **Architectural styles and the design of network-based software architectures**. 2000. Tese (Doutorado em Ciência da Computação) - University of California, Irvine, 2000.

JESTJS. **Jest Framework**. Disponível em: <https://jestjs.io/>. Acesso em: 2 ago. 2025.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 7. ed. São Paulo: Atlas, 2022.

JONAS, Hans. **O princípio responsabilidade, ensaio de uma ética para a civilização tecnológica.** Rio de Janeiro: Editora PUC RIO, 2006.

MYSQL. **The world's most popular open source database**. Disponível em: <https://www.mysql.com/>. Acesso em: 2 ago. 2025.

NODEJS. **Run JavaScript Everywhere**. Disponível em: <https://nodejs.org/pt>. Acesso em: 2 ago. 2025.

PRISMA. **Instant Postgres plus an ORM for simpler db workflows**. Disponível em: <https://www.prisma.io/>. Acesso em: 2 ago. 2025.

SANTOS, Joseanny et al. Desenvolvimento de app mobile para adoção de animais recuperados pela ONG Unidos pelas Patinhas-Pau dos Ferros/RN. **Conjecturas**, v. 22, n. 16, p. 704-720, 2022.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. — São Paulo : Pearson. 2019.

VALOR ECONÔMICO. **Mais de 30 milhões de animais vivem em situação de abandono no Brasil, mostra pesquisa da Mars Petcare**. Disponível em: <https://valor.globo.com/conteudo-de-marca/mars-petcare/noticia/2024/12/13/mais-de-30-milhoes-de-animais-vivem-em-situacao-de-abandono-no-brasil-mostra-pesquisa-da-mars-petcare.ghtml>. 2024. Acesso em: 2 ago. 2025.