UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES

FELIPE FERREIRA DUMER

APLICATIVO MULTIPLATAFORMA PARA RASTREAMENTO DE ANIMAIS DE COMPANHIA

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2021

FELIPE FERREIRA DUMER

APLICATIVO MULTIPLATAFORMA PARA RASTREAMENTO DE ANIMAIS DE COMPANHIA

Trabalho de Conclusão de Curso de Graduação, apresentado ao Curso Superior de Tecnologia em Sistemas de Telecomunicações, do Departamento Acadêmico de Eletrônica – DAELN, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. M. Sc. Omero Francisco Bertol

CURITIBA

2021

FELIPE FERREIRA DUMER

APLICATIVO MULTIPLATAFORMA PARA RASTREAMENTO DE ANIMAIS DE COMPANHIA

Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas de Telecomunicações apresentado como requisito para obtenção do título de Tecnólogo em Sistemas de Telecomunicações da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 30/Agosto/2021

Omero Francisco Bertol
Mestrado
Universidade Tecnológica Federal do Paraná – Câmpus Curitiba, DAELN

Edenilson José da Silva
Doutorado
Universidade Tecnológica Federal do Paraná – Câmpus Curitiba, DAELN

Kleber Kendy Horikawa Nabas
Doutorado
Universidade Tecnológica Federal do Paraná – Câmpus Curitiba, DAELN

- A Folha de Aprovação assinada encontra-se na Secretaria do Programa –

CURITIBA
2021

À memória de

Gercely dos Santos Ferreira.

AGRADECIMENTOS

À minha mãe e ao meu tio, em especial, por todo apoio e pela ajuda. Às pessoas que com quem convivi ao longo dos anos, que sempre me incentivaram, apoiaram e que certamente tiveram grande impactado para a conclusão dessa etapa na minha vida.

Muitas pessoas devem a grandeza de suas vidas aos problemas que tiveram de vencer.

(BADEN-POWELL, Robert; 1922)

RESUMO

DUMER, Felipe Ferreira. **Aplicativo multiplataforma para rastreamento de animais de companhia**. 2021. 63 p. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

O objetivo deste trabalho é desenvolver um aplicativo para rastreamento de animais de companhia de fácil manuseio para usuário, utilizando integração a um dispositivo físico de rastreamento já existente. O acesso ao aplicativo será por meio de smartphone com conexão à internet. O aplicativo possibilita ao usuário cadastrar diversos rastreadores em sua conta, sendo possível ao mesmo visualizar na tela inicial a lista de rastreadores cadastrados. O aplicativo realiza a comunicação com os rastreadores através de mensagem de texto (SMS) e informa ao usuário em um mapa onde está a localização do mesmo.

Palavras-chave: Rastreamento. Animais de companhia. SMS. Aplicativo móvel.

ABSTRACT

DUMER, Felipe Ferreira. **Cross-platform app for tracking pet animals**. 2021. 63 p. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

The objective of this work is to develop an application for tracking pets that is easy to use for the user, using integration with an existing physical tracking device. Access to the application will be via a smartphone with an internet connection. The application allows the user to register several trackers in their account, being possible to view the list of registered trackers on the home screen. The application communicates with the trackers via text message (SMS) and informs the user on a map where it is located.

Keywords: Tracking. Company animals. SMS. App.

LISTA DE ILUSTRAÇÕES

LISTA DE LISTAGENS

LISTA DE SIGLAS

GPS	*Global Positioning System* (em português: Sistema de Posicionamento Global)
GSM	*Global System for Mobile Communications* (em português: Sistema Global para Comunicações Móveis)
iOS	*IPhone Operating System*, sistema Operacional para Smartphones criado e exclusivo da Apple
NPM	*Node Package Manager*
SMS	*Short Message Service* (em português: Serviço de mensagens curtas)

SUMÁRIO

1 INTRODUÇÃO

Os brasileiros passam desde o final do século passado por uma mudança no perfil familiar, com o maior foco na vida profissional e acadêmica, muitos casais optam por adiar cada vez mais os planos de terem filhos. No caso particular das mulheres com filhos, sobretudo em idade pré-escolar, as mesmas possuem uma maior probabilidade de conseguirem trabalhos precários e não conseguirem conciliar trabalho com vida familiar e acadêmica (RIBEIRO; GARCIA; FARIA, 2019).

Segundo Mendes *et al*. (2018, p. 2):

> É visível a mudança da sociedade nos dias de hoje, inclusive jovens colocando estudos, carreira e vida amorosa na frente do que antes era estimação. Considerado primeira instância, a família. Nota-se que as mulheres tendem a ser mães, quando por escolha, tardiamente; identificou-se que em 2015 apenas 25,14% das mães possuem idades entre 20 a 24 anos. Isso se dá principalmente pelo conforto e comodidade oferecido por outros artifícios que substituem os filhos, como os animais de estimação [...].

Essas transformações colocam os animais de estimação, em muitas vezes, como membros da família, seus donos sentem suas necessidades humanas como companhia, amizade, amor, preenchidas pelos pets, criando um apego emocional muito grande em seus donos (CARVALHO; PESSANHA, 2012).

Diante desse apego emocional e os sentimentos positivos gerados pelo animal de companhia em seus respectivos donos, a morte ou desaparecimento do mesmo pode causar um sentimento negativo na pessoa que adotou o pet e na família no qual o animal conviveu. Os proprietários podem sentir dor, culpa, tristeza, raiva, dentre outros sentimentos negativos ocasionados pelo ocorrido, há casos de apego tão forte, que podem ocasionar depressão em seus donos. Esses sentimentos são reais e, para os proprietários, a sensação é de perder um membro familiar (GARDEMANN *et al.*, 2009).

Por maior que seja o cuidado dos donos com o seu animal de companhia, em muitos casos, ocorre uma fuga do pet do local de habitação de seus proprietários. Conforme uma pesquisa realizada nos Estados Unidos, cerca de 14% e 15% dos donos de cachorros e gatos, respectivamente, afirmaram que perderam seus pets nos 5 anos anteriores à pesquisa (WEISS; SLATER; LORD, 2012).

Visando amenizar o problema da perca por fuga do animal de companhia, este projeto de pesquisa visa propor o desenvolvimento de um aplicativo móvel para smartphones, onde seja possível rastrear o animal de estimação por através de solução de rastreamento pré-existente de *Global Positioning System* (GPS).

1.1 OBJETIVOS

No desenvolvimento deste trabalho de conclusão de curso tem-se os seguintes objetivo geral e objetivos específicos.

1.1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver e apresentar uma solução para o rastreamento de animais de companhia totalmente funcional.

1.1.2 Objetivos Específicos

Para atender ao objetivo geral neste trabalho de conclusão de curso os seguintes objetivos específicos serão abordados:

a) Desenvolver uma interface agradável e de fácil utilização adequada para smartphones.
b) Desenvolver um aplicativo multiplataforma funcional compatível com Android.
c) Levantar soluções para persistência dos dados do aplicativo.
d) Conectar aplicativo com solução de banco de dados em nuvem.
e) Utilizar tecnologias pré-existentes para rastreamento através de GPS.
f) Integrar o aplicativo com a tecnologia de rastreamento GPS.

1.2 JUSTIFICATIVAS

Com a disseminação dos aparelhos denominados smartphones, cada vez mais a população em geral possui acesso a diversos tipos de novas tecnologias, sendo na sua grande maioria oferecidas por meio de aplicativos. Esses aplicativos em sua grande maioria possuem tecnologia de comunicação de dados, comunicação por GPS, e grande capacidade de processamento.

Visando diminuir o problema exemplificado na introdução, ou seja, a perca dos animais de companhia, este trabalho visa o desenvolvimento de um aplicativo

para rastreamento de animais de companhia, com foco no sistema operacional para dispositivos móveis Android.

1.3 METODOLOGIA

Com a intenção de alcançar os objetivos apresentados previamente, a metodologia foi desenvolvida conforme os subtópicos a seguir.

1.3.1 Elaboração da Interface

Visando obter uma interface de fácil utilização e simplificada, onde esteja disponível apenas as informações que o usuário precisa. Nessa fase do projeto será elaborado um protótipo de *layout*, afim de obter a melhor posição dos objetos na tela em diferentes resoluções de smartphones.

1.3.2 Desenvolvimento Multiplataforma

Os sistemas operacionais para dispositivos móveis dominantes no mercado atualmente são: iOS da Apple e o Android, da Google. Visando atender ambas as plataformas e diminuição de tempo de desenvolvimento, o projeto buscará uma tecnologia única que possa ser utilizada em ambos os sistemas operacionais.

1.3.3 Rastreamento

Atualmente, existem muitas soluções para rastreamento disponíveis no mercado, são rastreadores que seus tamanhos vão de pequeno a grande porte, devido a isso, e economizar tempo no projeto, será utilizado nesse trabalho uma solução de rastreamento de mercado.

Visando a integração entre o aplicativo e esses rastreadores, a comunicação entre os mesmos será por meio de um protocolo já conhecido.

1.4 ESTRUTURA DO TRABALHO

As atividades apresentadas no projeto estão divididas nos seguintes capítulos.

O Capítulo 2 contém a revisão bibliográfica, são introduzidos temas de tecnologias relacionados ao projeto como: Smartphones, Sistemas operacionais para dispositivos móveis, Android, iOS, comunicação cliente-servidor, GPS, dentre outras.

O processo de desenvolvimento do trabalho é descrito no Capítulo 3, onde em um primeiro momento elabora-se os requisitos imaginados do projeto, em seguida, é feito um planejamento da arquitetura do aplicativo, continuando com o banco de dados definido, a prototipação de telas utilizando ferramenta de design, definindo o dispositivo de rastreamento, indo até às tecnologias utilizadas.

No Capítulo 4 é apresentado o resultado do desenvolvimento das telas, com figuras do projeto finalizado.

Finalizando, no Capítulo 5 é feito a conclusão do trabalho apresentando os resultados do tema proposto, com as respectivas dificuldades e considerações finais.

2 REVISÃO DA LITERATURA

2.1 SMARTPHONE

Em tradução do inglês, o termo smartphone refere-se a telefone inteligente. Como descrição mais detalhada, Torres (2009, p. 393) o apresenta como “celular que oferece recursos avançados similares aos de um notebook”, ou seja, o mesmo possui uma capacidade para processamento de dados e conexão com a internet.

Segundo Meirelles (2020), existem no Brasil, 234 milhões de smartphones e conforme dados da Agência Nacional de Telecomunicações (ANATEL), em março de 2021, o país contava com 240,6 milhões de dispositivos móveis em funcionamento, desses, 213,7 milhões com acesso a banda larga móvel, atingindo uma densidade de 99,4 acessos por 100 habitantes (ANATEL, 2021).

2.2 SISTEMAS OPERACIONAIS PARA SMARTPHONES

Dados da empresa Statcounter (2021) apontam que em abril de 2021 os sistemas operacionais para dispositivos móveis, Android e iOS detinham, respectivamente, 86.51% e 13.19% do mercado brasileiro de smartphones, cerca de 99,7% do total. Dessa forma, parece plausível a escolha desses sistemas operacionais como foco para desenvolvimento de aplicativos, pois, sendo assim, grande parcela da população teria acesso aos mesmos sem complicações ou incompatibilidades de hardware.

2.3 ANDROID

O Android é um sistema operacional para dispositivos móveis criado inicialmente pela Android Inc. A plataforma foi apresentada como um sistema operacional baseado no núcleo Linux e que herdaria os padrões dessa plataforma oferecendo estabilidade e segurança aos usuários e desenvolvedores. Baseado em uma filosofia aberta, conceituada como software livre, a ferramenta possibilitaria a utilização simultânea em diversos aparelhos de vários fabricantes (QUERINO FILHO, 2017).

Atualmente mantido pela empresa de tecnologia Google, a plataforma requere que o desenvolvimento de aplicativos para a mesma seja feito utilizando as linguagens de programação Java ou Kotlin (ANDROID DEVELOPER GUIDE, 2021).

2.4 IPHONE OPERATING SYSTEM (IOS)

iPhone Operating System (iOS) é um sistema operacional criado pela empresa Apple Inc. É de código proprietário e exclusivo para smartphones modelo iPhone e para tablets modelo IPads.

Seguindo a documentação disponibilizada pela Apple (IOS DEVELOPER GUIDE, 2021), o desenvolvimento de qualquer aplicativo para iOS deverá utilizar a linguagem de programação Swift ou Objective-C.

2.5 COMUNICAÇÃO CLIENTE-SERVIDOR

Segundo Campos *et al*. (2016), o termo cliente-servidor se refere às aplicações computacionais de várias plataformas, os usuários (clientes) acessam uma interface da aplicação para manipulação de dados enquanto um único servidor detém os dados do serviço. Ainda conforme esse autor, o Cliente também é chamado de *front-end*, vários clientes se comunicam com um único servidor e não se comunicam entre si, o Servidor (também chamado de *back-end*) recebe e responde às solicitações de clientes. Dessa forma, nesse projeto, será adotado o modelo Cliente-servidor pela necessidade de centralização das informações entre vários dispositivos (CAMPOS *et al.*, 2016).

2.6 SISTEMA DE POSICIONAMENTO GLOBAL

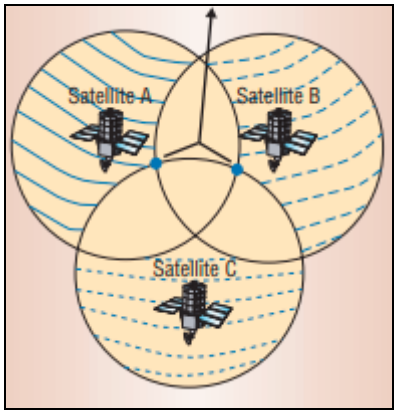
O Sistema de Posicionamento Global, do inglês *Global Positioning System* (GPS) é um sistema de localização com uma grande variedade de aplicações, como rastreamento de encomendas, exploração e resposta de emergência (BAJAJ; RANAWEERA; AGRAWAL, 2002).

Segundo Bajaj, Ranaweera e Agrawal (2002), esse sistema consiste de 30 satélites em seis diferentes órbitas de 12 horas de duração, espaçados de forma que, pelo menos cinco estejam visíveis de qualquer ponto do globo. Cada satélite emite um sinal de rádio que um receptor utiliza para estimar a posição do satélite,

bem como a distância entre os dois. O sinal de rádio contém dados do instante em que foi emitido, de forma que o receptor, utilizando um relógio interno, consegue determinar a diferença de tempo de emissão e recepção da mensagem. Multiplicando essa diferença de tempo pela velocidade da luz, o receptor determina a distância até o satélite.

Com quatro ou mais satélites visíveis o receptor consegue determinar a latitude, longitude e altitude com bastante precisão (BAJAJ; RANAWEERA; AGRAWAL, 2002). É possível visualizar uma ilustração do funcionamento através da Figura 1.

Figura 1 – Triangulação GPS



Fonte: Bajaj, Ranaweera e Agrawal (2002).

2.7 TECNOLOGIAS AUXILIARES

2.7.1 Javascript

Segundo Prescott (2016), Javascript é a linguagem de script do lado do cliente mais amplamente utilizada. É possível utilizar o Javascript para controlar o navegador, comunicar-se de forma assíncrona com o servidor, alterar de forma dinâmica o conteúdo de páginas *web*, desenvolver jogos, aplicações móveis e de *desktop*.

2.7.2 Node.js

Node.js é um interpretador de Javascript de código aberto (*open-source*), orientado a eventos e multiplataforma, executa sobre uma *engine* conhecida como Google V8 (NODE.JS, 2021). Tem como objetivo migrar a programação do Javascript do cliente (*front-end*) para os servidores (*back-end*).

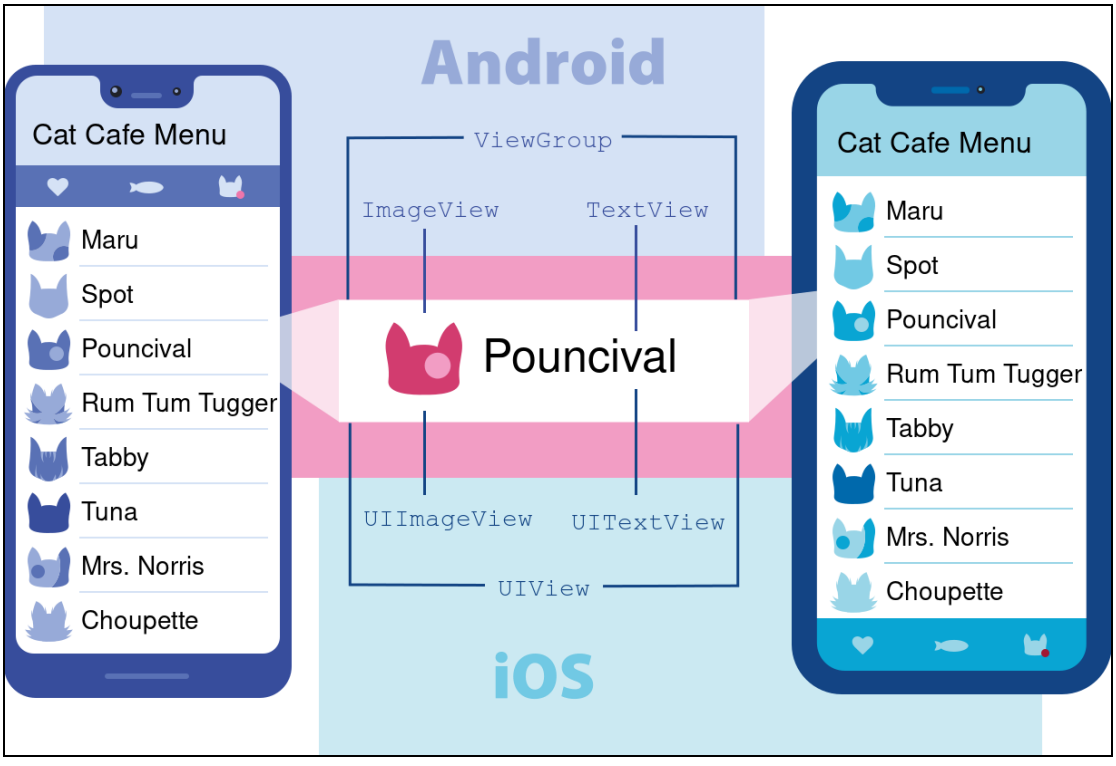
2.7.3 Node Package Manager (NPM)

Node Package Manage (NPM) é um gerenciador de pacotes e dependências de projeto (Listagem 1), utilizado para administrar as bibliotecas e *frameworks* de uma aplicação desenvolvida utilizando Node.js (NPM, 2021).

2.7.4 React Native

O React Native é um *framework* para desenvolvimento de aplicativos para dispositivos móveis lançado inicialmente em 2015 pela empresa Facebook. O *framework* foi lançado inicialmente com suporte apenas para desenvolvimento em smartphones com sistema operacional iOS, posteriormente sendo estendido o suporte para desenvolvimento em smartphones com sistema operacional Android. Destaca-se que utilizando o React Native, é possível desenvolver para as duas plataformas, Android ou iOS, utilizando o mesmo código (SILVA; SOUZA, 2019). O próprio React Native se encarrega de realizar uma compilação utilizando seus componentes para o componente nativo do sistema específico, na Figura 2 é possível visualizar um exemplo de componentes específicos de cada sistema operacional.

Figura 2 – Exemplo de componentes Android e iOS



Fonte: React Native (2021).

O React Native é um framework de código aberto (*open-source)*, ou seja, está disponível para utilização por qualquer pessoa ou empresa (REACT NATIVE, 2021).

2.7.5 Google Firebase

É um conjunto de soluções *back-end* na nuvem disponibilizado pela empresa Google para armazenamento, sincronização e leitura de dados em tempo real. Possui uma variedade de soluções que auxiliam o desenvolvimento de aplicativos e softwares. É uma infraestrutura totalmente alocada em nuvem e auto escalável, a comunicação com a mesma pode ser feita utilizando bibliotecas disponibilizadas e desenvolvidas pela própria empresa (MORONEY, 2017).

Tem por objetivo diminuir o esforço de trabalho do desenvolvedor, deixando o controle das soluções por parte do servidor a responsabilidade da empresa.

2.7.6 Figma

Figma é uma ferramenta de edição de design gráfico focada na criação de interfaces gráficas e experiência de usuário (NASCIMENTO *et al*., 2020). É utilizada para modelagem de telas para aplicativos, páginas web e softwares.

3 DESENVOLVIMENTO

Neste capítulo é descrito o desenvolvimento do projeto. A descrição dos assuntos é feita em subseções que visam englobar os seguintes temas: elaboração de requisitos, definição da arquitetura, visualização geral do banco de dados, apresentação da prototipação das telas, dispositivo de rastreamento utilizado, tecnologias utilizadas no trabalho como: linguagem de programação, soluções de tecnologia e ferramentas utilizadas no processo de desenvolvimento.

3.1 REQUISITOS

Nessa etapa foi elaborado o levantamento de requisitos com base na perspectiva do problema pelo autor. Para o monitoramento de dos animais de companhia, é necessário obter a localização em tempo real do mesmo, para isso, o requisito de localização foi levantado com base em soluções de mercado. Os demais requisitos estão agrupados em duas categorias: requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais são: 1) Possibilidade do usuário se cadastrar; 2) Persistência dos dados do usuário no banco de dados; 3) Possibilidade de o usuário cadastrar seus rastreadores; 4) Possibilidade de o usuário listar seus rastreadores; 5) Integração entre o rastreador e a aplicação; 6) Obter a localização do rastreador; e 7) Exibir a localização do rastreador para o usuário.

Já os requisitos não funcionais são: 1) A aplicação deve ser de fácil uso; 2) A aplicação deve consumir o mínimo possível de poder computacional em nuvem; e 3) A aplicação deve ser visualmente elegante.

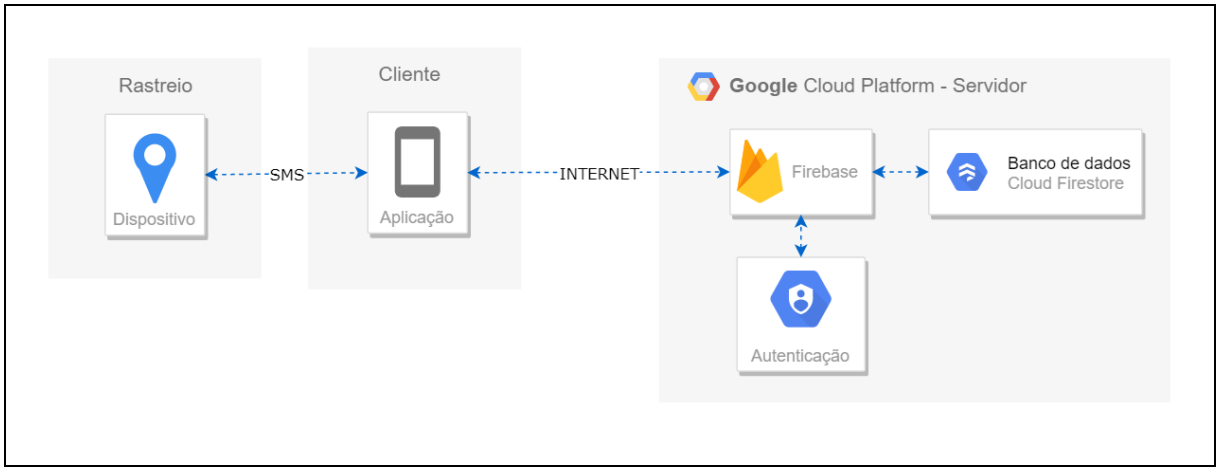
3.2 ARQUITETURA

A arquitetura de software para a aplicação foi definida como uma variação do modelo cliente-servidor, conforme visto na revisão da literatura, o termo cliente-servidor se refere às aplicações computacionais de várias plataformas, os usuários (clientes) acessam uma interface da aplicação para manipulação de dados, enquanto um único servidor detém os dados do serviço.

Para que fosse possível integrar a solução de rastreamento à aplicação, a comunicação entre o aplicativo e o rastreador não passará pelo servidor da aplicação, o servidor ficará encarregado apenas da persistência dos dados e do controle de autenticação do usuário.

Toda a infraestrutura será alocada no serviço firebase da empresa Google Cloud Platform, a comunicação entre a aplicação e o servidor será por meio da internet. A comunicação com a solução de rastreamento será feita por meio do protocolo *SMS*, uma ilustração da arquitetura pode ser visualizada na Figura 3.

Figura 3 – Arquitetura da aplicação

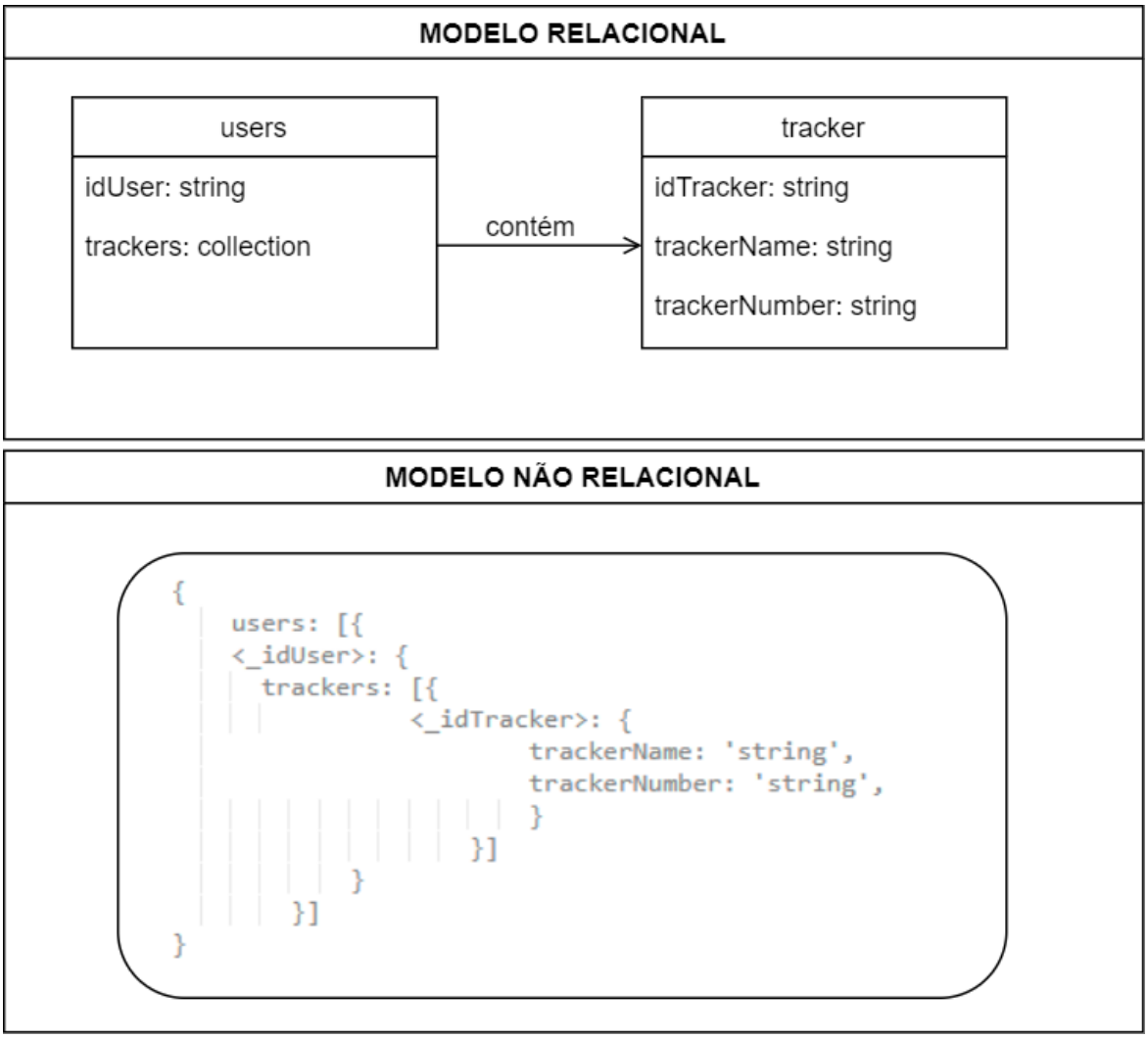


Fonte: Autoria própria.

3.3 BANCO DE DADOS

O serviço de banco de dados utilizado foi o Cloud Firestore, esse é um banco de dados não relacional disponibilizado pela Google, não existe uma modelagem de relações para esse banco, apenas hierarquia dos documentos e coleções. Na Figura 4, é possível visualizar o formato que os dados são armazenados.

Figura 4 – Modelagem de dados

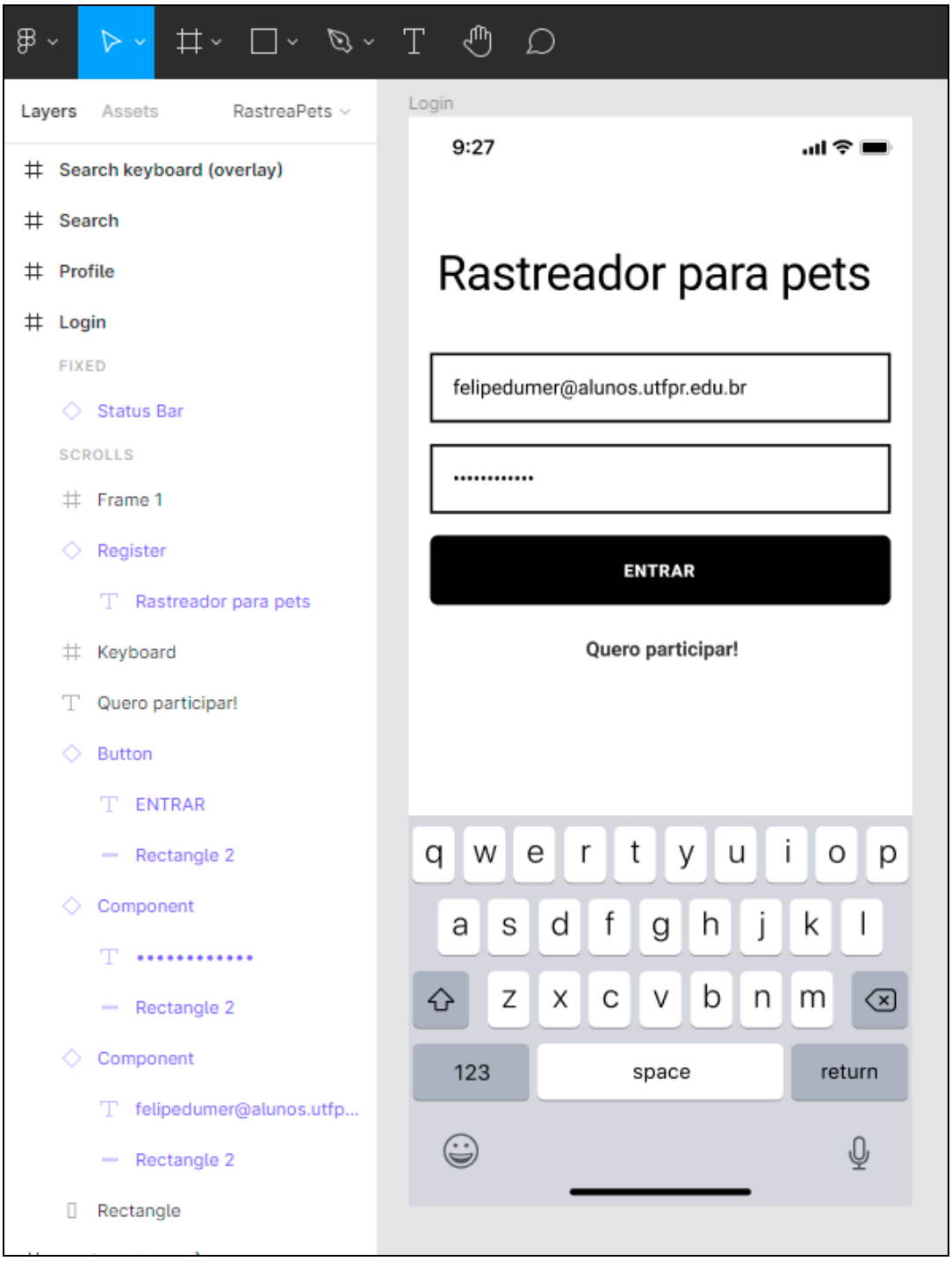


Fonte: Autoria própria.

3.4 PROTOTIPAÇÃO DE TELAS

No início do desenvolvimento do aplicativo foi feito um processo de prototipação para a visualização das telas. Utilizando a ferramenta Figma, foi elaborado um esboço inicial das telas que seriam posteriormente desenvolvidas, é possível visualizar na Figura 5 um modelo da tela inicial.

Figura 5 – Protótipo da Tela Inicial



Fonte: Autoria própria.

3.5 DISPOSITIVO DE RASTREAMENTO

No momento do desenvolvimento do projeto, existiam variadas soluções para rastreamento disponíveis no mercado nacional e global, visando utilizar a menor solução possível para conforto dos animais de companhia, foi optado pela utilização de mini rastreadores.

Para que a solução pudesse ser facilmente integrada ao aplicativo, o rastreador escolhido possuía comunicação por meio de *SMS* através da tecnologia de rede *GSM* e tecnologia de *GPS*.

Foram feitos testes utilizando o modelo GF-07 e o modelo GF-21, sendo esse último utilizado definitivamente no projeto, as especificações do modelo se encontram disponível no site: <https://org-info.mobi/manual/gf-21.html>. É possível visualizar o dispositivo na Fotografia 1.

Fotografia 1 – Dispositivo de rastreamento GF-21

Fonte: Autoria própria.

3.6 TECNOLOGIAS UTILIZADAS

Nas subseções a seguir são introduzidas as ferramentas utilizadas no desenvolvimento do aplicativo com um breve resumo de sua função.

- Javascript: Linguagem de programação definida para codificação do aplicativo, devido a sua sintaxe intuitiva e por ser uma linguagem com rica comunidade e *frameworks* para desenvolvimento de aplicações.
- Node.js: A ferramenta executa Javascript na parte do servidor, possibilitando desenvolvimento das aplicações fora do navegador *web*.
- NPM: Gerenciador de pacotes do Node.js, através dele foi utilizado bibliotecas da comunidade para facilitar o desenvolvimento da aplicação.

- React Native: O *framework* foi utilizado para o desenvolvimento do aplicativo, o mesmo se encarregou de transformar o código Javascript em código nativo para Android (Java) e iOS (Swift).
- Firebase: Utilizado no projeto para controle da parte do servidor do aplicativo, foi utilizada a solução de autenticação para armazenamento dos dados dos usuários e também o serviço de banco de dados *Firestore*.
- Dispositivo Android: O desenvolvimento do aplicativo foi focado no desenvolvimento para dispositivos Android, não sendo priorizado o desenvolvimento para dispositivos com iOS.

4 RESULTADOS

Neste capítulo é apresentado o aplicativo concluído. O capítulo se inicia descrevendo o aplicativo de maneira geral. Em seguida, de forma individual, é exibido cada tela disponível no aplicativo e sua respectiva função.

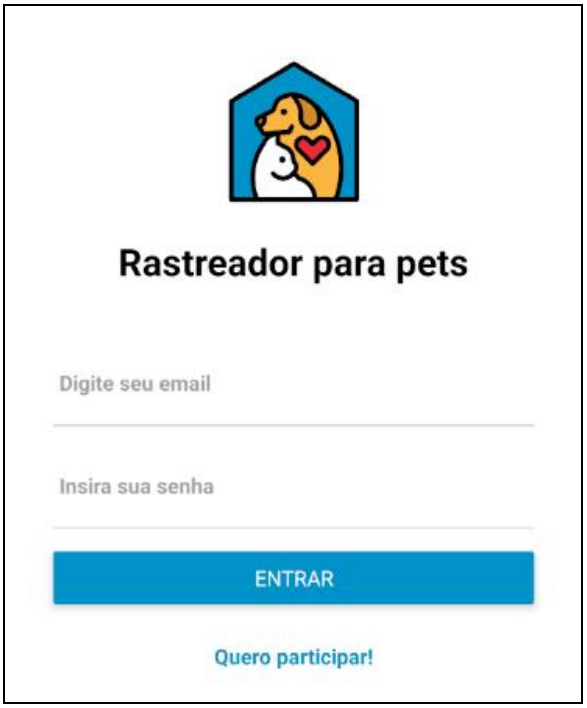
4.1 O APLICATIVO “RASTREADOR PARA PETS”

O aplicativo visa atuar como uma ferramenta para que os donos de animais de companhia encontrem os mesmos sem muitas dificuldades. A seguir são introduzidas as informações de cada tela presente no aplicativo.

4.2 TELA INICIAL

A Tela Inicial, apresentada na Figura 6 e com o código-fonte disponibilizado na Listagem 2, é o primeiro contato visual entre o usuário e a aplicação. Nela é possível entrar no aplicativo digitando seu usuário e senha (caso o mesmo já possua cadastro), e também navegar para a tela de participação selecionando a opção: Quero participar.

Figura 6 – Tela Inicial



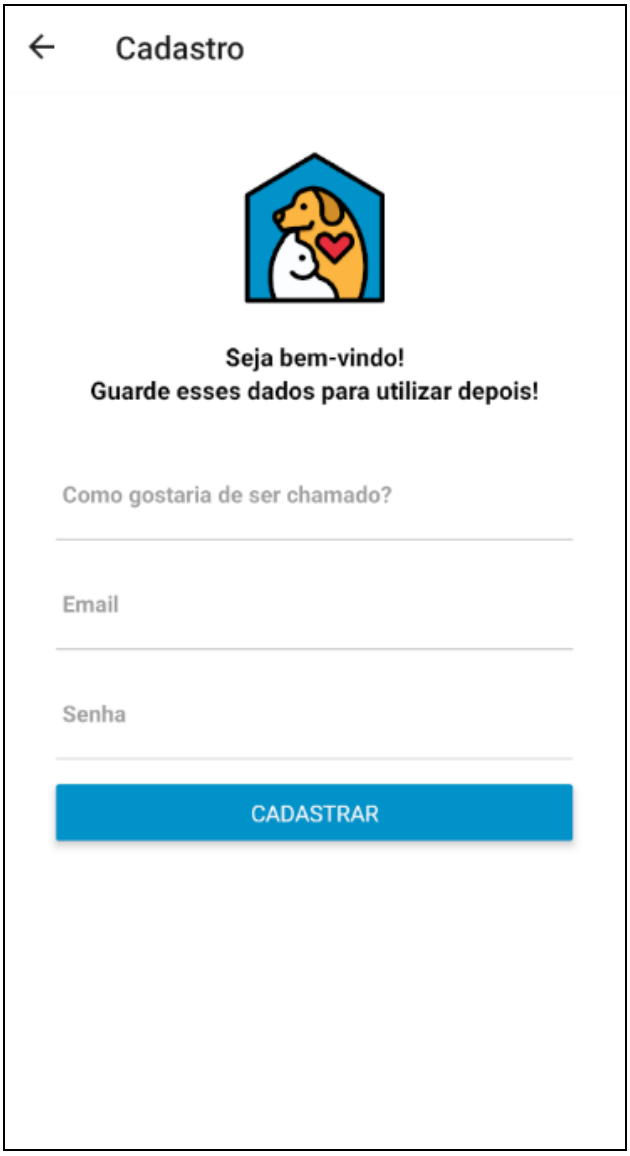
Fonte: Autoria própria.

4.3 TELA DE PARTICIPAÇÃO

A Tela de Participação possui a função de cadastrar o usuário para que o mesmo possa entrar posteriormente no aplicativo. Ao abrir a tela, é solicitado que o usuário insira suas informações para cadastro, como e-mail e senha.

Após a inserção dos dados, é feita uma comunicação com o serviço *Firestore* do *Firebase* salvando os dados do usuário, posteriormente, o usuário é direcionado de forma automática para a “Tela Principal”. Detalhes da Tela de Participação podem ser vistos na Figura 7 e o código-fonte foi disponibilizado na Listagem 3.

Figura 7 – Tela de Participação



Fonte: Autoria própria.

4.4 TELA PRINCIPAL

A Tela Principal, ou Tela de Início, exibe as informações do usuário que está no aplicativo no momento, e também as respectivas informações sobre os rastreadores que estão cadastrados.

Através dessa tela, é possível acessar a “Tela de Configurações” e cadastrar um novo rastreador com a opção “Adicionar Rastreador”. O layout da tela pode ser visualizado na Figura 8 e o código-fonte foi disponibilizado na Listagem 4.

A lista de rastreadores é listada e exibida conforme o nome que o usuário exibiu na “Tela de Cadastro de Rastreadores”, mais detalhes serão abordados na respectiva sessão da tela.

Figura 8 – Tela Principal

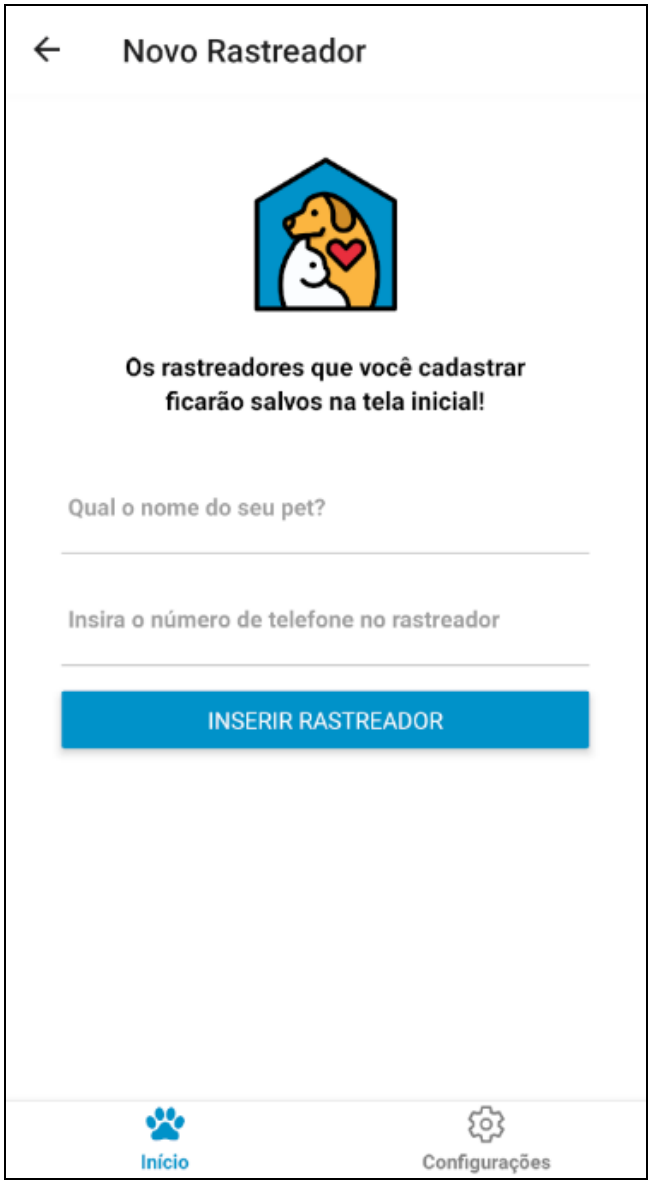
Fonte: Autoria própria.

4.5 TELA DE CADASTRO DE RASTREADORES

A Tela de Cadastro de Rastreadores é a tela que o usuário consegue cadastrar os novos rastreadores na aplicação. Ao abrir a tela, é solicitado que o usuário insira o número de telefone que está inserido no rastreador e o nome do animal de companhia que receberá esse rastreador. É possível visualizar mais detalhes da tela na Figura 9 e o código-fonte foi disponibilizado na Listagem 5.

Esses dados, assim como os dados do usuário, são salvos no serviço *Firebase*, sendo consultados depois quando o usuário abre novamente a aplicação.

Figura 9 – Tela de Cadastro de Rastreadores



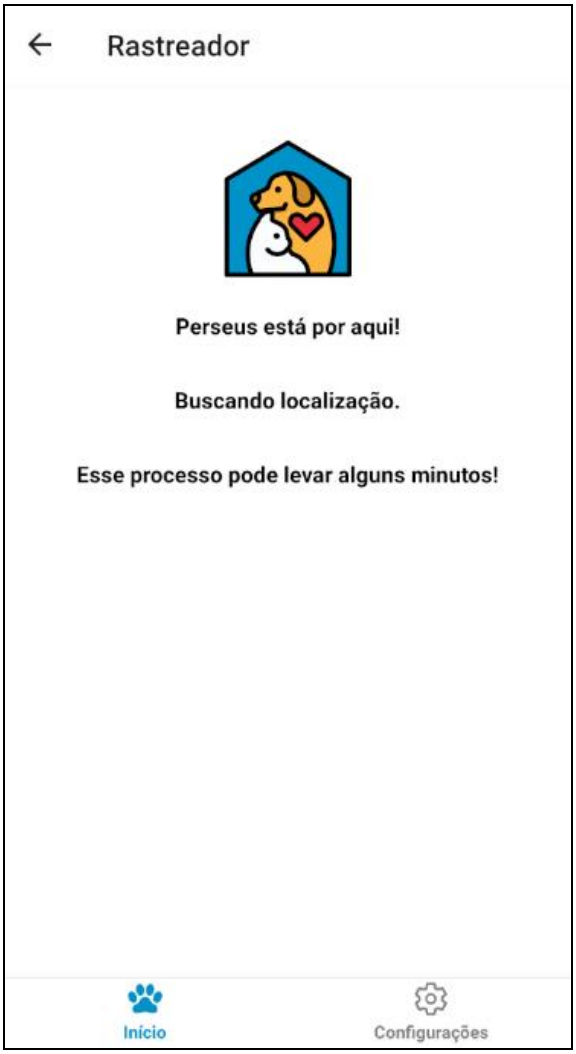
Fonte: Autoria própria.

4.6 TELA DE LOCALIZAÇÃO DO RASTREADOR

Na Tela de Localização do Rastreador é possível verificar onde se encontra a posição do animal de companhia. Essa tela pode ser acessada através da “Tela Principal”, ao pressionar o nome do animal de companhia cadastrado anteriormente, será aberto uma tela de carregamento. Esse processo leva um tempo variável para ser concluído, pois é feito uma comunicação por meio de SMS em segundo plano para o rastreador, a partir disso, o aplicativo aguarda o retorno do rastreador para que seja feito a listagem do mapa (Listagem 6).

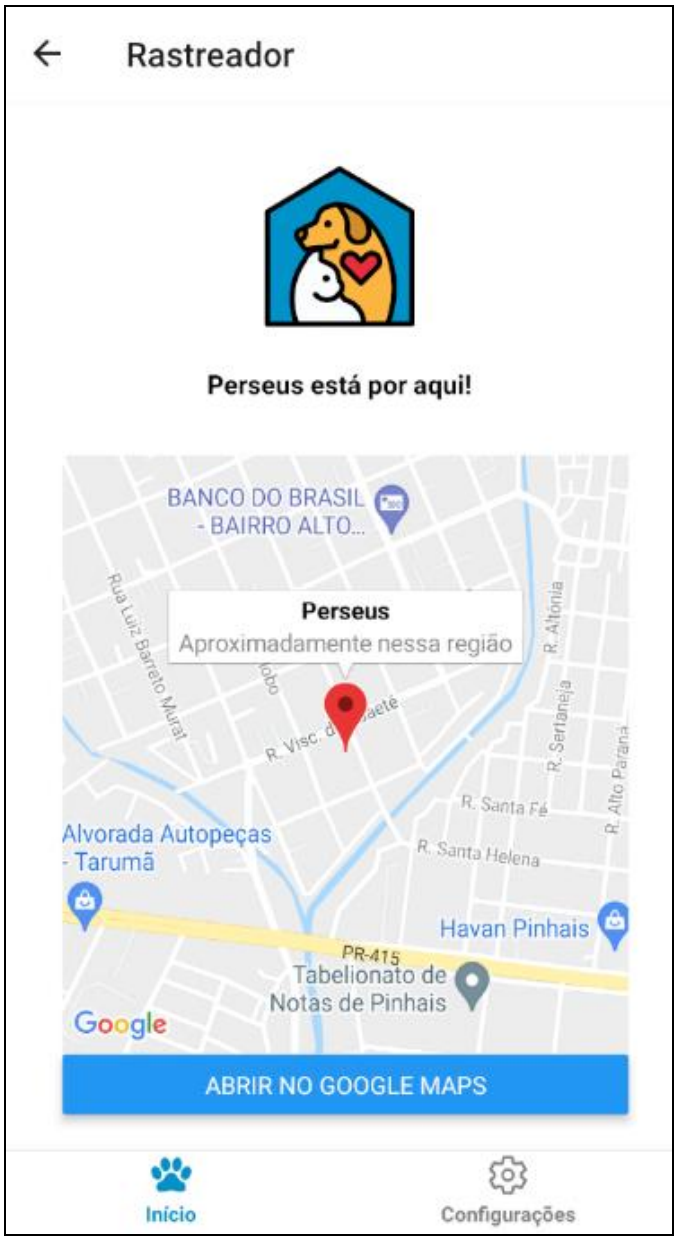
Os layouts da tela podem ser observados na Figura 10 que é a tela enquanto aguarda a reposta do rastreador, e a Figura 11 que se trata da tela final com a localização do animal de companhia.

Figura 10 – Tela de Localização do Rastreador (parte 1)



Fonte: Autoria própria.

Figura 11 – Tela de Localização do Rastreador (parte 2)



Fonte: Autoria própria.

Caso o usuário deseje traçar alguma rota para o ponto onde se encontra o animal de companhia, é possível navegar para o aplicativo do Google Maps, pressionando a opção “Abrir no Google Maps”, após isso, abrirá uma tela do aplicativo da Google com a respectiva localização no mapa, conforme a Figura 12.

Figura 12 – Tela do Google Maps

Fonte: Autoria própria.

4.7 TELA DE CONFIGURAÇÕES

Na Tela de Configurações é possível editar o nome do usuário que está com acesso ao aplicativo. É possível também, desvincular a conta do aplicativo, ou seja, sair do aplicativo para que seja possível entrar com outra conta, ou que os dados da pessoa e seus rastreadores não estejam mais salvos no smartphone utilizado.

O *layout* da tela é apresentado na Figura 13 e o código-fonte foi disponibilizado na Listagem 7.

Figura 13 – Tela de Configurações

Fonte: Autoria própria.

5 CONSIDERAÇÕES FINAIS

O trabalho apresentou uma aplicação multiplataforma para dispositivos móveis como resultado de um projeto para o rastreamento de animais de companhia. Destaca-se que o projeto necessitou de uma ampla base de conhecimento tecnológico, esse projeto utilizou diversos campos de conhecimento tais como: programação para dispositivos móveis, banco de dados, tecnologias de rastreamento, infraestrutura, design, dentre outros temas.

A aplicação foi testada em dispositivos com o sistema operacional Android. Foi obtido resultados satisfatórios em relação ao desempenho da mesma. O aplicativo utilizou a linguagem Javascript como código fonte, visando a integração entre diversos serviços e bibliotecas gratuitas, tais como o framework React Native, as soluções para armazenamento de dados utilizando o Firebase, dentre outros.

Visando a otimização do trabalho, optou-se por utilizar serviços de mercado disponíveis para o back-end, focando o trabalho para a parte de front-end e interação com o usuário. O serviço Firebase se comportou de forma muito satisfatória e nos testes realizados não apresentou falhas. O framework React Native não apresentou nenhum problema no dispositivo testado, não houve lentidão nem muitos erros para a conclusão do trabalho.

A flexibilidade do framework React Native e sua ampla comunidade permitiram que o trabalho fosse desenvolvido e adaptado sem muitas dificuldades ou obstáculos, como as bibliotecas utilizadas no projeto são utilizadas por um número elevado de usuários, testes acabam sendo feitos pelos mesmos, adiantando boa parte do tempo do projeto gastado com testes e adaptações.

Existiu uma dificuldade em relação à solução de GPS de mercado, inicialmente, o projeto foi concebido utilizando o modelo “GF-07”, após testes com 2 unidades, o mesmo não retornou as localizações por meio do protocolo *SMS*. Para contornar essa dificuldade, o rastreador modelo GF-07 foi substituído pelo modelo GF-21, adaptações precisaram ser feitas no projeto devido ao formato diferente de retorno dos dados do mesmo.

Após as adaptações por conta do modelo do rastreador, a comunicação entre o aplicativo e o rastreador ocorreu sem problemas, um ponto a se observar é a demora para a localização ser listada, por conta da tecnologia utilizada pelo rastreador, o mesmo pode levar até 3 minutos para retornar sua localização.

Podem ocorrer casos em que a localização não é retornada, devido ao mal tempo ou impossibilidade do rastreador de recuperar sua localização atual.

Outras funcionalidades podem ser incluídas no projeto, por utilizar muitas tecnologias de código aberto e a linguagem Javascript, é relativamente fácil implementar funcionalidades, respeitando a forma de trabalho do framework React Native.

REFERÊNCIAS

ANATEL. Acessos telefonia móvel. Agência Nacional de Telecomunicações (ANATEL). mar. 2021. Disponível em: <https://informacoes.anatel.gov.br/paineis/acessos/telefonia-movel>. Acesso em: 15 abr. 2021.

ANDROID DEVELOPER GUIDE. **Guias do desenvolvedor**. Google Developers, 2021. Disponível em: <https://developer.android.com/guide>. Acesso em: 28 abr. 2021.

BAJAJ, Rashmi; RANAWEERA, Samantha Lalinda; AGRAWAL, Dharma. **GPS**: location-tracking technology. Computer, v. 35, n. 4, mar. 2002, p. 92–94. Disponível em: <https://www.researchgate.net/publication/220477594_GPS_Location-Tracking_Technology>. Acesso em: 15 ago. 2021.

CAMPOS, Diego Passos Garcia; *et al*. **Aplicação de armazenamento em nuvem utilizando a plataforma node js**. UNIFENAS. v. 11, n.1, 2016. Disponível em: <http://revistas.unifenas.br/index.php/RE3C/article/view/160/105>. Acesso em: 10 mai. 2021.

CARVALHO, Roberto Luis da Silva; PESSANHA, Lavinia Davis Rangel Pessanha. **Relação entre famílias, animais de estimação, afetividade e consumo**: estudo realizado em bairros do Rio de Janeiro. SOCIAIS E HUMANAS, Santa Maria, v. 26, n. 03, set/dez 2013, p. 622-637. Disponível em <https://periodicos.ufsm.br/sociaisehumanas/article/view/6562/pdf>. Acesso em: 05 abr. 2021.

GARDEMANN, Patrícia Nascer; *et al*. **Aspectos emocionais gerados pela morte do animal de estimação**. Arq. Ciênc. Vet. Zool. Unipar, Umuarama, v. 12, n. 1, p. 33-36, jun. 2009. Disponível em <https://revistas.unipar.br/index.php/veterinaria/article/view/2932/2144>. Acesso em: 05 abr. 2021.

IOS DEVELOPER GUIDE. **SDK**: Bring your ideas to life. Apple, 2021. Disponível em: <https://developer.apple.com/develop/>. Acesso em: 28 abr. 2021.

MEIRELLES, Fernando de Souza. **31ª pesquisa anual do FGVCIA: Uso da TI nas empresas**. FGV EAESP, 2020, p. 48. Disponível em: <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/u68/fgvcia2020pesti-resultados_0.pdf>. Acesso em: 15 abr. 2021.

MENDES, Francielly Fontes; *et al*. **Comportamento das famílias brasileiras ante ao crescimento de pets como substituto do filho**. Revista da Graduação da Faculdade Paulus de Comunicação (FAPCOM). Ano 04, v. 08, 2018. Disponível em: <https://www.fapcom.edu.br/revista/index.php/revista-comfilotec/article/view/328/291>. Acesso em: 02 abr. 2021.

MORONEY, Laurence. **The definitive guide to Firebase**: Build Android Apps on Google's Mobile Platform. Apress; 1. ed. nov. 2017.

NASCIMENTO, Karla Angélica Silva do; *et al*. **Ferramenta de prototipagem para criação de um aplicativo para o ensino na saúde**. Workshop de Informática na Escola. Sociedade Brasileira de Computação, 2020. p. 509-513. Disponível em: <https://doi.org/10.5753/cbie.wie.2020.509>. Acesso em: 18 ago. 2021.

NODE.JS. **Documentação Node.js**. Node.js, 2021. Disponível em: <https://nodejs.dev/learn/introduction-to-nodejs>. Acesso em: 10 ago. 2021.

NPM. **Documentação Node Package Manager**. Node Package Manager (NPM), 2021. Disponível em: <https://docs.npmjs.com/about-npm>. Acesso em: 13 ago. 2021.

PRESCOTT, Preston. **Programação em JavaScript**. Babelcube Inc, ago. 2016.

QUERINO FILHO, Luiz Carlos. **Desenvolvendo seu primeiro aplicativo Android**: Entre de cabeça no mundo dos aplicativos móveis, criando e publicando seu próprio programa para o sistema líder do mercado. Novatec Editora, 2017.

REACT NATIVE. Documentação do React Native. React Native, 2021. Disponível em: <https://reactnative.dev>. Acesso em: 17 ago. 2021.

RIBEIRO, Adriana Miranda; GARCIA, Ricardo Alexandre; FARIA, Tereza Cristina de Azevedo Bernardes. **Baixa fecundidade e adiamento do primeiro filho no Brasil**. Revista brasileira de estudos de população. v. 36, São Paulo, set. 30, 2019. Disponível em: <https://www.scielo.br/scielo.php?pid=S0102-30982019000100155>. Acesso em: 02 abr. 2021.

SILVA, Denys Alves; SOUZA, Caio Frias de. **Construção de App com React Native**. Revista Tecnologias em Projeção. v.10, n.1, 2019, p.1. Disponível em: <http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/view/1316>. Acesso em: 17 ago. 2021.

STATCOUNTER. **Mobile Operating System market share Brazil**. **Statcounter** GlobalStats. abr. 2021. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/brazil/#monthly-202004-202104>. Acesso em: 18 abr. 2021.

TORRES, Cláudio. **A Bíblia do Marketing Digital**. Novatec editora Ltda, São Paulo, 2009.

WEISS, Emily; SLATER, Margaret; LORD, Linda. **Frequency of lost dogs and cats in the united states and the methods used to locate them**. Animals**.** v. 2, p. 301-315, 2012. Disponível em: <https://www.mdpi.com/2076-2615/2/2/301/htm>. Acesso em: 13 abr. 2021.

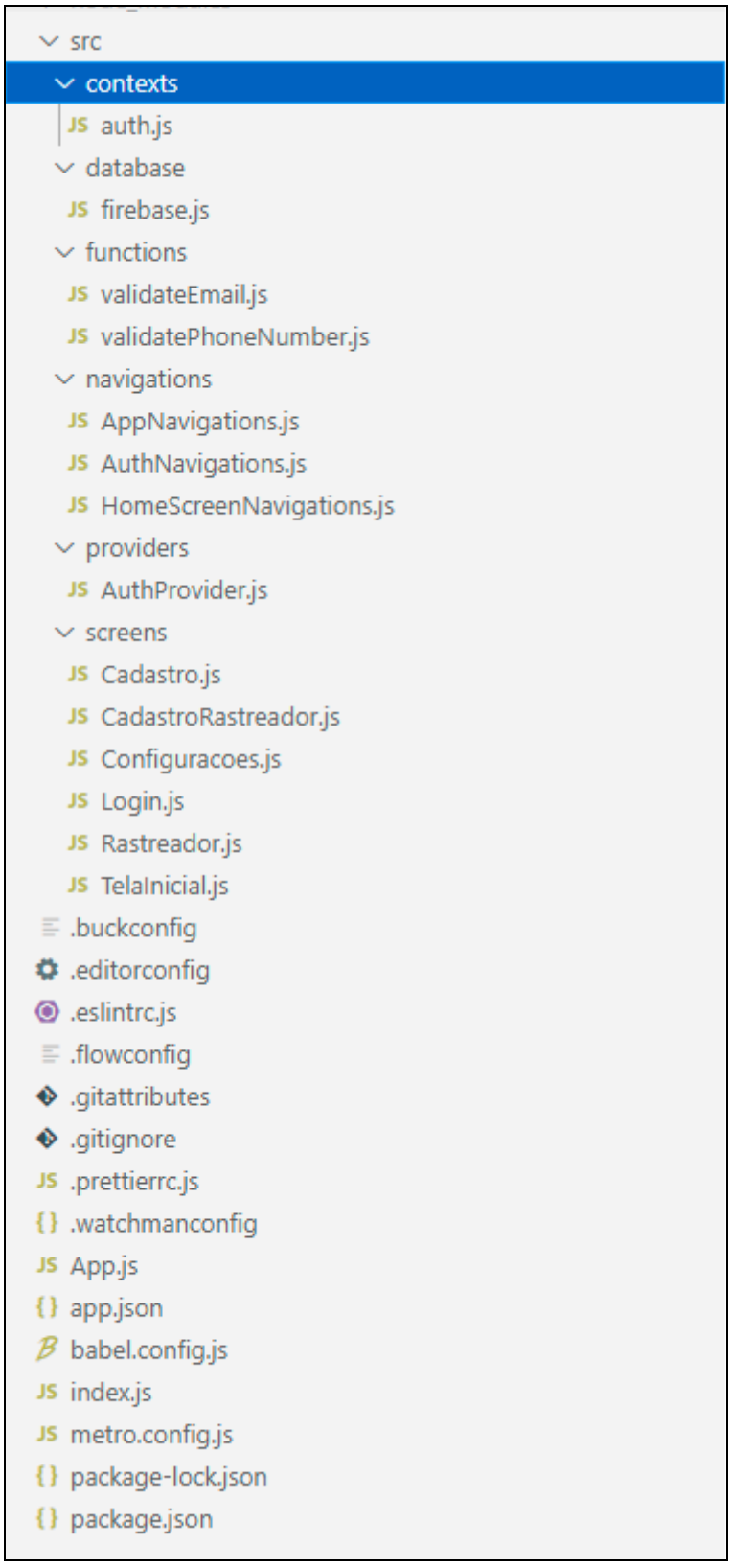
APÊNDICE A – DESTAQUES DA IMPLEMENTAÇÃO

Listagem 1 – Código-fonte das dependências do projeto

```
{
  "name": "rastreadorparapets",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "start": "react-native start",
    "test": "jest",
    "lint": "eslint ."
  },
  "dependencies": {
    "@react-native-community/masked-view": "^0.1.11",
    "@react-navigation/bottom-tabs": "^5.11.13",
    "@react-navigation/native": "^5.9.4",
    "@react-navigation/stack": "^5.14.5",
    "firebase": "^8.8.1",
    "react": "17.0.1",
    "react-native": "0.64.2",
    "react-native-gesture-handler": "^1.10.3",
    "react-native-get-sms-android": "^2.1.0",
    "react-native-maps": "^0.28.0",
    "react-native-reanimated": "^2.2.0",
    "react-native-safe-area-context": "^3.2.0",
    "react-native-screens": "^3.4.0",
    "react-native-vector-icons": "^8.1.0",
    "react-native-webview": "^11.13.0"
  },
  "devDependencies": {
    "@babel/core": "^7.12.9",
    "@babel/runtime": "^7.12.5",
    "@react-native-community/eslint-config": "^2.0.0",
    "babel-jest": "^26.6.3",
    "eslint": "7.14.0",
    "jest": "^26.6.3",
    "metro-react-native-babel-preset": "^0.64.0",
    "react-test-renderer": "17.0.1"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

Fonte: Autoria própria.

Figura 14 – Estrutura de pastas do projeto



Fonte: Autoria própria.

Listagem 2 – Código-fonte da “Tela de Login”

```
import React, {useState} from 'react';
import {
  SafeAreaView,
  ScrollView,
  StyleSheet,
  Text,
  View,
  TextInput,
  Image,
  Button,
  Alert,
  ActivityIndicator,
} from 'react-native';
import firebase from '../database/firebase';
import PetsIntro from '../../assets/images/pets_intro.png';
import validateEmail from '../functions/validateEmail';

const LoginScreen = ({navigation}) => {
  const [userData, setUserData] = useState({
    email: '',
    password: '',
    isLoading: false,
  });

  const handleUpdateInputVal = (val, prop) => {
    const state = {...userData};
    state[prop] = val;
    setUserData(state);
  };

  const handleUserLogin = async () => {
    if (userData.email === '' && userData.password === '') {
      Alert.alert('Ops...', 'Insira seus dados para entrar');
    } else if (!validateEmail(userData.email)) {
      Alert.alert('Ops...', 'Insira um endereço de e-mail válido');
    } else if (userData.password.length < 6) {
      Alert.alert('Ops...', 'Sua senha deve conter no mínimo 6 caracteres');
    } else {
      setUserData({
        ...userData,
        isLoading: true,
      });
      try {
        await firebase
          .auth()
          .signInWithEmailAndPassword(userData.email, userData.password);
      } catch (err) {
        if (err.code === 'auth/wrong-password') {
```

```
          Alert.alert('Ops...', 'Você inseriu uma senha inválida.');
          setUserData({
            ...userData,
            isLoading: false,
            password: '',
          });
        } else if (err.code === 'auth/user-not-found') {
          Alert.alert('Ops...', 'Seu usuário não foi encontrado no servidor');
          setUserData({
            ...userData,
            isLoading: false,
          });
        } else {
          Alert.alert('Erro', err.message);
          setUserData({
            ...userData,
            isLoading: false,
          });
        }
      }
    }
  };

  if (userData.isLoading) {
    return (
      <View style={styles.preloader}>
        <ActivityIndicator size="large" color="#0092c9" />
      </View>
    );
  }

  return (
    <SafeAreaView style={styles.safeContainer}>
      <ScrollView>
        <View style={styles.container}>
          <Image style={styles.petsImage} source={PetsIntro} />
          <Text style={styles.introText}>Rastreador para pets</Text>
          <TextInput
            style={styles.inputStyle}
            placeholder="Digite seu email"
            placeholderTextColor="#585858"
            value={userData.email}
            onChangeText={val => handleUpdateInputVal(val, 'email')}
          />
          <TextInput
            style={styles.inputStyle}
            placeholder="Insira sua senha"
            placeholderTextColor="#585858"
            value={userData.password}
```

```
            onChangeText={val => handleUpdateInputVal(val, 'password')}
            maxLength={15}
            secureTextEntry={true}
          />
          <Button
            color="#0092c9"
            title="Entrar"
            onPress={() => handleUserLogin()}
          />
          <Text
            style={styles.loginText}
            onPress={() => navigation.navigate('Cadastro')}>
            Quero participar!
          </Text>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  safeContainer: {
    flex: 1,
    backgroundColor: 'white',
  },
  container: {
    flex: 1,
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    padding: 35,
    backgroundColor: '#fff',
  },
  introText: {
    marginBottom: 40,
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 25,
  },
  petsImage: {
    marginBottom: 20,
    width: 100,
    height: 100,
    alignSelf: 'center',
  },
  inputStyle: {
    width: '100%',
    marginBottom: 15,
    paddingBottom: 15,
```

```
    alignSelf: 'center',
    color: '#0092c9',
    fontWeight: 'bold',
    borderColor: '#ccc',
    borderBottomWidth: 1,
  },
  loginText: {
    color: '#0092c9',
    fontWeight: 'bold',
    marginTop: 25,
    marginBottom: 55,
    textAlign: 'center',
  },
  preloader: {
    left: 0,
    right: 0,
    top: 0,
    bottom: 0,
    position: 'absolute',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: '#fff',
  },
});

export default LoginScreen;
```

Fonte: Autoria própria.

Listagem 3 – Código-fonte da “Tela de Cadastro”

```
import React, {useState} from 'react';
import {
  StyleSheet,
  Text,
  View,
  TextInput,
  Button,
  Alert,
  ActivityIndicator,
  Image,
  SafeAreaView,
  ScrollView,
} from 'react-native';
import firebase from '../database/firebase';

import PetsIntro from '../../assets/images/pets_intro.png';

import validateEmail from '../functions/validateEmail';

const CadastroScreen = ({navigation}) => {
  const [userForm, setUserForm] = useState({
    displayName: '',
    email: '',
    password: '',
    isLoading: false,
  });

  const handleUpdateInputVal = (val, prop) => {
    const state = {...userForm};
    state[prop] = val;
    setUserForm(state);
  };

  const handleRegisterUser = async () => {
    if (
      userForm.email === '' &&
      userForm.password === '' &&
      userForm.displayName === ''
    ) {
      Alert.alert('Ops...', 'Insira seus dados para entrar');
    } else if (userForm.displayName === '') {
      Alert.alert('Ops...', 'Nos informe como gostaria de ser chamado');
    } else if (!validateEmail(userForm.email)) {
      Alert.alert('Ops...', 'Insira um endereço de e-mail válido');
    } else if (userForm.password.length < 6) {
      Alert.alert('Ops...', 'Sua senha deve conter no mínimo 6 caracteres');
    } else {
      setUserForm({
        isLoading: true,
      });
      try {
        let res = await firebase
          .auth()
          .createUserWithEmailAndPassword(userForm.email, userForm.password);
        res.user.updateProfile({
```

```
          displayName: userForm.displayName,
        });
      } catch (err) {
        Alert.alert('Erro no cadastro', err.message);
        setUserForm({
          isLoading: false,
          displayName: '',
          email: '',
          password: '',
        });
      }
    }
  };

  if (userForm.isLoading) {
    return (
      <View style={styles.preloader}>
        <ActivityIndicator size="large" color="#0092c9" />
      </View>
    );
  }

  return (
    <SafeAreaView style={styles.safeContainer}>
      <ScrollView>
        <View style={styles.container}>
          <Image style={styles.petsImage} source={PetsIntro} />
          <Text style={styles.introText1}>Seja bem-vindo!</Text>
          <Text style={styles.introText2}>
            Guarde esses dados para utilizar depois!
          </Text>
          <TextInput
            style={styles.inputStyle}
            placeholder="Como gostaria de ser chamado?"
            placeholderTextColor="#585858"
            value={userForm.displayName}
            onChangeText={val => handleUpdateInputVal(val, 'displayName')}
          />
          <TextInput
            style={styles.inputStyle}
            placeholder="Email"
            placeholderTextColor="#585858"
            value={userForm.email}
            onChangeText={val => handleUpdateInputVal(val, 'email')}
          />
          <TextInput
            style={styles.inputStyle}
            placeholder="Senha"
            placeholderTextColor="#585858"
            value={userForm.password}
            onChangeText={val => handleUpdateInputVal(val, 'password')}
            maxLength={15}
            secureTextEntry={true}
          />
          <Button
            color="#0092c9"
```

```
            title="Cadastrar"
            onPress={() => handleRegisterUser()}
          />
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  safeContainer: {
    flex: 1,
    backgroundColor: 'white',
  },
  container: {
    flex: 1,
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    padding: 35,
    backgroundColor: '#fff',
  },
  introText1: {
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
  },
  introText2: {
    marginBottom: 30,
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
  },
  petsImage: {
    marginBottom: 20,
    width: 100,
    height: 100,
    alignSelf: 'center',
  },
  inputStyle: {
    width: '100%',
    marginBottom: 15,
    paddingBottom: 15,
    alignSelf: 'center',
    color: '#0092c9',
    fontWeight: 'bold',
    borderColor: '#ccc',
    borderBottomWidth: 1,
  },
  loginText: {
    color: '#0092c9',
    fontWeight: 'bold',
    marginTop: 25,
    marginBottom: 55,
    textAlign: 'center',
  },
```

```
  preloader: {
    left: 0,
    right: 0,
    top: 0,
    bottom: 0,
    position: 'absolute',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: '#fff',
  },
});

export default CadastroScreen;
```

Fonte: Autoria própria.

Listagem 4 – Código-fonte da "Tela Principal"

```
import React, {useEffect, useState, useContext} from 'react';
import AuthContext from '../contexts/auth';
import firebaseApp from '../database/firebase';
import Ionicons from 'react-native-vector-icons/Ionicons';
import {
  SafeAreaView,
  ScrollView,
  StyleSheet,
  Text,
  View,
  TouchableOpacity,
  Image,
} from 'react-native';
import PetsIntro from '../../assets/images/pets_intro.png';

const InicialScreen = ({navigation}) => {
  const {user} = useContext(AuthContext);
  const [trackers, setTrackers] = useState();

  const trackersFirebaseRef = firebaseApp
    .firestore()
    .collection('users')
    .doc(firebaseApp.auth().currentUser.uid)
    .collection('trackers');

  useEffect(() => {}, [user, user.displayName]);

  useEffect(() => {
    return trackersFirebaseRef.onSnapshot(data => {
      const passedValue = [];
      data.forEach(val => passedValue.push(val.data()));
      setTrackers(passedValue);
    });
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  return (
    <SafeAreaView style={styles.safeContainer}>
      <ScrollView>
        <View style={styles.container}>
          <Image style={styles.petsImage} source={PetsIntro} />
          <Text style={styles.displayNameText}>
            {`Olá, ${firebaseApp.auth().currentUser.displayName}`}
          </Text>
          <Text style={styles.descriptionScreenText}>
            {'Veja aqui a localização dos seus pets!'}
          </Text>
          {trackers !== undefined &&
            trackers.map(element => (
              <TouchableOpacity
                key={element.trackerNumber}
                onPress={() => {
                  navigation.navigate('Rastreador', {
                    trackerName: element.trackerName,
                    trackerNumber: element.trackerNumber,
```

```
                  });
                }}
                style={styles.rastreadorButtom}>
                <Text style={styles.rastreadorText}>
                  {`${element.trackerName}`}
                </Text>
                <Ionicons
                  style={styles.rastreadorIcon}
                  name={'location-outline'}
                  color={'white'}
                  size={30}
                />
              </TouchableOpacity>
            ))}
          <TouchableOpacity
            onPress={() => navigation.navigate('Novo Rastreador')}
            style={styles.newRastreadorButton}>
            <Text style={styles.rastreadorText}>Adicionar Rastreador</Text>
            <Ionicons
              style={styles.rastreadorIcon}
              name={'add-circle-outline'}
              color={'white'}
              size={30}
            />
          </TouchableOpacity>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  safeContainer: {
    flex: 1,
    backgroundColor: 'white',
  },
  container: {
    flex: 1,
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    padding: 35,
    backgroundColor: '#fff',
  },
  petsImage: {
    marginBottom: 20,
    width: 100,
    height: 100,
    alignSelf: 'center',
  },
  displayNameText: {
    fontSize: 25,
    color: '#0092c9',
    alignSelf: 'center',
  },
  descriptionScreenText: {
```

```
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
    marginBottom: 40,
    color: '#0092c9',
  },
  rastreadorButtom: {
    width: '65%',
    backgroundColor: '#41bd2e',
    borderRadius: 25,
    height: 60,
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    alignSelf: 'center',
    marginTop: 20,
    marginBottom: 10,
  },
  rastreadorText: {
    paddingLeft: 20,
    fontWeight: 'bold',
    color: 'white',
  },
  rastreadorIcon: {
    paddingRight: 20,
  },
  newRastreadorButton: {
    width: '80%',
    backgroundColor: '#0092c9',
    borderRadius: 25,
    height: 70,
    alignItems: 'center',
    alignSelf: 'center',
    flexDirection: 'row',
    justifyContent: 'space-between',
    marginTop: 20,
    marginBottom: 10,
  },
});

export default InicialScreen;
```

Fonte: Autoria própria.

Listagem 5 – Código-fonte da "Tela de Cadastro de Rastreadores"

```
import React, {useState} from 'react';
import {
  StyleSheet,
  SafeAreaView,
  ScrollView,
  View,
  Text,
  Button,
  TextInput,
  Image,
  ActivityIndicator,
  Alert,
} from 'react-native';
import firebaseApp from '../database/firebase';
import validatePhoneNumber from '../functions/validatePhoneNumber';
import PetsIntro from '../../assets/images/pets_intro.png';

const CadastroRastreadorScreen = ({navigation}) => {
  const [newTrackerForm, setNewTrackerForm] = useState({
    trackerNumber: '',
    trackerName: '',
    isLoading: false,
  });

  const trackersFirebaseRef = firebaseApp
    .firestore()
    .collection('users')
    .doc(firebaseApp.auth().currentUser.uid)
    .collection('trackers');

  const handleUpdateInputVal = (val, prop) => {
    if (prop === 'trackerNumber') {
      if (val.length > 15) {
        return;
      }
      val = val.replace(/\D/g, ''); //Remove tudo o que não é dígito
//Coloca parênteses em volta dos dois primeiros dígitos
      val = val.replace(/^(\d\d)(\d)/g, '($1) $2');
//Número com 9 dígitos. Formato: (99) 99999-9999
      val = val.replace/((\d{5})(\d)/, '$1-$2');
    }
    const state = {...newTrackerForm};
    state[prop] = val;
    setNewTrackerForm(state);
  };

  const handleSaveNewTracker = async () => {
    if (
      newTrackerForm.trackerNumber === '' &&
      newTrackerForm.trackerName === ''
    ) {
      Alert.alert(
        'Ops...',
        'Insira seus dados para cadastrar o novo rastreador',
      );
```

```
    } else if (newTrackerForm.trackerName === '') {
      Alert.alert(
        'Ops...',
        'Precisamos saber o nome do pet que usará o rastreador',
      );
    } else if (!validatePhoneNumber(newTrackerForm.trackerNumber)) {
      Alert.alert(
        'Ops...',
        'O número de telefone precisa ser no formato (xx) xxxxx-xxxx',
      );
    } else {
      setNewTrackerForm({
        ...newTrackerForm,
        isLoading: true,
      });

      try {
        await trackersFirebaseRef.add({
          trackerNumber: newTrackerForm.trackerNumber,
          trackerName: newTrackerForm.trackerName,
        });
        navigation.navigate('Início');
      } catch (err) {
        Alert.alert('Erro no cadastro', err.message);
        setNewTrackerForm({
          ...newTrackerForm,
          isLoading: false,
        });
      }
    }
  };

  return (
    <SafeAreaView style={styles.safeContainer}>
      <ScrollView>
        <View style={styles.container}>
          <Image style={styles.petsImage} source={PetsIntro} />
          <Text style={styles.introText1}>
            Os rastreadores que você cadastrar
          </Text>
        <Text style={styles.introText2}>ficarão salvos na tela inicial!</Text>
          <TextInput
            placeholder="Qual o nome do seu pet?"
            placeholderTextColor="#585858"
            value={newTrackerForm.trackerName}
            onChangeText={val => handleUpdateInputVal(val, 'trackerName')}
            style={styles.inputStyle}
          />
          <TextInput
            placeholder="Insira o número de telefone no rastreador"
            placeholderTextColor="#585858"
            value={newTrackerForm.trackerNumber}
            onChangeText={val => handleUpdateInputVal(val, 'trackerNumber')}
            style={styles.inputStyle}
          />
          {newTrackerForm.isLoading ? (
```

```
            <ActivityIndicator size="large" color="#0092c9" />
          ) : (
            <Button
              color="#0092c9"
              title="Inserir Rastreador"
              onPress={() => handleSaveNewTracker()}
            />
          )}
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};
const styles = StyleSheet.create({
  safeContainer: {
    flex: 1,
    backgroundColor: 'white',
  },
  container: {
    flex: 1,
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    padding: 35,
    backgroundColor: '#fff',
  },
  introText1: {
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
  },
  introText2: {
    marginBottom: 30,
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
  },
  petsImage: {
    marginBottom: 20,
    width: 100,
    height: 100,
    alignSelf: 'center',
  },
  inputStyle: {
    width: '100%',
    marginBottom: 15,
    paddingBottom: 15,
    alignSelf: 'center',
    color: '#0092c9',
    fontWeight: 'bold',
    borderColor: '#ccc',
    borderBottomWidth: 1,
  },
});
export default CadastroRastreadorScreen;
```

Fonte: Autoria própria.

Listagem 6 – Código-fonte da "Tela do Rastreador"

```
import React, {useEffect, useState} from 'react';
import {
  StyleSheet,
  SafeAreaView,
  ScrollView,
  Button,
  View,
  Text,
  Image,
  Linking,
} from 'react-native';
import PetsIntro from '../../assets/images/pets_intro.png';
import SmsAndroid from 'react-native-get-sms-android';
import MapView, {Marker} from 'react-native-maps';
import {WebView} from 'react-native-webview';

const RastreadorScreen = ({navigation, route}) => {
  const {trackerName, trackerNumber} = route.params;

  const [urlLocation, setUrlLocation] = useState(null);
  const [mapsCoordinate, setMapsCoordinate] = useState(null);

  const dateInMsNow = Date.now();
  const numeroSms = '+55' + trackerNumber.replace(/\D/g, '');

  useEffect(() => {
    const sendSms = () => {
      SmsAndroid.autoSend(
        numeroSms,
        '999',
        fail => {
          console.log('Failed with this error: ', fail);
        },
        success => {
          console.log('SMS sent successfully');
        },
      );
    };

    sendSms();
  }, [numeroSms]);

  const getSms = () => {
    let filter = {
      box: 'inbox',
      minDate: dateInMsNow,
      address: numeroSms,
    };
    SmsAndroid.list(
      JSON.stringify(filter),
      fail => {
        console.log('Failed with this error: ' + fail);
      },
      (count, smsList) => {
        let arr = JSON.parse(smsList);
```

```
        arr.forEach(function (object) {
          console.log('Object: ' + object);
          console.log('-->' + object.date);
          console.log('-->' + object.body);
          if (
            object.body.substring(0, object.body.indexOf(' ')) !== 'Torpedo'
          ) {
            setUrlLocation(object.body.substring(object.body.indexOf(',') + 1)
);
          }
        });
      },
    );
  };

  const openGps = (lat, lng) => {
    const browser_url = 'https://maps.google.com/?q=' + lat + ',' + lng;

    Linking.openURL(browser_url);
  };

  const INTERVAL_DELAY = 5000;
  useEffect(() => {
    if (urlLocation === null) {
      const interval = setInterval(() => {
        getSms();
      }, INTERVAL_DELAY);
      return () => clearInterval(interval);
    }
  });

  return (
    <SafeAreaView style={styles.safeContainer}>
      <ScrollView>
        <View style={styles.container}>
          <Image style={styles.petsImage} source={PetsIntro} />
          <Text style={styles.introText2}>{trackerName} está por aqui!</Text>
          {mapsCoordinate === null && (
            <View>
              <Text style={styles.introText2}>Buscando localização.</Text>
              <Text style={styles.introText2}>
                Esse processo pode levar alguns minutos!
              </Text>
            </View>
          )}
          <View style={styles.webView}>
            <WebView
              source={{
                uri: urlLocation,
              }}
              onNavigationStateChange={event => {
                if (event.url !== urlLocation && mapsCoordinate === null) {
                  let newEnd = event.url.replace('+', ',');

                  let mapsCoordinateFromUrl = newEnd.substring(
```

```
                    newEnd.indexOf('q=') + 2,
                    newEnd.indexOf('&'),
                  );

                  let xPos = mapsCoordinateFromUrl.substring(
                    0,
                    mapsCoordinateFromUrl.indexOf(',') + 1,
                  );
                  let yPos = mapsCoordinateFromUrl.substring(
                    mapsCoordinateFromUrl.indexOf(',') + 1,
                  );
                  setMapsCoordinate({
                    xPos: parseFloat(xPos),
                    yPos: parseFloat(yPos),
                  });
                }
              }}
              javaScriptEnabled={true}
              domStorageEnabled={true}
              injectedJavaScript={`
                getStaticGoogleMap(16);
              `}
              geolocationEnabled={true}
            />
          </View>
          {mapsCoordinate !== null && (
            <View style={{width: 330}}>
              <MapView
                style={{height: 350}}
                initialRegion={{
                  latitude: mapsCoordinate.xPos,
                  longitude: mapsCoordinate.yPos,
                  latitudeDelta: 0.0122,
                  longitudeDelta: 0.0121,
                }}>
                <Marker
                  key="01"
                  coordinate={{
                    latitude: mapsCoordinate.xPos,
                    longitude: mapsCoordinate.yPos,
                  }}
                  title={trackerName}
                  description={'Aproximadamente nessa região'}
                />
              </MapView>
              <Button
                onPress={() =>
                openGps(mapsCoordinate.xPos, mapsCoordinate.yPos, trackerName)
                }
                title={'ABRIR NO GOOGLE MAPS'}
              />
            </View>
          )}
        </View>
      </ScrollView>
    </SafeAreaView>
```

```
  );
};

const styles = StyleSheet.create({
  safeContainer: {
    flex: 1,
    backgroundColor: 'white',
  },
  webView: {
    width: 1,
    height: 1
  },
  container: {
    flex: 1,
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    padding: 35,
    backgroundColor: '#fff',
  },
  introText1: {
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
  },
  introText2: {
    marginBottom: 30,
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
  },
  petsImage: {
    marginBottom: 20,
    width: 100,
    height: 100,
    alignSelf: 'center',
  },
  inputStyle: {
    width: '100%',
    marginBottom: 15,
    paddingBottom: 15,
    alignSelf: 'center',
    color: '#0092c9',
    fontWeight: 'bold',
    borderColor: '#ccc',
    borderBottomWidth: 1,
  },
});

export default RastreadorScreen;
```

Fonte: Autoria própria.

Listagem 7 – Código-fonte da "Tela de Configurações"

```
import React, {useState, useContext} from 'react';
import {
  StyleSheet,
  SafeAreaView,
  ScrollView,
  View,
  Text,
  Button,
  TextInput,
  Image,
  ActivityIndicator,
} from 'react-native';

import AuthContext from '../contexts/auth';

import firebase from 'firebase';

import PetsIntro from '../../assets/images/pets_intro.png';

const CadastroRastreadorScreen = ({navigation}) => {
  const {user, setUser} = useContext(AuthContext);
  const [settings, setSettings] = useState({isLoading: false});

  const handleUpdateInputVal = (val, prop) => {
    const state = {...settings};
    state[prop] = val;
    setSettings(state);
  };

  const handleUpdateSettings = async () => {
    try {
      setSettings({
        ...settings,
        isLoading: true,
      });

      await firebase.auth().currentUser.updateProfile({
        displayName: settings.displayName,
      });

      setUser({...user, displayName: settings.displayName});

      setSettings({
        isLoading: false,
      });
    } catch (err) {
      console.log(err);
      setSettings({
        ...settings,
        isLoading: false,
      });
    }
  };

  const handleLogoff = () => {
```

```
    firebase.auth().signOut();
    navigation.navigate('Início');
  };

  return (
    <SafeAreaView style={styles.safeContainer}>
      <ScrollView>
        <View style={styles.container}>
          <Image style={styles.petsImage} source={PetsIntro} />
          <Text style={styles.displayNameText}>
            {`Olá, ${user.displayName}`}
          </Text>
          <Text style={styles.descriptionScreenText}>
            Aqui estão as suas configurações
          </Text>
          <TextInput
            placeholder="Como gostaria de ser chamado?"
            placeholderTextColor="#585858"
            value={settings.displayName}
            onChangeText={val => handleUpdateInputVal(val, 'displayName')}
            style={styles.inputStyle}
          />
          {settings.isLoading ? (
            <ActivityIndicator size="large" color="#0092c9" />
          ) : (
            <Button
              color="#0092c9"
              title="Alterar meu nome"
              onPress={() => handleUpdateSettings()}
            />
          )}
          <View style={{paddingTop: 25}}>
            <Button
              color="#ed5151"
              title="Sair do aplicativo"
              onPress={() => handleLogoff()}
            />
          </View>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  safeContainer: {
    flex: 1,
    backgroundColor: 'white',
  },
  container: {
    flex: 1,
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    padding: 35,
    backgroundColor: '#fff',
```

```
  },
  petsImage: {
    marginBottom: 20,
    width: 100,
    height: 100,
    alignSelf: 'center',
  },
  displayNameText: {
    fontSize: 25,
    color: '#0092c9',
    alignSelf: 'center',
  },
  descriptionScreenText: {
    alignSelf: 'center',
    fontWeight: 'bold',
    fontSize: 15,
    marginBottom: 40,
    color: '#0092c9',
  },
  inputStyle: {
    width: '100%',
    marginBottom: 15,
    paddingBottom: 15,
    alignSelf: 'center',
    color: '#0092c9',
    fontWeight: 'bold',
    borderColor: '#ccc',
    borderBottomWidth: 1,
  },
});

export default CadastroRastreadorScreen;
```

Fonte: Autoria própria.