UNIVERSIDADE DO SUL DE SANTA CATARINA

ARTUR MENDES BRASIL

ELIAS OLIVEIRA DA ROSA

PET'S HERO

DESENVOLVIMENTO DE UMA PLATAFORMA QUE VISA AUXILIAR INSTITUIÇÕES DEFENSORAS DOS ANIMAIS EM RECEBER DOAÇÕES E MOSTRAR SUAS NECESSIDADES

Tubarão

2021

ARTUR MENDES BRASIL

ELIAS OLIVEIRA DA ROSA

PET'S HERO

DESENVOLVIMENTO DE UMA PLATAFORMA QUE VISA AUXILIAR INSTITUIÇÕES DEFENSORAS DOS ANIMAIS EM RECEBER DOAÇÕES E MOSTRAR SUAS NECESSIDADES

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade do Sul de Santa Catarina como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Luciano José Sávio

Tubarão

2021

ARTUR MENDES BRASIL

ELIAS OLIVEIRA DA ROSA

PET'S HERO

DESENVOLVIMENTO DE UMA PLATAFORMA QUE VISA AUXILIAR INSTITUIÇÕES DEFENSORAS DOS ANIMAIS EM RECEBER DOAÇÕES E MOSTRAR SUAS NECESSIDADES

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade do Sul de Santa Catarina como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Tubarão, 19 de junho de 2021.

_____

Professor e orientador Luciano Sávio.
Universidade do Sul de Santa Catarina

_____

Prof. Clávison Zapelini
Universidade do Sul de Santa Catarina

_____

Prof. Osmar
Universidade do Sul de Santa Catarina

RESUMO

Este trabalho aborda o desenvolvimento de um sistema *web* voltado à otimização da comunicação entre doadores e instituições protetoras dos animais. Embora existam aplicações que ajudam as ONGs a mostrar seu trabalho, poucas possuem como foco ser uma ferramenta específica que atenda às necessidades observadas. Atualmente, este é um processo custoso para as instituições, sendo necessário visitar diversas ferramentas como *Instagram*, *Facebook* e outras, para estabelecer um fluxo de comunicação e centralização das informações recebidas de diferentes fontes. A ferramenta desenvolvida neste trabalho visa auxiliar as instituições a divulgarem seus trabalhos, facilitar o processo de adoção de animais, divulgar campanhas de arrecadação de recursos para o desenvolvimento de projetos dentro da ONG, como por exemplo: reforma de estrutura ou arrecadação de fundos para procedimentos médicos que algum dos animais esteja precisando. A ferramenta também pode ajudar os usuários a encontrarem animais perdidos. Futuramente, será viável oferecer funcionalidades extras, como calendário de eventos, agendar visitas pelo site, ser padrinho de um animal através de doações mensais fixas e loja online das instituições. Portanto, o objetivo geral deste trabalho é o desenvolvimento de uma ferramenta que, efetivamente contribua na organização das instituições protetoras de animais em risco e abandonados.

Palavras-chave: TypeScript. React. JavaScript. Web. PWA. Animais. ONG.

ABSTRACT

This work addresses the development of a web system aimed at optimizing communication between donors and animal protection institutions. Although there are applications that help NGOs show their work, few are focused on being a specific tool that meets the needs observed. Currently, this is a costly process for institutions, and it is necessary to visit various tools such as Instagram, Facebook and others, to establish a flow of communication and centralize information received from different sources. The tool developed in this article aims to help institutions disseminate their work, facilitate the process of adopting animals, publicize fundraising campaigns for the development of projects within the NGO, such as: structure reform or fundraising for procedures doctors that any of the animals are in need. The tool can also help users find lost animals. In the future, it will be viable to offer extra features, such as calendar of events, scheduling visits through the website, being the godfather of an animal through fixed monthly donations and the institutions' online store. Therefore, the general objective of this work is the development of a tool that effectively contributes to the organization of institutions that protect at-risk and abandoned animals.

Keywords: TypeScript. React. ReactNative. JavaScript. Web. PWA. Animal. NGO.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Um dos grandes problemas encontrados nas mais diversas cidades brasileiras é o grande número de animais abandonados. Segundo o Instituto Pet Brasil, em levantamento feito em 2019, o Brasil possui mais de 170 mil animais abandonados sob cuidado de ONGs e grupos de proteção animal. O levantamento também apurou a existência de 370 ONGs e instituições atuando na proteção animal em todo o país.

A maioria dessas organizações se apresentam apenas em redes sociais, e por meio destas, divulgam todo o seu trabalho, sendo eles: animais que estão para adoção e que já foram adotados, divulgação de eventos e campanhas para angariar recursos, pedidos de ajuda para animais que precisam de algum tratamento, cirurgia e outros.

O maior problema que a utilização das redes sociais possui é a falta de organização e escassez de ferramentas que facilitem o recebimento de doações, ao utilizar, se torna difícil identificar as informações que são mais recentes, onde fica os destaques, especificamente no *Instagram*, as publicações recentes e antigas ficam todas juntas e não é possível identificar se já foi solucionado o problema, no caso de uma adoção ou levantamento de recursos através das doações. Com essa desorganização, possíveis adotantes ou doadores, ao acessarem os perfis desses grupos, podem não conseguir encontrar as informações que estão procurando. Podemos dar como exemplo o perfil "segundachancesc" do *Instagram*. Esse perfil é de um grupo de voluntários de Tubarão/SC que atua principalmente com foco na castração de animais de rua, o grupo possui diversos animais para adoção, mas através do Instagram, que é o principal meio de divulgação do grupo, não é possível encontrar com facilidade, todos os animais que estão para a adoção hoje, ou que já foram adotados. Outras informações como prestação de contas, divulgação de animais resgatados, animais que estão precisando de ajuda, animais perdidos também são difíceis de encontrar por causa da grande quantidade de informações concentradas em apenas um lugar, sem nenhuma categorização ou filtro.

Esse trabalho tem como objetivo geral criar uma ferramenta que auxilie instituições defensoras dos animais a organizarem e divulgarem melhor as informações. Assim, possivelmente, esses grupos podem ter um aumento no número de adotantes ou até mesmo de doadores.

Para o alcance do objetivo geral são necessários os seguintes objetivos específicos:

a) Identificar os principais problemas das ONG's relacionados a organização e divulgação de suas necessidades (solicitar doações para os animais residentes na instituição, mostrar animais que estão para adoção, entre outras);
b) Conhecer as principais ferramentas de desenvolvimento, como, *React*, *Typescript* e *Node.js*;
c) Desenvolver uma aplicação responsiva que funcione em computadores e celulares.

O trabalho está estruturado em 7 capítulos. Sendo o primeiro, esta introdução.

No segundo capítulo apresentamos 2 trabalhos relacionados, que também abordam a criação de uma ferramenta que auxilia ONG’s e pessoas que se identificam com a causa animal. No terceiro capítulo, mostramos a metodologia utilizada. No quarto, apresentamos os requisitos funcionais e não funcionais do desenvolvimento do trabalho. No quinto, descrevemos como foi o processo de desenvolvimento da aplicação. No sexto, apresentamos todas as funcionalidades da aplicação já finalizada, com descrição e uma captura de tela para cada funcionalidade. No sétimo e último capítulo, discorremos sobre as conclusões do trabalho.

2 TRABALHOS RELACIONADOS

A seguir, apresentaremos dois trabalhos correlatos, que se assemelham com o objetivo do nosso trabalho. Ambos visam o desenvolvimento de uma plataforma que auxilie instituições protetoras de animais ou pessoas que se identificam com a causa. Também há uma similaridade com o nosso trabalho, em relação à linguagem utilizada: *JavaScript*.

2.1 PETAPPY – APLICATIVO PARA PROTEÇÃO DOS ANIMAIS

O objetivo deste trabalho foi criar um aplicativo que auxilia as pessoas a encontrar seus animais perdidos. Para isso, o autor optou por utilizar *React Native* no *front-end* por ser uma ferramenta que permite desenvolver tanto para *iOS* quanto para *Android* usando o mesmo código fonte. Para o *backend* foi utilizado *Ruby on Rails*, e o banco de dados escolhido foi *Postgresql*. A metodologia usada durante o desenvolvimento do trabalho foi o *Design Sprint*, que, conforme explicada por Soares (2017, p.16):

> É uma metodologia ágil focada na experiência do usuário e nos processos de design do produto, colocando também em si a lógica de negócio, mantendo especialmente o usuário como foco e centro de toda a operação de desenvolvimento.

Assim, de acordo com o autor, todo o desenvolvimento da ferramenta deve estar consonante com o usuário final.

2.2 APLICAÇÕES MÓVEIS NATIVAS COM REACT NATIVE E FIREBASE: UM ESTUDO DE CASO

Nessa monografia, Silva (2018), desenvolveu um aplicativo com o objetivo de ser um elo de comunicação entre ONG's e pessoas, ou ONG's e ONG's, para que assim, pudesse engajar novos doadores e/ou voluntários.

O aplicativo desenvolvido foi utilizado como estudo de caso sobre a capacidade do *JavaScript* na criação de aplicativos móveis.

3 METODOLOGIA

A metodologia adotada neste trabalho foi utilizar dados coletados através de pesquisas bibliográficas, análise de ferramentas correlatas e obtenção de *feedback* de voluntários de instituições defensoras dos animais. O público-alvo dessa proposta são instituições que buscam melhores formas de organizar e divulgar informações e pessoas que queiram ajudar na causa animal.

3.1 LEVANTAMENTO DE DADOS

Durante a validação da proposta desse trabalho, realizamos a coleta de informações referente às funcionalidades do sistema e o quão importante cada funcionalidade seria. A pesquisa foi realizada pelo *Google Forms* e enviada a pessoas próximas dos autores, que se identificam com a causa animal.

Para o levantamento das funcionalidades que foram mostradas no formulário, foi feito uma pesquisa exploratória de problemas nas principais redes sociais, assim foi possível propor soluções, ao colocar as funcionalidades do formulário e apresentar aos entrevistados, foi possível identificar quais funcionalidades apresentavam maior relevância para o desenvolvimento do projeto.

Quanto aos resultados do levantamento: a pesquisa era composta de diversas funcionalidades e o entrevistado respondia se considerava aquela funcionalidade importante ou não. No total 12 pessoas responderam a pesquisa, 4 delas sendo voluntárias em alguma instituição.

Tabela 1 – Resultados da pesquisa

FUNCIONALIDADE	**“É importante”**
Cadastrar/visualizar animais para adoção, com fotos e informações como idade, porte, sexo.	12 de 12
Espaço com publicações de imagem ou vídeo, mostrando as atividades da instituição.	7 de 12
Doações direto pelo sistema.	9 de 12
Agendar visitas à instituição.	4 de 12
Cadastrar/visualizar animais que estão desaparecidos.	12 de 12
Criar/visualizar campanhas de arrecadação para algum animal	12 de 12

Fonte: Elaborada pelos autores (2021).

Então, de acordo com a pesquisa, concluímos que a maioria das funcionalidades transmite importância efetiva aos usuários. Com exceção das funcionalidades “Espaço com publicações de imagem ou vídeo, mostrando as atividades da instituição” e “Agendar visitas à instituição”, que apenas 7 e 4 entrevistados consideraram importante, respectivamente.

3.2 PLANO DE TRABALHO

Para atingir todos os objetivos do projeto, as seguintes etapas foram definidas:

a) Pesquisa para definir as necessidades do projeto: levantamento de todas as funcionalidades necessárias para o sistema final.
b) Pesquisa bibliográfica: com as necessidades da ferramenta definidas, foi realizado um estudo relacionado às tecnologias adequadas para auxiliar na resolução do problema, como linguagens, integrações, *API's*, *frameworks* e técnicas de desenvolvimento.
c) Implementação: desenvolvimento do sistema conforme definições realizadas nas etapas anteriores.
d) Análise, correções e testes: análise da aplicação através de testes, ajustes das funcionalidades e melhorias.

No próximo capítulo, apresentaremos os requisitos funcionais e não funcionais do desenvolvimento do trabalho.

4 REQUISITOS

Os requisitos funcionais (RF) e não funcionais (RNF) mapeados para o desenvolvimento do projeto foram os seguintes:

4.1 REQUISITOS FUNCIONAIS (RF)

Os requisitos funcionais (RF) do projeto são os seguintes:

- RF01: O usuário deve ser capaz de criar uma conta do tipo doador ou instituição, informando nome, *WhatsApp*, e-mail e senha. Caso seja uma instituição deve informar também a chave PIX.
- RF02: Todos os usuários devem ser capazes de editar as informações do seu perfil, cadastrar foto e endereço.
- RF03: Usuários do tipo doador devem ser capazes de cadastrar animais perdidos, informando nome, idade, sexo, porte, espécie, raça, descrição e foto do animal.
- RF04: Usuários do tipo doador devem ser capazes de editar informações de seus animais perdidos, e alterar o status dos animais para "encontrado".
- RF05: ONG's devem ser capazes de cadastrar animais para adoção, informando nome, idade, sexo, porte, espécie, raça, descrição e foto do animal.
- RF06: ONG's devem ser capazes de editar informações de seus animais para adoção, e alterar o status dos animais para "adotado".
- RF07: ONG's devem ser capazes de cadastrar campanhas de arrecadação, informando título, descrição, meta de arrecadação, valor arrecadado e foto. A campanha pode ou não, ter um animal da ONG atrelado a ela.
- RF08: ONG's devem ser capazes de editar informações das suas campanhas, e alterar o status para "inativa".
- RF09: Permitir que qualquer usuário entre em contato com as ONG's e com os donos de animais perdidos.
- RF10: Todos os usuários devem conseguir visualizar os animais e campanhas cadastradas por determinada ONG.

- RF11: Todos os usuários devem conseguir visualizar:
  - RF11.1: Todos os animais perdidos, encontrados, adotados ou para adoção;
  - RF11.2: Todas as ONG's cadastradas;
  - RF11.3: Todas as campanhas de arrecadação;
  - RF11.4: A chave PIX das ONG's cadastradas.

4.2 REQUISITOS NÃO FUNCIONAIS (RNF)

Os requisitos não funcionais (RNF) do projeto são os seguintes:

- RNF01: O *front-end* deve ser desenvolvido utilizando *React*.
- RNF02: O *back-end* deve ser desenvolvido utilizando *Node.js*.
- RNF03: A aplicação deve ser um *Progressive Web Application* (PWA);
- RNF04: A aplicação deve suportar os navegadores web mais utilizados e as versões mais recentes dos mesmos.

No capítulo a seguir descrevemos como foi o processo de desenvolvimento da aplicação, desde a escolha das linguagens e ferramentas até os principais desafios que encontramos conforme o projeto evoluía.

5 DESENVOLVIMENTO

Para o desenvolvimento do projeto foram utilizadas diferentes tecnologias para *front-end* e *back-end*.

Utilizamos o banco de dados *PostgreSQL* rodando dentro um container no *Docker*, o *Docker* é uma plataforma open source, que possibilita efetuar instalações isoladas, esse isolamento é chamado de container, que facilita a remoção ou atualização sem afetar todo o projeto, nesses containers são instalados ferramentas de um servidor para facilitar a execução no ambiente de desenvolvimento. Para testar as requisições REST desenvolvidas no *back-end* usamos o *Insomnia*, uma aplicação *open-source* que auxilia no desenvolvimento e testes de *API's*.

5.1 ANÁLISE E ESCOLHA DAS LINGUAGENS

Para desenvolver a ideia principal do projeto, foi necessário analisar o ambiente de desenvolvimento e quais linguagens de programação o mercado oferece para solucionar determinados problemas. Tendo em mente que a plataforma é um sistema *web*, optamos por utilizar como linguagem de programação central o *JavaScript* junto com o conjunto de ferramentas *TypeScript* para adicionar tipagem estática ao desenvolvimento, assim podemos definir o tipo de uma variável como *String, Number, Boolean*, *Array*, entre outros exemplos. Para o desenvolvimento da aplicação mobile, foi aplicado a tecnologia *PWA*, que possibilita a utilização de uma implementação *web* em um dispositivo móvel, simulando uma aplicação nativa.

5.1.1 Front-end web

Na programação, o *front-end* pode ser definido como a parte visível de um projeto, é a parte que será apresentada para o cliente: as telas do sistema, botões, tabelas, imagens, animações, etc. É a partir daí que o usuário consegue efetuar cadastros, inserir informações, entre outros dados, que serão enviados para o *back-end,* onde as informações serão processadas e armazenadas no banco de dados, podendo retornar ou não, alguma mensagem ou informação relevante para o usuário.

Para podermos utilizar a linguagem de programação *JavaScript* no desenvolvimento do o *front-end* do projeto, analisamos três opções atuais de *framework*, sendo eles o *Angular, React* e o *Vue*:

1. O *Angular* traz para o ambiente de desenvolvimento, dentro de seus componentes, todas as funcionalidades que um desenvolvedor necessita para pôr em prática a sua ideia, mas ele é geralmente usado para sistemas maiores e complexos, para projetos menores ele pode tornar a aplicação lenta, pelo fato de trazer junto na instalação todos as suas funcionalidades embarcadas.
2. A programação utilizando o *React*, traz para o ambiente de desenvolvimento a velocidade que o *Angular* não disponibiliza, pois não traz na sua instalação todos os seus componentes disponíveis, para que o programador utilize de funcionalidades específicas, é necessário baixar de terceiros, essa opção agrega performance no projeto pois reduz o tamanho final do projeto significativamente.
3. O *Vue* é um *framework* que foi produzido com base no *template* do *Angular* e agrega os principais pontos positivos do *React* além de não deixar o projeto lento, ele possui uma interface amigável para escolher os componentes a ser utilizado no projeto a ser desenvolvido. (FRIAS, 2020)

> O ponto fraco do Vue é que ele ainda é novo, e a versão 2, que se consolidou entre os grandes frameworks, é mais recente ainda, isso traz insegurança aos clientes. O Vue também não tem nomes como Facebook e Google em seu rodapé, o que o torna “menos seguro” aos clientes. Mas entre os desenvolvedores Front-end, o Vue se destaca na frente dos outros.
> (FRIAS, 2020)

5.1.2 Front-end mobile

O próximo passo do foi a escolha da tecnologia para o desenvolvimento do *front-end* da aplicação *mobile*. Analisamos o *React Native, Flutter* e o PWA *(Progressive Web App)*.

O *Flutter* foi criado pela *Google*, e o *React Native* foi criado pelo *Facebook*. Ambos trabalham com o padrão *Single Page Application* (SPA), a aplicação roda inteiramente em apenas uma única página, quando a aplicação precisa atualizar

alguma informação o SPA não carrega toda a página, ele carrega apenas a informação que está sendo atualizada.

Segundo Lima (2020), o *Flutter* possui dois pontos principais que se diferenciam do *React Native*. O primeiro, é a linguagem utilizada no desenvolvimento, enquanto o *React Native* utiliza o *JavaScript*, o *Flutter* usa o *Dart*, uma linguagem criada pela *Google* em 2011, que tinha como objetivo substituir o *JavaScript*. O segundo ponto é a forma como ele se comunica com o dispositivo: além de utilizar as *API's* nativas dos dispositivos, o *Flutter* usa uma biblioteca de interface gráfica, também da *Google*, chamada *Skia*. Segundo o site da própria *Skia* (2002), alguns produtos que a utilizam são: *Google Chrome*, *Chrome OS*, *Android*, *Mozilla Firefox*.

Com *React Native* é possível desenvolver aplicativos utilizando *Javascript* e/ou *Typescript*. Por acessar *API's* nativas do aparelho, o *React Native* consegue executar as tarefas de uma forma muito mais rápida e performática que as tecnologias híbridas. (LIMA, Victor, 2020)

Ele consegue entregar os atributos característicos de cada dispositivo, como botões, *input's*, *datePickers*, etc, assim, dando uma maior impressão de estar usando um aplicativo de desenvolvimento nativo. (LIMA, 2020)

Já o PWA, possibilita a integração de *API's* dos dispositivos móveis no desenvolvimento de um aplicação *web*, tornando uma metodologia híbrida entre *web* e *mobile*. Ao se utilizar dessa tecnologia no desenvolvimento, conseguimos reduzir o tempo de produção, pois exclui a necessidade de se utilizar uma linguagem nativa.

No PWA o usuário tem a opção de adicionar um ícone do site na tela inicial do smartphone e acessar a aplicação, assim, é possível ter a mesma experiência de usabilidade que uma aplicação nativa possui.

5.1.3 Back-end

Para selecionar a tecnologia utilizada para efetuar a comunicação do *front-end* com o banco de dados, analisamos três linguagens: *Python, PHP e Node.js*.

5.1.3.1 Python

O *Python* é uma linguagem de programação de alto nível, orientada a objetos, dinâmica e multiuso. A sintaxe e a digitação dinâmica do *Python*, com natureza interpretada, o tornam uma linguagem ideal para scripts. (GURU99, 2020)

Ele suporta vários padrões de programação, incluindo programação orientada a objetos, programação funcional ou estilos de procedimentos. Além disso, é um interpretador, o que significa que não pode ser convertido em código legível por computador antes de ser executado em tempo de execução. (GURU99, 2020)

Permite a inclusão de módulos de baixo nível no interpretador *Python*. Estes módulos permitem que os programadores adicionem ou personalizem suas ferramentas. Fornece interfaces para todos os principais bancos de dados comerciais. Oferece tipos de dados dinâmicos de alto nível e suporta verificação de tipo dinâmico. Pode ser facilmente integrado com C, C++, COM, ActiveX, CORBA e Java. (GURU99, 2020)

Uma das principais vantagens de se utilizar *Python*, é que a linguagem é extremamente amigável para desenvolvedores iniciantes.

5.1.3.2 PHP

Segundo a própria documentação do PHP, ele é definido como: "é uma linguagem de *script open source* de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento *web* e que pode ser embutida dentro do HTML.".

Sua principal função é a criação de páginas dinâmicas. Ou seja, dependendo de determinadas informações ou condições inseridas pelo usuário, o PHP pode exibir uma mesma página, de formas diferentes.

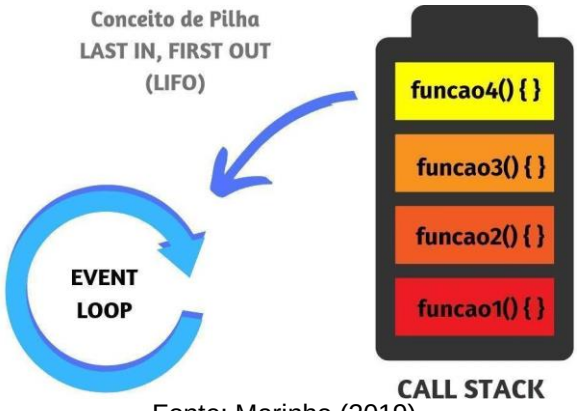
O PHP recebe constantemente atualizações contínuas. Os desenvolvedores dessa linguagem sempre agregam novos recursos e conceitos para proporcionar uma alta velocidade de execução.

5.1.3.3 Node.js

O *Node.js* é uma plataforma do lado do servidor que utiliza *JavaScript* no *back-end*. Foi construída em cima do motor de interpretação de *Javascript* do *Chrome*. Entre as principais características do *Node.js*, podemos citar:

A. *Event Loop*: uma arquitetura baseada em eventos. O *Node.js* está o tempo todo ouvindo eventos. Quando uma função (evento) é disparada ela entra na *Call Stack*, uma pilha de eventos. Todo código que entra nessa pilha, é executado no processo do *event loop*.

Figura 1 – Funcionamento do Event Loop e Call Stack



Fonte: Marinho (2019).

B. *Single Thread*: o *Node.js* sempre é executado em apenas um *core* do processador.
C. *Non-blocking I/O*: isso significa que o *Node.js* pode receber requisições a qualquer momento. (MARINHO, 2019)

O *Node.js* está em constante aprimoramento. Esta tecnologia também vem com mais uma vantagem. É composto por todas as funcionalidades atuais em uma distribuição. Essa é uma das grandes vantagens para os programadores que precisam gastar tempo zero na reformulação completa do código. (MARTIN, 2019)

O *Node.js* é mais adequado para programação assíncrona. Seu princípio de programação é o *JavaScript* puro, seus padrões de codificação não são limpos, não

sendo indicado para projetos de grande porte, já para aplicações *web* que possui execução em tempo real, o *Node.js* é ideal, o seu interpretador é o próprio *JavaScript* e suporta retorno de chamada. Sua programação é baseada em evento/retorno de chamada que o torna mais rápido. (GURU99, 2020)

5.1.4 Tecnologias escolhidas

No *front-end web* optamos por trabalhar com o *React.js*, principalmente por estarmos desenvolvendo um sistema simples e por ser um *framework* extremamente performático. No *back-end* utilizamos o *Node.js,* assim temos *web*, *mobile* e *back-end* usando a mesma linguagem de programação: *JavaScript*.

Para transformar a aplicação *web*, em mobile, no início do desenvolvimento do projeto, decidimos utilizar o *React Native*, mas durante o desenvolvimento percebemos que não havia necessidade, era possível fazer tudo com PWA, com um custo e esforço muito menor. Essa metodologia de desenvolvimento é vista como uma evolução híbrida entre páginas *web* e aplicativos móveis, possibilitando a comunicação entre ferramentas mobile que antes uma página *web* não tinha permissão de acesso, um exemplo prático é a utilização da câmera do smartphone.

Acreditamos que a principal vantagem que a escolha dessas tecnologias traz, é o fato de ter que dominar apenas uma linguagem de programação para conseguir atingir três ambientes diferentes. Uma outra vantagem que podemos citar é que segundo uma pesquisa realizada pelo *Stack Overflow* em 2020, pelo oitavo ano seguido o *JavaScript* foi a linguagem de programação mais popular entre os usuários do site. A pesquisa é realizada anualmente e no ano de 2020 teve 47.184 respostas de profissionais que atuam no mercado de desenvolvimento. Quase 70% do público do *Stack Overflow* utiliza o *JavaScript*. Por ter uma comunidade tão grande, é cada vez mais fácil encontrar bibliotecas, conteúdos sobre a linguagem que auxiliam no desenvolvimento de um projeto.

5.2 ESTRUTURA DE DADOS

Em uma aplicação orientada a objetos que possuem uma base de dados relacional, existe um problema de distanciamento de paradigmas entre as duas

frentes. Isso, em grande escala, torna a aplicação suscetível a inconsistências em relação a base de dados, ou vice e versa. (BESSA, 2020)

Devido a esse distanciamento de paradigmas entre uma aplicação OOP e a base de dados relacional, surge conceito de ORM, *Object-Relational Mapping*, que atua como um intermediador entre as duas frentes, mapeando os objetos e entidades da aplicação para a base de dados. (BESSA, 2020)

No *back-end*, o ORM que utilizamos para o mapeamento foi o *TypeORM,* que é focado em *Typescript*.

5.2.1 Entidades

No *TypeORM* uma entidade é responsável por fazer um mapeamento entre uma classe *TypeScript* e uma tabela de banco de dados. Para criar uma entidade é necessário adicionar a anotação *@Entity()* no topo da definição da classe e, para cada campo da classe, adicionar a anotação *@Column* (ou similar). Essas anotações fornecem ao *TypeORM* as informações necessárias para configurar a tabela do banco de dados.

Quadro 1 - Entidade Usuário

```
import {
  Entity,
  Column,
  PrimaryGeneratedColumn,
  CreateDateColumn,
  UpdateDateColumn,
  OneToOne,
  JoinColumn,
} from 'typeorm';
import { Exclude, Expose } from 'class-transformer';
import uploadConfig from '../config/upload';

import Address from './Address';

@Entity('users')
class User {
  @PrimaryGeneratedColumn('uuid')
  id: string;

  @Column()
  name: string;
```

```
  @Column()
  email: string;

  @Column()
  @Exclude()
  password: string;

  @Column()
  address_id: string;

  @OneToOne(() => Address) // 1 user tem apenas 1 endereco
  @JoinColumn({ name: 'address_id' })
  address: Address;

  @Column()
  whatsapp: string;

  @Column()
  pix: string;

  @Column()
  avatar: string;

  @Column()
  is_ong: boolean;

  @CreateDateColumn()
  created_at: Date;

  @UpdateDateColumn()
  updated_at: Date;

  @Expose({ name: 'avatar_url' })
  getAvatarUrl(): string | null {
    if (!this.avatar) {
      return null;
    }
    return `${process.env.APP_API_URL}/files/${this.avatar}`;
  }
}

export default User;
```

Fonte: Elaborado pelos autores (2021).

5.2.2 Migrations

Durante o desenvolvimento de uma aplicação é comum ter o surgimento de demandas de migração da base de dados com a aplicação, para que ambas estruturas estejam sincronizadas. Esse processo feito de forma manual, depende de uma comunicação estreita com a equipe de desenvolvimento para que não haja desencontros. (BESSA, 2020)

Uma das estratégias que muitos ORM's oferecem, para aperfeiçoar esse processo, é o uso de *migrations*. No *TypeORM* isso é possível através do seu CLI. As *migrations* aparecem em nossa estrutura relacional como uma tabela com esse nome. São esses arquivos que o *TypeORM* usa para criar, deletar ou alterar as tabelas no banco de dados.

Quadro 2 – Migration de Usuário

```
import {
  MigrationInterface,
  QueryRunner,
  Table,
  TableForeignKey,
} from 'typeorm';

export default class CreateUsers1598486614140 implements MigrationInterface {
  public async up(queryRunner: QueryRunner): Promise<void> {
    await queryRunner.createTable(
      new Table({
        name: 'users',
        columns: [
          {
            name: 'id',
            type: 'uuid',
            isPrimary: true,
            generationStrategy: 'uuid',
            default: 'uuid_generate_v4()',
          },
          {
            name: 'name',
            type: 'varchar',
          },
          {
            name: 'email',
            type: 'varchar',
            isUnique: true,
          },
```

```
        {
          name: 'password',
          type: 'varchar',
        },
        {
          name: 'address_id',
          type: 'uuid',
          isNullable: true,
        },
        {
          name: 'whatsapp',
          type: 'varchar',
        },
        {
          name: 'pix',
          type: 'varchar',
          isNullable: true,
        },
        {
          name: 'avatar',
          type: 'varchar',
          isNullable: true,
        },
        {
          name: 'is_ong',
          type: 'boolean',
          default: false,
        },
        {
          name: 'created_at',
          type: 'timestamp',
          default: 'now()',
        },
        {
          name: 'updated_at',
          type: 'timestamp',
          default: 'now()',
        },
      ],
    }),
  );

  await queryRunner.createForeignKey(
    'users',
    new TableForeignKey({
      name: 'UserAddress',
      columnNames: ['address_id'],
      referencedColumnNames: ['id'],
      referencedTableName: 'addresses',
      onDelete: 'SET NULL',
```

```
      onUpdate: 'CASCADE',
    }),
  );
}

public async down(queryRunner: QueryRunner): Promise<void> {
  await queryRunner.dropForeignKey('users', 'UserAddress');
  await queryRunner.dropTable('users');
}
```

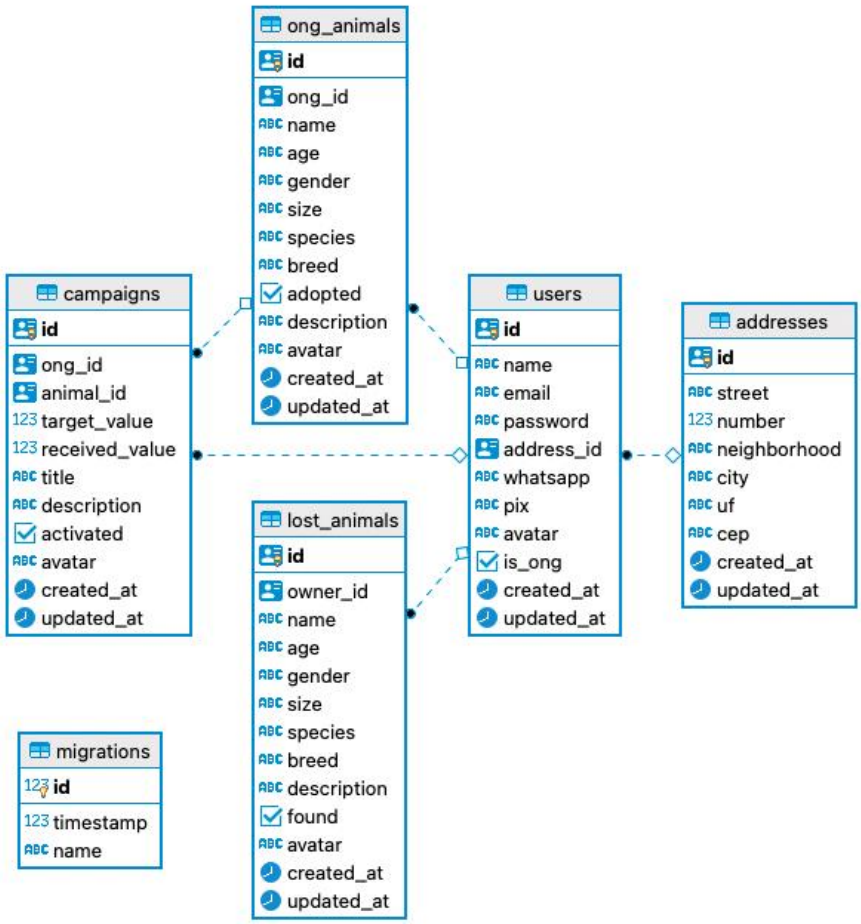
Fonte: Elaborado pelos autores (2021).

5.2.3 Diagrama entidades relacionamento (ER)

Um diagrama ER é um tipo de fluxograma que ilustra como “entidades” se relacionam entre si dentro de um sistema. No Pet’s Hero, por exemplo, temos um relacionamento entre a tabela *users* (coluna *id*) e *ong_animals* (coluna *ong_id*).

Utilizamos uma chave primária em cada tabela, para servir de identidade única e através dessa chave, é possível definir os relacionamentos entre as tabelas, assim conseguimos otimizar a quantidade de tabelas que foram utilizadas no banco de dados.

Figura 2 – Diagrama ER



Fonte: Elaborado pelos autores (2021).

5.3 AUTENTICAÇÃO

Para o sistema de login do Pet's Hero, optamos por desenvolver uma autenticação com *JSON Web Token* (JWT).

O JWT é um método padrão da indústria, utilizado para realizar autenticação entre duas partes por meio de um token assinado que autentica uma requisição *web*. O token não é criptografado, ele é apenas assinado, dessa forma somente o servidor (que possui a chave) pode verificar se a assinatura é autêntica. O fluxo desse tipo de autenticação funciona da seguinte forma:

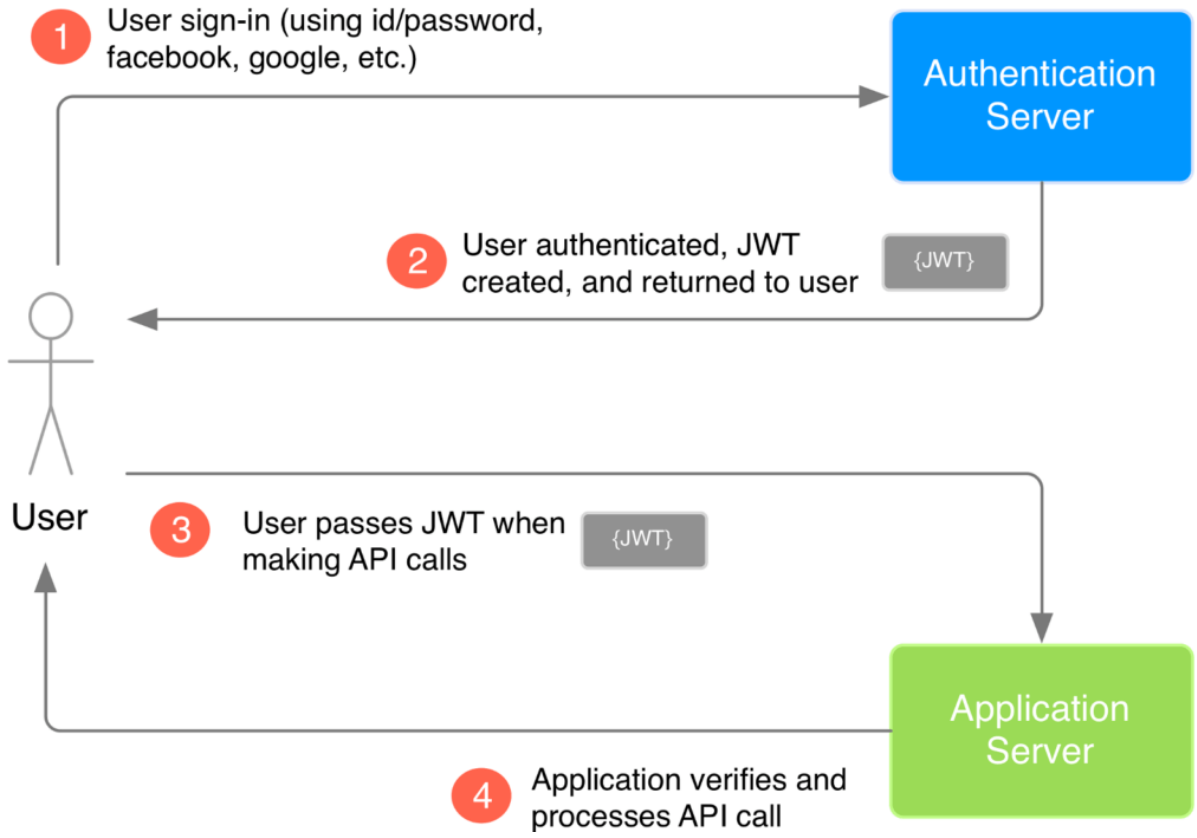
1- Usuário faz uma requisição de login;

2- Se login e senha estiverem corretos, a aplicação gera e retorna um JWT;

3- Com o token gerado, o usuário pode fazer requisições à rotas privadas;

4- A aplicação verifica se o JWT enviado é válido. Se for válido, retorna o que foi requisitado. Se for inválido, retorna um erro.

Figura 3 – Diagrama de fluxo de autenticação



Fonte: Duarte (2020).

Para fazer a verificação do token nas rotas privadas, criamos um *middleware*. De acordo com Janones (2017), "*Middleware* no *Node* é todo o tipo de função que está entre um pedido HTTP e a resposta final que o servidor envia de volta para o cliente".

Com o *middleware* de autenticação criado, todas as requisições em rotas privadas, ou seja, rotas que só podem ser acessadas por usuários logados, primeiro executam as funções desse *middleware*, antes de executar a função solicitada. É no *middleware* que fazemos as verificações se o token de autenticação está presente, e se é um token válido. Caso o token não exista, esteja vencido, ou inválido por qualquer outro motivo, a requisição retorna um erro. Caso exista um token válido, o usuário está logado e pode fazer as requisições normalmente nesta rota privada.

Quadro 3 – Middleware de autenticação

```
import { Request, Response, NextFunction } from 'express';
import { verify } from 'jsonwebtoken';

import authConfig from '../config/auth';

interface TokenPayload {
  iat: number;
  exp: number;
  sub: string;
}
export default function ensureAuthenticated(
  request: Request,
  response: Response,
  next: NextFunction,
): void {
  // Validação do token JWT
  const authHeader = request.headers.authorization;

  if (!authHeader) {
    throw new Error('JWT token is missing.');
  }

  // Dividir o token em 2 -> formato normal: Bearer token
  const [, token] = authHeader.split(' ');

  try {
    const decoded = verify(token, authConfig.jwt.secret);

    const { sub } = decoded as TokenPayload;

    // Pode ser usado em qualquer rota
    request.user = {
      id: sub,
    };

    return next();
  } catch (err) {
    throw new Error('Invalid JWT token.');
  }
}
```

Fonte: Elaborado pelos autores (2021).

5.4 DOAÇÕES

A ideia inicial era ter uma forma de fazer doações às instituições diretamente pelo *site*. Essa funcionalidade chegou a ser criada no *back-end*, para isso utilizamos a *API* do Mercado Pago. Mas, durante o desenvolvimento nos deparamos com alguns problemas que nos fizeram desistir de implementar essa funcionalidade no *front-end*:

- A *API* do Mercado Pago tem uma taxa de recebimento de pagamentos que varia entre 0,99% a 4,99%.
- Todas as instituições cadastradas no Pet's Hero teriam que ter uma conta e uma chave para utilizar a *API* do Mercado Pago ou todas as doações caíram na nossa conta e só depois poderíamos distribuir as doações para as instituições.

Por se tratar de um projeto que visa ajudar e facilitar os processos das instituições, acreditamos que essa solução não seria a ideal.

Como alternativa para esse problema, escolhemos utilizar o PIX. Ao criar uma conta do tipo ONG, a instituição pode cadastrar sua chave PIX, que ficará visível para que todos os usuários possam ver e doar. Um problema que essa alternativa trouxe foi que dessa forma não temos como saber quanto cada ONG recebeu através do *site*. Para contornar isso, deixamos liberado para que as ONG's possam atualizar os valores recebidos em cada campanha criada.

No capítulo a seguir, mostraremos todas as funcionalidades da aplicação.

6 FUNCIONALIDADES

Nesse capítulo, apresentaremos cada uma das funcionalidades da aplicação, com descrições detalhadas de cada função e capturas de tela.

6.1 CRIAÇÃO DE CONTA

Para conseguir acessar o *site* Pet's Hero, o usuários precisam fazer um cadastro no sistema.

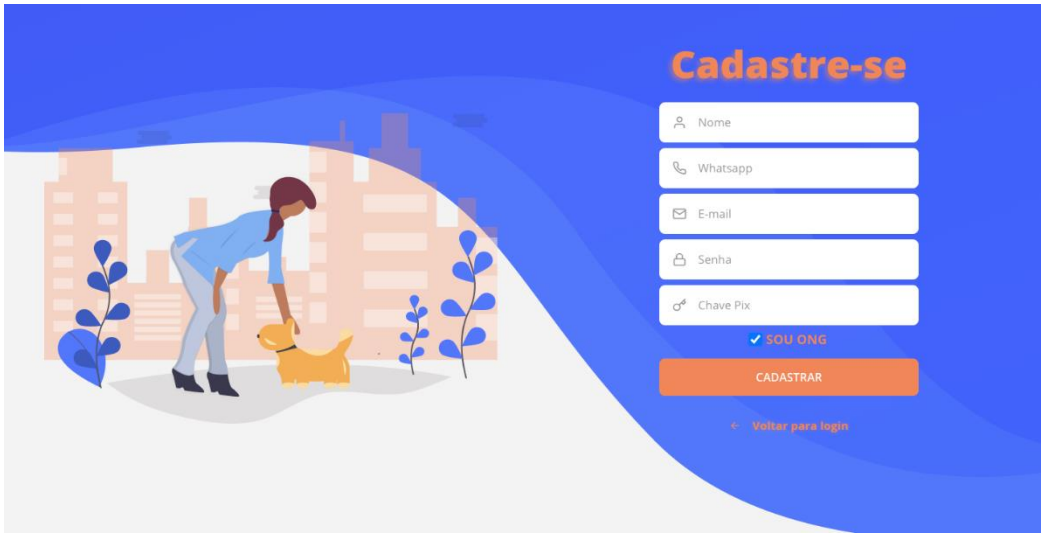
Ao clicar em criar conta, o usuário é redirecionado a página onde serão solicitadas algumas informações para efetuar o cadastro.

Na página de cadastro, o usuário possui a opção de se identificar como uma instituição, nesse caso ele também deve informar a sua chave PIX, caso a opção não seja selecionada, será considerado um usuário do tipo "doador".

Figura 4 – Cadastro de usuário do tipo doador

Fonte: Elaborada pelos autores (2021).

Figura 5 – Cadastro de usuário do tipo ONG

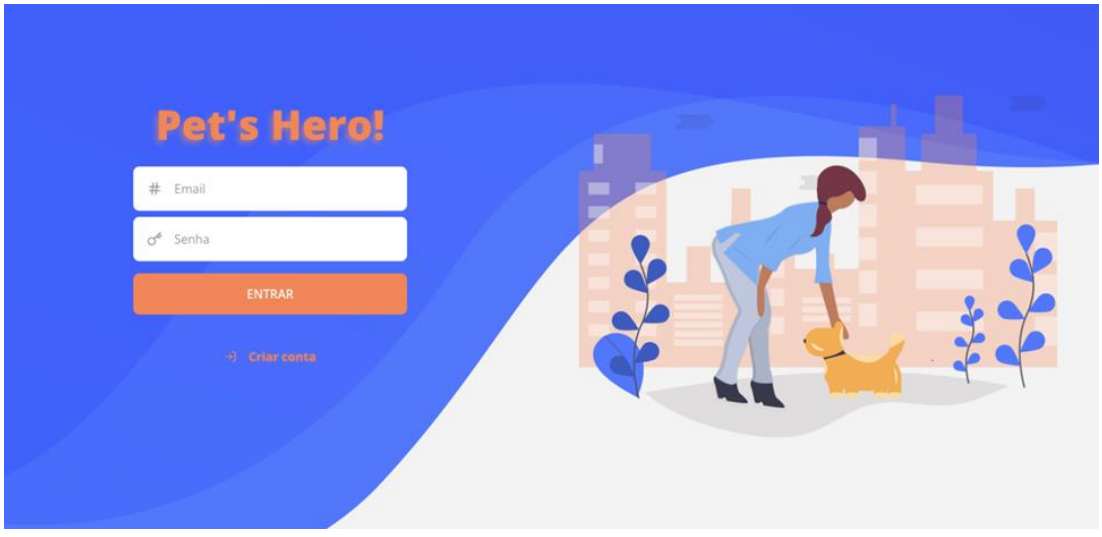


Fonte: Elaborada pelos autores (2021).

6.2 LOGIN

Após concluir o cadastro, o usuário tem permissão para acessar o sistema com e-mail e senha.

Figura 6 – Tela de login

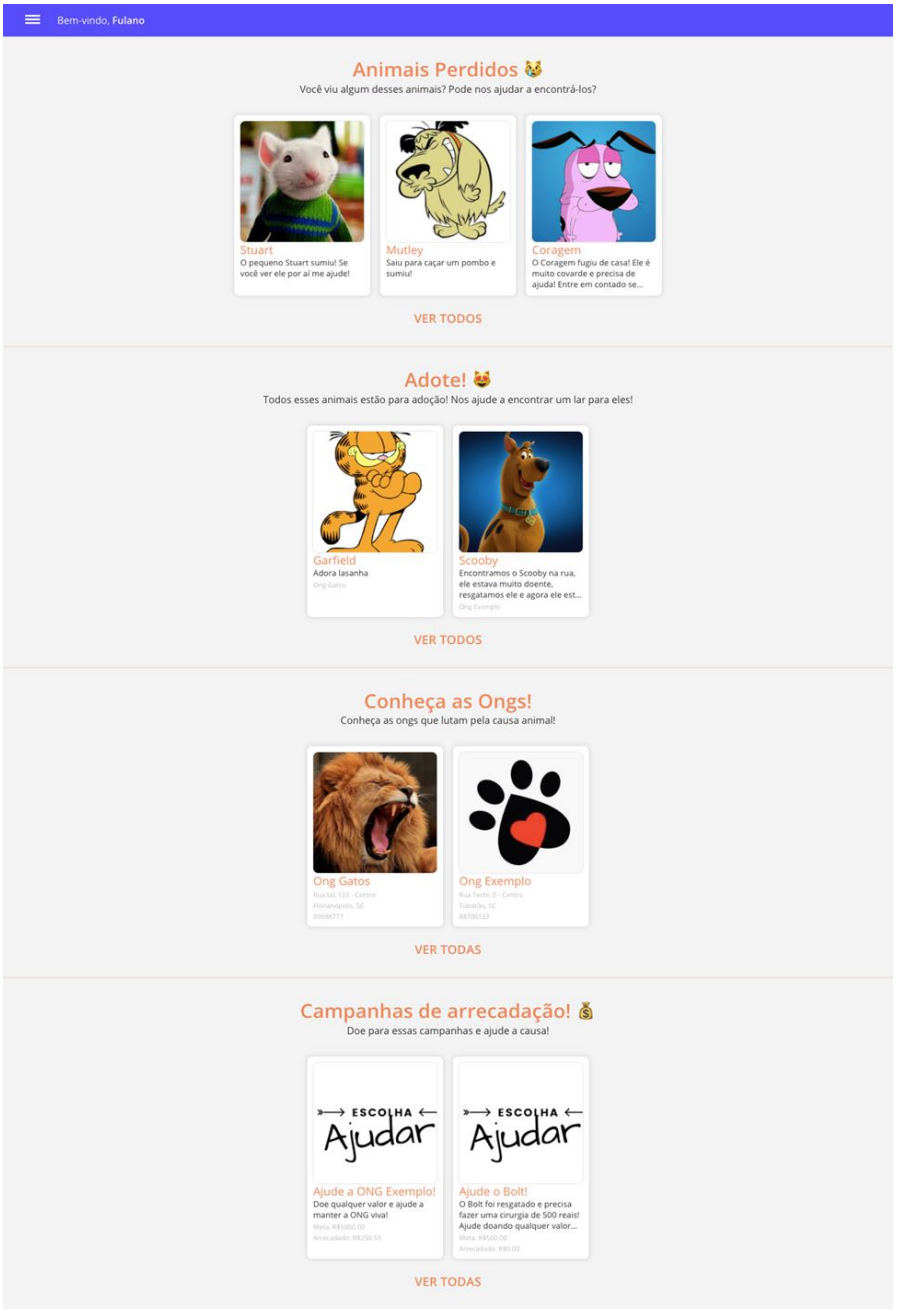


Fonte: Elaborada pelos autores (2021).

6.3 PÁGINA INICIAL

Ao acessar o sistema os usuários podem ver as publicações feitas, como, mural de animais perdidos, animais que estão para adoção, perfil das ONG'S cadastradas e campanhas de arrecadação onde é possível fazer as doações através do PIX cadastrado ou entrar em contato com a instituição.

Figura 7 – Página inicial



Fonte: Elaborada pelos autores (2021).

6.4 MENUS

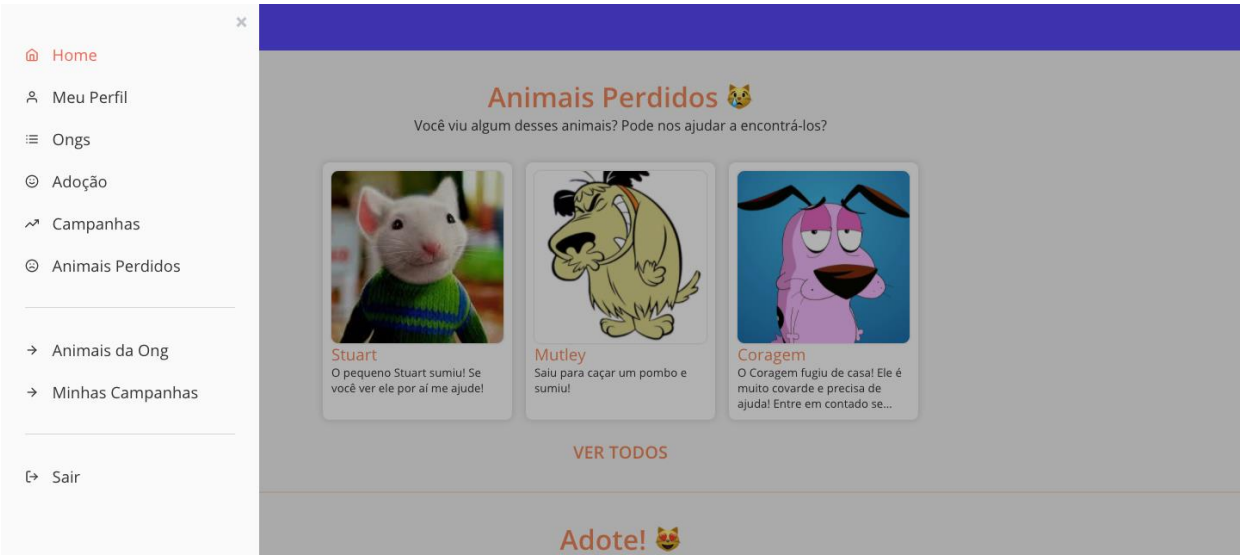
Clicando no ícone de menu no canto superior esquerdo, as opções de navegação do *site* serão abertas.

Através do menu, o usuário tem acesso a todas as páginas do sistema, dependendo do tipo de usuário (ONG ou doador). A seguir mostraremos as diferenças entre as funções que estão disponíveis para cada perfil.

6.4.1 Menu do usuário do tipo ONG

O menu de uma ONG apresenta duas páginas exclusivas, a primeira página é dos "Animais da ONG", onde a instituição poderá visualizar, cadastrar e editar seus animais que estão para adoção ou já foram adotados. A segunda página é "Minhas Campanhas", nela a instituição poderá visualizar, cadastrar ou editar campanhas de arrecadação onde mostrará a necessidade da instituição a fim de levantar recursos por meio de doações.

Figura 8 – Menu do usuário (ONG)

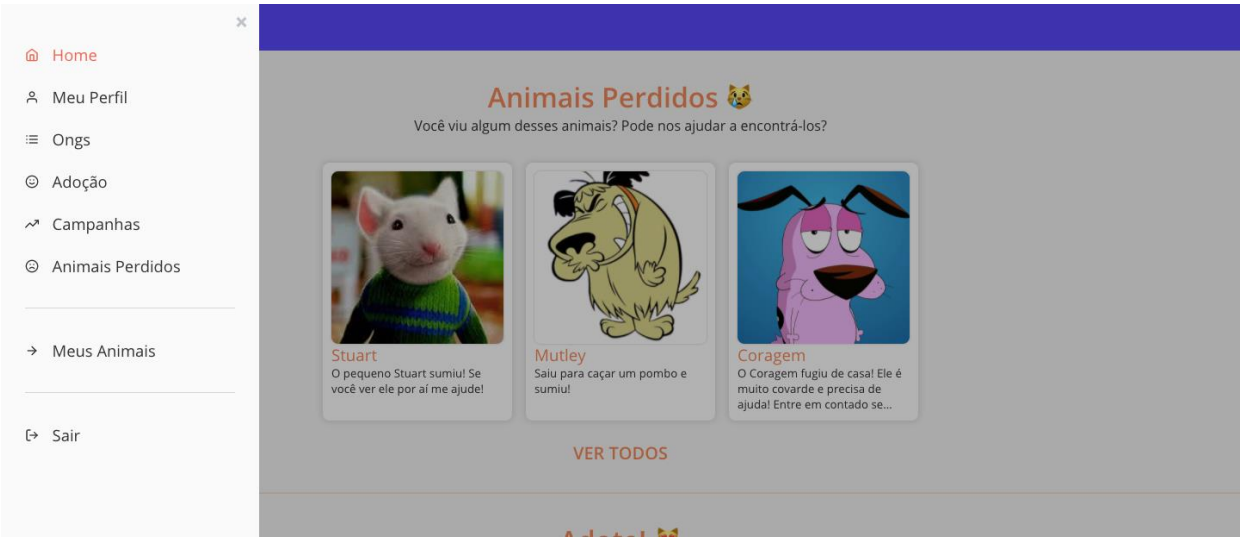


Fonte: Elaborada pelos autores (2021).

6.4.2 Menu do usuário do tipo doador

Para os usuários que não são ONG's, as primeiras páginas que aparecem no menu lateral são as mesmas. Esse tipo de usuário tem apenas uma página exclusiva, chamada "Meus Animais", onde poderá visualizar, cadastrar ou editar informações de animais perdidos.

Figura 9 – Menu do usuário (doador)



Fonte: Elaborada pelos autores (2021).

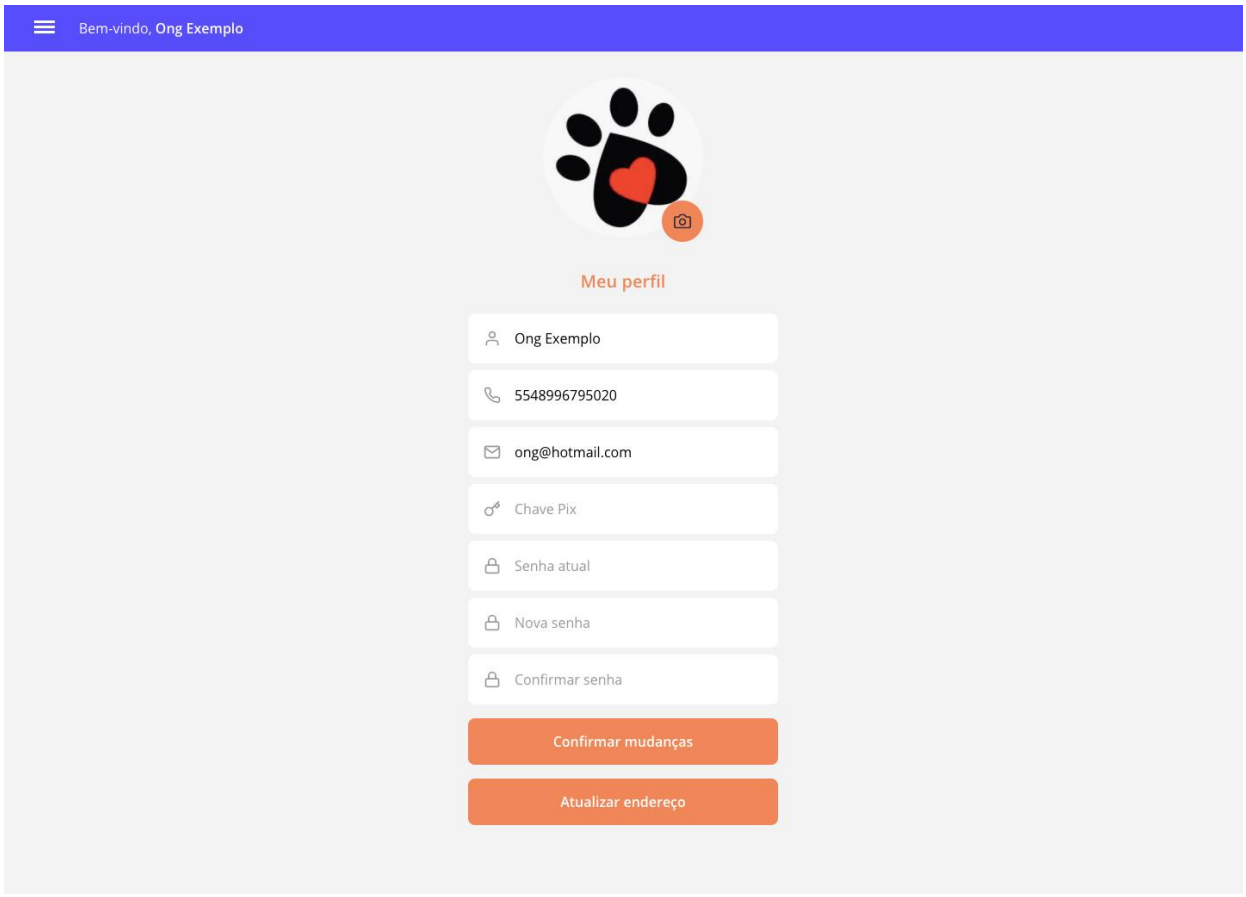
6.5 ALTERAÇÃO DAS INFORMAÇÕES DO PERFIL

Ao navegar até a página do perfil, o usuário pode alterar as informações de cadastro.

6.5.1 Informações do perfil da ONG

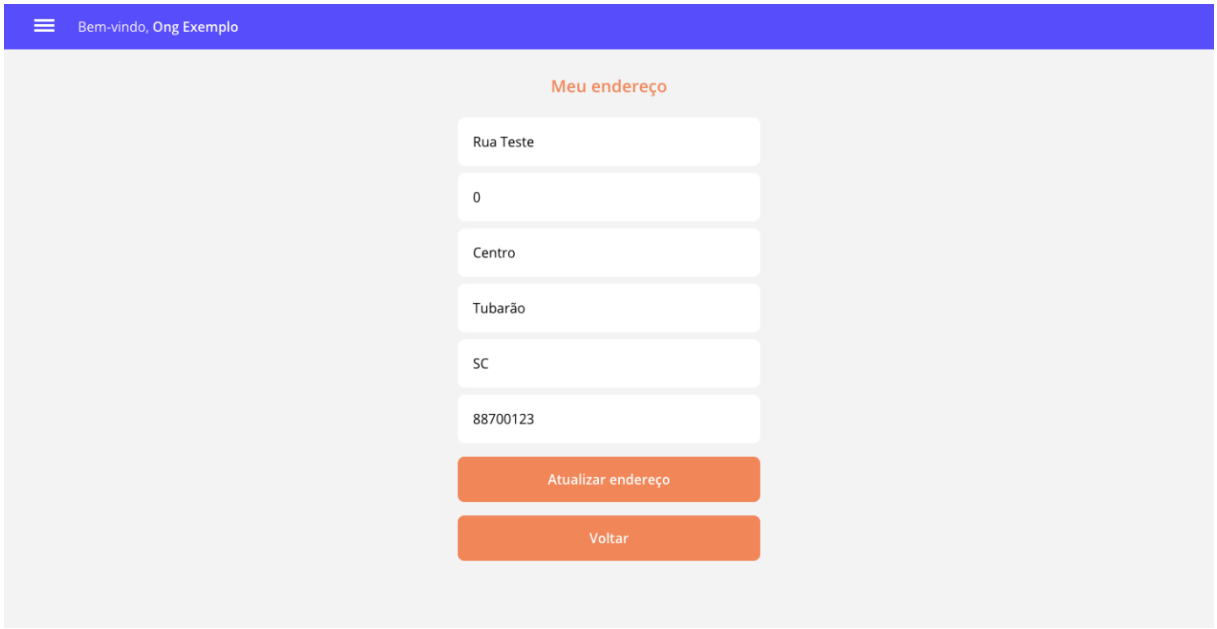
É possível alterar todas as informações do usuário, tais como: nome, telefone, e-mail, PIX, senha e endereço.

Figura 10 – Meu perfil (ONG)



Fonte: Elaborada pelos autores (2021).

Figura 11 – Cadastro de endereço

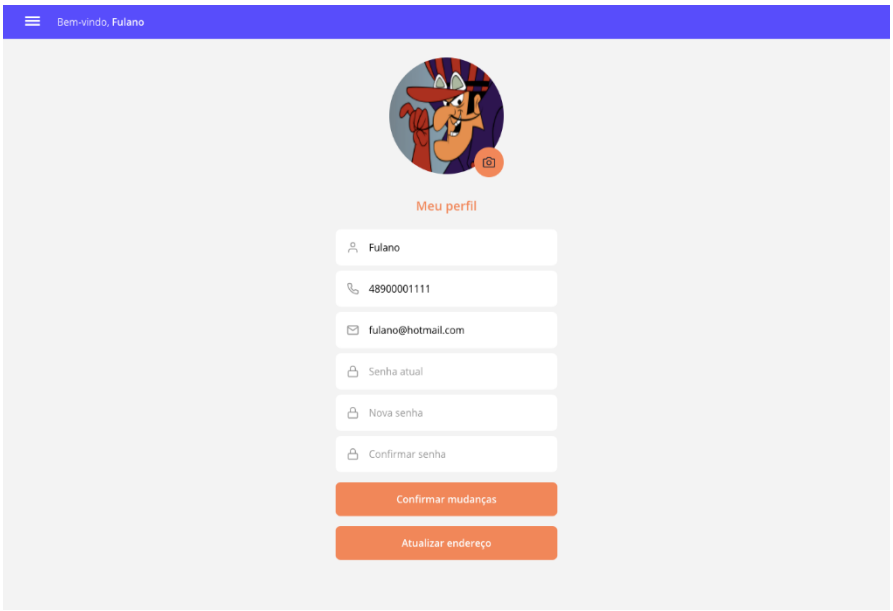


Fonte: Elaborada pelos autores (2021).

6.5.2 Informações do perfil do doador

O perfil do doador possui as mesmas informações do perfil da ONG, exceto a chave PIX.

Figura 12 – Meu perfil (doador)

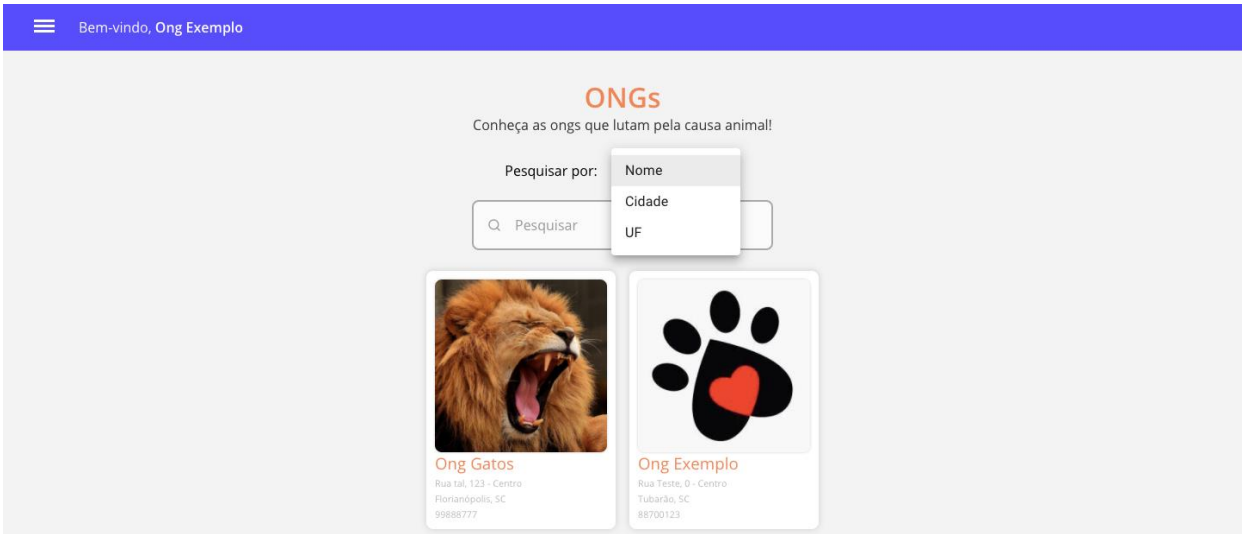


Fonte: Elaborada pelos autores (2021).

6.6 ONG'S CADASTRADAS

Acessando o menu lateral e clicando no item “Ong's”, é possível ver todas as instituições cadastradas no *site* Pet’s Hero. Os usuários podem filtrar as ONG's por: nome, cidade ou estado.

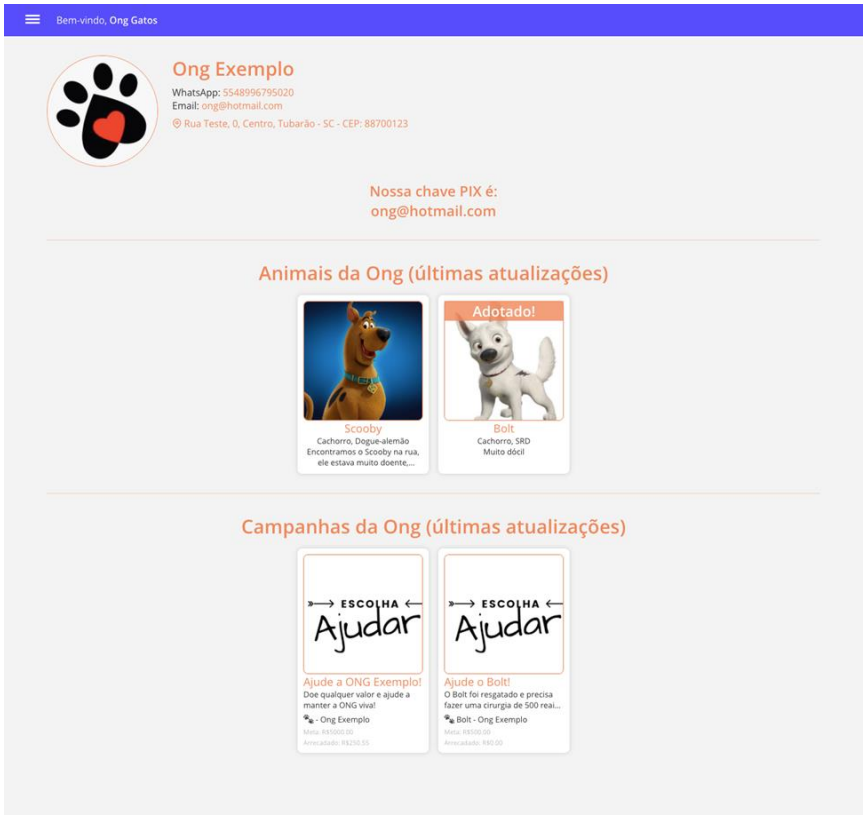
Figura 13 – Lista de ONG's cadastradas



Fonte: Elaborada pelos autores (2021).

Ao clicar na ONG, é possível ver as informações da mesma, como: nome, e-mail, PIX, contato, endereço e suas últimas postagens de animais para adoção e campanhas em andamento.

Figura 14 – Detalhes da ONG



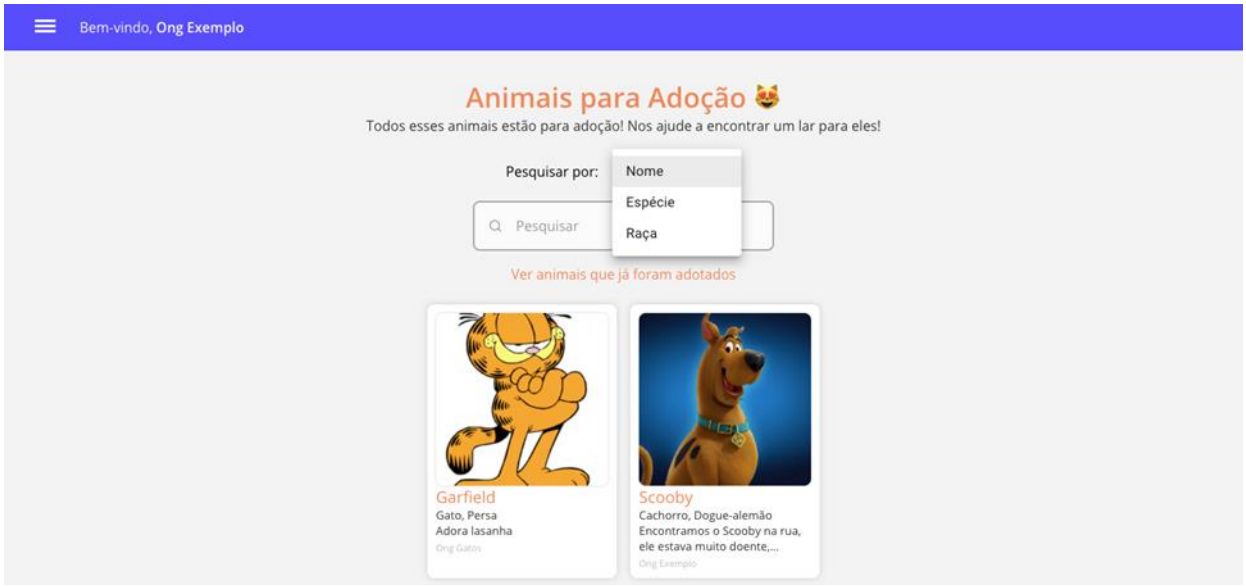
Fonte: Elaborada pelos autores (2021).

6.7 MURAL DE ADOÇÃO

Ao acessar o item "Adoção" do menu, o *site* mostra todos os animais que estão aguardando serem adotados, é possível personalizar a busca com alguns filtros disponíveis: nome, espécie e raça.

Cada animal cadastrado no mural de adoção, é apresentado com uma imagem, nome, raça, uma breve descrição, e o nome da instituição que o cadastrou.

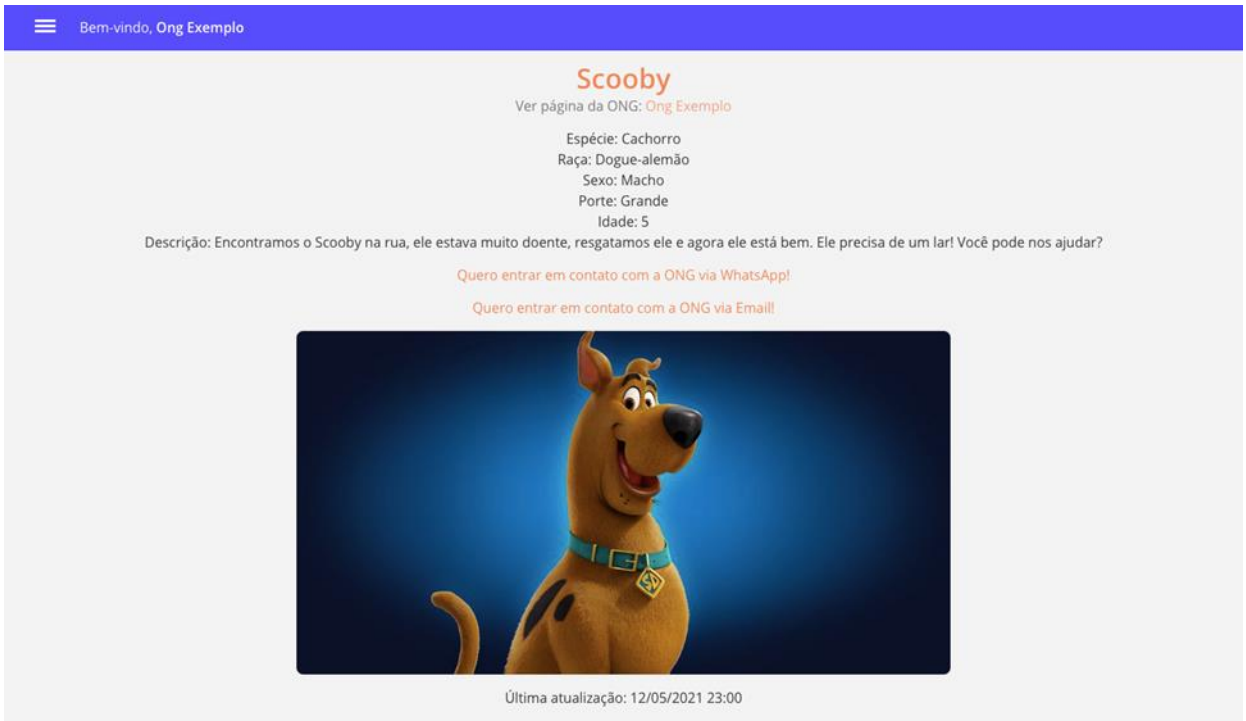
Figura 15 – Lista de animais para adoção



Fonte: Elaborada pelos autores (2021).

Ao clicar em um animal disponível para adoção, o *site* mostra todas as informações. O usuário poderá entrar em contato através do *WhatsApp* ou e-mail da instituição através do link disponível.

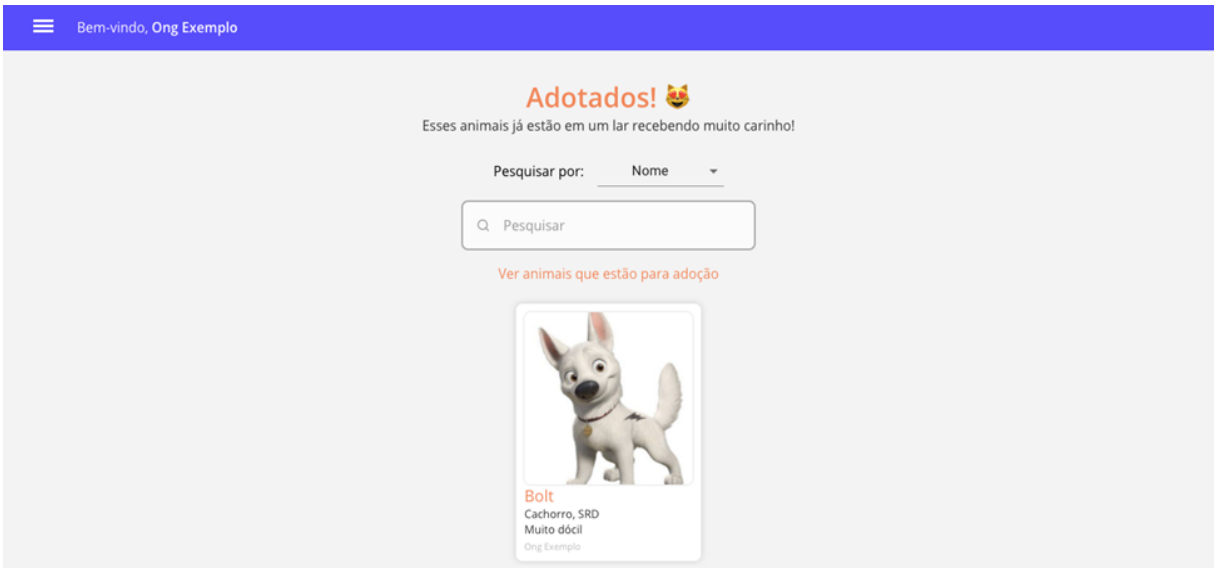
Figura 16 – Detalhes do animal para adoção



Fonte: Elaborada pelos autores (2021).

Ainda no mural de adoção, abaixo da barra de pesquisa, existe um link que possibilita filtrar somente os animais que já foram adotados, podendo ser filtrados da mesma forma mencionada anteriormente.

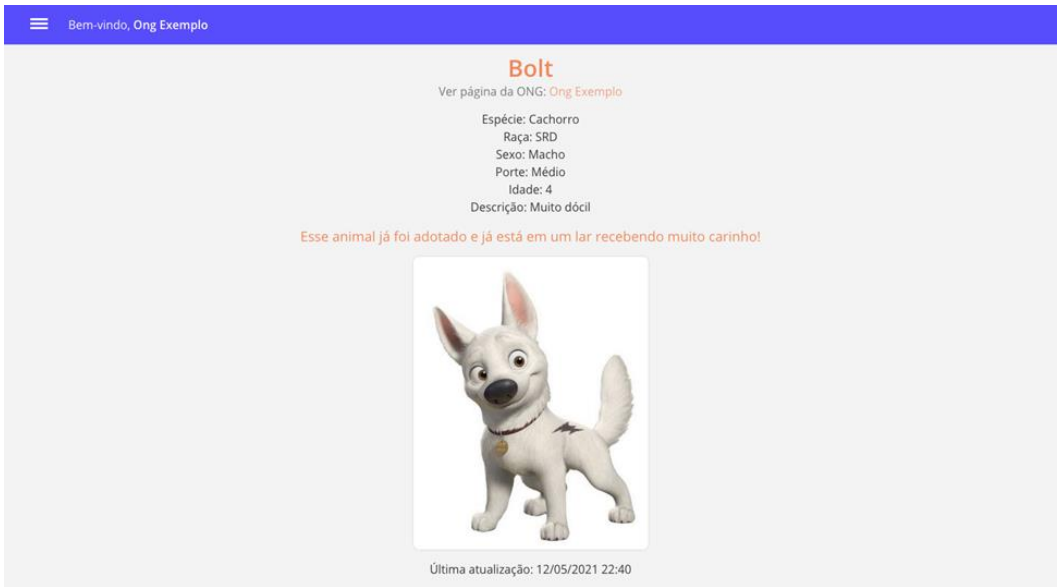
Figura 17 – Lista de animais adotados



Fonte: Elaborada pelos autores (2021).

Ao clicar sobre um animal que já foi adotado, o *site* mostrará uma página com os detalhes do animal, o nome da instituição onde ele estava antes de ser adotado e uma mensagem dizendo que animal já encontrou um lar.

Figura 18 – Detalhes do animal adotado



Fonte: Elaborada pelos autores (2021).

6.8 CAMPANHAS

Através do menu lateral também é possível acessar uma página com todas as campanhas cadastradas. Nesta página as instituições apresentam as suas necessidades para levantar recursos. As campanhas podem ser feitas para ajudar um animal específico ou não.

As campanhas podem ser filtradas por título, nome da ONG ou caso a campanha seja para um determinado animal, é possível filtrar pelo seu nome.

Figura 19 – Lista de campanhas

Fonte: Elaborada pelos autores (2021).

Ao clicar em uma das campanhas, o usuário é direcionado para a página com todos os detalhes: título, descrição, meta de arrecadação, valor arrecadado, nome da ONG, forma que o doador pode contribuir e caso a campanha seja para um animal específico, também mostra as informações desse animal.

Figura 20 – Detalhes de uma campanha sem animal atrelado

Fonte: Elaborada pelos autores (2021).

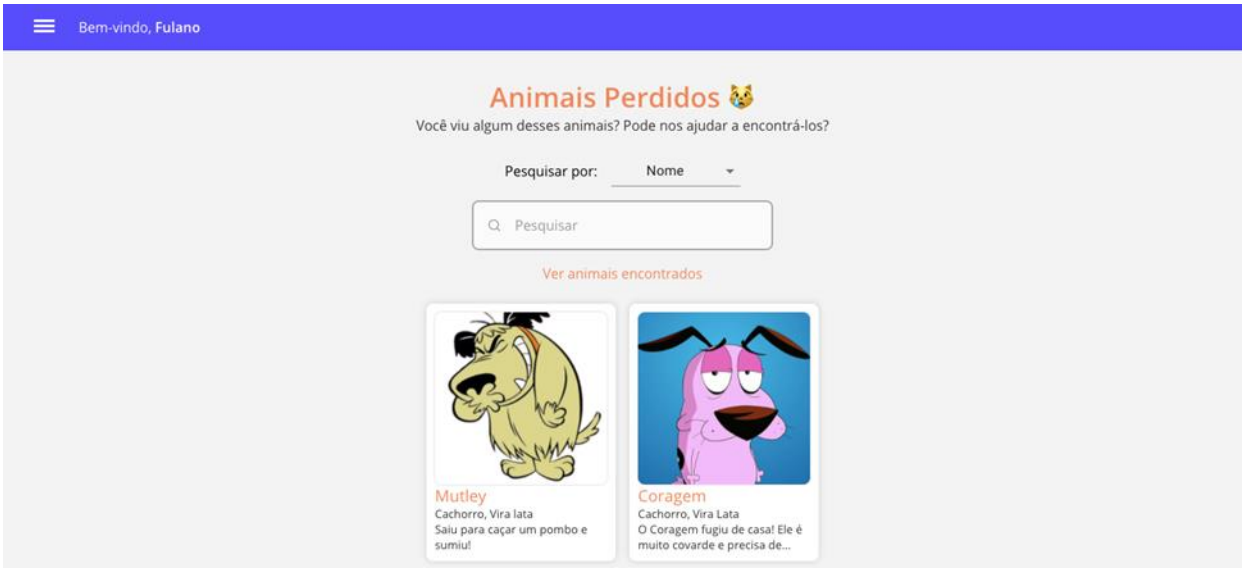
Figura 21 – Detalhes de uma campanha com animal atrelado

Fonte: Elaborada pelos autores (2021).

6.9 ANIMAIS PERDIDOS

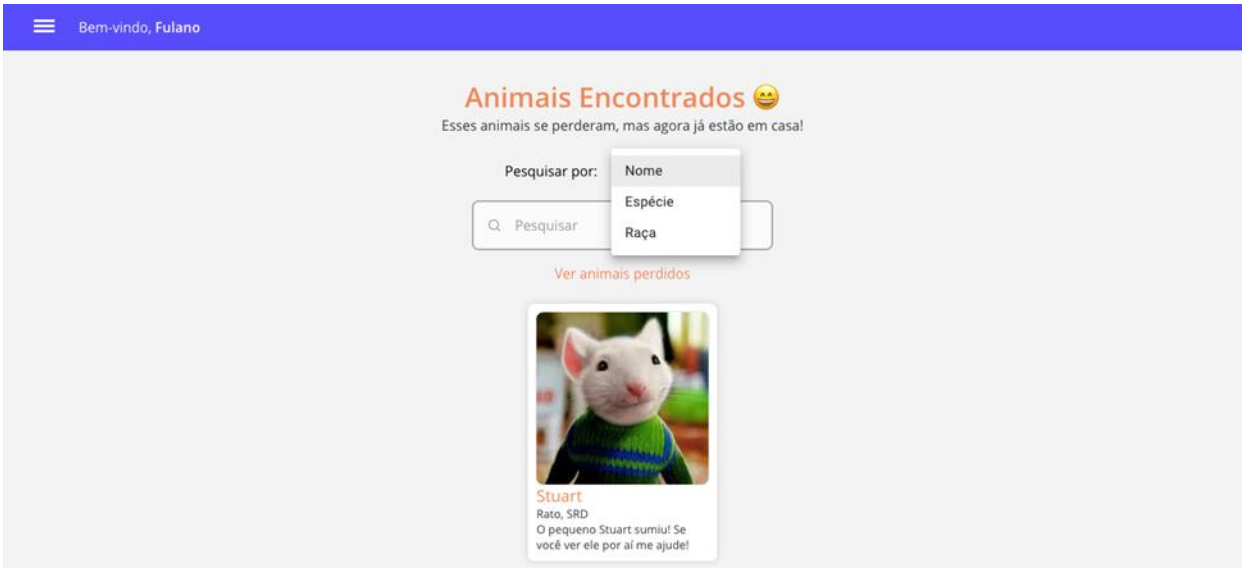
Na página de animais perdidos, há um filtro por nome, espécie ou raça. Também existe um link para mostrar apenas os animais já encontrados.

Figura 22 – Lista de animais perdidos



Fonte: Elaborada pelos autores (2021).

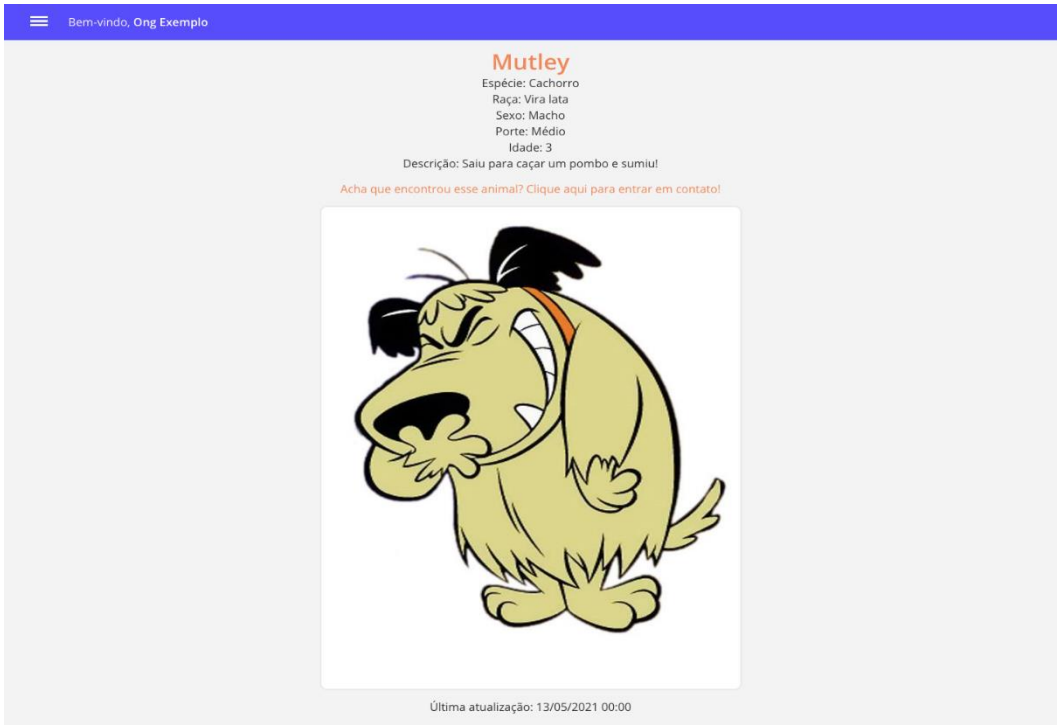
Figura 23 – Lista de animais encontrados



Fonte: Elaborada pelos autores (2021).

Ao clicar em um animal perdido, apresentadas as informações do animal e um link para entrar em contato com o dono.

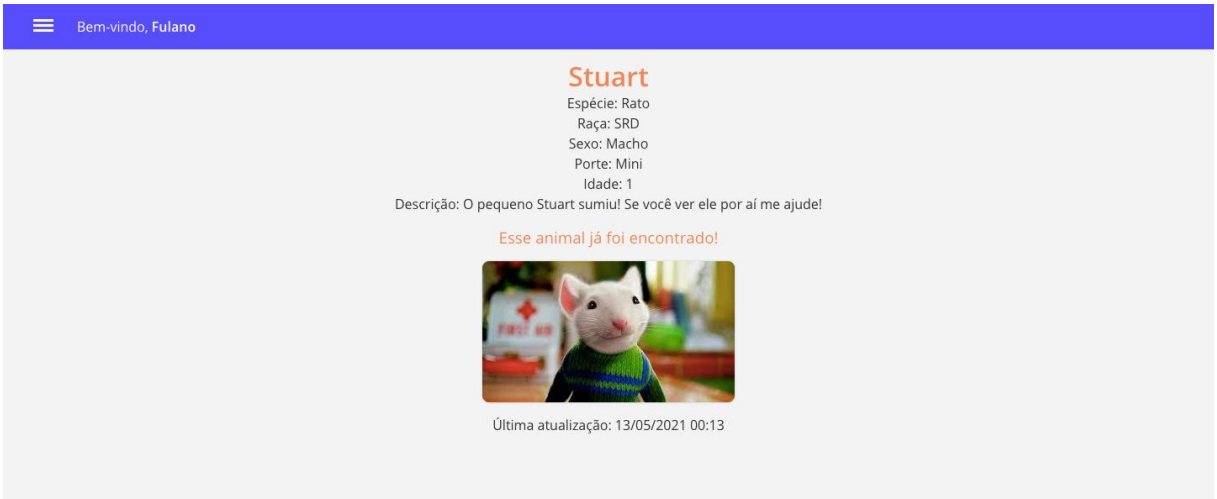
Figura 24 – Detalhes de um animal encontrado



Fonte: Elaborada pelos autores (2021).

Ao clicar em um animal encontrado, são apresentadas as informações do animal e uma mensagem dizendo que o animal já foi encontrado.

Figura 25 – Detalhes de um animal perdido



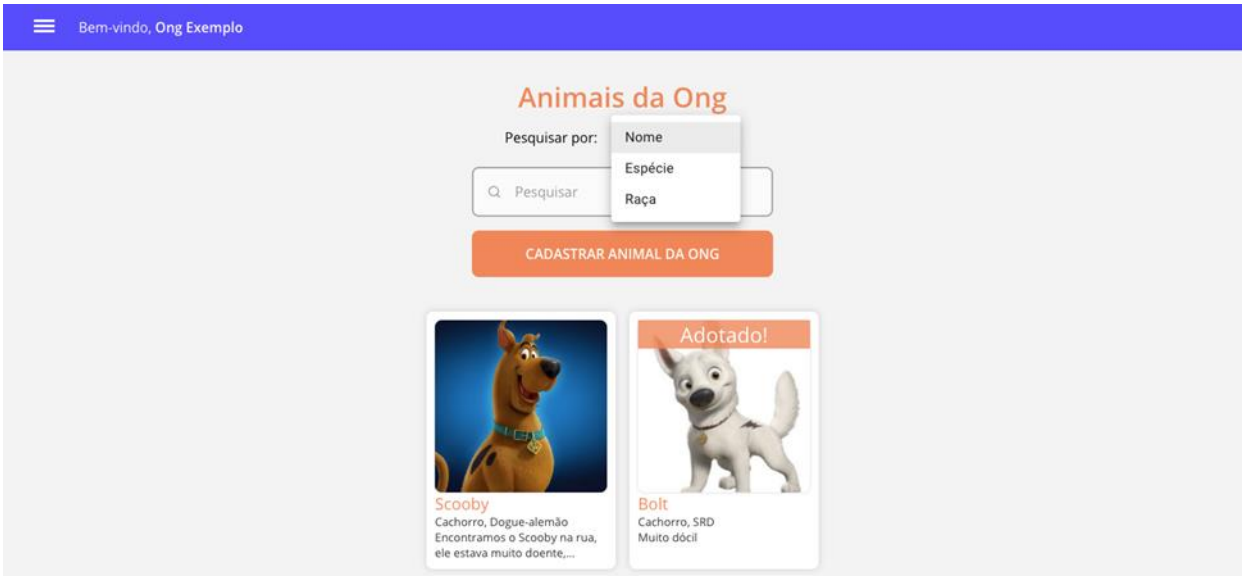
Fonte: Elaborada pelos autores (2021).

6.10 PÁGINAS EXCLUSIVAS DAS ONG'S

6.10.1 Animais da ONG

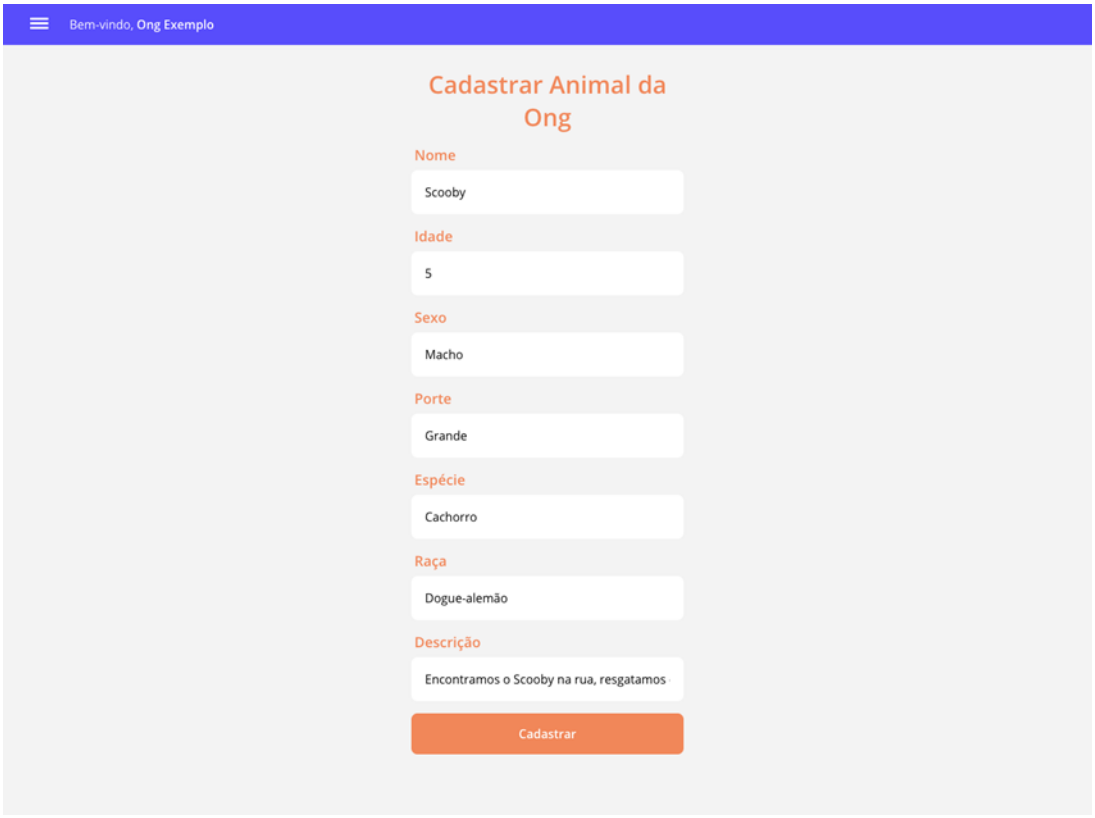
A ONG pode visualizar todos os seus animais adotados ou que ainda estão para adoção. Também pode cadastrar novos animais, e alterar informações de animais já cadastrados. Na edição de um animal, é possível alterar o seu status (adotado ou para adoção).

Figura 26 – Lista de animais cadastrados pela ONG



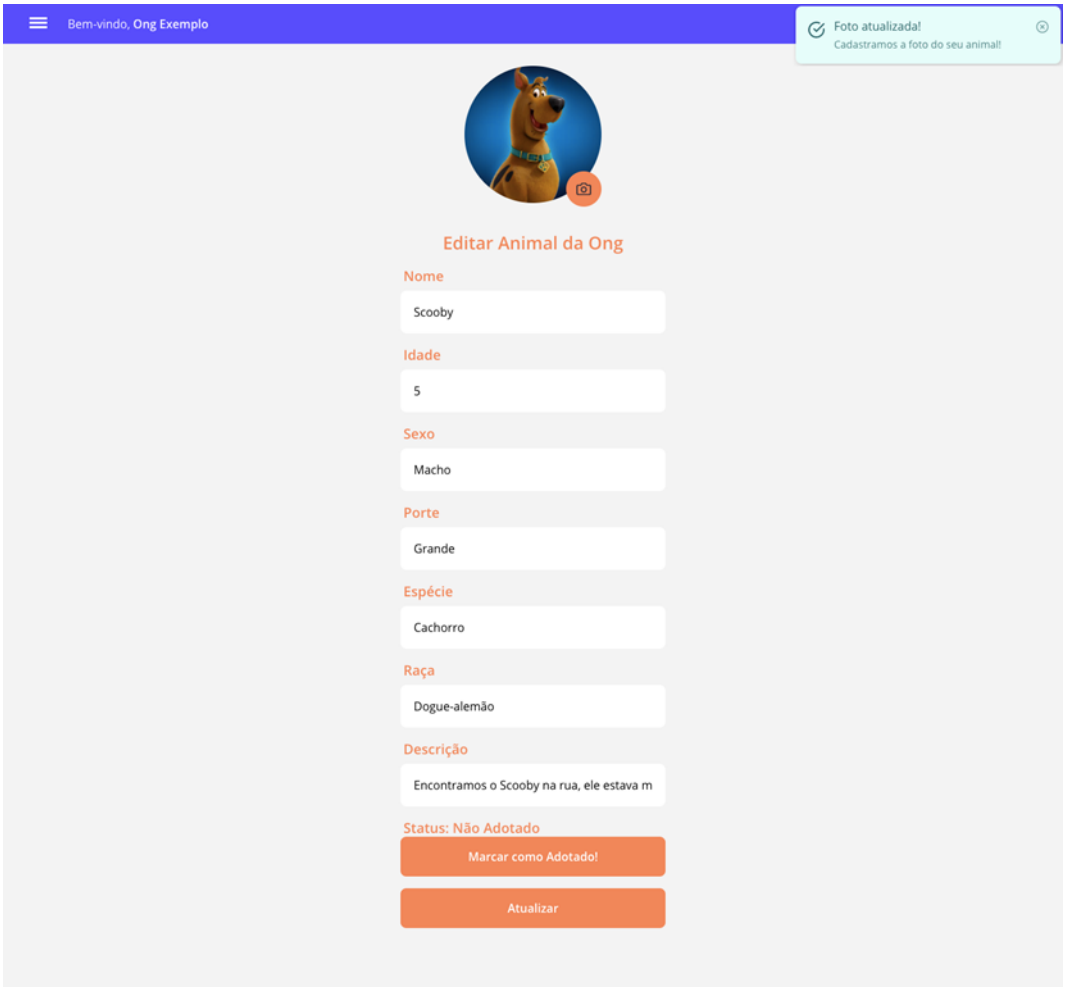
Fonte: Elaborada pelos autores (2021).

Figura 27 – Cadastro de animais da ONG



Fonte: Elaborada pelos autores (2021).

Figura 28 – Edição de cadastro de animal da ONG



Fonte: Elaborada pelos autores (2021).

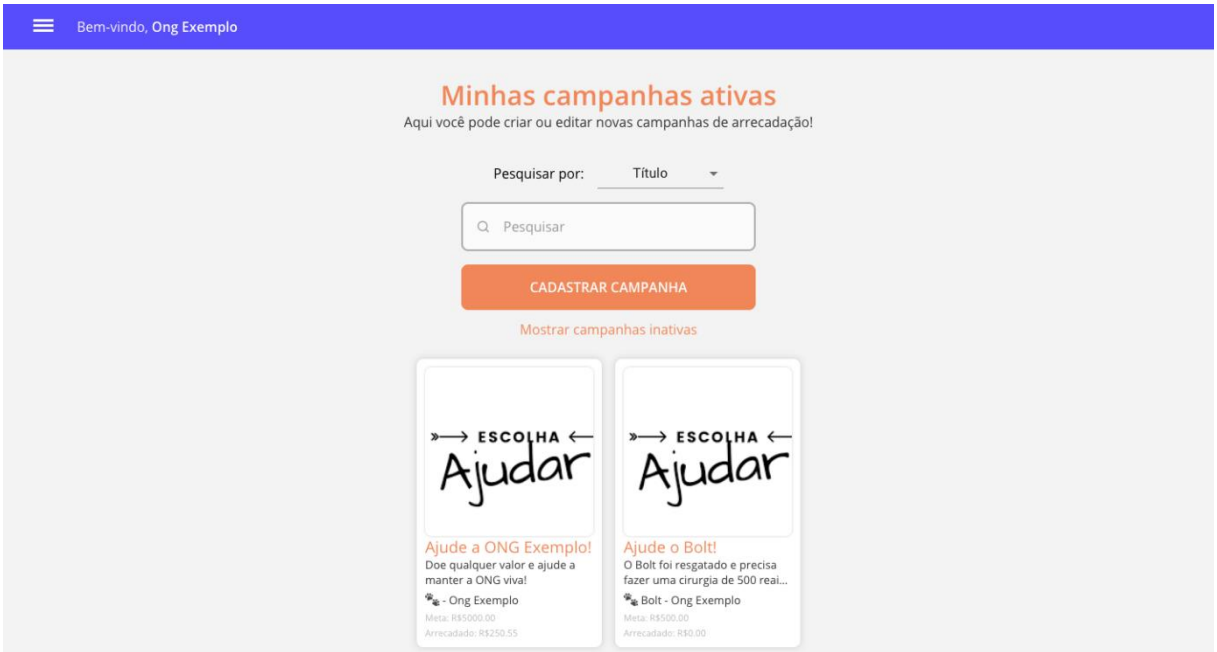
6.10.2 Campanhas da ONG

Para que as instituições tenham condições de fazer os seus trabalhos, são necessários recursos monetários, e isso só é possível através das doações.

Na página “Minhas campanhas” a ONG pode cadastrar, editar e visualizar todas as suas campanhas de arrecadação.

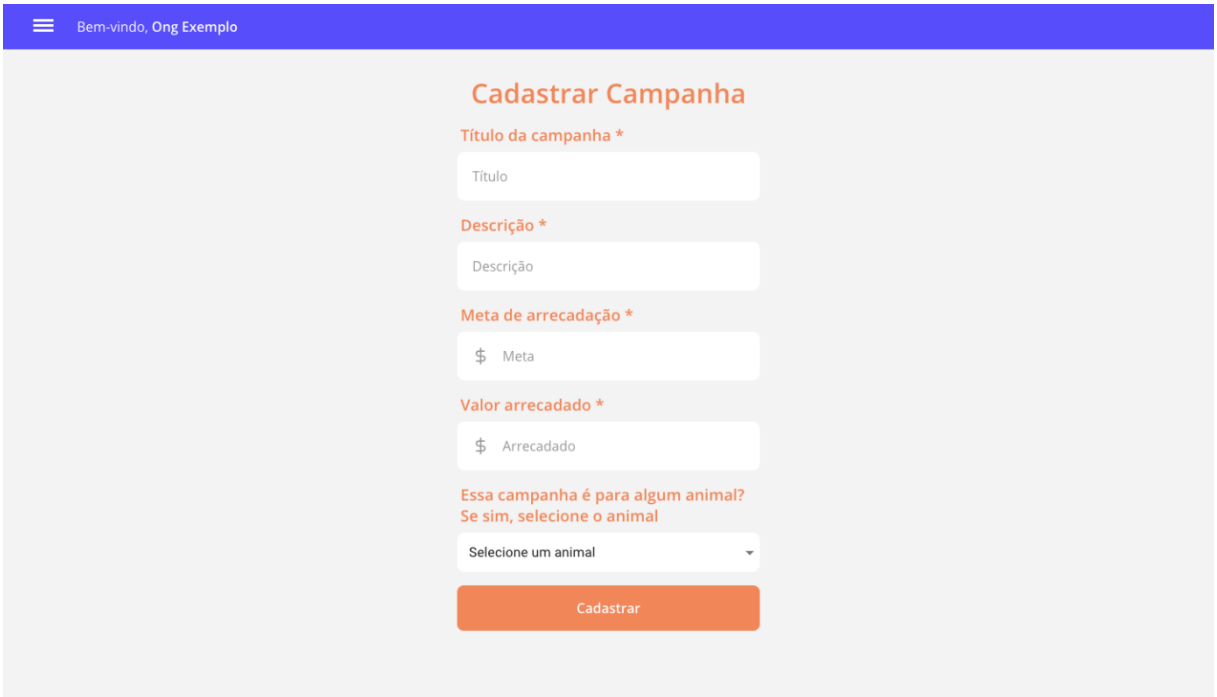
Essas campanhas podem ser criadas para ajudar um animal específico ou não.

Figura 29 – Lista de campanhas cadastradas pela ONG



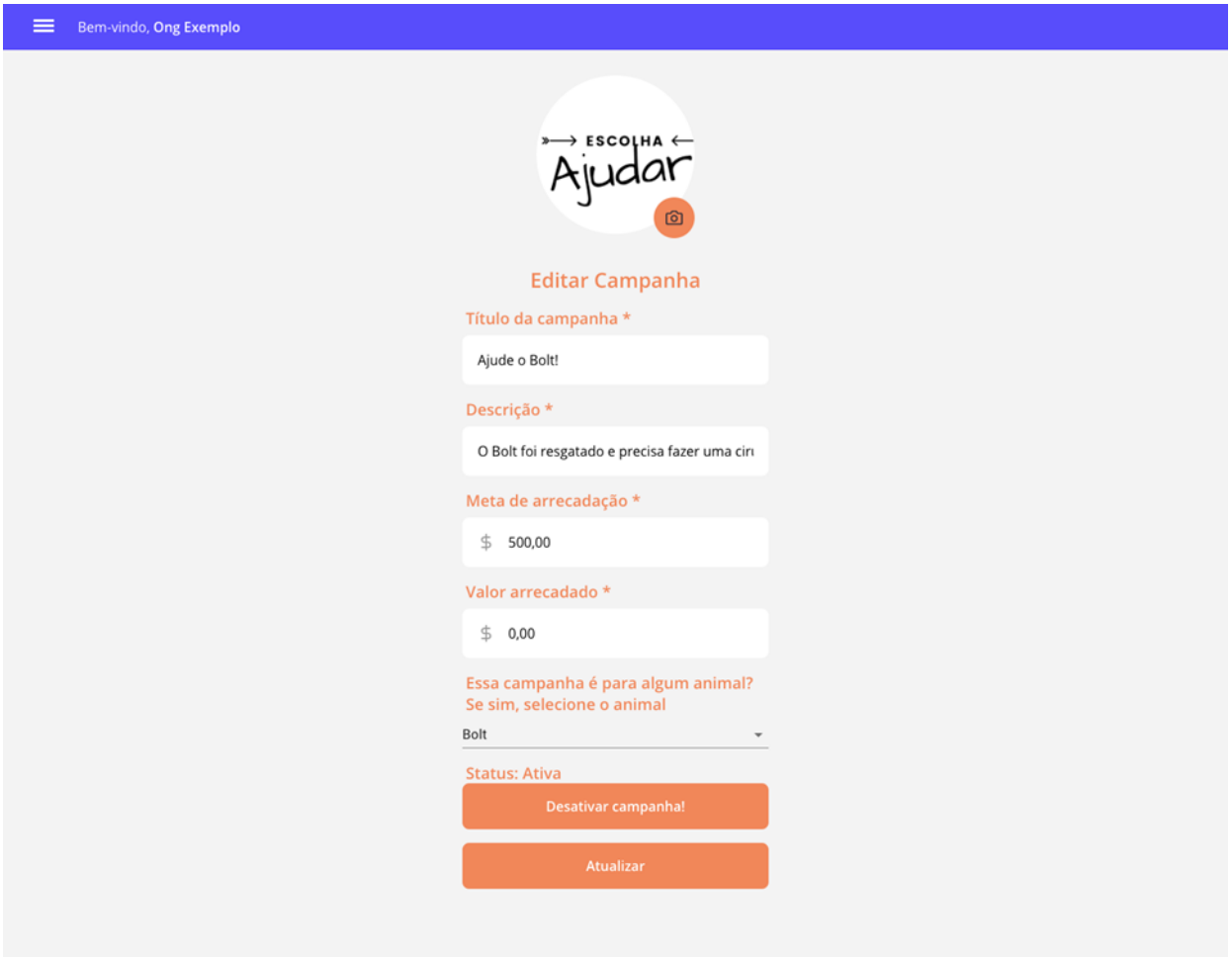
Fonte: Elaborada pelos autores (2021).

Figura 30 – Cadastro de campanhas



Fonte: Elaborada pelos autores (2021).

Figura 31 – Edição de campanhas

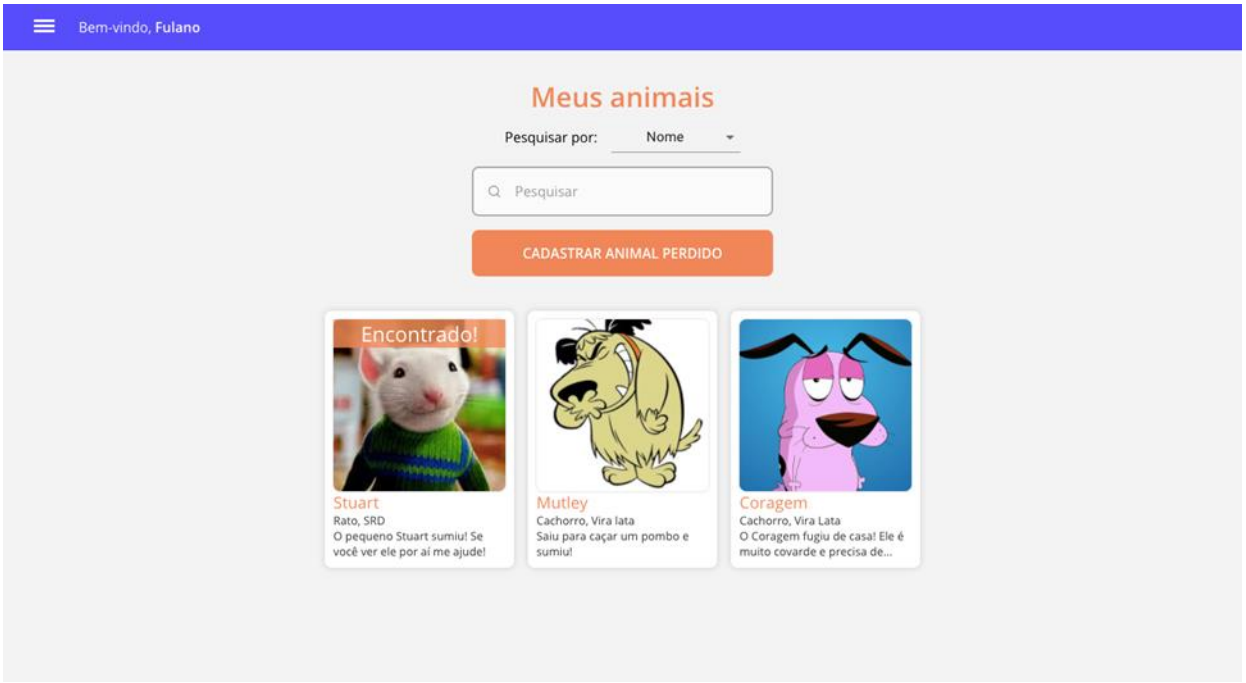


Fonte: Elaborada pelos autores (2021).

6.11 PÁGINA EXCLUSIVA DE DOADOR

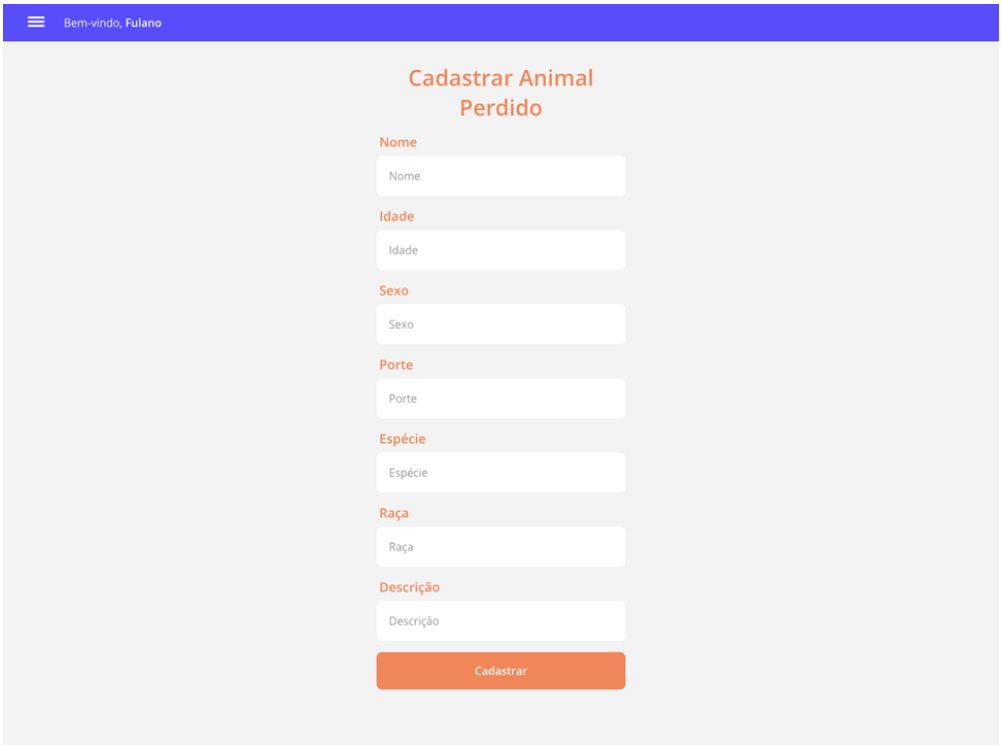
Usuários do tipo doador, têm acesso a uma página onde podem cadastrar, editar e visualizar seus animais perdidos. Quando o animal for encontrado, é possível alterar o status para "Encontrado".

Figura 32 – Lista de animais do usuário do tipo doador



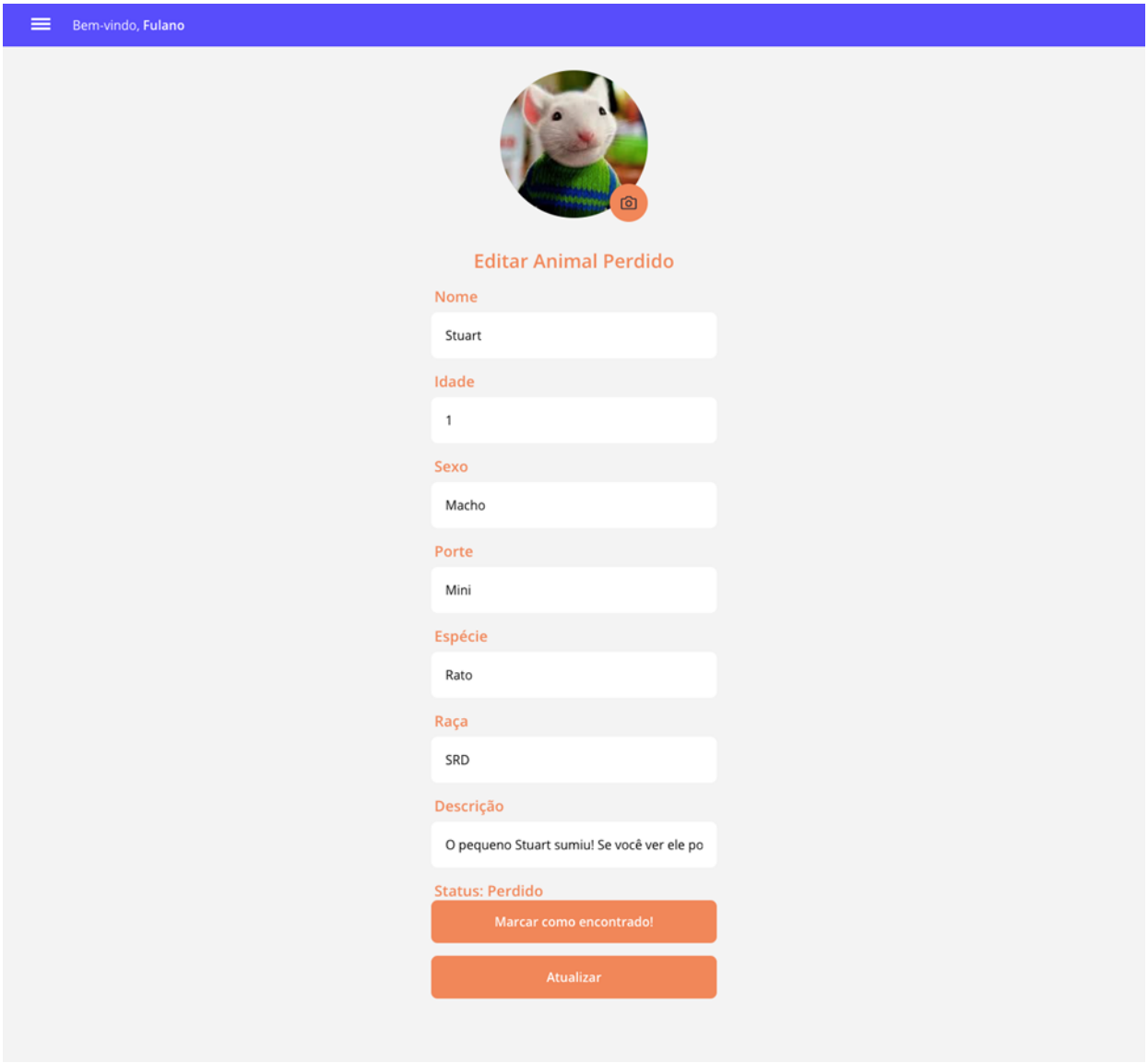
Fonte: Elaborada pelos autores (2021).

Figura 33 – Cadastro de animais perdidos



Fonte: Elaborada pelos autores (2021).

Figura 34 – Edição de um animal perdido



Fonte: Elaborada pelos autores (2021).

6.12 PWA

Quando o *site* é acessado a partir de um celular,uma mensagem é mostrada, ensinando o usuário a instalar o Pet's Hero.

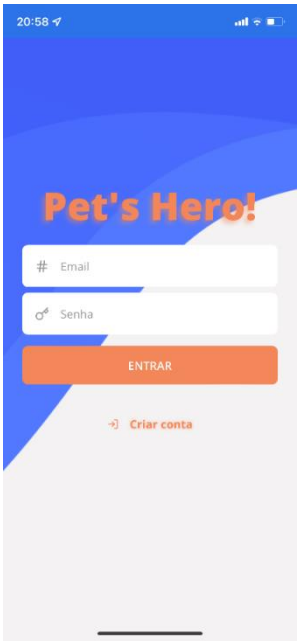
Figura 35 – Mensagem mostrada ao acessar o *site* a partir de um celular

Fonte: Elaborada pelos autores (2021).

Figura 36 – Ícone do aplicativo ao ser instalado

Fonte: Elaborada pelos autores (2021).

Figura 37 – Tela inicial da aplicação rodando como se fosse um aplicativo



Fonte: Elaborada pelos autores (2021).

7 CONSIDERAÇÕES FINAIS

Com o desenvolvimento do *site* Pet's Hero durante este trabalho, é possível citar implementações futuras com a finalidade de aumentar o leque de funcionalidades, como por exemplo: calendário de eventos, agendamento de visitas às instituições, vendas de produtos diretamente no *site*, assinatura mensal com a funcionalidade de apadrinhar animais das instituições.

Outro ponto interessante e mais complexo que poderíamos explorar em implementações futuras é a *gamificação* e *tokenização* do sistema. Através dessa ideia, poderíamos utilizar a tecnologia *blockchain* para criar NFTs (*No Fungible Tokens*) com fotos dos animais nas instituições, criando assim, novas formas de arrecadação de fundos.

Com a finalização do desenvolvimento desse projeto, tivemos como resultado, uma aplicação otimizada que consegue centralizar em um único sistema as principais necessidades das instituições protetoras dos animais, conseguindo alcançar os objetivos gerais e específicos apresentados no início do trabalho, foi possível implementar uma ferramenta que organiza em um único sistema, as campanhas para levantar recursos, mostrar de forma simplificada os animais a serem adotados e os animais perdidos em que o dono quer reencontrar novamente e mostra de forma simples onde está localizada a instituição possibilitando o doador ou o adotante visitar a instituição e conhecer de perto os trabalhos feitos, o sistema não se limitando a um pequeno número ou determinada área geográfica, possibilitando as instituições de todo o Brasil utilizar.

Apresentamos o projeto a voluntários de instituições cuidadoras de animais, veterinários e pessoas que se identificam com a causa animal. Tivemos *feedbacks* positivos, tal qual, confirmando o objetivo geral do projeto que visa ser uma ferramenta que aproxima as instituições protetoras dos animais e seus doadores.

Através do sistema é possível aproximar as instituições e seus colaboradores, centralizando em um único local o que antes era feito em mais de uma ferramenta, por exemplo: *Facebook* e *Instagram*, que são ferramentas que não tem funcionalidades específicas e focadas nas necessidades de uma instituição (criar campanhas de arrecadação, criar murais de adoção e animais perdidos de uma forma organizada e intuitiva).

Por fim, para o futuro, além da disponibilização do *site* na *internet*, ainda existem diversas possibilidades de implementações e melhorias para a evolução do projeto, possibilitando integrar novas formas de arrecadação de recursos, adaptando-se às novas tecnologias que estão surgindo e cada vez mais aproximando e conscientizando pessoas e instituições protetoras dos animais.

REFERÊNCIAS

BESSA, Mateus. **ORM no NodeJS com TypeORM.** 2020. Disponível em: <https://medium.com/@matheusbessa_44838/orm-no-nodejs-com-typeorm-a3b3d8a22240>. Acesso em: 19/05/2021.

DE SIQUEIRA, Fernando. **Modelo entidade e relacionamentos**. Disponível em: <https://sites.google.com/site/uniplibancodedados1/aulas/aula-4---modelo-entidade-e-relacionamentos>. Acesso em: 25/05/2021.

DUARTE, Luiz. **Autenticação JSON Web Token (JWT) em Node.js.** 2020. Disponível em: <https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/>. Acesso em: 25/05/2021.

FERNANDES, André. **O que é API?** Entenda de uma maneira simples. 2018. Disponível em: <https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>. Acesso em: 01/07/2020.

FERNANDES, Diego. **NodeJS:** Vale a pena? Vantagens, vagas e salário. 2018. Disponível em: <https://blog.rocketseat.com.br/nodejs-vale-a-pena-vantagens/>. Acesso em: 04/04/2020.

FERNANDES, Diego. **PWA:** O que é? Vale a pena? Quando utilizar?. 2019. Disponível em: <https://blog.rocketseat.com.br/pwa-o-que-e-quando-utilizar/>. Acesso em: 15/05/2021.

JANONES, Ramos de Souza. **O que são middlewares em Node JS?**. 2017. Disponível em: <https://blog.rocketseat.com.br/pwa-o-que-e-quando-utilizar/>. Acesso em: 15/05/2021.

FRIAS, Thiago. **React vs Vue vs Angular: qual escolher?**, 2020. Disponível em: <https://blog.geekhunter.com.br/react-vs-vue-vs-angular-qual-escolher/#React_vs_Vue_vs_A ngular_qual_a_conclusao_O_que_e_melhor>. Acesso em: 16/03/2020.

GURU99. **Node.js Vs Python: What's the Difference?** 2020. Disponível em: <https://www.guru99.com/node-js-vs-python.html#:~:text=KEY%20DIFFERENCE,dynamic%20and%20multipurpose%20programming%20language.&text=Node%20is%20best%20suited%20for,best%20option%20for%20asynchronous%20programming.>. Acesso em: 17/03/2020.

LIMA, Victor**. Desenvolvimento para mobile:** Ionic, React-Native ou Flutter, qual usar? 2020. Disponível em: <https://blog.geekhunter.com.br/desenvolvimento-para-mobile- reactnative-flutter-ionic/>. Acesso em: 16/03/2020.

MARINHO, Thiago. **Conceitos do Node**, 2019. Disponível em:<https://www.tgmarinho.com/conceitos-do-node/>. Acesso em: 04/04/2020.

MERCADOPAGO. **Quanto custa receber pagamentos**, 2015. Disponível em: <https://www.mercadopago.com.br/ajuda/custo-receber-pagamentos_453>. Acesso em: 05/04/2020.

PHP. **O que é o PHP?**, 2009. Disponível em: <https://www.php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 04/04/2020.

SKIA. **Skia Graphics Library.** 2002. Disponível em: <https://skia.org/>. Acesso em: 04/04/2020.

SOARES, Felipe Luiz. **PetAppy:** aplicativo para proteção dos animais. 2017. Trabalho de Conclusão de Curso - Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Sorocaba, Sorocaba, 2017.

STACKOVERFLOW. **Technology programming scripting and markup languages professional developers**. 2020. Disponível em: <https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-mar kup-languages-professional-developers>. Acesso em: 04/04/2020.

TAVARES, Lucas. **Conheça o PHP e suas principais funções**. 2019. Disponível em: <https://www.melhoreshospedagemdesites.com/o-que-e-php/>. Acesso em: 04/04/2020.

TYPEORM. **TypeORM**. 2021. Disponível em: <https://typeorm.io/>. Acesso em: 25/05/2021.

VELASCO, Clara. **Brasil tem mais de 170 mil animais abandonados sob cuidado de ONGs, aponta instituto.** 2019. Disponível em: <https://g1.globo.com/sp/sao-paulo/noticia/2019/08/18/brasil-tem-mais-de-170- mil-animais-abandonados-sob-cuidado-de-ongs-aponta-instituto.ghtml>. Acesso em: 30/03/2020.

WERLITON, Silva. **Aplicações móveis nativas com React Native e Firebase:** um estudo de caso. 2018. Monografia - Ciência da Computação, Universidade Federal do Maranhão, São Luis, 2018.