UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

LUIZ ANDRÉ VENTURA GOSLING

Pet Savior: um aplicativo para dispositivos móveis que visa ajudar cães e gatos perdidos, abandonados ou vítimas de maus tratos.

Florianópolis
2024

LUIZ ANDRÉ VENTURA GOSLING

Pet Savior: um aplicativo para dispositivos móveis que visa ajudar cães e gatos perdidos, abandonados ou vítimas de maus tratos.

Trabalho de conclusão de curso submetido ao corpo docente do Departamento de Informática e Estatística como um dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal de Santa Catarina(UFSC).

Orientador:

_____

Prof. Frank Augusto Siqueira

Banca Examinadora:

_____

Prof. José Eduardo De Lucca

_____

Prof. Cristian Koliver

DEDICATÓRIA

Ser aprovado para cursar Sistemas de Informação na UFSC, em 2016, foi uma das maiores conquistas que tive. Tal acontecimento mudou o rumo de minha vida em vários sentidos. Durante este período, experimentei altos e baixos, vitórias e reveses, mas, por fim, pude desenvolver a resiliência necessária para superar os desafios e com eles aprender. Muito do que vivi nesses últimos anos, só pude por ter uma ligação com a UFSC e, consequentemente, Florianópolis.

Dedico esse trabalho de conclusão de curso à minha amada companheira Letícia, que está sempre ao meu lado, apoiando-me em todos os momentos, com muita lealdade, cumplicidade e amor.

À minha amada mãe Maria Lúcia que acreditou e investiu em mim desde o princípio, jamais medindo esforços para me proporcionar tudo que estava ao seu alcance, sendo um porto seguro.

Ao meu amado pai Luiz Renato sempre disposto a me ouvir e aconselhar em momentos difíceis, tendo moldado muito do meu caráter e me ensinado tanto através de sua própria vivência.

Ao meu grande parceiro nessa jornada da UFSC, José Ribamar, por todo apoio, conversas e espírito de equipe. Triunfamos juntos. Tenho muito orgulho do que fizemos. Obrigado!

À Fabio, Marcos, Jeferson e Eduardo, grandes amigos que de algum modo, nos últimos anos, me apoiaram, ouviram e certamente contribuíram para essa conquista.

Aos meus avós, todos já falecidos, mas que ajudaram em minha educação e encheram-me de amor e incentivo no início da vida. Rubens e Maria Otília. Renato e Maria Lúcia.

Menciono honrosamente aqui também o colega Alan, que em momentos oportunos colaborou com dicas e incentivo. Obrigado!

Muito obrigado também ao meu orientador Frank Siqueira por ter acreditado na ideia que propus, por todo incentivo, apoio e orientação. Seu norteamento foi essencial.

Por fim, agradeço à UFSC e a todos que, de algum modo, participaram de forma positiva desta jornada tão importante em minha vida.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
APP	Aplicativo
ORM	Object-Relational Mapping
CRUD	Create, Read, Update, Delete
IOS	Iphone Operating System
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SDK	Software Development Kit
AVD	Android Virtual Device
SO	Sistema Operacional
JSON	JavaScript Object Notation
JWT	Json Web Token
MVC	Model-View-Controller
MVCC	Multiversion Concurrency Control
RDBMS	Relational Database Management System
GPS	Global Positioning System
TCC	Trabalho de Conclusão de Curso

SUMÁRIO

RESUMO

Este projeto objetiva o desenvolvimento de um aplicativo para dispositivos móveis que servirá como um espaço digital de anúncios de animais perdidos ou para adoção, entre outras ações beneficentes, tendo como finalidade principal prover uma plataforma na qual usuários dispostos a ajudar possam encontrar animais desamparados, de modo funcional e simplificado.

O aplicativo foi desenvolvido utilizando a arquitetura cliente-servidor e segue o padrão MVC *(Model-View-Controller*). O front-end, criado com React Native, oferece uma interface intuitiva e amigável para os usuários, facilitando a navegação e interação com as funcionalidades do aplicativo. O back-end, implementado em Node.js, desempenha um papel essencial na integração dos dados provenientes do cliente. Ele gerencia informações de localização no mapa, perfis de animais cadastrados e outras funcionalidades críticas. Entre as principais funcionalidades do aplicativo estão: a exibição da localização dos animais no mapa, permitindo aos usuários encontrem facilmente os pets, a criação e visualização de perfis detalhados de animais, um espaço dedicado para que os usuários denunciem casos de maus tratos e uma sessão de parceiros que ofertam produtos e serviços para pets, conectando prestadores de serviço com tutores.

Palavras-chave: Aplicativos Móveis, React Native, Typescript, Node.js, Express.js, Adoção de cães e gatos, Animais de rua, Pets.

ABSTRACT

This project aims to develop a mobile application that will serve as a digital space for posting announcements about lost or adoptable animals, among other charitable actions. Its main purpose is to provide a functional and straightforward platform where users willing to help can find homeless animals.

The application was developed using the client-server architecture and follows the MVC (*Model-View-Controller*) programming pattern. The front-end, created with React Native, offers an intuitive and user-friendly interface, making it easy for users to navigate and interact with the application's features. The back-end, implemented in Node.js, plays a crucial role in integrating data from the client. It manages information about map locations, registered animal profiles, and other critical functionalities.

Key features of the application include displaying the locations of animals on the map, allowing users to easily find pets; creating and viewing detailed animal profiles; a dedicated space for users to report cases of animal abuse; and a session of partners that offer products and services for pets, connecting service providers with pet owners.

Keywords: Mobile Apps, React Native, Typescript, Node.js, Express.js, Dogs and cats's adoption, Street animals, Pets.

1. INTRODUÇÃO

Nessa seção serão apresentados os objetivos e a justificativa do projeto.

1.1 Justificativa

Diante do crescente número de animais em situação de abandono e vulnerabilidade no Brasil, torna-se evidente a necessidade de um aplicativo que facilite a adoção, o financiamento e o reencontro de animais desaparecidos com suas famílias. Além de promover o bem-estar animal, um aplicativo com essas funcionalidades pode aproveitar o significativo mercado pet brasileiro, que faturou R$ 40,8 bilhões em 2020. Portanto, o desenvolvimento do aplicativo "Pet Savior" visa atender a essa demanda, utilizando tecnologias modernas e conceitos aprendidos durante o curso de Sistemas de Informação.

1.2 Solução Proposta

Diante dos cenários descritos e dado o contexto histórico e social, é inegável que o problema contemporâneo quanto a cães e gatos abandonados e vítimas de maus tratos é relevante. A tecnologia vem em parte melhorando a vida de seres humanos e, consequentemente, de seus animais de estimação, há muito. Nas últimas décadas, com a popularização da internet e smartphones, é possível fazer mais por causas como a anteriormente exposta. Sendo assim, uma solução proposta, mas que de forma alguma pode ser considerada (ao menos de início) ampla e definitiva, é o desenvolvimento de um aplicativo funcional e bem estruturado, que atue como uma ferramenta facilitadora na melhora da qualidade de vida de animais de estimação negligenciados.

1.3 Objetivos

A seção de objetivos deste projeto de conclusão de curso define as metas a serem alcançadas. Os objetivos gerais estabelecem a finalidade principal do projeto, enquanto os objetivos específicos detalham as ações necessárias para atingir esse propósito.

1.3.1 Objetivos Gerais

Este Trabalho de Conclusão de Curso (TCC) visa desenvolver um aplicativo para dispositivos móveis (com foco em *smartphones*) que colete informações sobre animais que estão em situação de vulnerabilidade nas ruas, que foram abandonados ou que estejam

desaparecidos. O intuito é promover adoção, financiamento e eventualmente ser um facilitador para o reencontro entre animais desaparecidos e suas famílias.

1.3.2 Objetivos Específicos

Elencamos como objetivos específicos deste trabalho:

- Elaborar fundamentação teórica que evidencie tanto a relevância de animais de estimação para membros da sociedade, quanto a necessidade do uso da tecnologia através de projetos como este, para que ao menos parte dos problemas expostos possa ser revertido através da filantropia, empatia e ações beneficentes e voluntárias dos usuários.

- Desenvolver um protótipo aplicativo levando em conta conceitos de arquitetura de software e usabilidade.

- Implementar o desenvolvimento do aplicativo utilizando tecnologias indicadas para tal e com boas práticas de programação.

- Avaliar o aplicativo desenvolvido com testes e casos de uso.

1.4. Metodologia

A metodologia proposta para o desenvolvimento do projeto tem o intuito de ser abrangente e sequencial, visando permitir o desenvolvimento de um aplicativo sólido, embasado em teorias e práticas recomendadas, além de possibilitar a inovação e a adequação às necessidades específicas dos usuários. Para tanto, seguirá as seguintes etapas:

<u>Etapa 1 - Fundamentação Teórica</u>

- Revisão bibliográfica sobre o abandono e maus-tratos de animais de estimação no Brasil.
- Identificação de conceitos-chave e fundamentos de cunho tecnológico necessários para embasar o desenvolvimento do aplicativo.

Etapa 2 - Estado da Arte

- Pesquisa de aplicativos móveis existentes no mercado relacionados à busca de animais perdidos e adoção responsável.
- Análise das funcionalidades e características dos aplicativos encontrados.
- Identificação de pontos fortes e limitações dos aplicativos existentes.
- Levantamento de oportunidades de inovação e diferenciação em relação aos aplicativos concorrentes.

Etapa 3 - Desenvolvimento do Aplicativo

- Modelagem do aplicativo, incluindo a definição de fluxos de navegação e estrutura de informações, requisitos funcionais, requisitos não funcionais e casos de uso.
- Desenvolvimento do *back-end*, que envolve a criação do banco de dados e a implementação da lógica de funcionamento do aplicativo.
- Desenvolvimento do *front-end*, com a criação da interface gráfica e elementos interativos para os usuários.
- Integração de recursos externos, como APIs de localização e notificações, para melhorar a funcionalidade do aplicativo.

Etapa 4 - Avaliação do Aplicativo

- Realização de testes de usabilidade para identificar possíveis problemas na interação do usuário com o aplicativo.
- Análise dos resultados da avaliação para identificar pontos fortes e áreas de melhoria do aplicativo.
- Iterações e ajustes com base nos resultados da avaliação, visando aprimorar a usabilidade e a eficácia do aplicativo.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos básicos e tecnologias basilares no desenvolvimento do aplicativo móvel.

2.1 Humanos e animais de estimação

O vínculo entre humanos e animais de estimação é algo complexo e extremamente profundo. Os primeiros sinais de domesticação de caninos datam mais de 12.000 anos atrás (Larson et al., 2012). Ao longo dessa trajetória, os cães, por exemplo, exerceram papéis de enorme utilidade para a raça humana, como pastores, vigilantes, parceiros de caça e, nos tempos contemporâneos, atuando como guias de deficientes visuais e ajudantes na detecção de câncer através do faro (Guest et al., 2021). Entretanto, a conexão estabelecida entre as duas espécies ultrapassa uma relação apenas utilitária, onde os cães obtêm alimentos de seus tutores e os humanos se beneficiam dos papéis que os cães desempenham. Pesquisas indicam uma poderosa relação de afeto e empatia, como demonstrado em um estudo de 2014 que aponta que cães experimentam uma resposta fisiológica ao choro de bebês humanos, sendo a primeira evidência clara de empatia entre espécies (Yong e Ruffman, 2014).

O que há muito se desconfiava em relação à convivência com um animal de estimação vem sendo demonstrado por meio de pesquisas científicas. Um estudo aponta que as interações homem-animal decorrentes da posse de animais de estimação têm uma ampla gama de benefícios para a saúde emocional e física, e que esses fatores também tendem a melhorar a cognição (McDonough et al., 2022).

2.2 Animais em situação de vulnerabilidade

Segundo um censo do Instituto Pet Brasil (IPB) de 2021, o Brasil é o terceiro país do mundo em número de animais domésticos, com 149,6 milhões de animais de estimação. Esta significativa presença de animais em lares brasileiros impacta diretamente o mercado pet. Em 2020, o mercado pet brasileiro concluiu o ano com um faturamento de R$ 40,8 bilhões, considerando Pet Shops, clínicas veterinárias e outros estabelecimentos relacionados. A estimativa mais atual do IPB indica que há 58,1 milhões de cães no país, um número impressionante. Destaca-se também o cômputo de animais abandonados ou resgatados por maus-tratos, totalizando 184.960 animais sob a tutela de ONGs e grupos de protetores, dos quais 177.562 (96%) são cães e 7.398 (4%) são gatos.

De acordo com um levantamento publicado pelo Instituto Pet Brasil, entre 2018 e

2020, o número de pets em situação de vulnerabilidade subiu de 3,9 para 8,8 milhões. A pesquisa, chamada ACV (Animais em Condição de Vulnerabilidade), é realizada a cada dois anos e considera animais que vivem sob a tutela de famílias classificadas abaixo da linha de pobreza ou animais que moram nas ruas e recebem algum tipo de ajuda de pessoas ao redor. Do total da população ACV, verificou-se que cães representam 69,4% (6,1 milhões) e gatos 30,6% (2,7 milhões). Em 2018, cães eram 69% (2,69 milhões), enquanto os gatos correspondiam a 31% (1,21 milhão). Esses números revelam que os impactos causados pela pandemia de COVID-19 também afetaram os animais de estimação, agravando a situação. Mais animais abandonados e sem cuidados adequados pode significar uma menor possibilidade de castração, resultando no nascimento de outros animais em condições inadequadas para sobreviver, perpetuando o ciclo de vulnerabilidade.

2.3 Sistemas Operacionais Móveis

De acordo com Maziero (2019), os sistemas operacionais são elementos fundamentais para o funcionamento de praticamente qualquer sistema de computação, desde minúsculos sistemas embarcados e telefones celulares até gigantescos centros de processamento de dados de grandes empresas. Apesar da imensa diversidade de sistemas operacionais existentes, eles tentam resolver problemas de mesma natureza e seguem basicamente os mesmos princípios.

Sistemas operacionais móveis são softwares projetados especificamente para dispositivos móveis, como smartphones e tablets. Eles são responsáveis por gerenciar os recursos físicos do dispositivo, exibir a interface de usuário e executar aplicativos. Além dos recursos já citados, esses sistemas oferecem conectividade de rede, acesso à internet, suporte a aplicativos de terceiros e integração com serviços online. Os sistemas operacionais móveis mais populares são Android e iOS.

2.3.1 Android

O Android é um sistema operacional móvel desenvolvido pelo Google, que foi lançado inicialmente em setembro de 2008. Desde então, passou por várias versões e atualizações, adicionando constantemente novos recursos e aprimorando sua funcionalidade. A plataforma Android se tornou uma das mais populares e amplamente utilizadas em dispositivos móveis em todo o mundo, com uma ampla gama de fabricantes adotando-o para seus smartphones e tablets.

Segundo Morimoto (2009), uma parte significativa da estratégia do Android está focada no desenvolvimento de aplicativos por empresas e programadores independentes. O

Google reconheceu que, assim como nos computadores, as plataformas de smartphones estão se tornando consolidadas, e o aspecto mais importante passou a ser os aplicativos, não apenas o hardware ou as funções básicas do sistema.

2.3.2 iOS

O iOS é o sistema operacional desenvolvido pela Apple, conhecido como iPhone Operating System. Projetado especificamente para atender às necessidades dos dispositivos móveis da empresa, o iOS oferece uma camada de abstração entre o hardware e os aplicativos. Sua arquitetura é composta por quatro camadas principais: CocoaTouch, Media, Core Services e Core OS (MENDONÇA e BITTAR, 2011).

A denominação "iOS" deriva da combinação do termo "OS" (sistema operacional, em inglês) com o icônico prefixo "i" presente em diversos produtos da Apple. Vale ressaltar que o iOS é exclusivo para dispositivos da marca e não é compatível com hardware de outras empresas. Seu principal concorrente é o sistema operacional Android (MARQUES, 2019).

2.4 Aplicativos móveis

Os aplicativos móveis são programas desenvolvidos para *smartphones* e tablets. Podem ser instalados a partir das lojas de aplicativos e oferecem diferentes funcionalidades e serviços aos usuários. São fonte de auxílio em diversas tarefas, entretenimento e conexão com outros usuários em seus dispositivos móveis.

De acordo com El-Kassas et al. (2015), a criação de aplicativos móveis apresenta características e restrições distintas em comparação ao desenvolvimento de outros tipos de software. Os desenvolvedores devem levar em consideração fatores como as capacidades dos dispositivos móveis, a mobilidade, as especificações dos aparelhos, o design e a navegabilidade da interface do usuário, bem como a segurança e a privacidade do usuário.

2.4.1 Aplicativos Nativos

De acordo com Fowler e Lewis (2014), aplicativos nativos são programas desenvolvidos especificamente para um sistema operacional móvel, como Android ou iOS. Eles oferecem um desempenho superior, acesso total aos recursos do dispositivo e uma interface de usuário nativa. Além disso, estão disponíveis nas lojas de aplicativos dedicadas, facilitando sua distribuição e descoberta pelos usuários. No entanto, seu desenvolvimento requer conhecimento específico da plataforma e implica em custos adicionais para suportar diferentes sistemas operacionais.

2.4.2 Aplicativos Híbridos

Aplicativos híbridos são aqueles desenvolvidos com tecnologias web e encapsulados em uma camada nativa para acessar recursos do dispositivo. Tal abordagem permite que um único código seja executado em diferentes plataformas, como Android e iOS, proporcionando economia de tempo e recursos durante o desenvolvimento.

Para criar um aplicativo nativo que possa ser executado em várias plataformas, é necessário desenvolver uma versão específica para cada plataforma, o que requer tempo, investimento e uma equipe com conhecimento nas diferentes tecnologias utilizadas por essas plataformas. No entanto, soluções *cross-platform* permitem a implementação de um único aplicativo que pode ser executado em diversas plataformas (EL-KASSAR et al., 2015).

3. ESTADO DA ARTE

Neste capítulo há uma análise de outros aplicativos com objetivos semelhantes ao desenvolvido neste projeto. São comentados e avaliados diferentes aplicativos, destacando suas funcionalidades e propostas. Ao final, apresentaremos uma tabela com as diferenças entre eles. Isso nos ajudará a entender o cenário atual, identificar oportunidades e embasar o desenvolvimento do aplicativo proposto pelo autor.

3.1 Pupz

Pupz é um conhecido aplicativo brasileiro voltado para adoção de cães e oferta de produtos e serviços relacionados à pets. Apresenta-se como uma plataforma onde pessoas podem encontrar cães para adoção em suas regiões, conectando adotantes e protetores de animais. O aplicativo possui recursos como filtros de busca, perfis de cães disponíveis para adoção, opções de contato com os protetores responsáveis e informações sobre o processo de adoção. Há também a opção de criar uma conta.

O aplicativo está disponível tanto na Google Play (Android) quanto na App Store (iOS). Para a análise, o autor realizou o download do app em seu *smartphone* Android. Na página inicial, é possível verificar três sessões principais: *Produtos, Serviços* e *Meus Pupz*. No canto superior do app, que por padrão vem com “Não Informado”, é possível clicar e então o usuário informar o CEP para que a localidade seja registrada e os produtos e serviços da área sejam exibidos.

A seção de *Produtos* conta com anúncios em um slide no centro da tela e um considerável número de categorias. São elas: Cosméticos, Brinquedos, Acessórios, Moda e Look, Alimentação, Medicina e Saúde, Caça e Pesca, Outlet, Higiene e Limpeza, Inovadores, Fazenda, Plantas e Flores, Casa e Jardim, Piscina e Camping e Lazer.

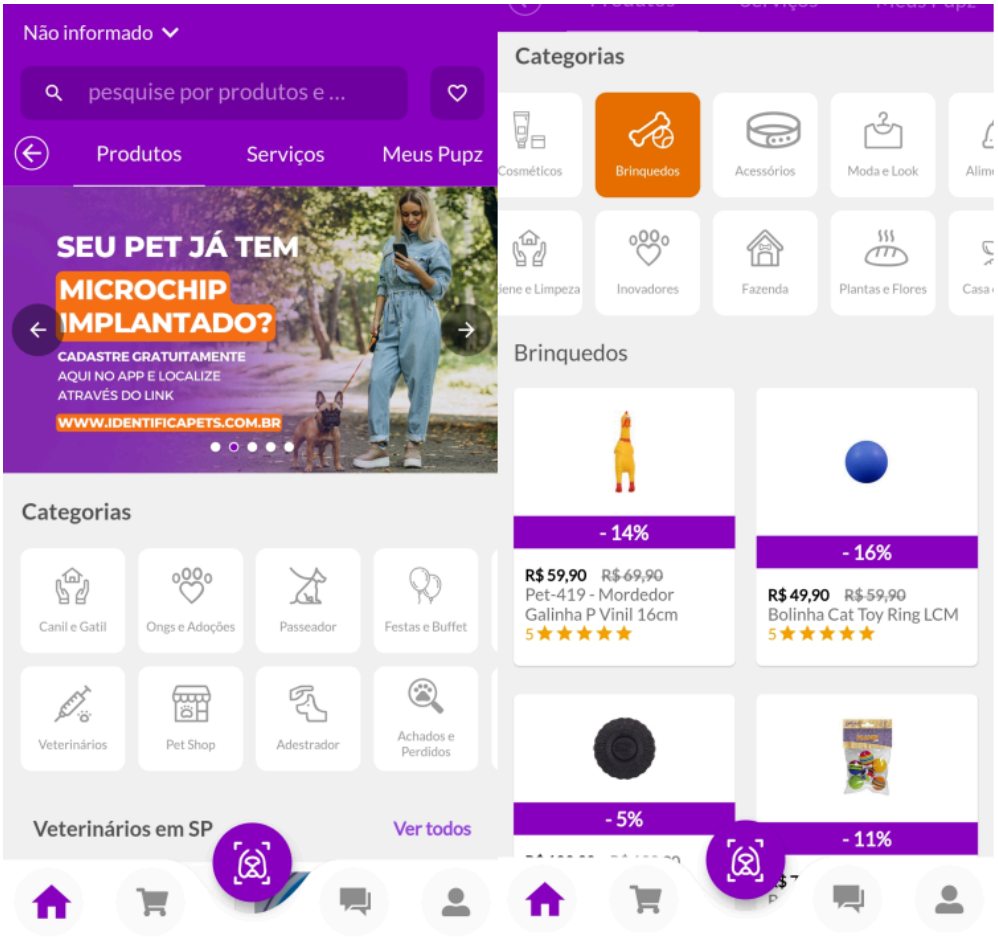

Figura 1. Tela inicial e Tela categoria ‘Brinquedos’ selecionada

Já na seção de *Serviços*, é possível encontrar as seguintes categorias: Canil e Gatil, ONGs e Adoções, Passeador, Festas e Buffet, Memorial, Veterinários, Pet Shop, Adestrador, Achados e Perdidos, Despedida.

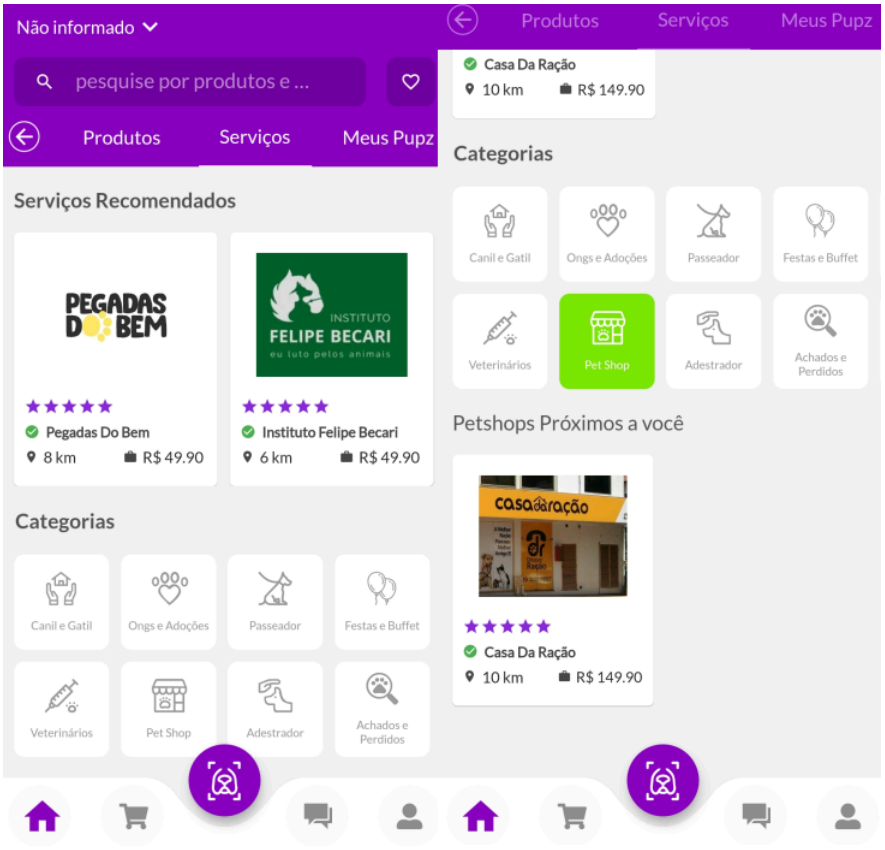

Figura 2. Tela de *Serviços* e Tela com a categoria "*Pet Shop*" selecionada

Por fim, a seção *Meu Pupz* trata-se de uma página específica onde pode-se realizar uma autenticação do animal e utilizar um reconhecimento facial. Segundo os criadores do app, quando um animal se encontra perdido, o aplicativo permite que qualquer pessoa que o encontre tire uma foto do *pet* para localizar seu tutor. A comunicação entre o dono do animal e a pessoa que o encontrou é facilitada por meio de um chat integrado dentro do aplicativo. Essa abordagem proporciona maior segurança e privacidade para ambas as partes, seguindo as diretrizes estabelecidas pela Lei Geral de Proteção de Dados (LGPD). Dessa forma, o aplicativo oferece uma solução eficiente e confiável para reunir animais perdidos e seus tutores de forma segura.

3.2 Animal de Estimação Perdido

Este é um aplicativo presente somente na Google Play (Android). Assim como no caso anterior, o autor fez o download do app para seu *smartphone* Android e o testou. O design é um tanto minimalista e não possui uma série de produtos e serviços ofertados. Conta com três seções principais: *Ausente*, *Encontrado* e *Seguindo.*

Diferentemente do Pupz, basta autorizar o aplicativo quanto ao uso do GPS do *smartphone* que a localização é registrada. A função 'Reportar Estimação' que aparece centralizada na parte inferior da tela na figura, não funciona.

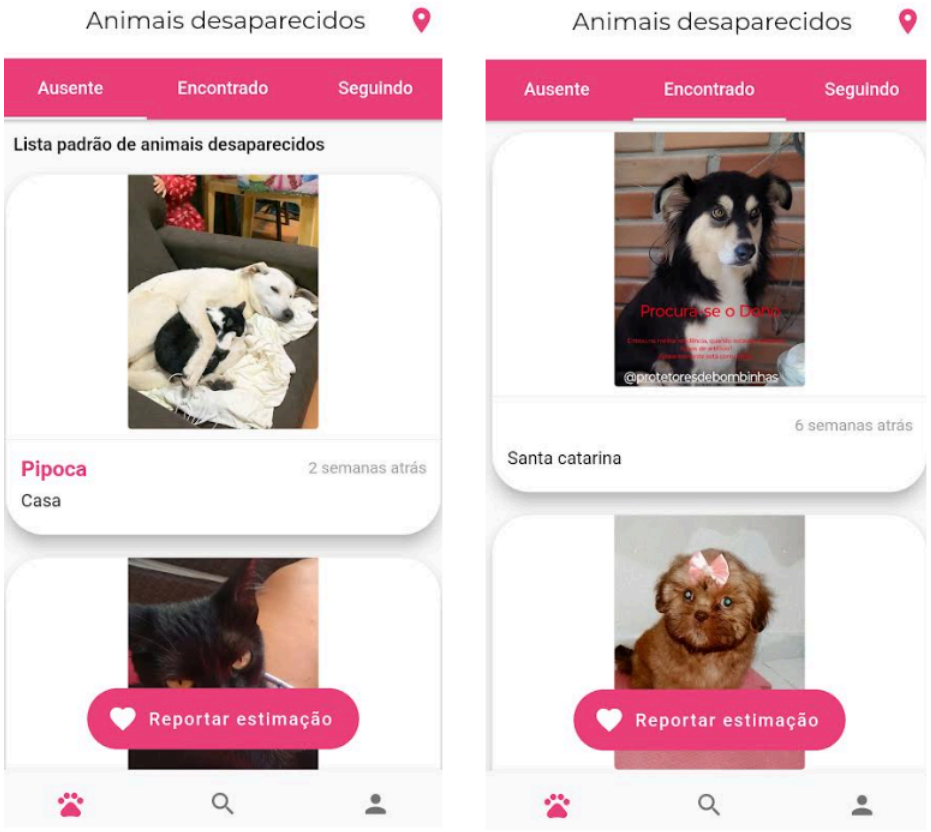

Figura 3. Tela de Ausente e Tela de Encontrado

Na seção *Ausente* aparecem listados alguns *pets* que estão perdidos. Ao clicar no card com a foto de um deles, há um redirecionamento para uma próxima tela que dá detalhes como a data que o animal desapareceu, o sexo, o contato dos donos (email e telefone) a recompensa e uma breve mensagem. Existe também a opção "Ver no mapa" que indica o registro do local onde o animal foi visto pela última vez.

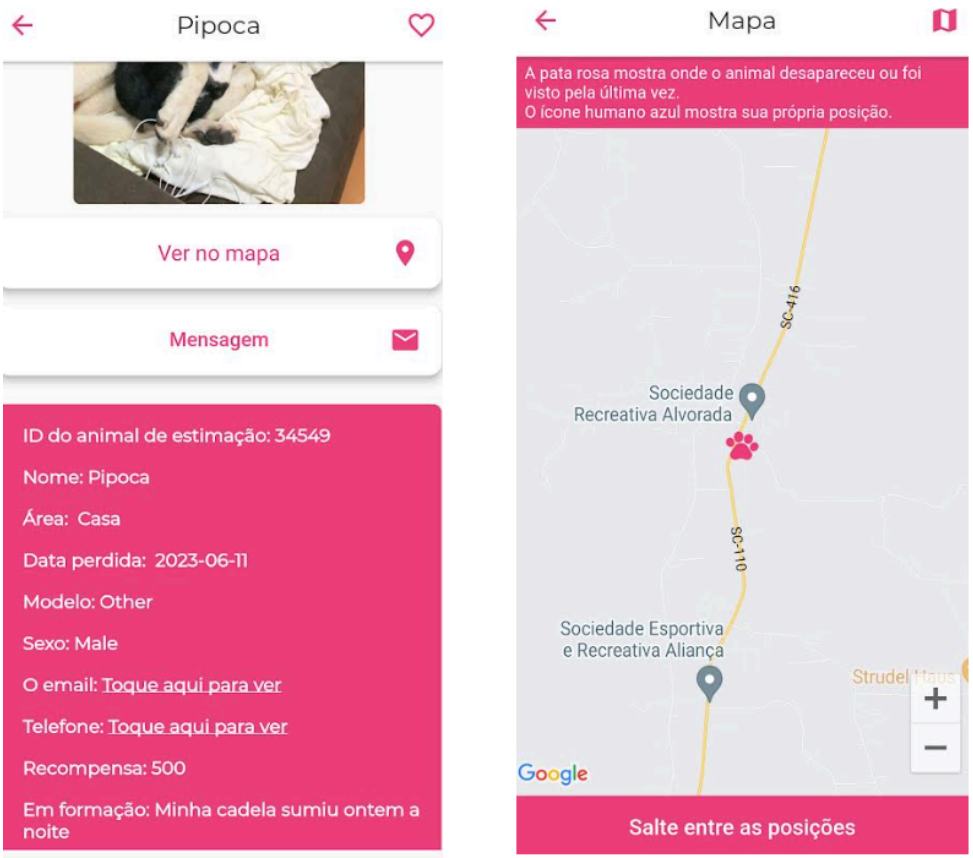

Figura 4. Tela de descrição do animal perdido e Tela da funcionalidade de indicação no mapa

Já na seção *Encontrado* presume-se que estão listados os animais que já foram localizados e estão com suas famílias.

3.3 Tabela Comparativa

APP	Adoção	Buscar animal perdido	Serviços oferecidos	Denúncia de maus tratos	Nota na Google Play	Nota na App Store
Pupz	Sim	Sim	Sim	Não	4,1	3,3
Animal de Estimação Perdido	Nao	Sim	Não	Não	3,8	-
Pet Savior	Sim	Sim	Sim	Sim	-	-

Tabela 1. Comparativo com aplicativos que já estão publicados

A tabela comparativa apresenta algumas diferenças significativas entre os aplicativos *Pet Savior*, *Pupz e Animal de Estimação Perdido*. Uma observação importante é que o *Pet Savior* está disponível na App Store e na Google Play, o que o diferencia dos outros dois aplicativos que estão restritos a apenas uma das plataformas. Além disso, a tabela revela que o *Pet Savior* é o único dentre os três aplicativos que oferece a funcionalidade de denúncia de maus tratos, demonstrando um compromisso em lidar com esta importante e urgente questão. Tal característica exclusiva do *Pet Savior* destaca a preocupação com o bem-estar dos animais e a promoção de uma cultura de proteção. Tais diferenças apontam para o grande potencial do aplicativo *Pet Savior* em se destacar no mercado e oferecer uma solução abrangente para ajudar animais perdidos e combater os maus tratos

4. DESENVOLVIMENTO DO APLICATIVO

Neste capítulo são apresentados os requisitos funcionais e não funcionais, bem como uma explanação acerca das ferramentas e tecnologias que foram utilizadas para o desenvolvimento.

4.1 Requisitos

Nesta seção, são abordados os elementos essenciais do aplicativo, incluindo os requisitos funcionais, não funcionais e as regras de negócios que o envolvem. De acordo com Sommerville (2018), os requisitos de um sistema representam as necessidades dos clientes em termos dos serviços que o sistema deve oferecer e das restrições que afetam sua operação. Esses requisitos são fundamentais para que o sistema atenda ao propósito específico para o qual foi desenvolvido. Além disso, Sommerville (2018) ressalta que os requisitos não são independentes, podendo influenciar ou ser limitados por outros requisitos. Eles não se limitam apenas a descrever as características ou serviços do sistema, mas também englobam as funcionalidades necessárias para garantir o correto funcionamento desses serviços e características, conforme especificado.

4.1.1 Requisitos Funcionais

Os requisitos funcionais desempenham um papel crucial no desenvolvimento do sistema, uma vez que descrevem as funcionalidades que estarão presentes na aplicação. Conforme definido por REINEHR (2020), um requisito funcional é uma funcionalidade necessária solicitada pelos stakeholders para atingir um objetivo de negócio. Ele representa o que os desenvolvedores devem implementar para que os usuários possam realizar suas atividades no sistema. Esses requisitos são essenciais para garantir que a aplicação atenda às necessidades e expectativas dos usuários.

Os requisitos funcionais deste projeto foram levantados com base na experiência do autor como usuário de outros aplicativos com propósitos similares ou não, além de uma pesquisa de campo envolvendo contato com soluções que catarinenses utilizam para lidar com o desaparecimento de animais e processos de adoção.

Requisito	Descrição
RF001	O aplicativo deve permitir o cadastro de usuários com informações básicas como nome, email e senha.
RF002	O aplicativo deve permitir a autenticação de usuários para garantir o acesso seguro.
RF003	O aplicativo deve permitir a recuperação de senha para os usuários.
RF004	O aplicativo deve permitir a busca por pets perdidos.
RF005	O aplicativo deve permitir o cadastro de animais encontrados para ajudar na identificação e reunião com seus donos.
RF006	O aplicativo deve permitir que os usuários cadastrem pets perdidos.
RF007	O aplicativo deve permitir a exibição de detalhes sobre animais perdidos, como descrição e localização.
RF008	O aplicativo deve permitir a criação de alertas para receber notificações sobre animais perdidos na área
RF009	O aplicativo deve permitir o compartilhamento de informações sobre animais perdidos através de redes sociais ou outras plataformas.
RF010	O aplicativo deve permitir que os usuários façam denúncias de maus tratos a animais, fornecendo um canal de comunicação seguro e confidencial.
RF011	O aplicativo deve permitir o cadastro de cães disponíveis para adoção, incluindo informações sobre espécie, raça,

	idade e fotos
RF012	O aplicativo deve permitir que o usuário veja os pets disponíveis para adoção
RF013	O aplicativo deve conter um mapa onde mostrará as últimas localizações de pets perdidos.
RF014	O aplicativo deve permitir que usuários insiram no mapa as localizações de pets.

Tabela 2. Requisitos funcionais

4.1.2 Requisitos Não Funcionais

Os requisitos não funcionais desempenham um papel essencial no sistema, pois definem o comportamento e as características gerais que o sistema deve apresentar. Enquanto os requisitos funcionais se concentram nas funcionalidades específicas oferecidas aos usuários, os requisitos não funcionais estão relacionados a propriedades mais amplas e abrangentes do sistema.

De acordo com SOMMERVILLE (2011), os requisitos não funcionais abordam aspectos como confiabilidade, tempo de resposta e uso de recursos. Eles podem influenciar a arquitetura geral do sistema e podem surgir como resultado de requisitos funcionais ou como restrições impostas ao sistema. Dessa forma, os requisitos não funcionais são essenciais para garantir que o sistema atenda aos padrões de desempenho, segurança, usabilidade e outros critérios relevantes, proporcionando uma experiência satisfatória aos usuários e cumprindo as expectativas estabelecidas.

Os requisitos não funcionais deste projeto foram levantados com base na experiência profissional e acadêmica do autor.

Requisito	Descrição
RNF001	O aplicativo deve ser desenvolvido utilizando o framework React Native
RNF002	O *back-end* deve ser escrito utilizando o Node.js

	e Express.js
RNF003	As senhas dos usuários devem ser criptografadas para garantir a segurança no armazenamento.
RNF004	O aplicativo deve ser desenvolvido de forma híbrida, garantindo compatibilidade com os sistemas operacionais Android e iOS.
RNF005	O aplicativo deve seguir a arquitetura cliente/servidor, permitindo uma comunicação eficiente entre o dispositivo do usuário e o servidor de backend.

Tabela 3. Requisitos não funcionais

4.2 Casos de Uso

Casos de uso são descrições concisas do comportamento do sistema, especificando uma sequência de ações que resultam em um valor observável para os atores envolvidos. Segundo Booch, Rumbaugh e Jacobson (2005), os casos de uso são criados de maneira compreensível para desenvolvedores e usuários finais, mesmo antes de terem sido implementados, facilitando uma compreensão comum entre as partes interessadas.

Nesta seção são exibidos alguns dos principais casos de uso e suas correlações com os requisitos funcionais anteriormente listados. Futuramente, o fluxo completo será informado.

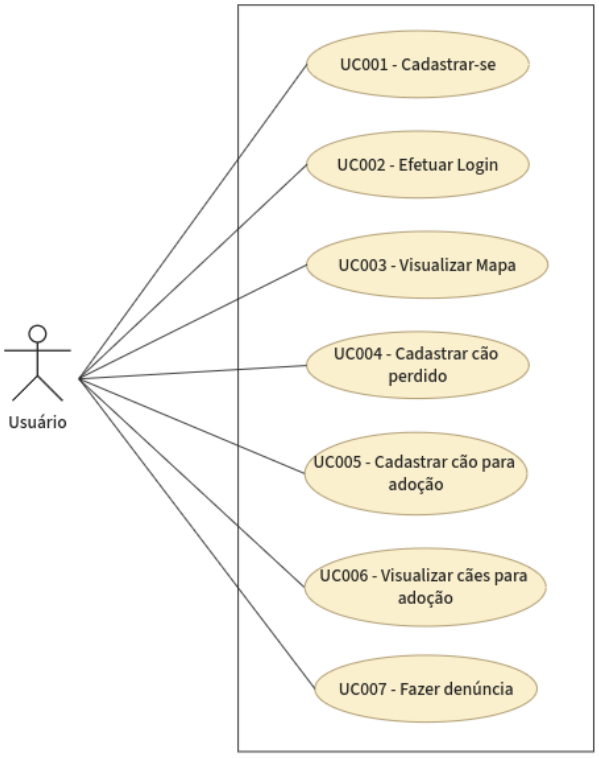

Figura 5. Diagrama de caso de uso

Caso de Uso	Descrição	Requisitos funcionais
UC001 - Cadastrar-se	Cadastro de usuário no app	RF001
UC002 - Efetuar Login	Login de usuário no app	RF002
UC003 - Visualizar Mapa	Visualizar seção de mapa do app, podendo inserir localizações	RF013, RF014
UC004 - Cadastrar pet perdido	Efetuar cadastro de um pet perdido informando principais dados	RF006
UC005 - Cadastrar pet para adoção	Efetuar cadastro de um pet para adoção	RF011

UC006 - Visualizar pets para adoção	Visualizar lista de perfis de cães para adoção no app	RF012
UC007 - Fazer denúncia	Realizar denúncia de maus tratos na seção que tem tal finalidade no app	RF010

Tabela 4. - Relação de casos de uso com requisitos funcionais

4.3 Ferramentas de Desenvolvimento

Nesta seção estão apresentadas as tecnologias que farão parte do desenvolvimento do aplicativo.

4.3.1 HTML

Segundo Flatschart (2011):

> HTML (Hypertext Markup Language - Linguagem de Marcação de Hipertexto) é a principal linguagem utilizada na web. Ela permite a criação de documentos estruturados em títulos, parágrafos, listas, links, tabelas, formulários e em muitos outros elementos nos quais podem ser incorporadas imagens e objetos como, por exemplo, uma animação ou um vídeo.

Podemos também definir HTML como uma linguagem de marcação utilizada para estruturar e apresentar conteúdo. Fornece elementos e tags que definem a estrutura e o formato do conteúdo. No aplicativo desenvolvido, o HTML desempenha um papel fundamental na construção do layout. Suas sintaxes são utilizadas para estruturar e formatar o conteúdo, permitindo a criação de uma interface visual atraente e funcional.

4.3.2 CSS

Enquanto o HTML é responsável por estruturar e formatar as páginas web, o CSS desempenha um papel essencial em melhorar sua aparência. De acordo com Silva (2007), a

definição mais precisa e simples para folha de estilo encontra-se na homepage da CSS no site do W3C e diz: "*Folha de estilo em cascata é um mecanismo simples para adicionar estilos(por exemplo: fonte, cores, espaçamentos) aos documentos web."*
De acordo com Munzlinger (2011), o CSS traz consigo algumas vantagens significativas, incluindo a facilidade de manutenção do código das páginas do site, a possibilidade de reutilizar estilos de formatação e a separação entre o conteúdo e a aparência do site, em conformidade com os padrões internacionais de desenvolvimento web estabelecidos pelo W3C.

No contexto do aplicativo desenvolvido, o CSS desempenha um papel fundamental, permitindo moldar o design da aplicação. O mesmo oferece a capacidade de criar um visual personalizado e adequado para proporcionar a melhor experiência possível.

4.3.3 JavaScript

De acordo com Flanagan (2014), JavaScript é uma linguagem de programação versátil, dinâmica e interpretada, que oferece suporte a estilos de programação orientados a objetos e funcionais. Sua sintaxe foi influenciada pela linguagem Java, pelas funções de primeira classe Scheme e pela herança baseada em protótipos da linguagem Self. JavaScript é uma linguagem de alto nível e não tipada, o que significa que não é necessário declarar explicitamente os tipos de dados das variáveis. Essas características tornam o JavaScript uma opção conveniente e poderosa para o desenvolvimento de aplicativos web interativos.

4.3.4 TypeScript

TypeScript é uma linguagem de programação que se baseia no JavaScript e oferece recursos adicionais, como tipagem estática e suporte a recursos avançados de programação. Criada pela Microsoft, é uma escolha popular para o desenvolvimento no *back-end*, devido à sua escalabilidade e ampla adoção em frameworks *front-end*. Utilizar TypeScript em projetos FullStack permite que os desenvolvedores trabalhem com uma única linguagem de programação, aumentando a produtividade. Além disso, a vasta quantidade de bibliotecas disponíveis no ecossistema do TypeScript ajuda a resolver uma ampla variedade de problemas. Conforme mencionado por Cherny (2019), *"Ao aproveitar ferramentas, bibliotecas e frameworks já existentes para desenvolver tanto o frontend quanto o back-end, podemos avançar rapidamente e construir nossa aplicação sobre uma base sólida."*.

No aplicativo desenvolvido neste projeto, o TypeScript é amplamente utilizado. Está presente tanto no *back-end* através do Node.js, quanto no *front-end* através do React Native.

4.3.5 MVC

MVC é um padrão arquitetural amplamente adotado no desenvolvimento de software. Esse padrão divide a arquitetura do software em três componentes principais, o que facilita a manutenção, a reutilização e a segurança do sistema (LUCIANO; ALVES, 2017). Para garantir a independência dos componentes, é essencial organizar o sistema em camadas, o que promove escalabilidade e eficiência.

No padrão MVC, o Modelo é responsável por manipular e armazenar os dados da aplicação. A Visão lida com a interface do usuário, capturando ações e apresentando dados, enquanto o Controlador gerencia a interação entre a Visão e o Modelo, processando as requisições e controlando a lógica da aplicação (LUCIANO; ALVES, 2017).

4.3.6 React Native

O React Native é um framework de desenvolvimento de aplicativos móveis, desenvolvido por engenheiros do Facebook, que permite criar aplicativos móveis utilizando JavaScript (ou Typescript). Sua abordagem é de desenvolvimento híbrido, permitindo que os desenvolvedores escrevam código uma vez e o executem em múltiplas plataformas, como iOS e Android. O React Native oferece uma ampla variedade de componentes pré-construídos e permite a integração com bibliotecas nativas, proporcionando uma experiência de desenvolvimento eficiente e rápida. Danielsson (2016) afirma que:

> O principal objetivo do React Native é simples, um desenvolvedor não deve exigir conhecimento ou precisar gastar tempo supérfluo para criar um aplicativo móvel desde que pelo menos dois aplicativos precisam ser desenvolvidos para suportar iOS e Android. Como plataformas diferentes têm aparências, sensações e recursos diferentes, não pode haver um aplicativo que é homogêneo em todos os sistemas operacionais. Porém como é a interface gráfica que difere, o desenvolvimento poderia se basear na mesma linguagem mas ter os gráficos ser renderizados de forma diferente dependendo da plataforma de destino e serem componentes nativos reais.
> O Facebook chama essa abordagem de "aprenda uma vez, escreva em qualquer lugar", que descreve o que é o React Native.

O React Native foi o framework para desenvolvimento *mobile* escolhido para desenvolver o *front-end* (ou cliente) do aplicativo deste projeto.

4.3.6.1 JSX

JSX, que significa JavaScript XML, é uma extensão de sintaxe especial utilizada no React. Ela permite escrever código HTML dentro de arquivos JavaScript. Em projetos React, não se cria arquivos HTML separados, pois o JSX possibilita combinar HTML e JavaScript no mesmo arquivo. Isso torna o desenvolvimento mais prático, já que podemos executar código JavaScript diretamente no HTML usando chaves *{ }*. No início, o JSX pode parecer estranho, mas ele é muito útil para a criação de interfaces de usuário interativas. Além disso, há algumas regras a seguir, como sempre fechar as tags HTML e escrever atributos em *camelCase* (EYGI, 2020).

4.3.6.2 Expo

De acordo com Hayani(2020), Expo é uma ferramenta que simplifica a construção de aplicativos React Native ao reduzir a complexidade de configuração e instalação do ambiente de desenvolvimento. Ele permite que os desenvolvedores criem aplicativos móveis usando JavaScript e React Native com maior facilidade, sem a necessidade de configurações complicadas de build. Expo fornece um cliente que pode ser usado para visualizar o desenvolvimento do aplicativo em tempo real em dispositivos móveis. Além disso, suporta múltiplas funcionalidades nativas através de bibliotecas e plugins, facilitando o acesso a APIs nativas como câmera, localização, notificações, e mais.

4.3.7 Leaflet

Leaflet é uma biblioteca JavaScript open source usada para criar mapas interativos. Foi desenvolvida em 2011 por Volodymyr Agafonkin, um ucraniano com formação matemática. Leaflet pode funcionar com o código incorporado diretamente em um documento HTML, desde que o código esteja dentro da tag <*script*> (NGUGI, 2023). No contexto do aplicativo desenvolvido, a biblioteca Leaflet foi utilizada na criação do mapa.

4.3.8 JSON

JSON, ou JavaScript Object Notation, é um formato de dados amplamente utilizado para troca de informações entre sistemas. Embora o nome faça referência ao JavaScript, o JSON pode ser utilizado com diversas linguagens de programação (NOLETO, 2022). Na aplicação em questão, o JSON desempenha um papel fundamental ao permitir o envio e recebimento de informações entre o aplicativo móvel e o back-end. É uma ferramenta

essencial para garantir a comunicação eficiente e precisa entre as partes envolvidas no sistema.

4.3.9 Node.js

Node.js é um ambiente de execução de código JavaScript no lado do servidor, focado no desenvolvimento de aplicações back-end. Permite que os desenvolvedores utilizem JavaScript para criar servidores web, APIs e outras soluções de back-end. Uma das vantagens do Node.js é a possibilidade de utilizar o TypeScript, o que ajuda a evitar erros e facilita o desenvolvimento de grandes projetos.A principal proposta do Node.js é possibilitar a criação de aplicações de rede escaláveis, utilizando um modelo orientado a eventos, em contraste com outras soluções que adotam um modelo orientado a threads, conforme mencionado por Tilkov et al (2010).

Devido à sua eficiência e escalabilidade, o Node.js foi escolhido como a tecnologia principal para construir o back-end do aplicativo.

4.3.10 Express.js

Express.js é um framework web rápido, minimalista e não opinativo para Node.js. Ele fornece uma camada robusta de funcionalidades essenciais para aplicativos web e APIs, simplificando o desenvolvimento ao reduzir a quantidade de código necessário. Construído sobre Node.js, Express.js permite a criação de backends de maneira eficiente, sendo uma escolha popular para desenvolver APIs RESTful. O propósito do Express.js é facilitar a criação de servidores e a definição de rotas, permitindo que os desenvolvedores tenham total controle sobre as requisições e respostas. Ele é amplamente utilizado tanto para aplicativos renderizados no servidor quanto para APIs e microsserviços.

Com Express.js, é possível integrar facilmente com frameworks de frontend como React, Angular ou Vue, criando aplicações full-stack. Além disso, Express.js é extremamente leve e rápido, o que o torna adequado para aplicações de grande escala que exigem alta performance. A comunidade vibrante e o rico ecossistema de pacotes e bibliotecas também contribuem para a sua popularidade, tornando o desenvolvimento de aplicações web com Node.js muito mais fácil e eficiente (YAKUBU, 2024).

4.3.11 Prisma

Prisma é uma ferramenta moderna de mapeamento objeto-relacional (ORM) que funciona com Node.js e TypeScript. Ela facilita a construção e gerenciamento de bancos de

dados, gerando automaticamente os tipos das entidades e proporcionando uma definição de esquema fácil de entender. É compatível com PostgreSQL, MySQL e SQLite. Seu objetivo é ajudar desenvolvedores a construir aplicações mais rapidamente e com menos erros (THOMAS, 2021).

4.3.12 PostgreSQL

O PostgreSQL é um poderoso sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto. Inicialmente criado como um sucessor do sistema de banco de dados Ingres, o PostgreSQL é conhecido por sua robustez, confiabilidade e escalabilidade, tornando-o uma escolha popular para aplicações de banco de dados grandes e complexas. Ele oferece suporte avançado para tipos de dados não relacionais, como JSON, e para dados espaciais (OYAMA, 2023).

4.3.13 Firebase

O Firebase é uma plataforma de desenvolvimento de aplicativos da Google que simplifica a criação de apps para iOS, Android e Web. Ele oferece uma série de ferramentas úteis para desenvolvedores, como análise de desempenho, autenticação de usuários, envio de mensagens e armazenamento de dados em tempo real. Com Firebase, os desenvolvedores podem facilmente implementar sistemas de login que suportam e-mail, senha, telefone e contas de redes sociais como Google e Facebook. Além disso, Firebase permite o envio e recebimento de mensagens em dispositivos móveis e web de forma gratuita, e oferece um banco de dados NoSQL que sincroniza dados em tempo real entre usuários e funciona mesmo offline. A plataforma também inclui ferramentas para monitorar e melhorar o desempenho dos aplicativos, bem como uma infraestrutura de teste na nuvem para testar apps em diversos dispositivos e configurações. Firebase é ideal para desenvolvedores que procuram uma solução completa e integrada para desenvolver, testar, monitorar e aprimorar aplicativos de maneira eficiente (CHOUGALE et al, 2022).

4.3.14 Android SDK

O Android SDK (Software Development Kit) é um conjunto de ferramentas fornecido pela Google para o desenvolvimento de aplicativos para o sistema operacional Android. Ele inclui um compilador, depurador, emuladores de dispositivos e bibliotecas necessárias para a

construção de aplicativos Android. O SDK facilita a criação, teste e depuração de aplicativos, oferecendo também uma API (Application Programming Interface) que permite aos desenvolvedores acessar as funcionalidades e recursos do sistema Android.

4.3.15 Android Virtual Device

O Android Virtual Device (AVD) é uma configuração que define um dispositivo Android virtual usado no Android Emulator. Ele permite testar e depurar aplicativos em um ambiente simulado, replicando características de dispositivos físicos como tamanho de tela, versão do Android e recursos de hardware. AVD é integrado ao Android Studio, facilitando a execução e teste de apps sem necessidade de dispositivos físicos.

4.3.16 Visual Studio Code

Visual Studio Code é um editor de texto leve e open-source disponível para Windows, Mac e Linux. Desenvolvido pela Microsoft, ele é ideal para a codificação em várias linguagens de programação, com suporte embutido para JavaScript, TypeScript e Node.js. É altamente extensível através de plugins, muitos dos quais são desenvolvidos pela comunidade, permitindo adicionar suporte a outras linguagens e funcionalidades. Ao contrário do Visual Studio, que é um ambiente de desenvolvimento integrado (IDE) completo, o VS Code é mais leve e ocupa menos espaço no disco, sendo ideal para desenvolvedores que preferem uma ferramenta rápida e customizável para escrever e editar código (CHRIS, 2023).

O Visual Studio Code foi o editor escolhido para ser utilizado no desenvolvimento do código do projeto.

4.3.17 GitHub

O GitHub é uma plataforma online que permite o versionamento e desenvolvimento colaborativo de código. Utilizando o sistema de controle de versão Git, o GitHub hospeda milhões de projetos, abrangendo desde trabalhos escolares até sistemas de código aberto renomados, como o Linux. É um ambiente onde desenvolvedores podem compartilhar, colaborar e contribuir para projetos, facilitando a colaboração em equipe e promovendo a transparência e o compartilhamento de conhecimento na comunidade de desenvolvimento de software (CONTA, 2018). É amplamente utilizado pela comunidade de desenvolvedores para compartilhar e colaborar em projetos de código aberto, além de servir como repositório central para muitos projetos de software. O Github foi escolhido para hospedar o código-fonte do aplicativo deste projeto.

5. RESULTADOS OBTIDOS

Nesta seção são apresentados os resultados obtidos no desenvolvimento do aplicativo, ou seja, sua arquitetura, estrutura, módulos e telas.

5.1 Arquitetura do Aplicativo

O aplicativo desenvolvido adota uma arquitetura cliente-servidor, separando a interface de usuário da lógica de negócios em camadas distintas. O cliente, ou *front-end*, é desenvolvido com React Native. A parte do servidor, ou *back-end*, é construída com Node.js e Express.js para execução da lógica e também utiliza o Prisma como ORM para facilitar a interação com o banco de dados PostgreSQL, que armazena dados estruturados como informações de usuários e animais. O Firebase é integrado para funcionalidades adicionais, como autenticação e armazenamento de dados em tempo real, aprimorando a segurança e a funcionalidade do aplicativo.

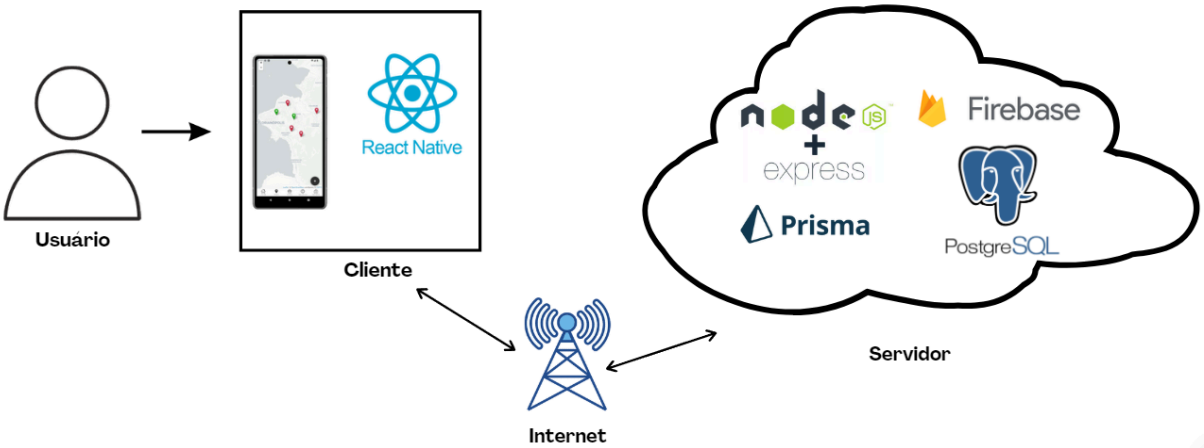

Figura 6. Arquitetura do Aplicativo

5.2 Estrutura do Aplicativo

A figura 7 é uma ilustração da estrutura de diretórios do projeto desenvolvido, juntamente com os arquivos contidos em cada diretório.

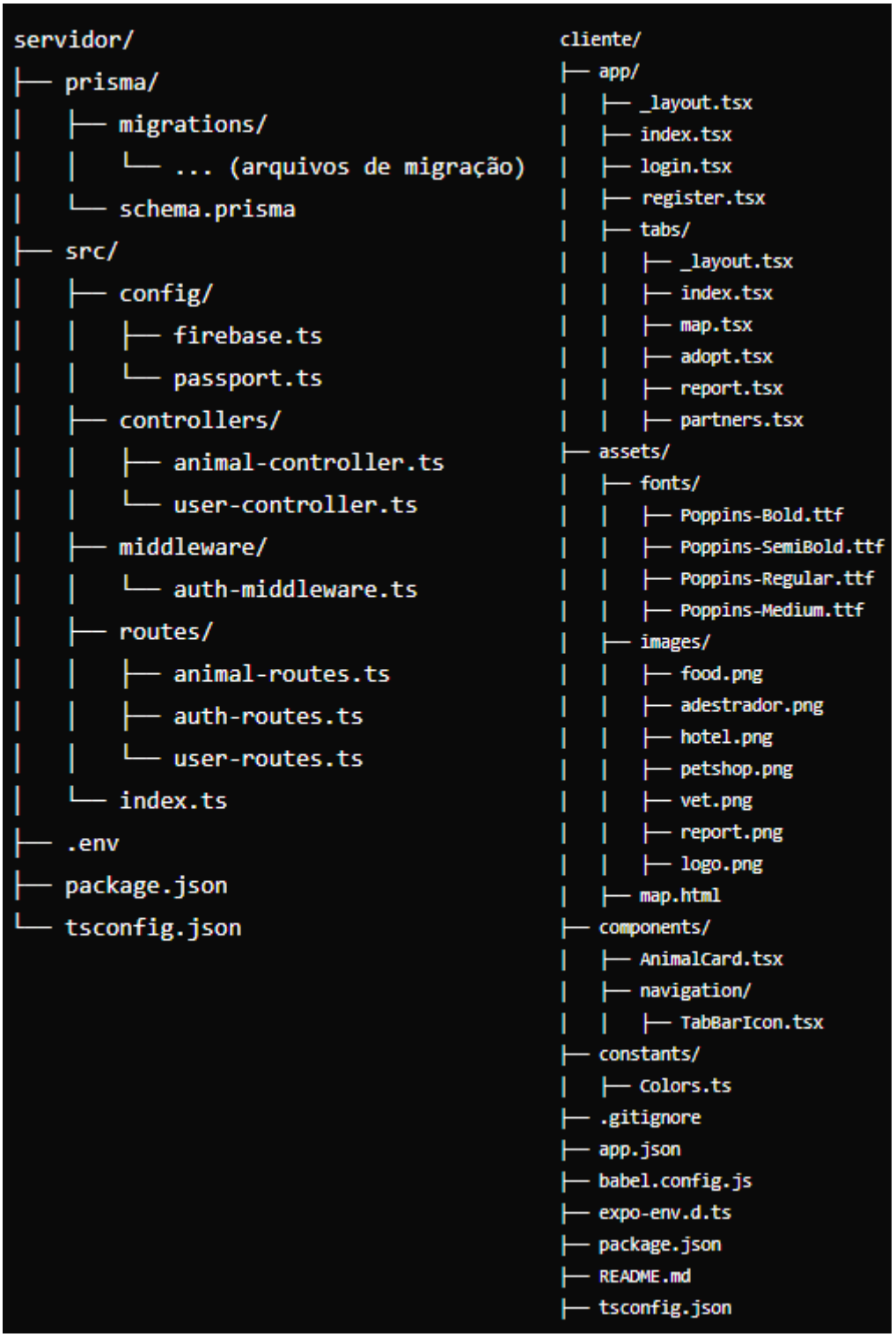

Figura 7. Estrutura do Aplicativo

A Tabela 5, apresentada abaixo, contém uma descrição detalhada dos itens presentes na estrutura do servidor, conforme ilustrado na Figura 7.

Item	Descrição
servidor/	Diretório raiz do *back-end*.
prisma/	Diretório que contém as configurações e migrações do Prisma.
migrations/	Subdiretório que contém os arquivos de migração gerados pelo Prisma.
schema.prisma	Arquivo de esquema do Prisma que define os modelos de dados e a configuração do banco de dados.
src/	Diretório que contém o código-fonte do servidor.
config/	Diretório para arquivos de configuração, como Firebase e Passport.
firebase.ts	Arquivo de configuração do Firebase.
passport.ts	Arquivo de configuração do Passport para autenticação.
controllers/	Diretório para controladores que contêm a lógica de negócios.
animal-controller.ts	Arquivo que contém a lógica de negócios para operações relacionadas aos animais.
user-controller.ts	Arquivo que contém a lógica de negócios para operações relacionadas aos usuários.
middleware/	Diretório para middlewares que interceptam e processam as requisições HTTP.
auth-middleware.ts	Middleware para verificação de autenticação.
routes/	Diretório para definir as rotas da aplicação.
animal-routes.ts	Arquivo que define as rotas relacionadas aos animais.
auth-routes.ts	Arquivo que define as rotas relacionadas à autenticação.
user-routes.ts	Arquivo que define as rotas relacionadas aos usuários.
index.ts	Arquivo principal que configura e inicia o servidor Express.
.env	Arquivo que contém variáveis de ambiente, como configurações do banco de dados e chaves de API.
package.json	Arquivo que lista as dependências e scripts do

	projeto.
tsconfig.json	Arquivo de configuração do TypeScript.

Tabela 5. Estrutura de arquivos e diretórios do servidor

A Tabela 6, apresentada abaixo, contém uma descrição detalhada dos itens presentes na estrutura do cliente, conforme ilustrado na Figura 7.

Item	Descrição
cliente/	Diretório raiz do projeto.
app/	Diretório contendo as telas principais do aplicativo.
_layout.tsx	Arquivo de layout geral para o aplicativo, define a estrutura comum para todas as telas.
index.tsx	Arquivo que direciona para a tela de login inicialmente.
login.tsx	Tela de login do usuário.
register.tsx	Tela de cadastro de novos usuários.
tabs/	Subdiretório contendo as telas das abas principais do aplicativo.
_layout.tsx (tabs)	Arquivo de layout para as abas, define a estrutura comum para todas as telas dentro das abas.
index.tsx (tabs)	Tela inicial das abas, também conhecida como a tela "home" do aplicativo.
map.tsx	Tela do mapa.
adopt.tsx	Tela de adoção de pets.
report.tsx	Tela de denúncias de maus tratos.
partners.tsx	Tela de parceiros do aplicativo.
assets/	Diretório contendo arquivos estáticos, como fontes e imagens.
fonts/	Subdiretório contendo arquivos de fontes utilizadas no aplicativo
Poppins-Bold.ttf	Arquivo de fonte.
Poppins-SemiBold.ttf	Arquivo de fonte.
Poppins-Regular.ttf	Arquivo de fonte.
Poppins-Medium.ttf	Arquivo de fonte.
images/	Subdiretório contendo arquivos de imagens utilizadas

Item	Descrição
	no aplicativo.
food.png	Imagem relacionada aos parceiros do aplicativo.
adestrador.png	Imagem relacionada aos parceiros do aplicativo.
hotel.png	Imagem relacionada aos parceiros do aplicativo.
petshop.png	Imagem relacionada aos parceiros do aplicativo.
vet.png	Imagem relacionada aos parceiros do aplicativo.
report.png	Imagem relacionada aos parceiros do aplicativo.
logo.png	Logotipo do aplicativo.
map.html	Arquivo HTML contendo o mapa utilizado na aplicação, incluindo funcionalidades de localização e marcadores.
components/	Diretório contendo componentes reutilizáveis do aplicativo.
AnimalCard.tsx	Componente de cartão de animal, exibindo informações básicas.
navigation/	Subdiretório contendo componentes relacionados à navegação do aplicativo.
TabBarIcon.tsx	Componente de ícone da barra de abas, utilizado para exibir ícones nas abas de navegação.
constants/	Diretório contendo constantes utilizadas no aplicativo.
Colors.ts	Arquivo de constantes de cores, definindo a paleta de cores utilizada no aplicativo.
.gitignore	Arquivo que especifica quais arquivos e diretórios devem ser ignorados pelo Git.
app.json	Arquivo de configuração do aplicativo, contendo informações sobre o projeto, como nome, versão e dependências.
babel.config.js	Arquivo de configuração do Babel, um transpiler JavaScript.
expo-env.d.ts	Arquivo de definições de tipos para o Expo, fornecendo suporte de TypeScript para APIs do Expo.
package.json	Arquivo de configuração do npm, listando as dependências e scripts do projeto.
README.md	Arquivo de documentação do projeto, contendo uma visão geral do mesmo e instruções de como rodar o aplicativo.

Item	Descrição
tsconfig.json	Arquivo de configuração do TypeScript, especificando opções de compilação e diretórios incluídos no projeto.

Tabela 6. Estrutura de arquivos e diretórios do cliente

5.3 Modelo de dados

Os modelos de dados ajudam a organizar as informações de forma lógica, facilitando o acesso, a manipulação e a integridade dos dados. No contexto deste aplicativo, os modelos de dados específicos para representar cada entidade são ‘usuários’ e ‘animais’, ambos com seus respectivos campos e relações. O arquivo *schema.prisma* é uma peça fundamental nesse contexto, pois serve como a definição central de todos os modelos utilizados pelo Prisma, para gerar um cliente que fornece uma API para interagir com o banco de dados de forma segura e eficiente. Esse cliente abstrai as complexidades das consultas SQL, permitindo que os desenvolvedores realizem operações CRUD (Create, Read, Update, Delete) e outras interações com o banco de dados através de métodos intuitivos e tipados.

5.3.1 Modelo de dados de usuários

O modelo de dados de usuários no *back-end* do aplicativo foi projetado para armazenar informações essenciais e garantir a funcionalidade necessária para o gerenciamento de contas de usuário. Este modelo inclui campos como *id*, que é uma chave primária auto incrementada para identificar unicamente cada usuário, e *email*, que é obrigatório e único, assegurando que cada conta seja associada a um endereço de email.

Para garantir a segurança das senhas, o campo *password* é armazenado em formato hash, utilizando a biblioteca *bcryptjs*. Além disso, o modelo suporta autenticação via Google OAuth 2.0 com o campo *googleId*, que armazena o identificador único do Google, caso o usuário opte por fazer login com uma conta Google. O campo *name* armazena o nome do usuário. Já o campo *phone* foi incluído para possibilitar a integração com o WhatsApp. Para monitorar a criação e atualização de registros, os campos *createdAt* e *updatedAt* registram automaticamente as datas de criação e modificação de cada registro, respectivamente.

Além disso, o modelo de usuário mantém um relacionamento com o modelo de animais,

representado pelo campo *animals*, que é uma lista de todos os animais associados ao usuário, facilitando o gerenciamento de animais no contexto do aplicativo.

5.3.2 Modelo de dados de animais

Este modelo é estruturado para capturar todos os detalhes relevantes sobre cada animal registrado no sistema. Os animais são identificados de forma única pelo campo *id*, que é uma chave primária, assim como no caso do modelo de dados de usuários. Outros campos são o *name* é utilizado para armazenar o nome do animal, *age* que registra a idade, *breed* registra a raça do pet, *gender* que indica o sexo, além de size que é usado para descrever o porte físico do animal.

A localização do animal, que é crucial, especialmente para animais perdidos ou disponíveis para adoção, é armazenada no campo *location*. O campo *type* identifica se o animal é um cão ou um gato, ajudando na categorização e busca. Assim como o outro modelo de dados, há também os campos *createdAt* e *updatedAt* são automaticamente preenchidos para registrar as datas de criação e atualização dos registros.

5.4 Rotas

As rotas são essenciais para direcionar e tratar requisições HTTP no *back-end*, definindo como diferentes URLs são mapeadas para funções específicas. Neste projeto, as rotas são organizadas em arquivos dentro do diretório *routes*, o que ajuda a manter o código modular.

As rotas de usuário são definidas no arquivo *user-routes.ts*. A rota */register* é responsável por criar novos usuários, validando os dados fornecidos e registrando-os no banco de dados. A rota */login* autentica os usuários, verificando suas credenciais (e-mail e senha) e retornando uma resposta adequada, que pode incluir um token de autenticação.

Já as rotas de animais estão no arquivo *animal-routes.ts*. A rota */register* lida com o registro de novos animais, validando as informações recebidas e criando os registros no banco de dados, de modo similar ao que ocorre nas rotas de usuários. A rota */pets* recupera a lista de animais cadastrados, fornecendo uma visão completa dos animais disponíveis no sistema.

Em *auth-routes.ts* são definidas as rotas de autenticação que tratam da autenticação via Google OAuth. A rota */google* inicia o processo de autenticação, redirecionando o usuário para o Google, enquanto que a */google/callback* lida com a resposta do Google e autentica o usuário no sistema, permitindo acesso às funcionalidades protegidas do aplicativo.

5.5 Servidor

Baseado no que foi exposto em seções anteriores, pode-se concluir que o lado do servidor, ou *back-end*, possui uma estrutura composta por modelos de dados, controladores, rotas, integração com Firebase e o uso do banco de dados PostgreSQL através do ORM Prisma, tudo isso sobre o framework Express.js. No arquivo *index.ts*, as principais operações incluem a configuração do Express para processar JSON, a configuração das sessões de usuário com *express-session*, a inicialização do *Passport* para autenticação, e a definição das rotas usando os arquivos de rotas importados.

5.6 Cliente

O lado cliente trata-se de um aplicativo que utiliza React Native com Expo. A navegação entre as telas é gerenciada através de uma estrutura de abas, permitindo uma interação intuitiva para o usuário. Diversos componentes do React Native são utilizados para construir as funcionalidades do aplicativo, garantindo um desenvolvimento modular e reutilizável. A comunicação com o servidor é feita utilizando a biblioteca Axios, que facilita tanto a obtenção de informações quanto o envio de dados necessários. O aplicativo possui telas que serão detalhadas na seção posterior.

5.7 Telas do Aplicativo

O aplicativo desenvolvido possui um total de sete telas. As telas de ‘Login’ e ‘Cadastro’ estão localizadas diretamente no diretório ‘app’, seguindo a estrutura padrão do Expo. As outras telas, que são navegadas através de um sistema de abas (tabs), estão localizadas no diretório *‘tabs’*. Esse modo de navegação permite que o usuário alterne facilmente entre diferentes seções do aplicativo. Os estilos de cada tela estão definidos no próprio arquivo de cada uma, permitindo uma organização clara e modular.

5.7.1 Login

Figura 8. Tela de Login

A tela de Login permite que os usuários autentiquem suas contas. A biblioteca *React Native Paper* foi utilizada para criar os componentes de entrada de dados (*TextInput*) e botões (*Buttons*). Há também a implementação do hook *useState* para gerenciar estados. O campo de email utiliza o *useState* para armazenar o email digitado pelo usuário, atualizando-o através do método setEmail, enquanto o campo de senha segue a mesma lógica, utilizando o hook para armazenar a senha e atualizá-la através do método *setPassword*.

O botão de login chama a função *handleNavigateToHome*, que utiliza o roteador (*router*) para navegar para a tela principal do aplicativo. Há também uma opção de login com Google, que permite autenticação via conta Google. Por fim, há um link para cadastro (*Cadastre-se*), implementado como um botão que, se clicado, ativa a função *handleNavigateToRegister*, navegando para a tela de cadastro caso o usuário ainda não possua uma conta.

5.7.2 Cadastro

Figura 9. Tela de Cadastro

Nesta tela novos usuários podem criar uma conta no aplicativo para utilizá-lo. Assim como na tela de Login, a biblioteca *React Native Paper* foi utilizada para criar os componentes de entrada de dados e botões. O hook *useState* é empregado para gerenciar o estado dos campos de nome de usuário, email e senha. Quando o botão de cadastro é pressionado, a função *handleRegistration* é chamada. Esta função coleta os dados inseridos (nome de usuário, email e senha) e faz uma requisição *POST* para o endpoint *"/register"* do servidor, utilizando a biblioteca axios. Se o registro for bem-sucedido, uma mensagem de sucesso é exibida e o usuário é redirecionado para a tela de login através do roteador (*router*). Caso ocorra algum problema durante o processo de registro, uma mensagem de erro é exibida para o usuário.

5.7.3 Home

Figura 10. Tela Home

A tela Home serve como o ponto de partida para o usuário, após efetuar o login. É a primeira aba e também a aba padrão. O usuário é saudado com uma mensagem dando boas vindas e uma breve explicação sobre algumas sessões pelas quais é possível navegar.

Figura 11. Modal de cadastro para adoção

A funcionalidade central da tela Home é o modal "Cadastrar para Adoção" que permite que o usuário cadastre animais para adoção, preenchendo diversos campos com informações relevantes sobre o pet. Os campos disponíveis no modal incluem nome do animal, idade, raça, bairro onde o animal se encontra, motivo do animal está sendo posto para adoção, foto, tipo (se é cão ou gato), porte e gênero. Após preenchimento, os dados são validados e enviados ao servidor através de uma requisição *POST* para a rota *'/register'* que, através do controlador de animais, reconhece que a função a ser executada no servidor é a *registerAnimal*. Caso o fluxo ocorra sem nenhum erro, é criado o perfil de um animal para adoção, que será listado na tela 'Adotar'.

5.7.4 Mapa

Esta é uma das telas mais cruciais do aplicativo. O mapa exibido possibilita ao usuário visualizar a localização de animais avistados e desaparecidos, bem como criar um aviso de avistamento de um animal perdido que eventualmente cruzou seu caminho, além do desaparecimento e última localização conhecida de seu próprio pet. Esta funcionalidade é implementada utilizando dois arquivos principais: *map.html* e *map.tsx*.

O arquivo *map.html* está localizado no diretório '*assets*' e utiliza a biblioteca Leaflet para renderizar e gerenciar o mapa. A camada de ‘tiles’ (*tileset*) utilizada é do CartoDB Positron, que fornece uma visualização leve e clara do mapa, evitando excesso de informações. O mapa captura eventos de clique para obter coordenadas e enviá-las de volta ao WebView no React Native usando *window.ReactNativeWebView.postMessage*. Funções adicionais incluem *addMarker*, que adiciona marcadores no mapa com uma mensagem pop-up, e *recenterMap*, que centraliza o mapa na localização atual do usuário. Também há a função *openWhatsApp* para abrir o WhatsApp com uma mensagem predefinida.

Figura 12. Tela do Mapa

Já o arquivo *map.tsx*, localizado no diretório '*tabs*', serve como a interface React Native que incorpora o mapa através do componente *WebView*. Ele administra vários estados usando o hook *useState*, incluindo visibilidade de modais e dados de localização. Dois modais são usados: um para selecionar o tipo de relatório (avistamento ou desaparecimento) e outro para upload de imagens. As funções *showModal* e *handleReport* gerenciam os modais, enquanto *pickImage* permite selecionar uma imagem da galeria e *addMarker* envia dados ao *WebView* para adicionar marcadores no mapa. A comunicação entre *map.html* e *map.tsx* é feita através

de *postMessage*. Eventos de clique no mapa enviam coordenadas ao React Native, que abre os modais apropriados. Relatórios enviados pelo usuário são processados e adicionam novos marcadores no mapa, integrando a interface de usuário do React Native com a visualização dinâmica do Leaflet.

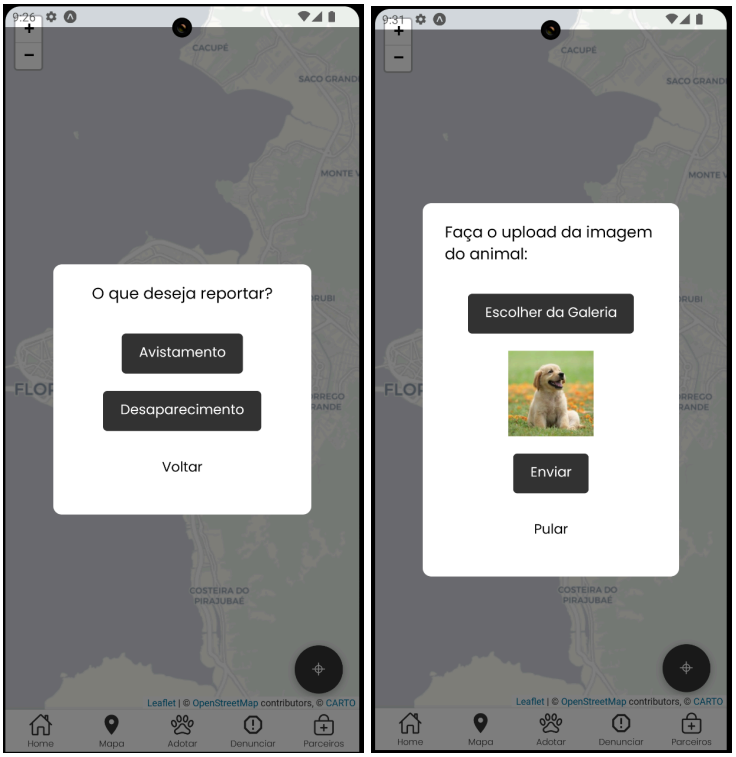

Figura 13. Reportando evento no mapa

Ao clicar em algum local no mapa, surge um modal exibindo as opções para reportar o avistamento ou o desaparecimento de um animal . Selecionando uma delas é solicitado o carregamento de imagem do animal para constar nas informações do *'marker'* que aparecerá no mapa.

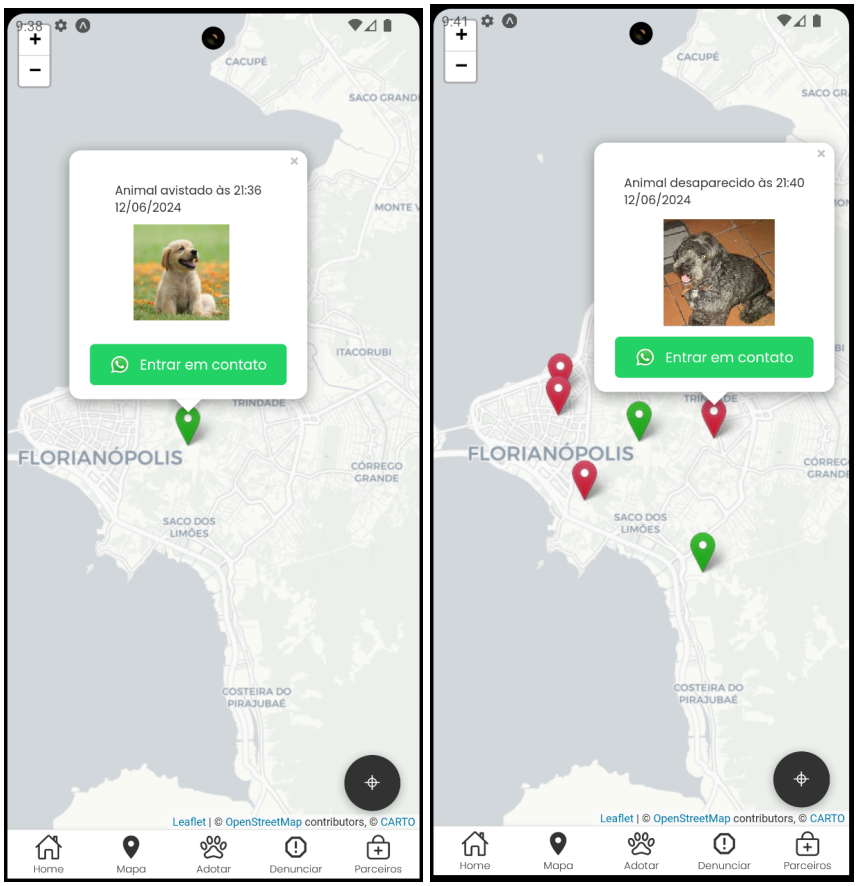

Figura 14. Observando eventos no mapa

Cada relato é representado no mapa com um marcador (*'marker'*): verde para avistamentos e vermelho para avisos de animais desaparecidos. Ao clicar em um marcador, um pop-up é exibido, contendo o horário e a data em que o aviso foi criado, uma foto do animal (se houver) e um botão com o texto 'Entrar em contato'. Este botão, quando clicado, abre o aplicativo WhatsApp do usuário, permitindo que estabeleça-se contato com o criador do aviso. Esta funcionalidade proporciona uma maneira eficiente e interativa para os usuários reportarem e encontrarem animais, promovendo a colaboração dentro da comunidade.

5.7.5 Adotar

Na tela Adotar (*adopt.tsx*) o usuário pode encontrar animais disponíveis para adoção e aplicar filtros para refinar suas buscas. A tela é dividida em duas principais seções: filtros de busca e a lista de animais disponíveis para adoção.

A parte superior da tela contém opções de refinamento de busca que ajudam o usuário a encontrar animais que atendam às suas preferências específicas. Essas opções são representadas por botões com ícones para selecionar entre cães e gatos, o tamanho do animal (pequeno, médio ou grande) e o gênero do animal (macho ou fêmea). Os estados desses filtros são gerenciados usando o hook *useState*. Quando um filtro é selecionado, as funções *handleAnimalTypePress*, *handleSizePress*, e *handleGenderPress* atualizam os estados correspondentes.

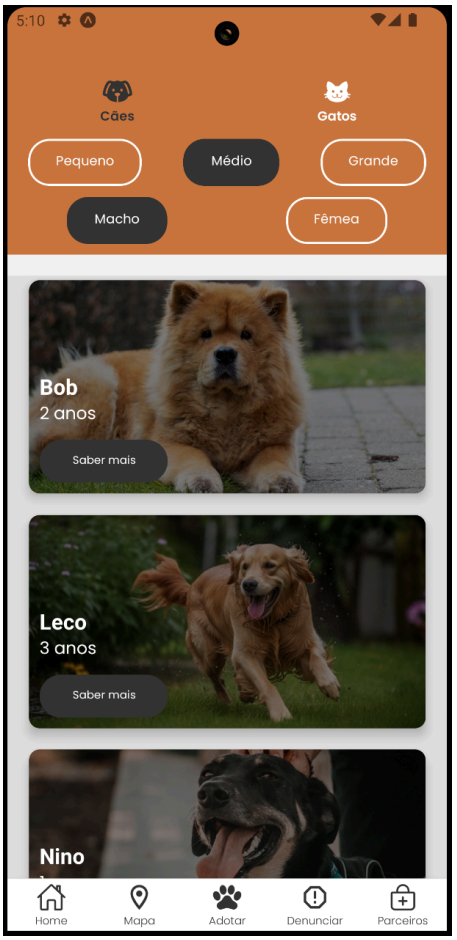

Figura 15. Tela Adotar

Abaixo dos filtros, a lista de animais disponíveis para adoção é exibida, utilizando um componente *ScrollView*. Cada animal é representado por um componente *AnimalCard*

(*AnimalCard.tsx*), que exibe informações como nome, idade e foto.

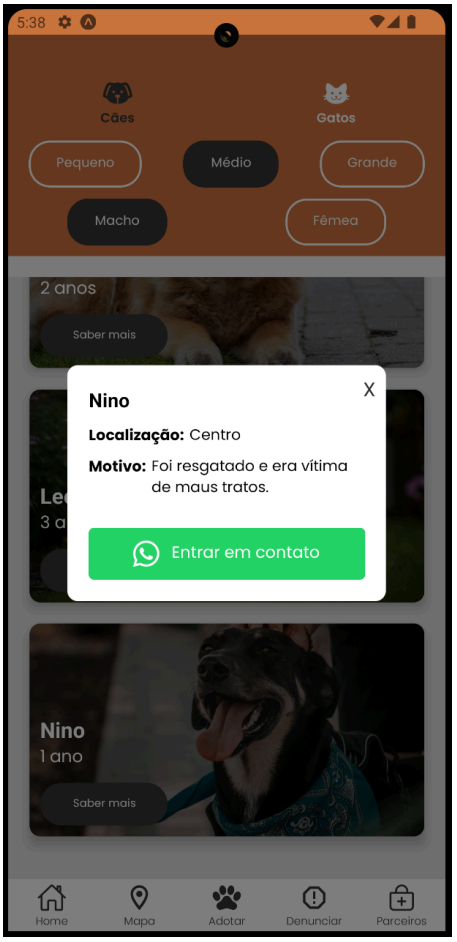

Figura 16. Modal de contato

Ao clicar no botão ‘Saiba mais’ é exibido um modal com a localização do pet e o motivo da disponibilidade para adoção. Através do botão ‘Entre em contato’ , o usuário é redirecionado para uma conversa no WhatsApp com o anunciante do animal, permitindo uma comunicação direta e imediata. Este comportamento é similar ao que ocorre na tela do mapa, onde o botão no pop-up dos marcadores também abre o WhatsApp para facilitar o contato.

5.7.6 Denunciar

Figura 17. Tela Denunciar

A tela Denunciar oferece a possibilidade do usuário relatar casos de maus tratos a animais. O código desta tela está presente no arquivo report.tsx e direciona as denúncias para o órgão responsável pela proteção animal em Florianópolis, o Dibea. Ao clicar no botão "Denunciar", a função *handlePress* é chamada, a qual utiliza o módulo Linking do React Native para abrir o URL da página de denúncias do site da Dibea no navegador do smartphone do usuário. Esta página contém instruções detalhadas sobre como realizar uma denúncia.

5.7.7 Parceiros

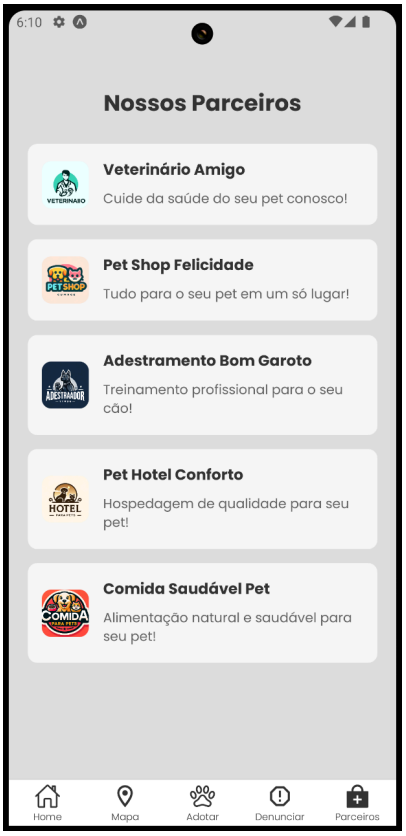

Figura 18. Tela de Parceiros

A tela Parceiros apresenta os parceiros e apoiadores do aplicativo, oferecendo aos usuários informações sobre serviços e produtos relacionados à pets. O código desta tela está localizado no arquivo *partners.tsx*. Ao destacar e promover esses parceiros, encontra-se uma maneira eficaz de monetizar a plataforma através de anúncios pagos.

6. CONCLUSÃO E TRABALHOS FUTUROS

O resultado deste projeto foi o desenvolvimento do aplicativo "Pet Savior", cujo propósito é contribuir para a proteção e o bem-estar de animais vulneráveis. Inicialmente, o escopo de funções do aplicativo, como a funcionalidade de denúncia, foi limitado a Florianópolis. Esta abordagem foi adotada para iniciar em um contexto mais restrito e, posteriormente, expandir para outras regiões. No entanto, funcionalidades cruciais, como o mapa e a adoção, podem ser utilizadas por usuários de outros estados do país.

Conforme observado nesta monografia, atualmente não existem opções populares que sirvam a esse propósito, o que serviu de incentivo para o tema ser escolhido. É importante mencionar que o projeto utilizou conceitos e conhecimentos adquiridos durante o curso de Sistemas de Informação, incluindo programação orientada a objetos, desenvolvimento de dispositivos móveis e embarcados, segurança da informação, gerência de projetos, informática e sociedade, engenharia de usabilidade, entre outros. Quanto aos objetivos estipulados no início do projeto, de fato foi construído um aplicativo com interface intuitiva e funcionalidades úteis para os propósitos pré-estabelecidos. No entanto, alguns objetivos, como testes e avaliações feitas por usuários, ainda estão pendentes e representam uma etapa importante para futuros desenvolvimentos e melhorias do aplicativo.

Para trabalhos futuros, seria interessante considerar o uso de computação em nuvem (cloud computing), utilizando serviços como AWS ou Azure. Esta abordagem faria mais sentido com um aumento na demanda e no número de usuários do aplicativo, proporcionando escalabilidade e maior capacidade de processamento. Novas funcionalidades poderiam ser desenvolvidas para enriquecer a experiência do usuário. Uma ideia é a integração com redes sociais, permitindo que os usuários compartilhem perfis de animais perdidos ou disponíveis para adoção diretamente em suas redes, aumentando assim a visibilidade e as chances de reencontro ou adoção. Outra funcionalidade interessante seria a implementação de notificações baseadas em geolocalização que alertem os usuários quando um animal é reportado como perdido ou encontrado em uma área próxima. Adicionalmente, um sistema de mensagens dentro do aplicativo facilitaria a comunicação direta entre os usuários e os responsáveis pelos anúncios de animais. Além das novas funcionalidades, a realização de testes com usuários para obter *feedback* é crucial. Esses testes ajudariam a identificar possíveis melhorias na usabilidade e no desempenho do aplicativo, garantindo que ele atenda às necessidades e expectativas dos usuários de maneira eficaz.

7. REFERÊNCIAS

Greger Larson, Elinor K. Karlsson, Angela Perri, Matthew T. Webster, Simon Y. W. Ho, Joris Peters, Peter W. Stahl, Philip J. Piper, Frode Lingaas, Merete Fredholm, Kenine E. Comstock, Jaime F. Modiano, Claude Schelling, Alexander I. Agoulnik, Peter A. Leegwater, Keith Dobney, Jean-Denis Vigne, Carles Vilà, Leif Andersson, and Kerstin Lindblad-Toh ***"Rethinking dog domestication by integrating genetics, archeology, and biogeography"*** , 2012.
Disponível em: https://doi.org/10.1073/pnas.1203005109
Acesso em 2. Dez. 2022

GUEST et al., ***"Feasibility of integrating canine olfaction with chemical and microbial profiling of urine to detect lethal prostate cancer,"*** PLOS ONE, 2021.
Disponível em: https://doi:10.1371/journal.pone.0245530
Acesso em 2. Dez. 2022

YONG, Hooi Yong . RUFFMAN, Ted. ,***"Emotional contagion: Dogs and humans show a similar physiological response to human infant crying"***, 2014.
Disponível em: https://doi.org/10.1016/j.beproc.2014.10.006
Acesso em 3. Dez. 2022

MCDONOUGH, Ian M. ERVWIN, Hillary B. SIN, Nancy L. ALLEN, Rebecca S. ***"Pet ownership is associated with greater cognitive and brain health in a cross-sectional sample across the adult lifespan"*** , 2022.
Acesso em 3. Dez. 2022
INSTITUTO PET BRASIL.
"Dados IPB: em 2020, mercado pet faturou R$ 40,8 bilhões", 2021.
Disponível em: http://institutopetbrasil.com/fique-por-dentro/mercado-pet-faturou/
Acesso em 3. Dez. 2022

INSTITUTO PET BRASIL.
"Mercado pet brasileiro: como o amor pelos animais impulsiona os negócios", 2022.

Disponível em:
https://institutopetbrasil.com/fique-por-dentro/amor-pelos-animais-impulsiona-os-negocios/
Acesso em 3. Dez. 2022

INSTITUTO PET BRASIL.
"Número de animais de estimação em situação de vulnerabilidade mais do que dobra em dois anos, aponta pesquisa do IPB", 2022.
Disponível em:
http://institutopetbrasil.com/fique-por-dentro/numero-de-animais-de-estimacao-em-situacao-de-vulnerabilidade-mais-do-que-dobra-em-dois-anos-aponta-pesquisa-do-ipb/
Acesso em 3. Dez. 2022

BALTHAZAR, Artur Donadel. PIRES, Thaina Alves. PAZMINO, Ana Veronica.
"Superpopulação, abandono e maus-tratos de cães: um estudo de design social", 2019.
Disponível em:
https://repositorio.ufsc.br/bitstream/handle/123456789/244904/VOLUME-5-301-312.pdf?sequence=1
Acesso em 15. Abril. 2023

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. ***"UML: guia do usuário"***, 2005.

FLATSCHART, Fábio "***HTML5 : embarque imediato"*** , 2011
Disponível em: https://editorabrasport.com.br/html5-ebook
Acesso em 10. Junho. 2023

SILVA, Maurício Samy "***Construindo sites com CSS e (X)HTML"***, 2007.
Disponível em: https://novatec.com.br/livros/csshtml/
Acesso em 10. Junho. 2023

MUNZLINGER, E. "***Introdução à Tecnologia Web"***. 2011.
Disponível em: .
http://www.elizabete.com.br/site/Outros/Entradas/2011/2/20_Tecnologia_Web_files/18-CSS-Sintaxe.pdf
Acesso em: 10.Junho.2023.

FLANAGAN, David. “***JavaScript : o guia definitivo. 6”*** , 2014.
Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788565837484/pageid/17
Acesso em: 11. Junho. 2023

CHERNY, Boris. ***“Programming TypeScript: Making Your JavaScript Applications Scale”***, 2019.
Disponível em: https://www.amazon.com/Programming-TypeScript-Making-JavaScript-Applications/dp/1492037656
Acesso em: 11. Junho. 2023

DANIELSSON, William**.** ***“React Native application development – A comparison between native Android and React Native”***, 2016.
Disponível em: http://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02
Acesso em: 12. Junho. 2023

NOLETO, Cairo***.“JSON: o que é e como funciona? O guia do básico ao avançado!”***, 2022.
Disponível em: https://blog.betrybe.com/json/
Acesso em 12. Junho. 2023

TILKOV, Stefan; VINOSKI, Steve. Node. js: “***Using JavaScript to build high-performance network programs”***, 2010.
Disponível em: https://ieeexplore.ieee.org/document/5617064
Acesso em 12. Junho. 2023

CONTA, Gabriel Sebastian von. “***Identificação e visualização de rastros entre artefatos no GitHub”.*** 2018.
Disponível em: https://repositorio.ufrn.br/handle/123456789/34178
Acesso em: 13. Junho.2023

SOMMERVILLE, Ian. ***“Engenharia de software”***, 2018.
Disponível em: https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/

Livro/engenhariaSoftwareSommerville.pdf
Acesso em: 15. Junho. 2023

REINEHR, Sheila. ***"Engenharia de requisitos"***, 2020.
Disponível em:
https://loja.grupoa.com.br/eb-ead-engenharia-requisitos9786556900674-p1007418
Acesso em: 15. Junho. 2023

FOWLER, M.; LEWIS, J. ***"Patterns of Enterprise Application Architecture"***, 2014.

MENDONÇA, Vinícius Lobo De; BITTAR, Thiago Jabur; DE, Márcio; et al. "***Um estudo dos Sistemas Operacionais Android e iOS para o desenvolvimento de aplicativos"***, 2011.
Disponível em:
https://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011_submission_54.pdf
Acesso em: 20. Junho. 2023

MARQUES, J. ***"O que significa iOS? Conheça cinco fatos sobre o sistema do iPhone"***.
Disponível em:
https://www.techtudo.com.br/listas/2019/07/o-que-significa-ios-conheca-cinco-fatos-sobre-o-sistema-do-iphone.ghtml
Acesso em: 20. Junho. 2023.

EL-KASSAS, W. S. et al. **"*Taxonomy of Cross-Platform Mobile Applications Development Approaches"***, 2015.
Disponível em: https://www.sciencedirect.com/science/article/pii/S2090447915001276
Acesso em: 24. Junho.2023

MORIMOTO, Carlos. ***"SmartPhones, Guia Prático"***, 2009.
Disponível em: http://www.hardware.com.br/livros/smartphones.
Acesso em: 24.Junho.2023

Android Developers, 2024.
Disponível em: https://developer.android.com/

Acesso em 18. Abril. 2024

HAYANI, Said . ***"How to Create a Camera App with Expo and React Native"*** , 2020.
Disponível em:
https://www.freecodecamp.org/news/how-to-create-a-camera-app-with-expo-and-react-native/
Acesso em 20. Abril. 2024

NGUGI, Samuel. ***"Leaflet in Practice: Create webmaps using the JavaScript Leaflet library"***, 2023. University of Nairobi.
Disponível em:
https://www.researchgate.net/publication/372720325_Leaflet_in_Practice_Create_webmaps_using_the_JavaScript_Leaflet_library
Acesso em: 25. Maio. 2024.

MAZIERO, Carlos Alberto. **"Sistemas operacionais: conceitos e mecanismos"** [recurso eletrônico] / Carlos Alberto Maziero. – Curitiba : DINF - UFPR, 2019.
Disponível em: https://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-livro.pdf
Acesso em: 3. Junho. 2024.

LUCIANO, Josué; ALVES, Wallison Joel Barberá. ***"Padrão de arquitetura MVC: Model-view-controller"***. EPeQ Fafibe, v. 1, n. 3a, p. 102-107, 2017.
Disponível em:
https://unifafibe.com.br/revistasonline/arquivos/revistaepeqfafibe/sumario/20/16112011142249.pdf
Acesso em 3.Junho. 2024.

EIGY, Cem. ***"React Functional Components, Props, and JSX – React.js Tutorial for Beginners"***, 2020.
Disponível em:
https://www.freecodecamp.org/news/react-components-jsx-props-for-beginners/
Acesso em: 08. Junho. 2024

THOMAS, Gaël. **"How to Build a Node.js Database using Prisma and SQLite"**, 2021.

Disponível em:
https://www.freecodecamp.org/news/build-nodejs-database-using-prisma-orm/
Acesso em: 08.Junho. 2024

OYAMA, Faith. ***"PostgreSQL and JSON – How to Use JSON Data in PostgreSQL"***, 2023.
Disponível em:
https://www.freecodecamp.org/news/postgresql-and-json-use-json-data-in-postgresql/
Acesso em: 08. Junho. 2024

CHRIS, Kolade. ***"Visual Studio vs Visual Studio Code – What's The Difference Between These IDE Code Editors?"***, 2023.
Disponível em: https://www.freecodecamp.org/news/visual-studio-vs-visual-studio-code/
Acesso em: 08. Junho. 2024

Chougale, Pankaj & Yadav, Vaibhav & Gaikwad, Anil & Student, Bharati & Vidyapeeth,. (2022). FIREBASE -OVERVIEW AND USAGE. Journal of Engineering and Technology Management. 2582-5208.
Disponível em:
https://www.researchgate.net/publication/362539877_FIREBASE_-OVERVIEW_AND_USAGE
Acesso em: 09. Junho. 2024

YAKUBU, Victor. ***"How to Create a CRUD API – NodeJS and Express Project for Beginners"***, 2024.
Disponível em: https://www.freecodecamp.org/news/create-crud-api-project/#what-is-express
Acesso em: 09. Junho. 2024

APÊNDICE A: Código fonte

O código-fonte do aplicativo desenvolvido neste Trabalho de Conclusão de Curso está disponível na plataforma GitHub. Para acessar o repositório completo, incluindo toda a implementação e documentação do projeto, utilize o seguinte link:

https://github.com/goslingla/pet-savior

No repositório, pode ser encontrado:

- **Código-Fonte:** Todos os arquivos e diretórios do projeto.
- **Instruções de Configuração:** Passos para configurar o ambiente e executar o aplicativo.
- **Documentação Adicional:** Informações detalhadas sobre a arquitetura, tecnologias utilizadas e funcionalidades do aplicativo.

Esta disponibilização visa facilitar o acesso ao código para fins de revisão, replicação e possível contribuição futura.

APÊNDICE B: Artigo SBC

Pet Savior: um aplicativo para dispositivos móveis que visa ajudar cães e gatos perdidos, abandonados ou vítimas de maus tratos.

Luiz André Ventura Gosling

Universidade Federal de Santa Catarina (UFSC)
Campus Universitário Reitor João David Ferreira Lima, R. Delfino Conti, s/n - Trindade, Florianópolis - SC, 88040-900

INE - Departamento de Informática e Estatística

a.gosling@grad.ufsc.br

Abstract. *This project aimed to develop a mobile application for posting lost or adoptable animals, as well as other charitable actions. The platform simplifies finding abandoned animals for users willing to help, providing a functional and straightforward solution. Using a client-server architecture and the MVC pattern, the front-end was created with React Native, offering an intuitive interface. The back-end, implemented in Node.js, integrates data and manages crucial information such as location on the map and animal profiles. The application includes features like displaying animals on the map, creating profiles, reporting animal abuse, and a section for partners offering products and services for pets.*

Resumo. *Este projeto visou desenvolver um aplicativo móvel para anúncios de animais perdidos ou para adoção, além de outras ações beneficentes. A plataforma facilita a localização de animais desamparados por usuários dispostos a ajudar, de maneira funcional e simplificada. Utilizando a arquitetura cliente-servidor e o padrão MVC, o front-end foi criado com React Native, proporcionando uma interface intuitiva. O back-end, em Node.js, integra dados e gerencia informações cruciais como localização no mapa e perfis de animais. O aplicativo inclui funcionalidades como exibição de animais no mapa, criação de perfis, denúncias de maus-tratos e uma sessão de parceiros que ofertam produtos e serviços para pets.*

1. Introdução

O convívio com animais de estimação traz benefícios significativos para a saúde emocional e física, além de melhorar a cognição. No entanto, o Brasil enfrenta um problema crescente de abandono e vulnerabilidade animal. Entre 2018 e 2020, o número de pets em situação de vulnerabilidade aumentou de 3,9 para 8,8 milhões, conforme levantamento do Instituto Pet Brasil (IPB). Esses animais vivem sob tutela de famílias abaixo da linha de pobreza ou nas ruas, recebendo algum tipo de ajuda. A pandemia de COVID-19 agravou essa situação, resultando em mais animais abandonados e menos possibilidades de castração, perpetuando o ciclo de vulnerabilidade.

Apesar da grande presença de animais de estimação e o potencial do mercado pet brasileiro, que faturou R$ 40,8 bilhões em 2020, há uma carência de plataformas digitais eficientes para buscar animais perdidos ou reportar desaparecimentos e promover adoção. Diante deste cenário, este artigo propôs o desenvolvimento de um aplicativo funcional e bem estruturado que facilite a melhoria da qualidade de vida de animais negligenciados. O aplicativo atuará como uma ferramenta para promover adoção, financiamento e reencontro de animais desaparecidos com suas famílias, aproveitando a tecnologia moderna e a popularização de smartphones e internet para o sucesso em tal empreitada.

2. Objetivos

O objetivo geral é desenvolver um aplicativo para dispositivos móveis, com foco em smartphones, que colete informações sobre animais em situação de vulnerabilidade nas ruas, que foram abandonados ou que estejam desaparecidos. O intuito é promover adoção, financiamento e eventualmente facilitar o reencontro entre animais desaparecidos e suas famílias.

Os objetivos específicos incluem: elaborar uma fundamentação teórica que evidencie tanto a relevância de animais de estimação para membros da sociedade quanto a necessidade do uso da tecnologia em projetos como este, visando reverter ao menos parte dos problemas expostos através da filantropia, empatia e ações beneficentes e voluntárias dos usuários; desenvolver um protótipo de aplicativo levando em conta conceitos de arquitetura de software e usabilidade; implementar o desenvolvimento do

aplicativo utilizando tecnologias adequadas e boas práticas de programação; e, finalmente, avaliar o aplicativo desenvolvido por meio de testes e casos de uso para garantir sua eficácia e atender às necessidades dos usuários de maneira satisfatória.

3. Tecnologias Utilizadas

Para o desenvolvimento do aplicativo "Pet Savior", foram utilizadas diversas tecnologias, com destaque para React Native no front-end. A escolha do React Native deve-se à sua capacidade de permitir o desenvolvimento de aplicativos móveis de alta performance para múltiplas plataformas, como Android e iOS, a partir de uma única base de código. Além do fato do autor possuir familiaridade com esta tecnologia, o que facilitou o processo de criação.

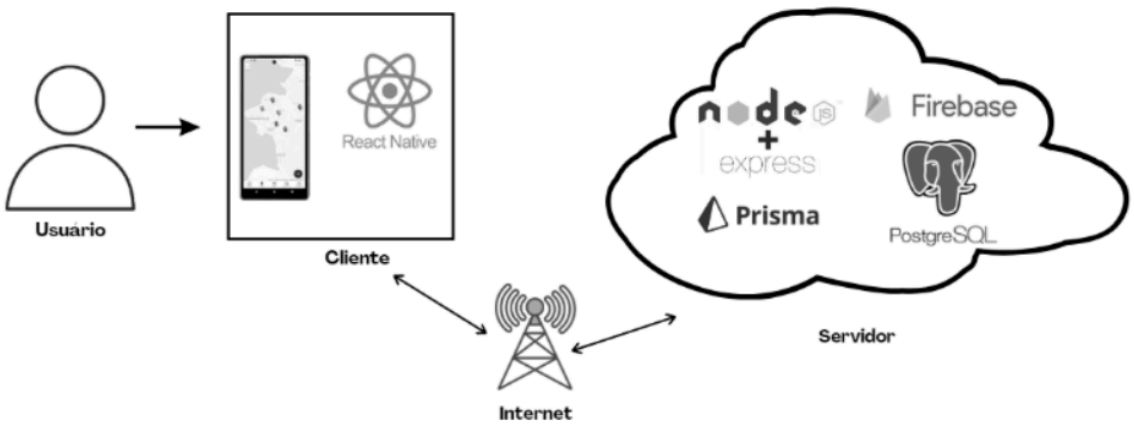

Figura 1. Arquitetura do aplicativo

No back-end, foram utilizadas tecnologias como Node.js e Express.js pela eficiência em manipular conexões simultâneas e simplificar a criação de APIs. Para o gerenciamento do banco de dados, optou-se pelo PostgreSQL com Prisma, garantindo integridade e eficiência. O Firebase foi integrado para autenticação e armazenamento de dados em tempo real, devido à sua infraestrutura.Essas ferramentas formam a estrutura principal do aplicativo.

4. Estrutura do Aplicativo

Nesta seção, discutiremos a estrutura do aplicativo, composta pelo Cliente, que gerencia a interface do usuário, e pelo Servidor, responsável pelo processamento e armazenamento dos dados.

4.1 Cliente

A estrutura do Cliente é organizada por arquivos e diretórios específicos para cada funcionalidade. As principais telas incluem *index.tsx* (inicial), *login.tsx* (login), *register.tsx* (registro), *adopt.tsx* (adoção de animais), *report.tsx* (denúncias), map.tsx (mapa interativo) e partners.tsx (parceiros). Os componentes incluem *AnimalCard.tsx* para exibição de animais e *TabBarIcon.tsx* para ícones de navegação. Recursos estão em assets, contendo fontes, imagens e o arquivo HTML do mapa.

A comunicação com o servidor é realizada com o *axios*. O registro de usuários envia uma requisição *POST* para */register*. A tela de adoção obtém animais com GET em */animals* e aplica filtros com *POST* em */animals/filter*. A tela de denúncias envia relatórios de maus-tratos para armazenamento. O mapa interativo usa Leaflet.js para mostrar localizações, obtendo e enviando dados via rotas específicas.

4.2 Servidor

O lado do Servidor utiliza Node.js e Express.js para criar uma API RESTful, gerenciando rotas para usuários, animais, autenticação e *reports*. O PostgreSQL, junto com Prisma, armazena dados estruturados, enquanto o Firebase Storage guarda imagens dos *reports*. A autenticação é feita com Passport.js, suportando login via Google OAuth e sessões com express-session. O cadastro e login de usuários envolvem validação, hash de senhas e armazenamento no PostgreSQL. Relatórios são criados com upload de imagens para o Firebase Storage e armazenamento dos dados no PostgreSQL. A recuperação de relatórios é feita consultando o PostgreSQL, incluindo URLs das imagens.

5. Telas

Nesta seção, discorreremos acerca das telas do aplicativo "Pet Savior", descrevendo suas funcionalidades principais e como contribuem para a experiência do usuário. Cada tela oferece recursos específicos que ajudam a alcançar os objetivos do aplicativo.

5.1 Login

A tela de Login permite que os usuários autentiquem suas contas. Utiliza *hooks* para gerenciar o estado dos campos de *email* e *senha*. Há opções de login via conta Google e um link para cadastro, que direciona para a tela de cadastro caso o usuário ainda não possua uma conta.

5.2 Cadastro

Nesta tela, novos usuários podem criar uma conta no aplicativo. Semelhante à tela de Login, utiliza *hooks* para gerenciar o estado dos campos. Quando o botão de cadastro é pressionado, uma requisição *POST* é enviada ao servidor para registrar o novo usuário. Em caso de sucesso, o usuário é redirecionado para a tela de login.

5.3 Home

A tela Home é o ponto de partida após o login, saudando o usuário com uma mensagem de boas-vindas. A funcionalidade central é um modal "Cadastrar para Adoção", onde o usuário pode cadastrar animais para adoção, preenchendo campos como *nome*, *idade*, *raça*, *localização* e outros detalhes relevantes. Os dados são enviados ao servidor para criar o perfil do animal para adoção.

5.4 Mapa

Esta tela permite ao usuário visualizar a localização de animais avistados e desaparecidos, e criar avisos de avistamento ou desaparecimento. Utiliza a biblioteca de código aberto Leaflet para renderizar o mapa e React Native para gerenciar a interface e a comunicação entre o mapa e o usuário. “*Reports*” são representados com marcadores no mapa, que podem ser clicados para mais detalhes e contato via *WhatsApp* com o criador do *report*.

5.5 Adotar

A tela Adotar permite ao usuário encontrar animais disponíveis para adoção, aplicando filtros para refinamento da busca. Utiliza componentes de botão para filtros e um *ScrollView* que exibe uma lista de formada por instâncias do componente *AnimalCard*, onde encontram-se informações e fotos de cada *pet*. Ao clicar em "Saiba mais", um modal exibe detalhes do animal e oferece a opção de contato via *WhatsApp*.

5.6 Denunciar

A tela Denunciar tem a opção de redirecionar para a página de denúncias do Dibea no navegador do usuário, onde podem ser encontradas instruções detalhadas sobre como realizar uma denúncia.

5.7 Parceiros

A tela Parceiros apresenta os parceiros e apoiadores do aplicativo, oferecendo informações sobre serviços e produtos relacionados à *pets*. Facilita a monetização da plataforma através de anúncios pagos.

6. Referências

Greger Larson, Elinor K. Karlsson, Angela Perri, Matthew T. Webster, Simon Y. W. Ho, Joris Peters, Peter W. Stahl, Philip J. Piper, Frode Lingaas, Merete Fredholm, Kenine E. Comstock, Jaime F. Modiano, Claude Schelling, Alexander I. Agoulnik, Peter A. Leegwater, Keith Dobney, Jean-Denis Vigne, Carles Vilà, Leif Andersson, and Kerstin Lindblad-Toh "Rethinking dog domestication by integrating genetics, archeology, and biogeography" , 2012.

GUEST et al., "Feasibility of integrating canine olfaction with chemical and microbial profiling of urine to detect lethal prostate cancer," PLOS ONE, 2021.

YONG, Hooi Yong . RUFFMAN, Ted. ,"Emotional contagion: Dogs and humans show a similar physiological response to human infant crying", 2014.Disponível em: https://doi.org/10.1016/j.beproc.2014.10.006

MCDONOUGH, Ian M. ERVWIN, Hillary B. SIN, Nancy L. ALLEN, Rebecca S. "Pet ownership is associated with greater cognitive and brain health in a cross-sectional sample across the adult lifespan" , 2022.

INSTITUTO PET BRASIL.
"Dados IPB: em 2020, mercado pet faturou R$ 40,8 bilhões", 2021.Disponível em: http://institutopetbrasil.com/fique-por-dentro/mercado-pet-faturou/

INSTITUTO PET BRASIL.
"Mercado pet brasileiro: como o amor pelos animais impulsiona os negócios", 2022.
Disponível em:
https://institutopetbrasil.com/fique-por-dentro/amor-pelos-animais-impulsiona-os-negocios/

INSTITUTO PET BRASIL.
"Número de animais de estimação em situação de vulnerabilidade mais do que dobra em dois anos, aponta pesquisa do IPB", 2022.
Disponível em:

http://institutopetbrasil.com/fique-por-dentro/numero-de-animais-de-estimacao-em-situacao-de-vulnerabilidade-mais-do-que-dobra-em-dois-anos-aponta-pesquisa-do-ipb/

BALTHAZAR, Artur Donadel. PIRES, Thaina Alves. PAZMINO, Ana Veronica. "Superpopulação, abandono e maus-tratos de cães: um estudo de design social", 2019. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/244904/VOLUME-5-301-312.pdf?sequence=1

CHERNY, Boris. "Programming TypeScript: Making Your JavaScript Applications Scale", 2019.
Disponível em: https://www.amazon.com/Programming-TypeScript-Making-JavaScript-Applications/dp/1492037656

DANIELSSON, William. "React Native application development – A comparison between native Android and React Native", 2016.
Disponível em: http://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02

LUCIANO, Josué; ALVES, Wallison Joel Barberá. "Padrão de arquitetura MVC: Model-view-controller". EPeQ Fafibe, v. 1, n. 3a, p. 102-107, 2017.
Disponível em: https://unifafibe.com.br/revistasonline/arquivos/revistaepeqfafibe/sumario/20/16112011142249.pdf

EIGY, Cem. "React Functional Components, Props, and JSX – React.js Tutorial for Beginners", 2020.
Disponível em: https://www.freecodecamp.org/news/react-components-jsx-props-for-beginners/

THOMAS, Gaël. "How to Build a Node.js Database using Prisma and SQLite", 2021.
Disponível em: https://www.freecodecamp.org/news/build-nodejs-database-using-prisma-orm/

OYAMA, Faith. “PostgreSQL and JSON – How to Use JSON Data in PostgreSQL”, 2023.Disponível em: https://www.freecodecamp.org/news/postgresql-and-json-use-json-data-in-postgresql/