Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Alisson Alves Silva

Felipe Rodrigues de Oliveira

Matheus Pequeno Linares

Victor Rodrigues de Oliveira

PROTÓTIPO DE DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA ADOÇÃO DE ANIMAIS

Americana

2022

Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Alisson Alves Silva

Felipe Rodrigues de Oliveira

Matheus Pequeno Linares

Victor Rodrigues de Oliveira

PROTÓTIPO DE DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA ADOÇÃO DE ANIMAIS

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Esp. Antônio Alfredo Lacerda.

Área de concentração: Engenharia de Software

Americana
2022

Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

SILVA, Alisson Alves

Protótipo de desenvolvimento de um aplicativo mobile para adoção de animais. / Alisson Alves Silva, Felipe Rodrigues de Oliveira, Matheus Pequeno Linares, Victor Rodrigues de Oliveira. – Americana, 2022.

55f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientadores: Prof. Antonio Alfredo Lacerda

1 Desenvolvimento de software I. OLIVEIRA, Felipe Rodrigues de II. LINARES, Matheus Pequeno III. OLIVEIRA, Victor Rodrigues de IV. LACERDA, Antonio Alfredo V. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.3.05

Alisson Alves Silva

Felipe Rodrigues de Oliveira

Matheus Pequeno Moraes

Victor Rodrigues de Oliveira

Protótipo – Desenvolvimento de Aplicativo Mobile para Adoção de Animais

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Americana como parte dos requisitos para obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza.
Área de Atuação : Engenharia de Software

Americana, 23 de junho de 2022.

Banca Examinadora:

Antonio Alfredo Lacerda
Especialista

Diogenes de Oliveira
Mestre

Thiago Salhab
Mestre

Resumo

Este Projeto trata-se do desenvolvimento de um protótipo de um aplicativo mobile para adoção de pets realizando o intermédio entre doador e donatário apresentando as informações completas do animal, fotos, sua localização e disponibilizando, pelo menos, 3 meios de contato sendo eles: um chat próprio, integração com *WhatsApp* e *e-mail*. O protótipo foi desenvolvido sem muitas dificuldades, para isso utilizamos tecnologias modernas como *React Native*, *Material Design* e *Java* com *Spring Boot*.

Palavras-Chave: Aplicativo; Animais; Adoção; *React Native*; *Java*; *JavaScript*.

Abstract

This work is about the development of a prototype of a mobile application for the adoption of pets, performing the intermediary between donor and donee, presenting the complete information of the animal, photos, its location and providing, at least, 3 means of contact, being them: a chat of its own, integration with WhatsApp and email. The prototype was developed without many difficulties, for this we use modern technologies such as React Native, Material Design and Java with *Spring Boot*.

Keywords: Application; Animals; Adoption; React Native; Java; JavaScript.

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Justificativa 10
1.2 Objetivos Específicos 11
2 SISTEMAS DE INFORMAÇÃO 12
3 ENGENHARIA DE SOFTWARE 13
3.1 Modelagem 13
3.2 Diagramas de Caso de Uso 13
3.3 Diagramas de Sequência 18
3.4 Diagrama de Entidade e Relacionamento 27
3.5 Dicionário de Dados 29
4 PROJETO PROPOSTO 35
4.1 API 35
4.1.1 API Google Maps 35
4.1.2 API ViaCEP 36
4.2 Tecnologia Empregada 36
4.2.1 Eclipse 36
4.2.2 Visual Studio Code 36
4.2.3 Java 37
4.2.4 Spring Boot 38
4.2.5 JavaScript 38
4.2.6 Figma 39
4.2.7 React Native 40
4.2.8 MYSQL 41
4.3 Telas do Projeto 42
4.3.1 Tela de login 42
4.3.2 Tela de Cadastro 44
4.3.3 Tela Home 45
4.3.4 Tela de Perfil 47
4.3.5 Tela de cadastro de animal 49
4.3.6 Tela de detalhes da adoção 50
4.3.7 Tela do Mapa 52
5 CONCLUSÃO 54
REFERÊNCIAS 55

LISTA DE FIGURAS

Figura 1 - Diagrama de caso de uso de login e registro do usuário. 14
Figura 2 - Diagrama de caso de uso da visualização de doações.....15
Figura 3 - Diagrama de caso de uso da visualização de detalhes da doação. 16
Figura 4 - Diagrama de caso de uso de criação de adoção 17
Figura 5 - Diagrama de caso de uso de edição de adoção. 17
Figura 6 - Diagrama de caso de uso de visualização de denúncias. 18
Figura 7 - Diagrama de Sequência de login. 19
Figura 8 - Diagrama de Sequência de cadastro. 20
Figura 9 - Diagrama de Sequência para exibir os detalhes de uma publicação. 21
Figura 10 - Diagrama de Sequência ao “curtir” uma publicação.....22
Figura 11 - Diagrama de Sequência ao clicar no botão "chat". 23
Figura 12 - Diagrama de Sequência ao clicar no botão “WhatsApp 23
Figura 13 - Diagrama de Sequência ao clicar no botão "e-mail". 24
Figura 14 - Diagrama de sequência ao clicar no botão "denunciar". 25
Figura 15 - Diagrama de Sequência ao clicar no botão "nova adoção". 26
Figura 16 - Diagrama de Entidade e Relacionamento 28
Figura 21 - Tela de Login.....43
Figura 22 - Tela de Cadastro. 44
Figura 23 - Tela Home. 46
Figura 24 - Tela de Perfil 48
Figura 25 - Tela de cadastro de adoção. 49
Figura 26 - Tela de detalhes da adoção 51
Figura 27 - Tela do Mapa. 53

LISTA DE TABELAS

Tabela 1 - Dicionário de Dados da entidade User. 29
Tabela 2 - Dicionário de Dados da entidade User_address.....30
Tabela 3 - Dicionário de Dados da entidade User_image.....30
Tabela 4 - Dicionário de Dados da entidade Chat. 31
Tabela 5 - Dicionário de Dados da entidade Message. 31
Tabela 6 - Dicionário de Dados da entidade Pet.....32
Tabela 7 - Dicionário de Dados da entidade Pet_image.....32
Tabela 8 - Dicionário de Dados da entidade Pet_post.....33
Tabela 9 - Dicionário de Dados da entidade Veterinary_care.....33
Tabela 10 - Dicionário de Dados da entidade Pet_post_report.....34

1 INTRODUÇÃO

Os amantes de animais sabem o quão triste é ver um animal sem um lar para o acolher, com frio, com fome e sede, e sabe-se que seria impossível uma instituição só, ou uma única pessoa acolher todos eles. É visto constantemente nas redes sociais, postagens relacionadas à adoção de pets por usuários que encontraram algum animal abandonado e/ou que não possuem condições de cuidar do animal, mas, como essa não é a finalidade principal dessas redes, o alcance se torna muito mais baixo e não possuem o suporte adequado para este fim.

Durante a pandemia do COVID-19, pesquisas indicam um aumento de 60% no número de abandono de animais[1]. Em contrapartida, dados da União Internacional Protetora dos Animais (**UIPA**) mostram que houve um crescimento de 400% na procura de animais para adoção. Mas como, no Brasil, não existem aplicativos referência nesse segmento, as vezes por limitação de funcionalidades, por serem pouco intuitivos ou até mesmo por falta de divulgação, acabam não contribuindo efetivamente para a melhora dessas pesquisas.

Pensando nisso, decidiu-se desenvolver um *software* que atenda essa finalidade, utilizando recursos modernos com funcionalidades que a maioria dos concorrentes não possuem, além de uma interface mais amigável, se baseando nos padrões do *Material Design*, trazendo algo mais intuitivo para o usuário.

1.1 Justificativa

Tendo em vista os problemas apresentados anteriormente, foi identificado uma oportunidade de contribuir positivamente a esse problema, conectando doadores, seja uma pessoa comum ou uma instituição, com donatários interessados através de um chat integrado ou de plataformas externas (*WhatsApp* e *e-mail*), trazendo uma lista de

[1] MARTUCCI, Mariana. Abandono de animais aumentou cerca de 60% durante a pandemia. EXAME. 2021. Disponível em: https://exame.com/bussola/abandono-de-animais-aumentou-cerca-de-60-durante-a-pandemia/. Acesso em: 13 jun. 2022.

animais a serem doados com suas informações completas, priorizando a localização atual do usuário para facilitar o encontro de animais nas proximidades.

1.2 Objetivos Específicos

- Desenvolver um protótipo de um aplicativo móvel para as plataformas *Android* e *iOS* utilizando conhecimentos adquiridos em sala e externamente;
- Disponibilizar informações a respeito do animal a ser doado, contendo obrigatoriamente pelo menos uma foto;
- Realizar o intermédio entre doador e donatário através de *chat* integrado e plataformas externas como *WhatsApp* e *e-mail*.

2 SISTEMAS DE INFORMAÇÃO

Um sistema de informação é “todo sistema que manipula dados e gera informação, usando ou não recursos de tecnologia da informação, pode ser genericamente considerado como um sistema de informação”. (GONÇALVES, 2021, p. 49).

Este é um conjunto organizado de pessoas, *hardware*, *software*, rede de comunicação e dados, que são coletados e transformados em informações dentro de um ambiente organizacional. (O’BRIEN, 2004).

Além de dar apoio à tomada de decisões, à coordenação e ao controle, esses sistemas também auxiliam os gerentes e trabalhadores a analisar problemas, visualizar assuntos complexos e criar produtos. (LAUDON & LAUDON, 2004).

3 ENGENHARIA DE SOFTWARE

A engenharia de *software* é um campo de engenharia e computação que se concentra na especificação, desenvolvimento, manutenção e criação de *software*, aplicando técnicas e práticas de gerenciamento de projetos e outras disciplinas, com o objetivo de organização, produtividade e qualidade. Hoje, essas tecnologias e práticas incluem linguagens de programação, bancos de dados, ferramentas, plataformas, bibliotecas, padrões de projeto de *software*, processos de *software* e qualidade de *software*. Além disso, a engenharia de *software* deve fornecer mecanismos para planejar e gerenciar o processo de desenvolvimento de sistemas computacionais de alta qualidade que atendam às necessidades dos solicitantes de *software*.

> A Engenharia de Software é uma disciplina de engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a sua manutenção desse sistema, depois que ele entrou em operação. (SOMMERVILLE, 2007, p. 5)

3.1 Modelagem

Na fase da modelagem é feita a documentação do aplicativo, se trata de diagramas que facilitam na compreensão do Projeto de forma padronizada. A documentação deste Projeto utilizará a linguagem de modelagem *Unified Modeling Language*[2] (**UML**) para modelar os casos de uso.

3.2 Diagramas de Caso de Uso

De acordo com DevMedia (2012) os diagramas de caso de uso é responsável por definir “as principais funcionalidades do sistema e a interação dessas

[2] *Unified Modeling Language* ou Linguagem Unificada de Modelagem (UML) é uma linguagem padrão para modelagem e documentar os sistemas orientados a objetos.

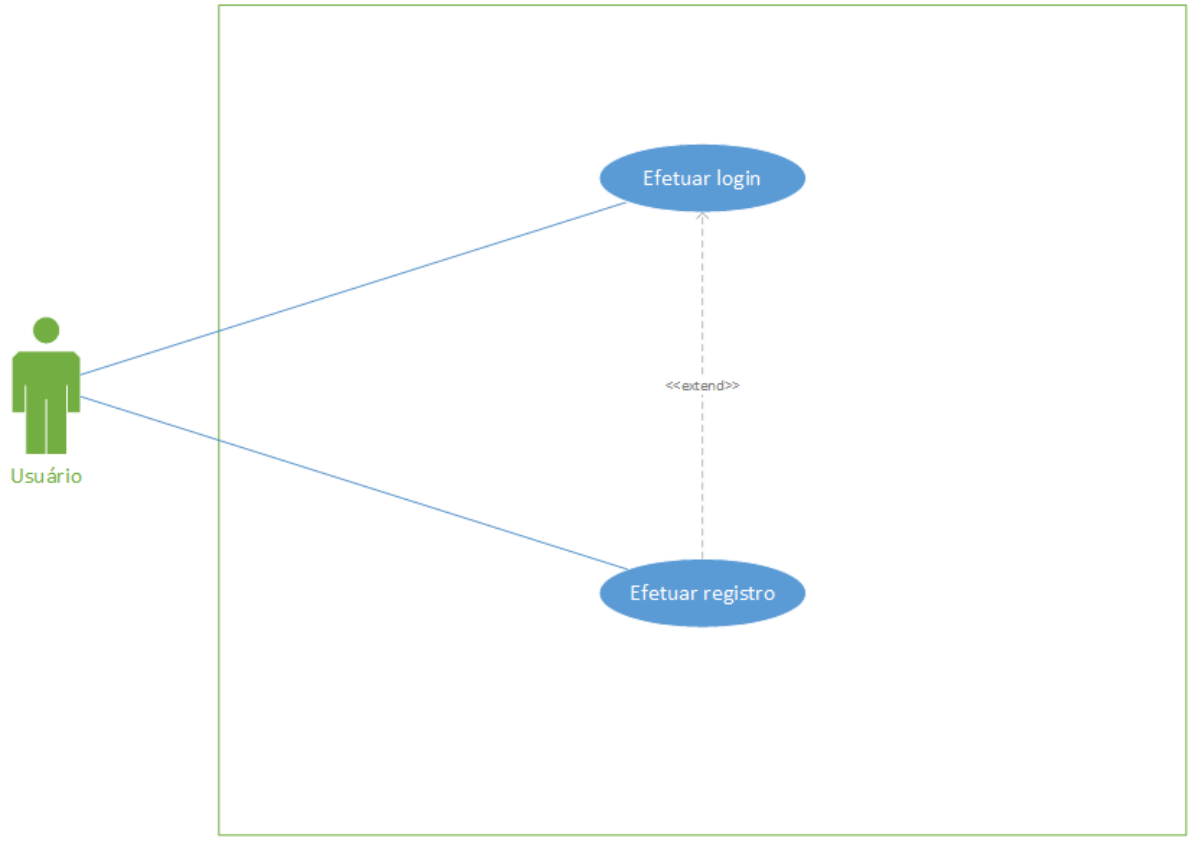
funcionalidades com os usuários do mesmo sistema. Nesse diagrama não nos aprofundamos em detalhes técnicos que dizem como o sistema faz.”.

Os atores que interagem com o sistema são o Usuário (Doador ou Donatário) e o Administrador. O sistema é um caso de uso explícito e se trata do sistema em si em que os casos de uso acontecem.

- **Usuário** é o ator que representa os utilizadores deste aplicativo. um usuário pode por exemplo visualizar os animais em adoção, visualizar animais próximos a ele, entrar em contato com outros usuários, entre outras coisas.
- O **Administrador** é o ator que administra o aplicativo, mantendo-o em ordem, assegurando-se que no aplicativo, não terá nada indevido.

A figura 1 representa o diagrama de caso de uso de login e registro de usuário, onde o usuário pode realizar seu login na aplicação ou navegar para a tela de registro e realizar o seu cadastro.

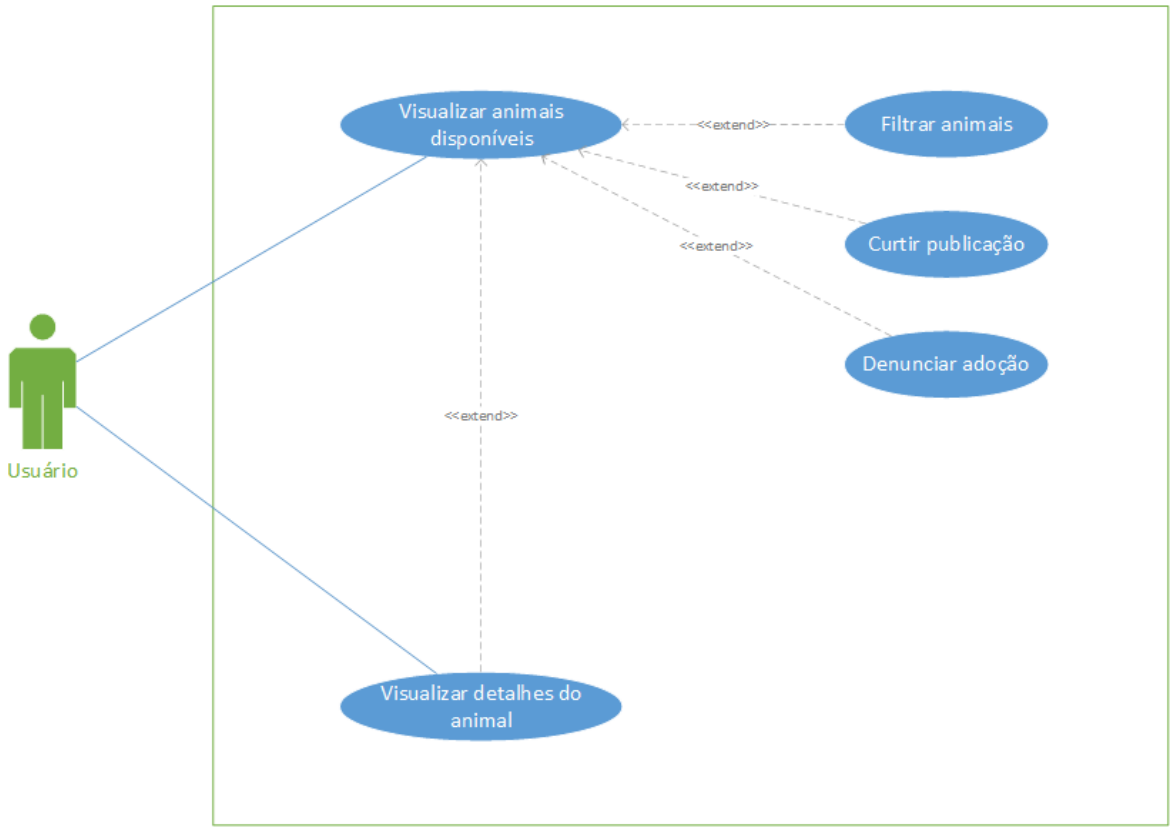
Figura 1 - Diagrama de caso de uso de login e registro do usuário.



Fonte: Elaborado pelos autores.

A figura 2 representa o diagrama de caso de uso da visualização de doações, onde o usuário pode visualizar as adoções disponíveis no momento e, opcionalmente, filtrar os animais que deseja ver, curtir a publicação para contribuir com impulsionamento da mesma, denunciar a doação caso exista algo que infrinja alguma regra da aplicação e navegar para a tela de detalhes do animal.

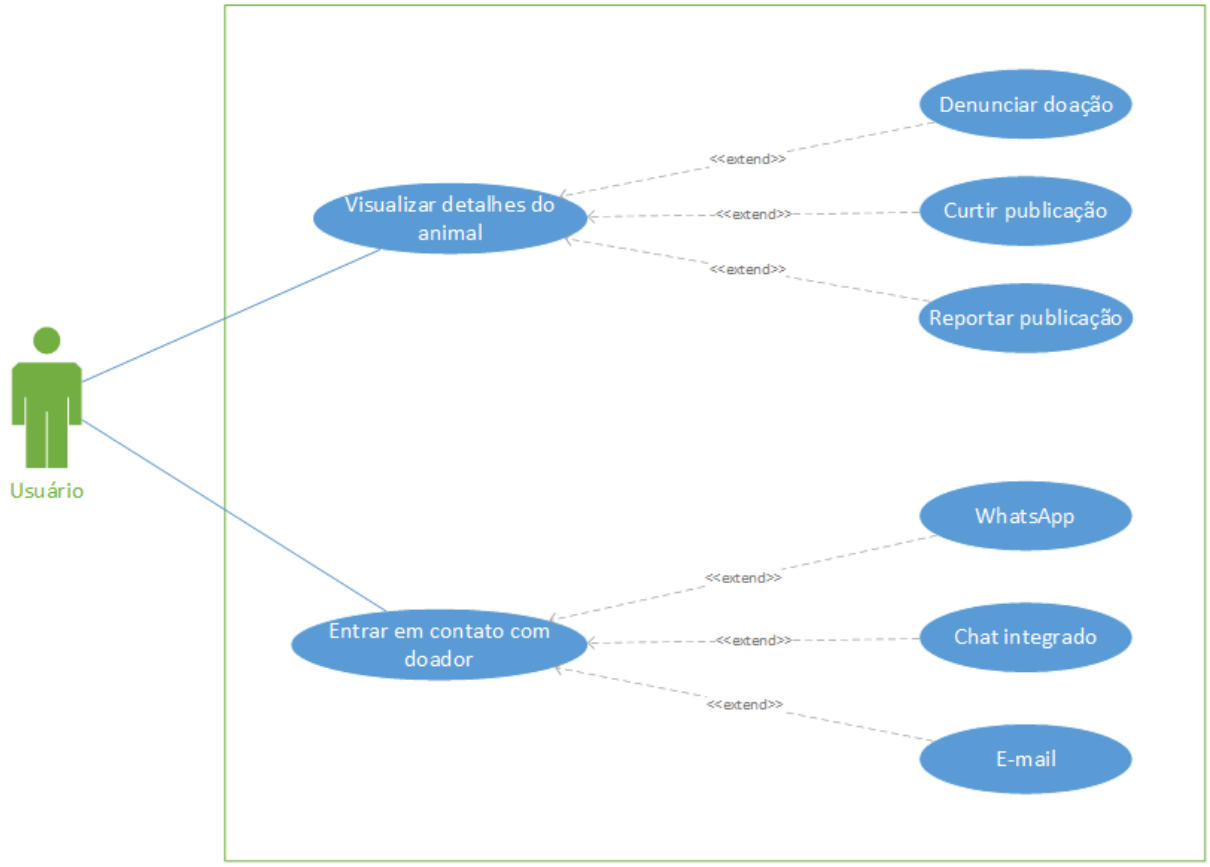
Figura 2 - Diagrama de caso de uso da visualização de doações.



Fonte: Elaborado pelos autores.

A figura 3 representa o diagrama de caso de uso da visualização de detalhes da doação, onde é possível visualizar os mesmos dados da figura 2, com o adicional de visualizar os meios de contatos disponíveis com o doador.

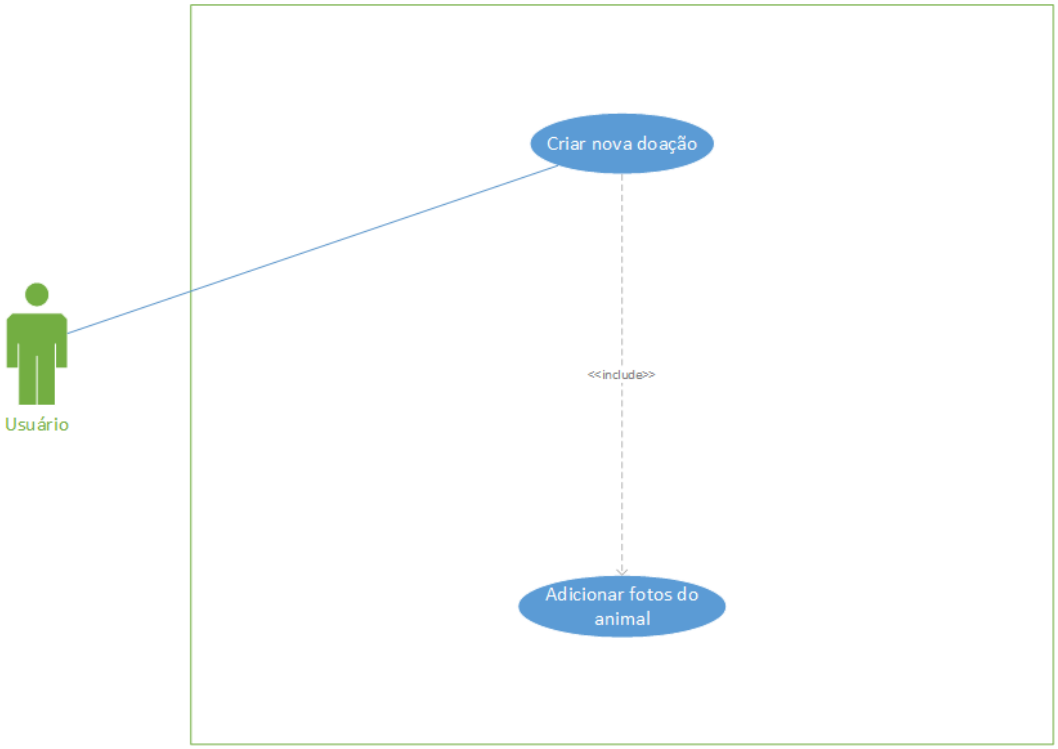
Figura 3 - Diagrama de caso de uso da visualização de detalhes da doação.



Fonte: Elaborado pelos autores.

A figura 4 representa o diagrama de caso de uso de criação de adoção, onde é necessário que o usuário insira ao menos uma imagem do animal a ser doado.

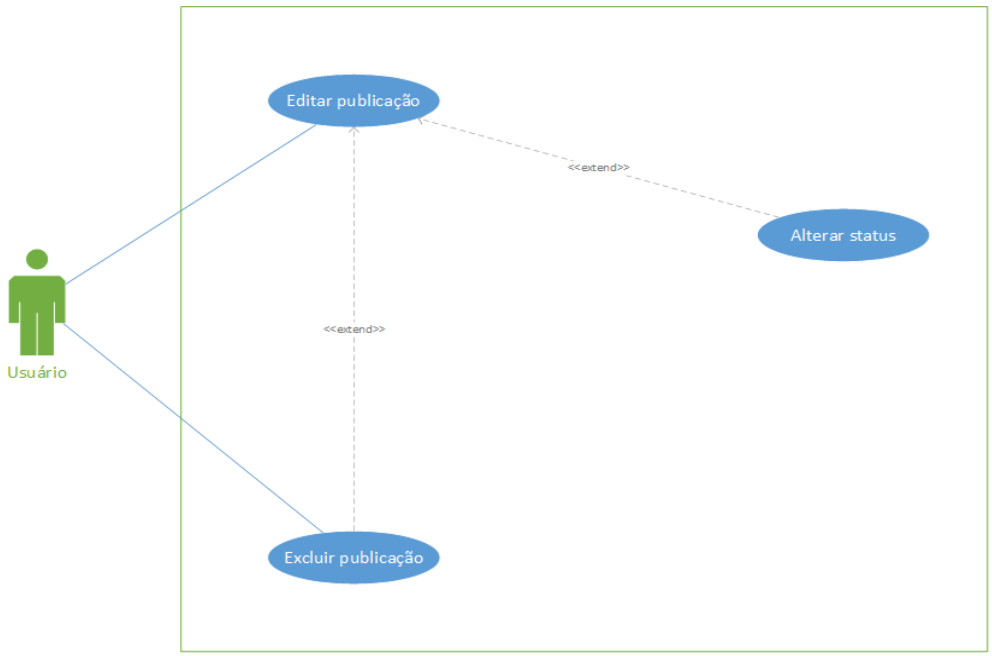
Figura 4 - Diagrama de caso de uso de criação de adoção.



Fonte: Elaborado pelos autores.

A figura 5 representa o diagrama de caso de edição de adoção, onde possibilita também a exclusão da publicação e a alteração do status da doação.

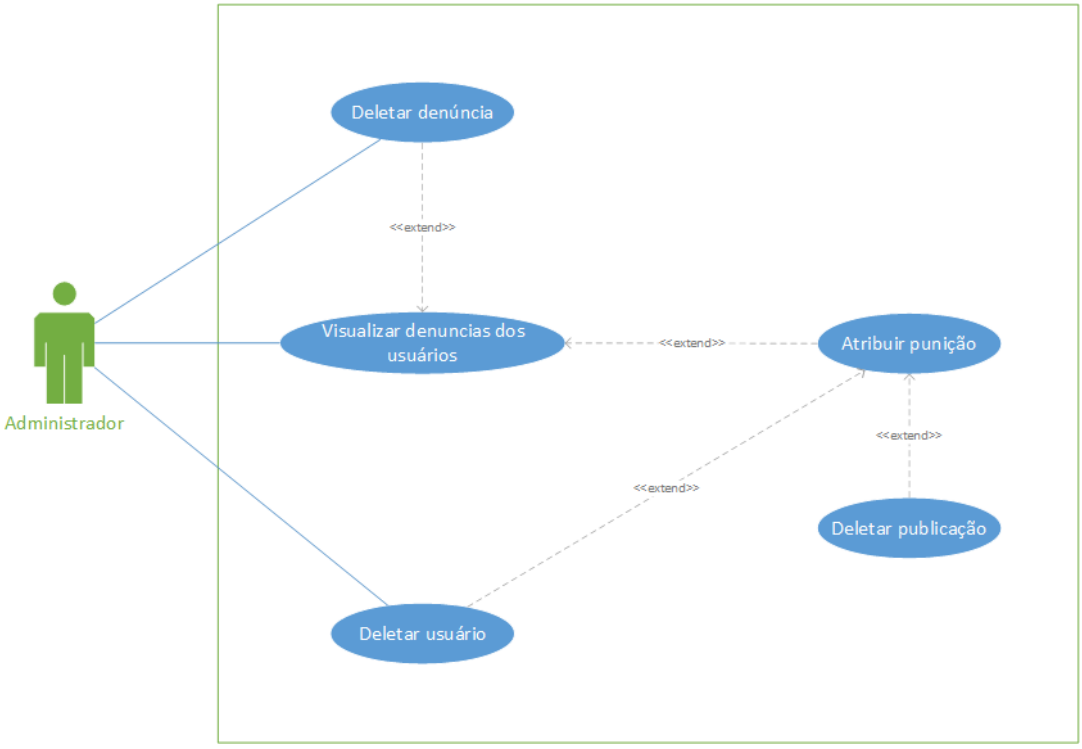
Figura 5 - Diagrama de caso de uso de edição de adoção.



Fonte: Elaborado pelos autores.

A figura 6 representa o diagrama de caso de uso de visualização de denúncias do aplicativo, onde possibilita a atribuição de uma punição, sendo elas deletar usuário e deletar publicação. Além disso, também permite deletar a denúncia registrada, caso seja identificada como inválida.

Figura 6 - Diagrama de caso de uso de visualização de denúncias.

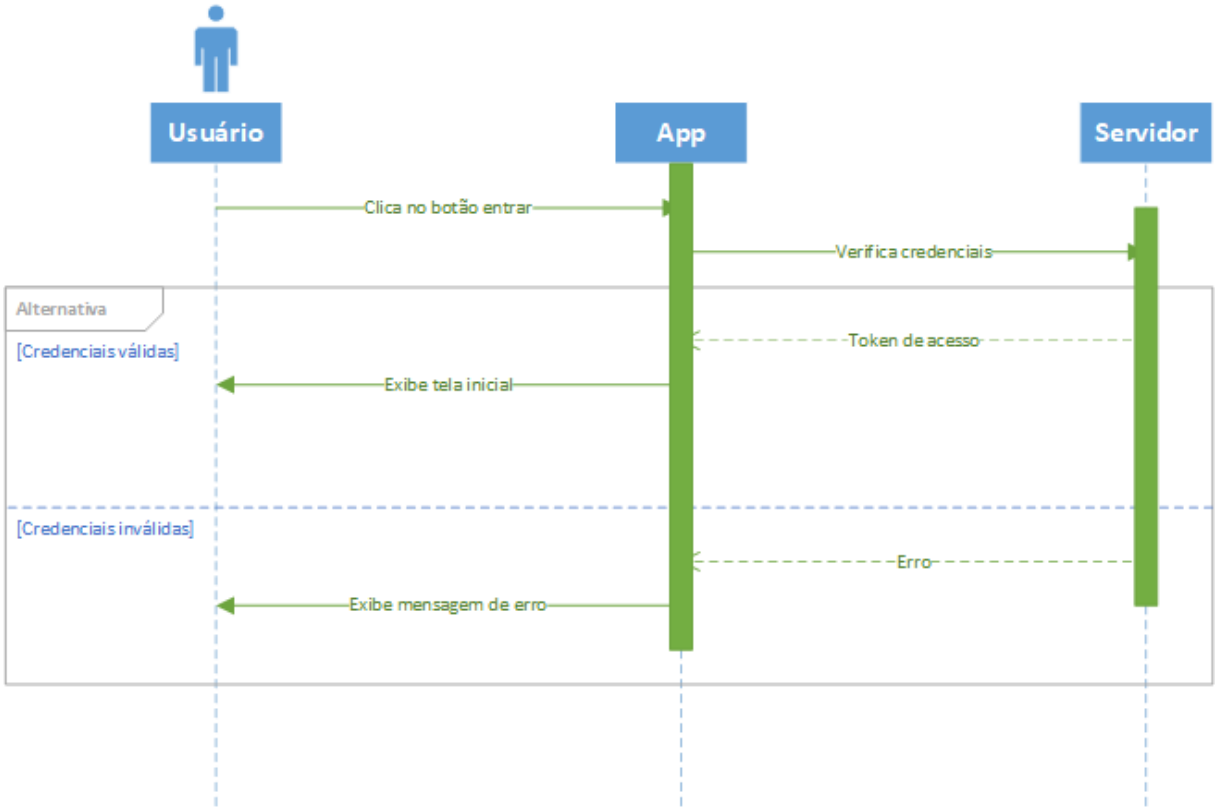


Fonte: Elaborado pelos autores.

3.3 Diagramas de Sequência

Diagrama de Sequência é um modelo UML primariamente utilizado para modelar interações entre atores e objetos em um sistema e interações entre os próprios objetos. Eles descrevem passo a passo o que acontece após uma interação.

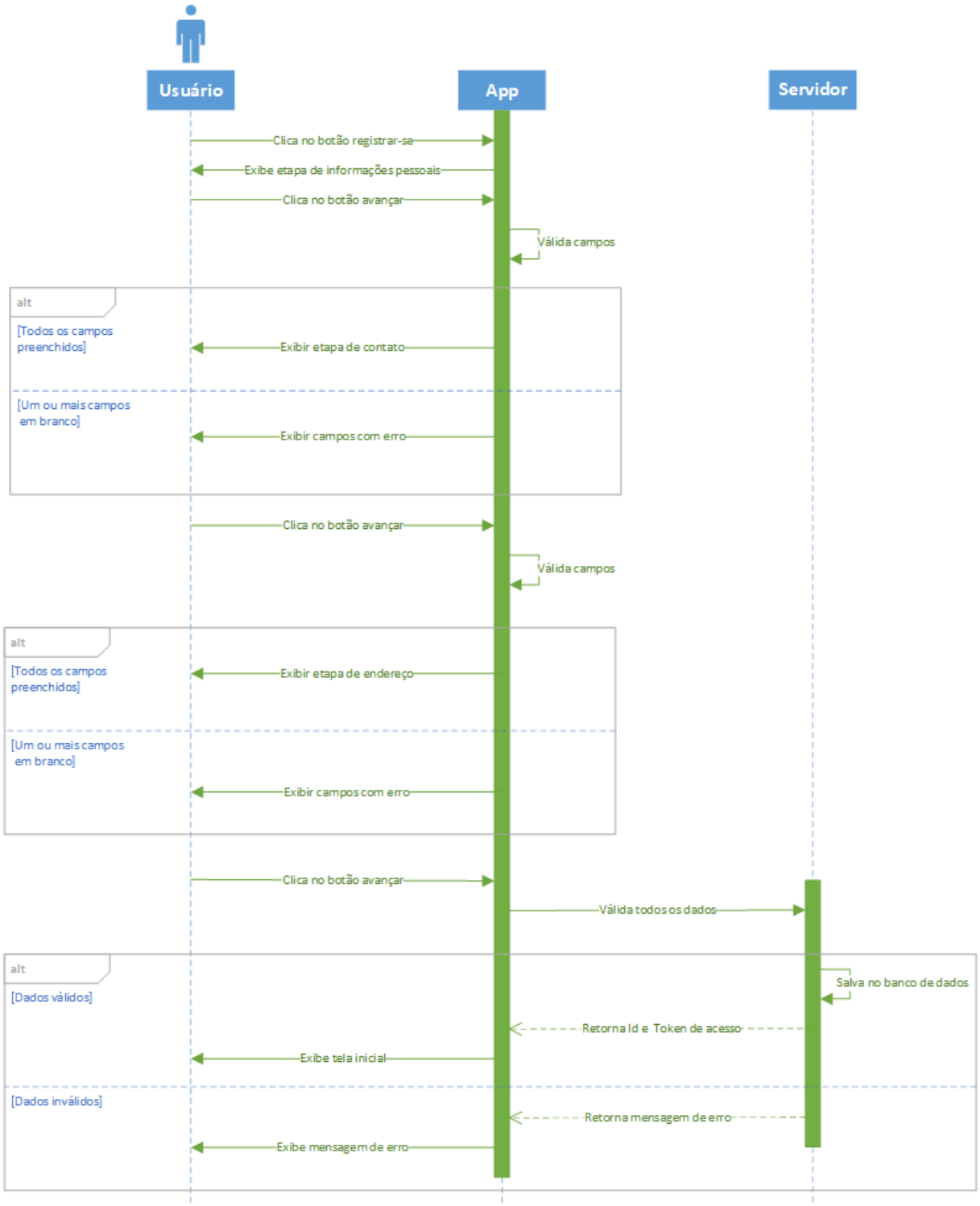
Figura 7 - Diagrama de Sequência de login.



Fonte: Elaborado pelos autores.

O diagrama de sequência de *login* (Figura 7) representa as interações entre o usuário, aplicativo e servidor. Ao usuário clicar em “entrar”, o aplicativo envia as credenciais de *login* para *o* servidor, que fará a validação dos dados. Em caso de sucesso, o servidor irá retornar o *token* de acesso para o aplicativo, que deverá enviá-lo em toda requisição para o servidor, pois o *token* identifica o usuário, e o aplicativo deve exibir a tela inicial. Em caso de erro, o servidor deve retornar o motivo do erro e o aplicativo deve informar ao usuário o que ocorreu.

Figura 8 - Diagrama de Sequência de cadastro.



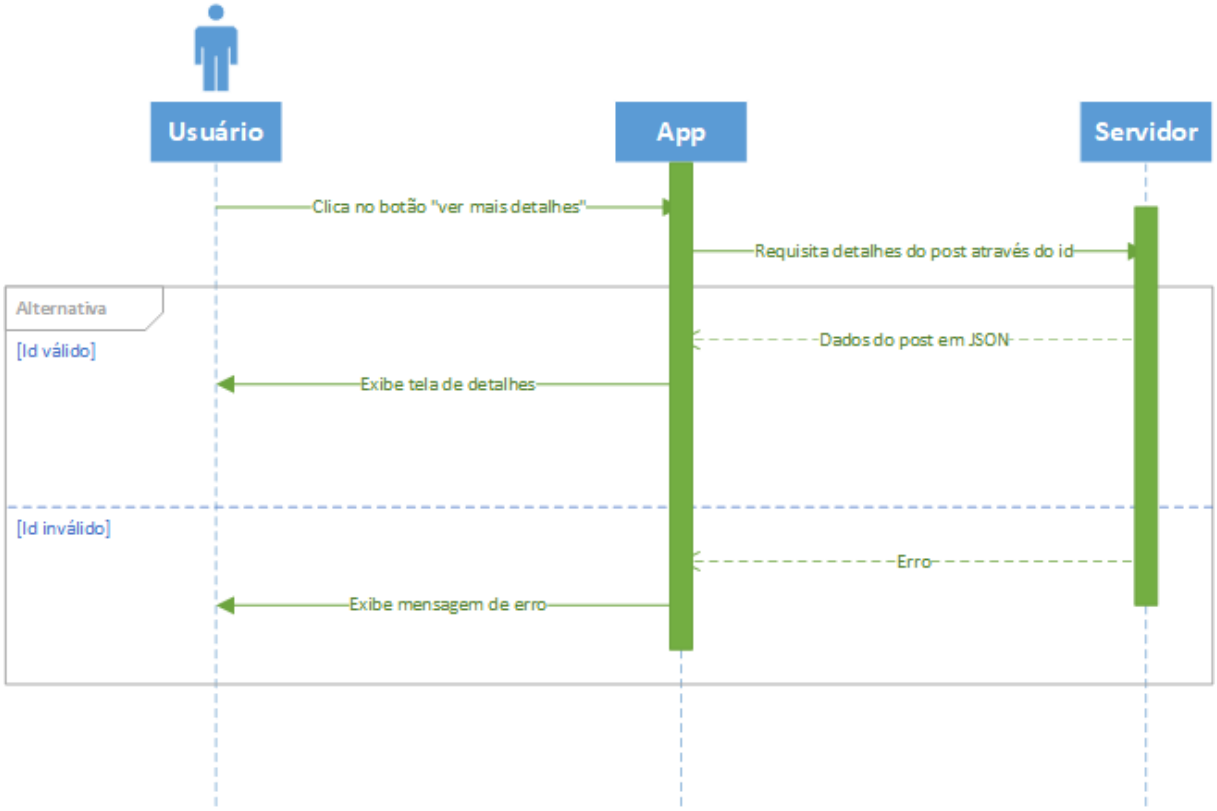
Fonte: Elaborado pelos autores.

O diagrama de sequência de cadastro (Figura 8) demonstra o passo a passo do formulário de cadastro do aplicativo. Ao usuário clicar em “registrar-se”, será exibido a primeira etapa do cadastro, onde o usuário deve informar seus dados

pessoais. Clicando em “avançar”, o aplicativo verifica se todos os campos obrigatórios foram preenchidos, em caso positivo, permite avançar para a próxima etapa, em caso negativo, exibe os campos que não foram informados. Esse fluxo se repete para todas as etapas seguintes.

Ao clicar em “avançar” na última etapa, o aplicativo envia os dados para o servidor, que fará outras validações, como verificar se o *e-mail* já foi utilizado, ou o número do celular. Caso todos os dados estejam corretos, o servidor deve salvar os dados no banco de dados e retornar o ID do novo usuário junto com o token de identificação. Em caso de erro, a mensagem deve ser exibida para o usuário.

Figura 9 - Diagrama de Sequência para exibir os detalhes de uma publicação.

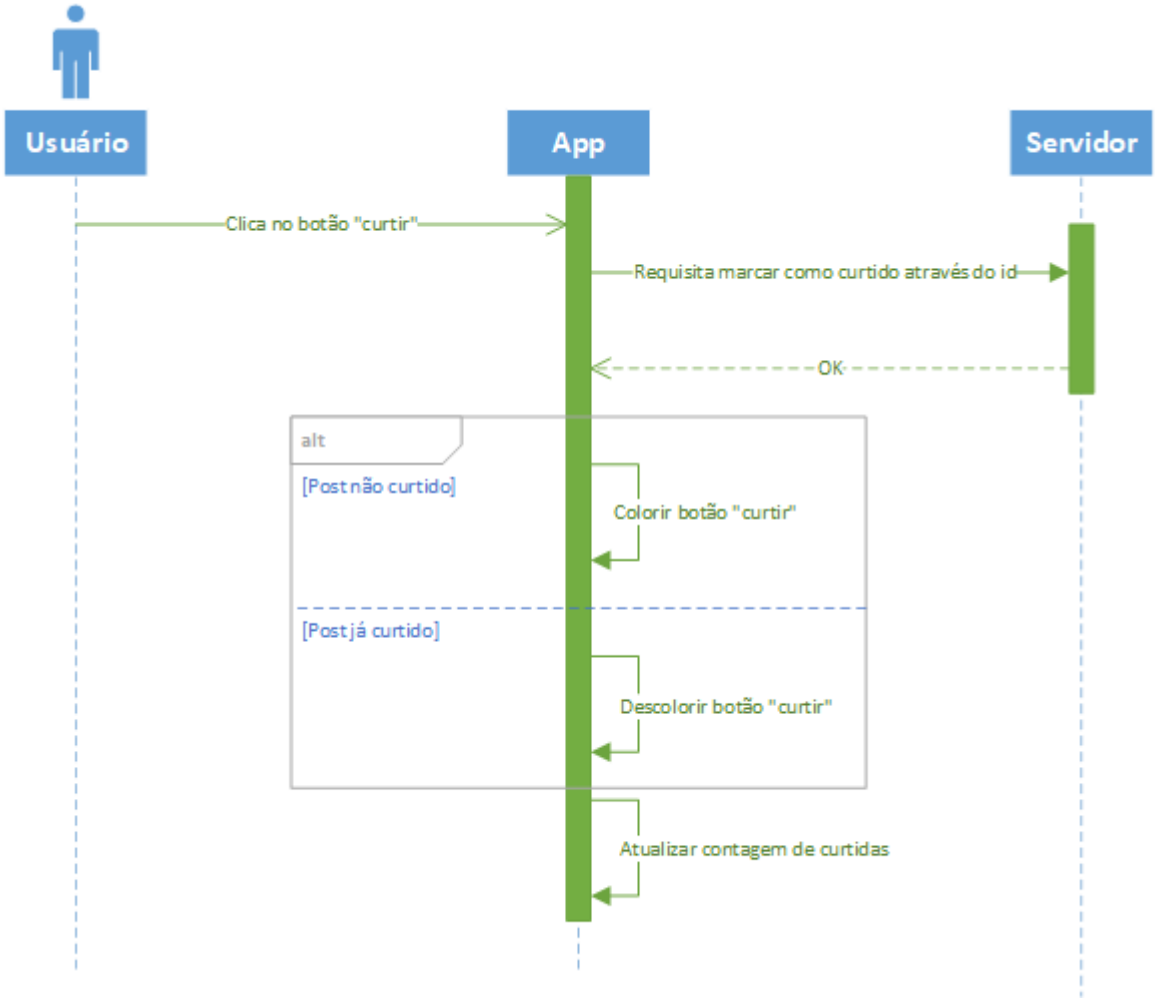


Fonte: Elaborado pelos autores.

O diagrama de sequência para exibir os detalhes de uma publicação etapa (Figura 9) demonstra o passo a passo do que deve ocorrer ao clicar em “ver mais detalhes” em uma publicação. O aplicativo envia para o servidor o ID da publicação que o usuário deseja ver mais detalhes e o servidor responde com um JSON contendo

as informações requisitadas. Ao receber a resposta, o aplicativo abre a tela de detalhes. Em caso de erro, a mensagem deverá ser exibida para o usuário.

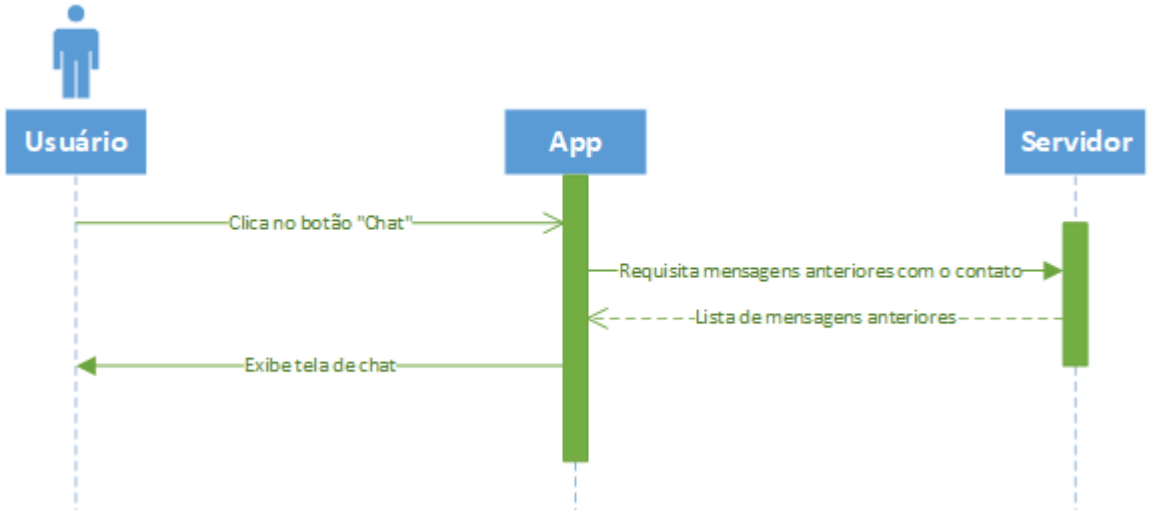
Figura 10 - Diagrama de Sequência ao “curtir” uma publicação.



Fonte: Elaborado pelos autores.

O diagrama de sequência ao “curtir” uma publicação (Figura 10) demonstra o que acontece ao “curtir” uma publicação. O aplicativo envia ao servidor o ID do post que o usuário curtiu de forma assíncrona e, ao receber a confirmação, irá colorir o botão “curtir” da publicação, caso não tenha sido curtida anteriormente, ou descolorir o botão, caso já tenha sido curtida e atualiza a contagem de “curtidas”.

Figura 11 - Diagrama de Sequência ao clicar no botão "chat".



Fonte: Elaborado pelos autores.

O diagrama de sequência ao clicar no botão "*chat*" (Figura 11) demonstra o passo a passo do que deve ocorrer ao abrir o *chat* com outro usuário. O aplicativo envia para o servidor o ID do usuário que deseja se comunicar e o servidor responde com uma lista de mensagens trocadas pelos usuários. Ao receber a resposta, o aplicativo exibe a tela de *chat*.

***Figura 12** - Diagrama de Sequência ao clicar no botão "WhatsApp".*

Fonte: Elaborado pelos autores.

O diagrama de sequência ao clicar no botão "*WhatsApp*" (Figura 12) representa a interação entre o usuário, o aplicativo e o aplicativo externo *WhatsApp*. Ao clicar no botão "*WhatsApp*", o aplicativo abre o *WhatsApp* do dispositivo do

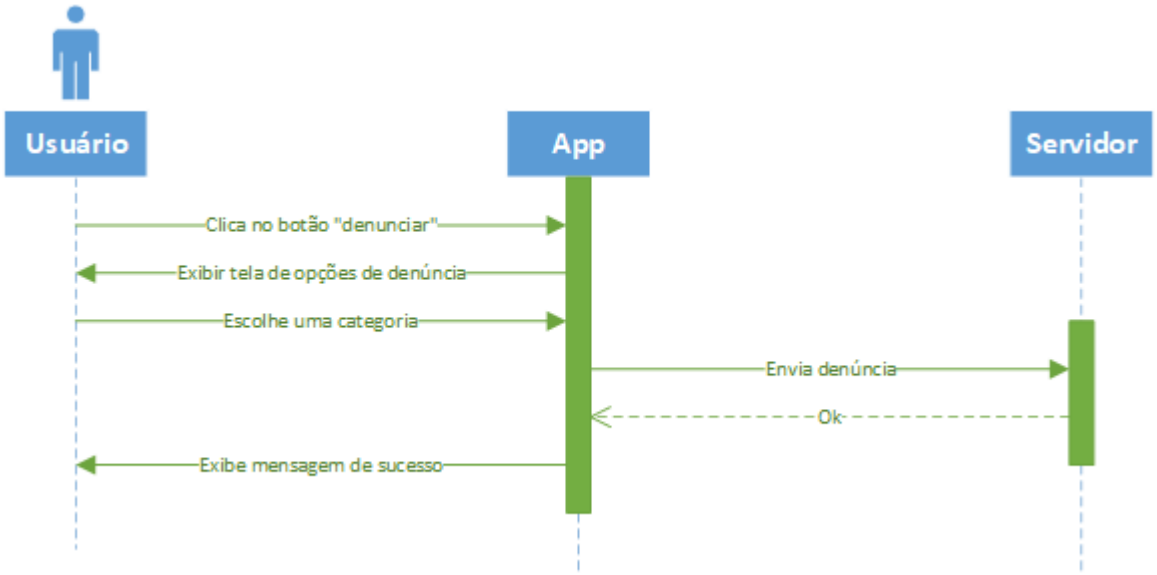
usuário, na conversa do número cadastrado na publicação e com uma mensagem pré-definida contendo informações básicas da publicação que o usuário interagiu e o nome de quem publicou.

Figura 13 - Diagrama de Sequência ao clicar no botão "e-mail".

Fonte: Elaborado pelos autores.

O diagrama de sequência ao clicar no botão “*e-mail*” (Figura 13) representa a interação entre o usuário, o aplicativo e o aplicativo externo de *e-mail*. Ao clicar no botão “*e-mail*”, o aplicativo abre o aplicativo de *e-mail padrão* do dispositivo do usuário com um *e-mail* de rascunho tendo o destinatário preenchido e uma mensagem pré-definida com as informações da publicação.

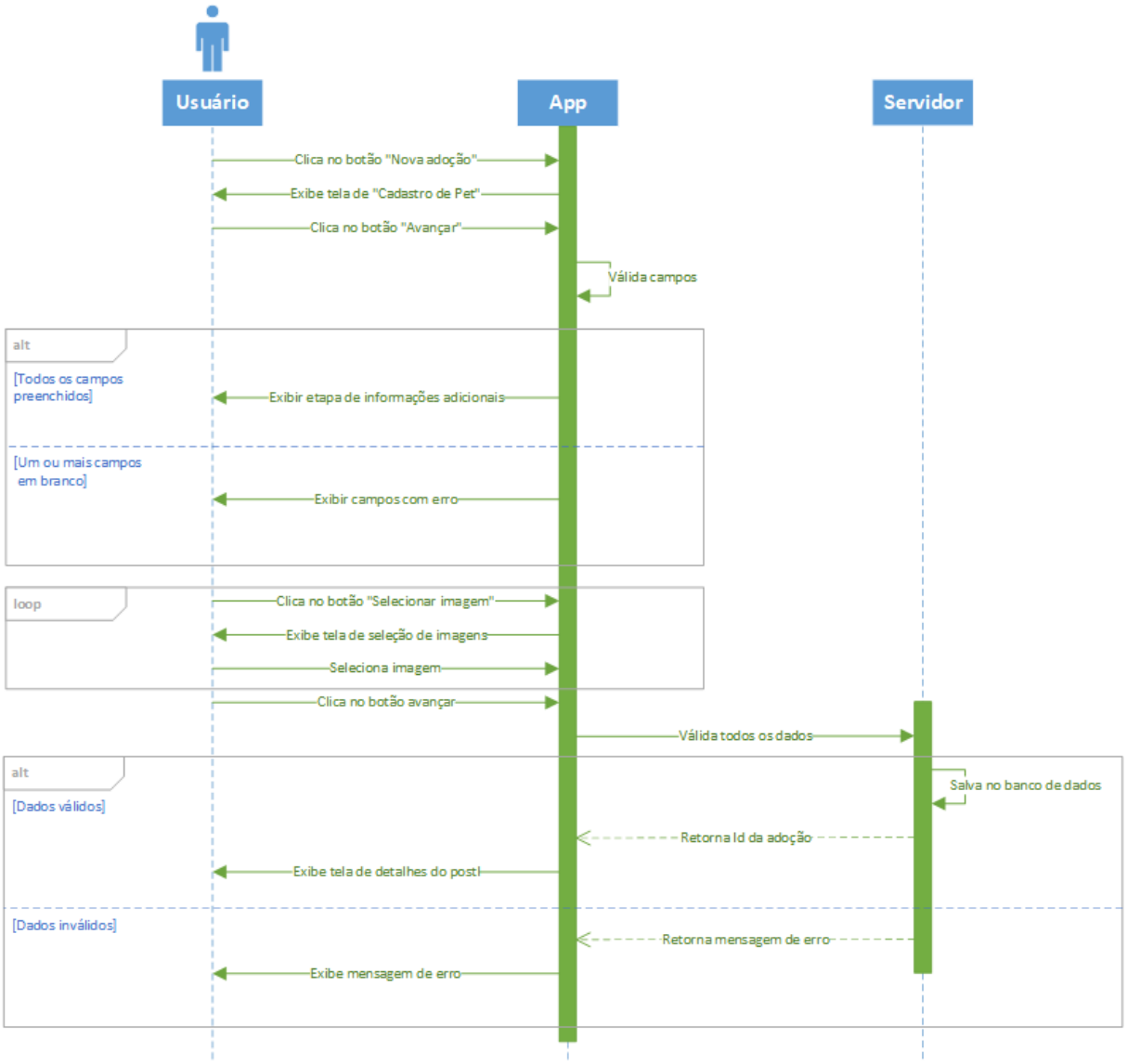
Figura 14 - Diagrama de sequência ao clicar no botão "denunciar".



Fonte: Elaborado pelos autores.

O diagrama de sequência ao clicar no botão “denunciar” (Figura 14) representa o que acontece ao denunciar uma publicação. No momento em que o usuário clicar no botão de denúncia, o aplicativo deve exibir uma lista de categorias ao qual essa denúncia pode se encaixar e junto a uma caixa de mensagens para que o usuário possa detalhar melhor o motivo. Ao clicar em “enviar”, o servidor recebe a denúncia e salva no banco de dados, retorna uma mensagem de sucesso e o aplicativo informa ao usuário que a denúncia foi enviada.

Figura 15 - Diagrama de Sequência ao clicar no botão "nova adoção".



Fonte: Elaborado pelos autores.

O diagrama de sequência ao clicar no botão “nova adoção” (Figura 15) representa o fluxo para o cadastro de um novo animal. Ao clicar no botão “nova adoção”, o aplicativo deve exibir a tela de cadastro de adoção, onde o usuário deve preencher os dados básicos na primeira etapa. Para avançar no cadastro, o aplicativo valida se todos os campos necessários foram preenchidos e então exibe a tela para seleção de fotos do animal e cadastro de informações complementares, como vacinação e cuidados do *pet*.

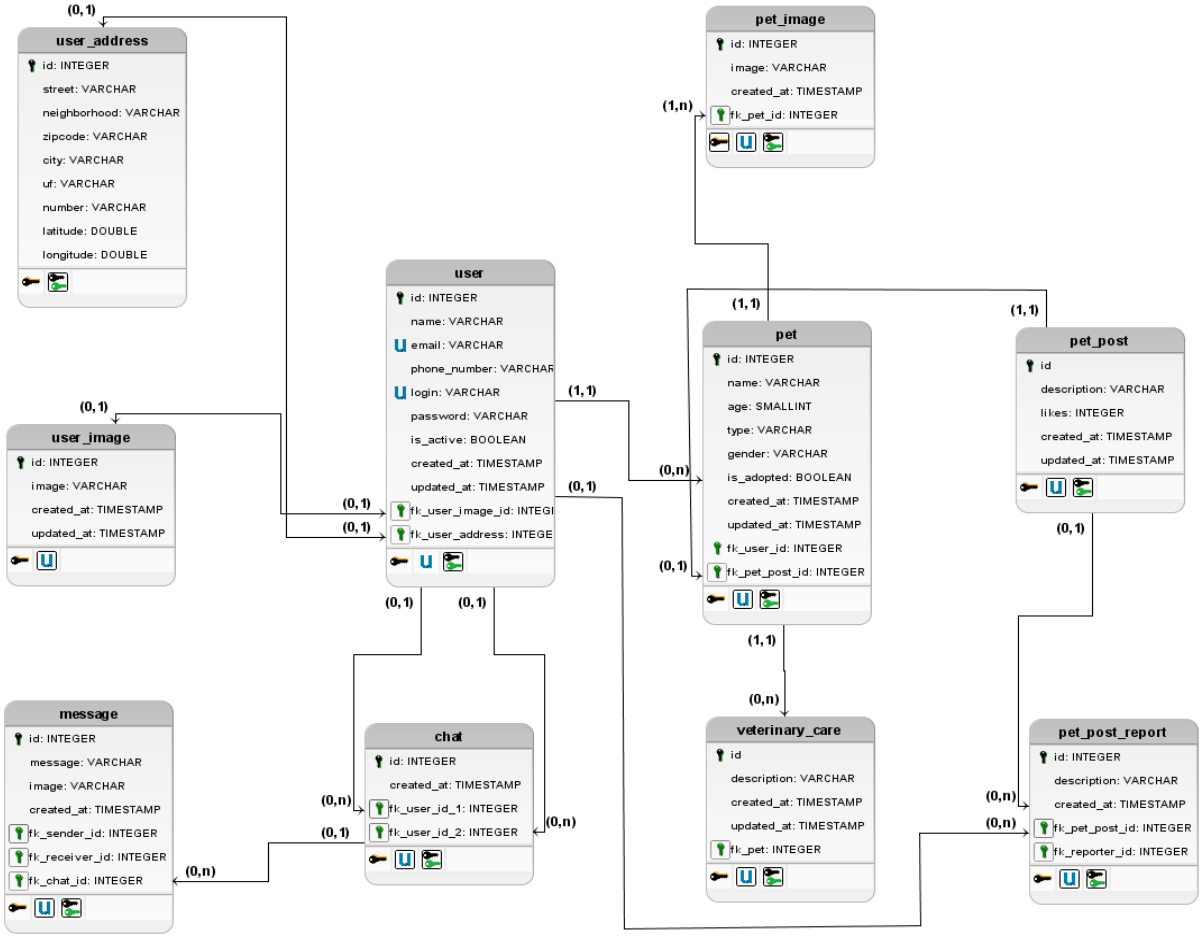
Pressionando o botão “avançar” na última etapa, o aplicativo envia os dados informados pelo usuário e o servidor verifica se as informações são válidas. Caso tudo esteja correto, os dados são salvos no banco de dados e o servidor retorna o ID

da publicação criada e o aplicativo exibe para o usuário a tela de detalhes da publicação. Em caso de erro, o aplicativo deve exibir uma mensagem contendo o motivo do erro.

3.4 Diagrama de Entidade e Relacionamento

Diagrama Entidade Relacionamento (DER) é um modelo diagramático que descreve o modelo de dados de um sistema com alto nível de abstração. Ele é a principal representação do Modelo de Entidades e Relacionamentos. Sua maior aplicação é visualizar o relacionamento entre tabelas de um banco de dados, no qual as relações são construídas através da associação de um ou mais atributos destas tabelas. (SOMMERVILLE, 2011). A Figura 16 apresenta o DER do sistema proposto.

Figura 16 - Diagrama de Entidade e Relacionamento



Fonte: Elaborado pelos autores.

3.5 Dicionário de Dados

O Dicionário de Dados (DD) consiste numa lista organizada de todos os elementos de dados que são pertinentes ao sistema.

Tabela 1 - Dicionário de Dados da entidade User.

Entidade: User				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
name	Simples	Varchar	50	Nome completo
email	Simples	Varchar	120	E-mail
phone_number	Simples	Varchar	64	Número de telefone
login	Simples	Varchar	32	Usuário para login
password	Simples	Varchar	32	Senha para login
is_active	Simples	Boolean	1	Identificador se a conta está ativa ou não
created_at	Simples	Timestamp	13	Data de criação do usuário
updated_at	Simples	Timestamp	13	Data da última modificação do usuário
fk_user_image	Determinante	Integer	32	Chave estrangeira para imagem do usuário
fk_user_address	Determinante	Integer	32	Chave estrangeira para endereço do usuário

Fonte: Elaborado pelos autores.

Tabela 2 - Dicionário de Dados da entidade User_address.

Entidade: User_address				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
street	Simples	Varchar	50	Rua
neighborhood	Simples	Varchar	120	Bairro
zipcode	Simples	Varchar	32	CEP
city	Simples	Varchar	32	Cidade
uf	Simples	Varchar	2	Estado (UF)
number	Simples	Boolean	1	Número da casa
latitude	Simples	Double	8	Latitude da localização do usuário
longitude	Simples	Double	8	Longitude da localização do usuário
created_at	Simples	Timestamp	13	Data de criação do endereço
updated_at	Simples	Timestamp	13	Data da última modificação do endereço

Fonte: Elaborado pelos autores.

Tabela 3 - Dicionário de Dados da entidade User_image.

Entidade: User_image				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
image	Simples	Varchar	1000	Caminho para imagem do usuário
created_at	Simples	Timestamp	13	Data de criação da imagem
updated_at	Simples	Timestamp	13	Data da última modificação da imagem

Fonte: Elaborado pelos autores.

Tabela 4 - Dicionário de Dados da entidade Chat.

Entidade: Chat				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
created_at	Simples	Timestamp	13	Data de criação da imagem
fk_user_id_1	Simples	Integer	32	Chave estrangeira para um dos participantes
fk_user_id_2	Simples	Integer	32	Chave estrangeira para um dos participantes

Fonte: Elaborado pelos autores.

Tabela 5 - Dicionário de Dados da entidade Message.

Entidade: Chat				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
message	Simples	Varchar	256	Mensagem enviada
image	Simples	Varchar	1000	Caminho para imagem enviada
created_at	Simples	Timestamp	13	Data de envio da mensagem
fk_sender_id	Simples	Integer	32	Chave estrangeira para o remetente
fk_receiver_id	Simples	Integer	32	Chave estrangeira para o receptor
fk_chat_id	Simples	Integer	32	Chave estrangeira para o chat

Fonte: Elaborado pelos autores.

Tabela 6 - Dicionário de Dados da entidade Pet.

Entidade: Pet_image				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
image	Simples	Varchar	1000	Caminho para imagem do pet
created_at	Simples	Timestamp	13	Data de criação do pet
fk_pet_id	Determinante	Integer	32	Chave estrangeira para o pet atrelado
is_adopted	Simples	Boolean	1	Identificador se o pet já foi adotado ou não
created_at	Simples	Timestamp	13	Data de criação do pet
updated_at	Simples	Timestamp	13	Data da última modificação do pet
fk_user_id	Determinante	Integer	32	Chave estrangeira para o usuário responsável
fk_post_id	Determinante	Integer	32	Chave estrangeira para o post do pet

Fonte: Elaborado pelos autores.

Tabela 7 - Dicionário de Dados da entidade Pet_image.

Entidade: Pet_image				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
image	Simples	Varchar	1000	Caminho para imagem do pet
created_at	Simples	Timestamp	13	Data de criação do pet
fk_pet_id	Determinante	Integer	32	Chave estrangeira para o pet atrelado

Fonte: Elaborado pelos autores.

Tabela 8 - Dicionário de Dados da entidade Pet_post.

Entidade: Pet_post				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
description	Simples	Varchar	256	Descrição
likes	Simples	Integer	32	Número de curtidas no post
created_at	Simples	Timestamp	13	Data de criação do pet
updated_at	Simples	Timestamp	13	Data da última modificação do pet

Fonte: Elaborado pelos autores.

Tabela 9 - Dicionário de Dados da entidade Veterinary_care.

Entidade: Veterinary_care				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
description	Simples	Varchar	256	Descrição
created_at	Simples	Timestamp	13	Data de criação do pet
updated_at	Simples	Timestamp	13	Data da última modificação do pet
fk_pet_id	Determinante	Integer	32	Chave estrangeira para o pet atrelado

Fonte: Elaborado pelos autores.

Tabela 10 - Dicionário de Dados da entidade Pet_post_report.

Entidade: Pet_post_report				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
id	Determinante	Integer	32	Identificador universal composto por 32 caracteres
description	Simples	Varchar	256	Descrição
created_at	Simples	Timestamp	13	Data de criação do pet
fk_pet_post_id	Determinante	Integer	32	Chave estrangeira para o post atrelado
fk_reporter_id	Determinante	Integer	32	Chave estrangeira para usuário que reportou

Fonte: Elaborado pelos autores.

4 PROJETO PROPOSTO

4.1 API

API – *Application Programming Interface* ou Interface de Programação da Aplicação é o nome utilizado para “especificação de protocolos, procedimentos e serviços que podem ser utilizados [...] para implementar um requerimento” (SAMOYLOV, 2018).

Uma interface de programação de aplicativos, ou API, permite que as empresas disponibilizem os dados e a funcionalidade de seus aplicativos para desenvolvedores externos de terceiros, parceiros de negócios e departamentos internos de suas empresas. Isso permite que serviços e produtos se comuniquem entre si e aproveitem os dados e a funcionalidade uns dos outros por meio de uma interface documentada. Os desenvolvedores não precisam saber como uma API é implementada; eles simplesmente usam a interface para se comunicar com outros produtos e serviços. O uso de APIs aumentou na última década, a ponto de muitos dos aplicativos da Web mais populares hoje não seriam possíveis sem APIs (IBM, 2020).

4.1.1 API Google Maps

Nessa aplicação foi utilizada a API do *Google Maps* para criação do sistema de geolocalização do sistema, permitindo que os usuários visualizem adoções próximas à sua localização atual num raio de 10 quilômetros por padrão. Essa API é criada e documentada pelo Google.

4.1.2 API ViaCEP

A API *ViaCEP* foi utilizada para consultar endereços a partir do CEP digitado pelo usuário, com o objetivo de preencher as informações automaticamente, facilitando o cadastro.

4.2 Tecnologia Empregada

Esta seção contempla as ferramentas de programação e os conceitos necessários para o desenvolvimento do sistema:

4.2.1 Eclipse

É um ambiente de desenvolvimento integrado (**IDE**) para o desenvolvimento de aplicativos em geral, mas com ênfase em *JAVA*, que oferece um ambiente unificado para o desenvolvimento de aplicativos no qual é possível desenvolver, realizar *debugs* e testes. A *IDE* foi utilizada para escrever os códigos do servidor do aplicativo, pois ela permite uma maior produtividade, com diversos facilitadores como preenchimento e indentação[3] automática de código e depurador integrado ao editor, permitindo um melhor acompanhamento da execução do programa.

4.2.2 Visual Studio Code

O *Visual Studio Code* é um editor de código-fonte leve, mas poderoso, que é executado em sua área de trabalho e está disponível para *Windows*, *macOS* e *Linux*. Ele vem com suporte integrado para *JavaScript*, *TypeScript* e *Node.js* e possui um

[3] neologismo derivado da palavra em inglês *indentation.*

rico ecossistema de extensões para outras linguagens (como *C++*, *C#*, *Java*, *Python*, *PHP*, *Go*) e *runtimes* (como *.NET* e *Unity*). (MICROSOFT, 2015).

4.2.3 Java

O *Java* é uma linguagem de programação orientada a objetos e é uma das linguagens mais utilizadas pelas empresas na atualidade no desenvolvimento de aplicações *web* e *mobile*. Foi criado em 1995 na empresa *Sun Microsystem* por uma equipe chefiada por James Gosling, conhecido como o pai do Java. Em 2008, o Java foi adquirido pela Oracle e vem crescendo bastante desde então no mercado de tecnologia. (ZUP, 2021).

A linguagem foi utilizada para o desenvolvimento do servidor da aplicação, pois disponibiliza um vasto material para estudos da tecnologia e diversas bibliotecas desenvolvidas pela comunidade para facilitar a construção do servidor. Na figura 17 é possível conferir um trecho de código utilizado para gerar o *token* de acesso do usuário através da biblioteca *open-source JJWT*.

Figura 17 - Trecho de código utilizado para geração de *token* de acesso do usuário.

```
public String generateToken( User user ){
    long expString = Long.valueOf(expirationMinutes);
    LocalDateTime dateHourExpiration = LocalDateTime.now().plusMinutes(expString);
    Instant instant = dateHourExpiration.atZone(ZoneId.systemDefault()).toInstant();
    Date date = Date.from(instant);

    return Jwts
            .builder()
            .setSubject(user.getLogin())
            .setExpiration(date)
            .signWith( SignatureAlgorithm.HS512, signatureKey )
            .compact();
}
```

Fonte: Elaborado pelos autores.

4.2.4 Spring Boot

É um framework *Java* de código aberto que tem como objetivo facilitar esse processo em aplicações *Java*. Consequentemente, ele traz mais agilidade para o processo de desenvolvimento, uma vez que os desenvolvedores conseguem reduzir o tempo gasto com as configurações iniciais. (ZUP, 2021).

O *Spring Boot* permite construir aplicações web e microsserviços em *Java* de maneira simplificada, disponibilizando diversas funcionalidades para criação de *endpoints* para APIs, drivers para comunicação com banco de dados, validação de acesso de usuários, dentre outras. Na figura 18 podemos conferir a criação de um *endpoint* para registrar um usuário na aplicação.

Figura 18 - Trecho de código para criação de um *endpoint* de registro de usuário com *Spring Boot*

```
@PostMapping("/add")
public ResponseEntity<?> create(@RequestBody User user) {
        Role roleUser = roleRepository.findByName("USER");
        try {
                user.addRole(roleUser);

                user.setPassword(encoder.encode(user.getPassword()));
                userRepository.save(user);
                return new ResponseEntity<>(user, HttpStatus.OK);

        } catch (Exception e) {
                e.printStackTrace();
                return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Create user failed");

        }

}
```

Fonte: Elaborado pelos autores.

4.2.5 JavaScript

É uma linguagem de *script* ou programação que permite implementar recursos complexos em páginas da web. Com o surgimento de *frameworks* como *React Native*,

a linguagem também passou a ser utilizada para o desenvolvimento de aplicativos móveis, o qual foi escolhido para desenvolver o protótipo.

Utilizando o *JavaScript* é possível consumir APIs de terceiros (como demonstrado na figura 19 onde é possível ver como é feito o consumo de uma API do serviço *ViaCep* para conseguir o endereço do usuário na aplicação), criar jogos e programas complexos como o *Google Maps* e trabalhar com Inteligência Artificial.

Figura 19 – Código JavaScript consumindo a API *ViaCep* para trazer o endereço do usuário.

```
const getAddressByZipCode = async () => {
  const zipcode = formRef.current.getFieldValue('adress.zipcode');
  if (zipcode.length < 9) return;
  try {
    const { data } = await api.get(`http://viacep.com.br/ws/${zipcode}/json/`);
    formRef.current.setData({
      ...formRef.current?.getData(),
      adress: {
        city: data.localidade,
        state: data.uf,
        neighbourhood: data.bairro,
        street: data.logradouro,
        longitude: formRef.current?.getFieldValue('adress.longitude'),
        latitude: formRef.current?.getFieldValue('adress.latitude'),
      }
    });
  } catch (err) {
    alert('CEP não encontrado')
  }
}
```

Fonte: Elaborado pelos autores.

4.2.6 Figma

É um editor de design gráfico online gratuito focado na criação de interfaces gráficas e experiências do usuário com foco na colaboração. A ferramenta permite que equipes de designers acessem e trabalhem no mesmo projeto ao mesmo tempo,

facilitando a vida de equipes de profissionais que trabalham remotamente em vários locais. O *Figma* foi utilizado para prototipar todas as telas do aplicativo.

4.2.7 React Native

É uma das tecnologias mais utilizadas *do JavaScript* e utilizada para construir uma interface de usuário (**IU**). Ela oferece uma resposta excelente para o usuário adicionar comandos usando um novo método de renderizar sites e aplicativos. É uma biblioteca criada pelo *Facebook*, desenvolvida para aplicações *mobile* (*Android* e *iOS*) reaproveitando o código para ambas as plataformas.

O *React Native* permite o desenvolvimento de uma interface componentizável, garantindo uma melhor padronização e reutilização de código. Na figura 20 é possível conferir um trecho onde foi utilizado um componente feito pela comunidade (*MapView*) para exibir o mapa dos animais próximos ao usuário.

Figura 20 - Utilização de um componente para exibir um mapa na tela.

```
<MapView
  style={{ flex: 1 }}
  onRegionChangeComplete={handleRegionChanged}
  initialRegion={currentRegion}
  initialCamera={{
    pitch: 45,
    heading: 90,
    altitude: 20,
    zoom: 2
  }}
>
  {pets.map((post, index) => (
      <Marker key={index} coordinate={{ longitude: post.longitude, latitude: post.latitude }}>
        <Image style={styles.avatar} source={{ uri: `http://192.168.50.126:8080/${post.image}` }} />
        <Callout onPress={() => {
          navigateToPetDetail(post.id);
        }} >
          <View style={styles.callout}>
            <Text style={styles.petName}>
              {post.name}
              <MaterialCommunityIcons
                name={post.gender && post.gender.toLocaleLowerCase() === 'm' ? 'gender-male' : 'gender-female'}
                size={16}
                color={post.gender && post.gender.toLocaleLowerCase() === 'm' ? '#00ADEF' : '#EA168F'}
              />
            </Text>
            <Text style={styles.petTechs}>Porte {translatePetSize(post.size)}</Text>
            <Text style={styles.petBio}>{post.description}</Text>
          </View>
        </Callout>
      </Marker>
    ))}
</MapView>
```

Fonte: Elaborado pelos autores.

4.2.8 MYSQL

MySQL é o banco de dados de código aberto mais popular do mundo. Com seu desempenho, confiabilidade e facilidade de uso comprovados, o *MySQL* tornou-se a principal escolha de banco de dados para aplicativos baseados na Web, usados por propriedades da *Web* de alto perfil, incluindo *Facebook*, *Twitter*, *YouTube*, *Yahoo!* e muitos mais. (ORACLE, 2013).

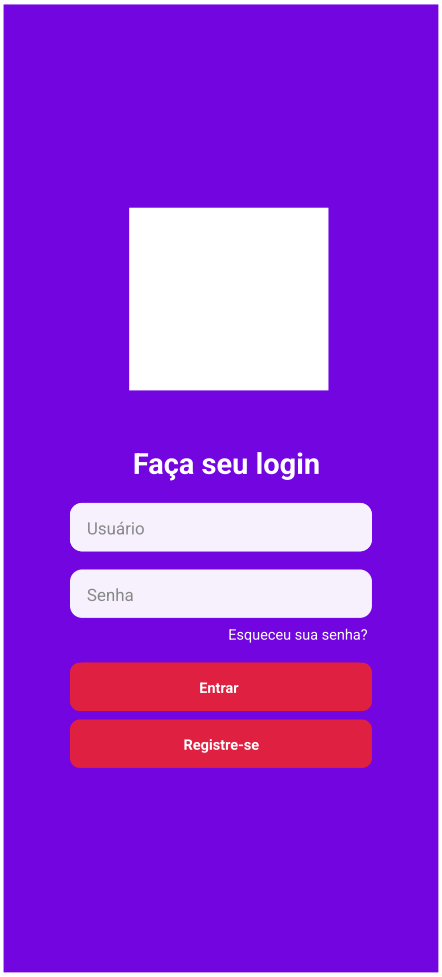
4.3 Telas do Projeto

A necessidade da construção de uma interface amigável ao usuário é fundamental em um sistema. A interface faz parte do sistema computacional e determina como as pessoas operam e controlam o sistema. Quando uma interface é bem projetada, ela é compreensível, agradável e controlável. Neste capítulo, será descrito e ilustrado as interfaces do protótipo desenvolvido.

4.3.1 Tela de login

A Figura 21 apresenta a tela de *login* do sistema, na qual são apresentados 2 campos de texto para usuário e senha, seguido de um botão para realizar a autenticação no sistema e outro para cadastrar uma nova conta. Há também um botão para recuperação de senha, caso necessário. Caso os dados não estejam cadastrados, o usuário recebe um alerta informando-o. Caso os dados estejam cadastrados, o usuário é redirecionado para a tela principal do aplicativo (*feed* de animais).

Figura 21 - Tela de Login



Fonte: Elaborado pelos autores.

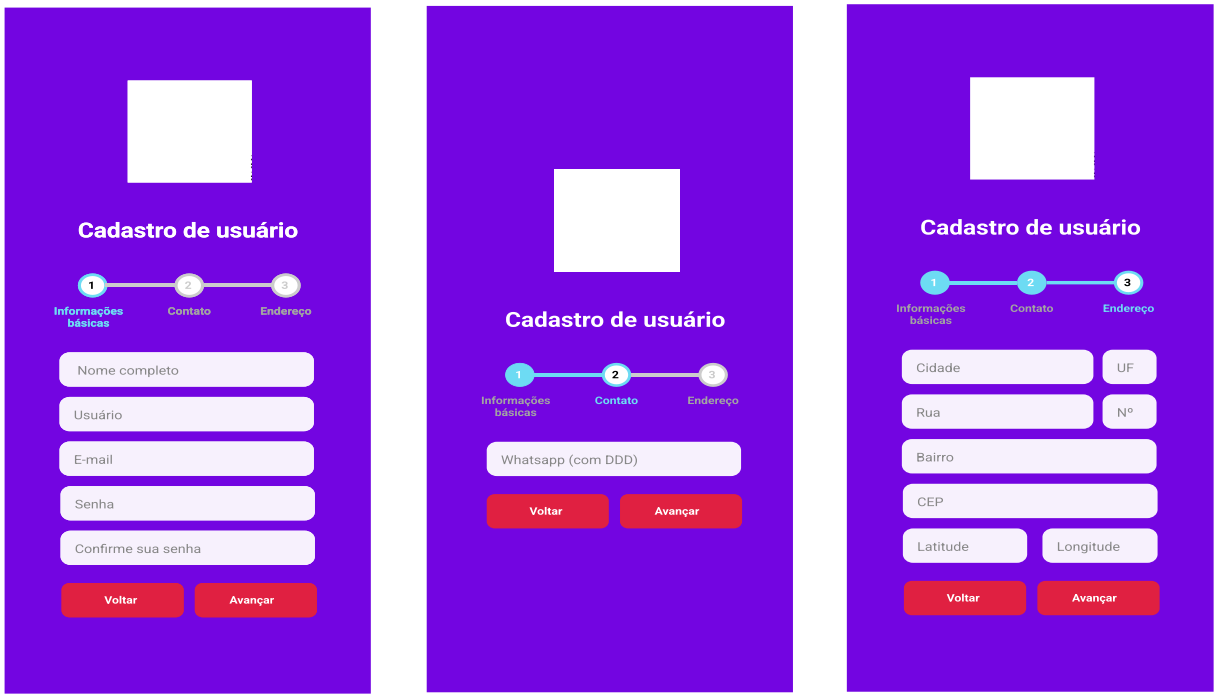
A tela de *login* apresentada na Figura 21 é composta por:

- **Campo usuário:** Para preenchimento do *login* do usuário.
- **Campo senha:** Para preenchimento da senha do usuário.
- **Botão Entrar:** Botão que realiza a autenticação na aplicação.
- **Botão Registre-se:** Botão que redireciona o usuário para a tela de cadastro.
- **Texto “Esqueceu sua senha?”:** Redireciona o usuário para a tela de recuperação de senha da aplicação.

4.3.2 Tela de Cadastro

A Figura 22 apresenta a tela de cadastro do sistema, na qual são apresentados vários campos separados em passos (Informações básicas, Contato e Endereço), sendo eles: Nome, usuário, e-mail, senha, confirmação de senha, WhatsApp, cidade, uf, rua, número, bairro, cep, latitude e longitude. Todos os passos possuem validação dos campos para impedir que o usuário avance sem preencher os obrigatórios. Ao entrar na tela, o usuário recebe um *pop-up* pedindo acesso à sua localização para preenchimento automático dos campos latitude e longitude (não obrigatórios).

Figura 22 - Tela de Cadastro.



Fonte: Elaborado pelos autores.

A tela de cadastro apresentada na Figura 22 é composta por:

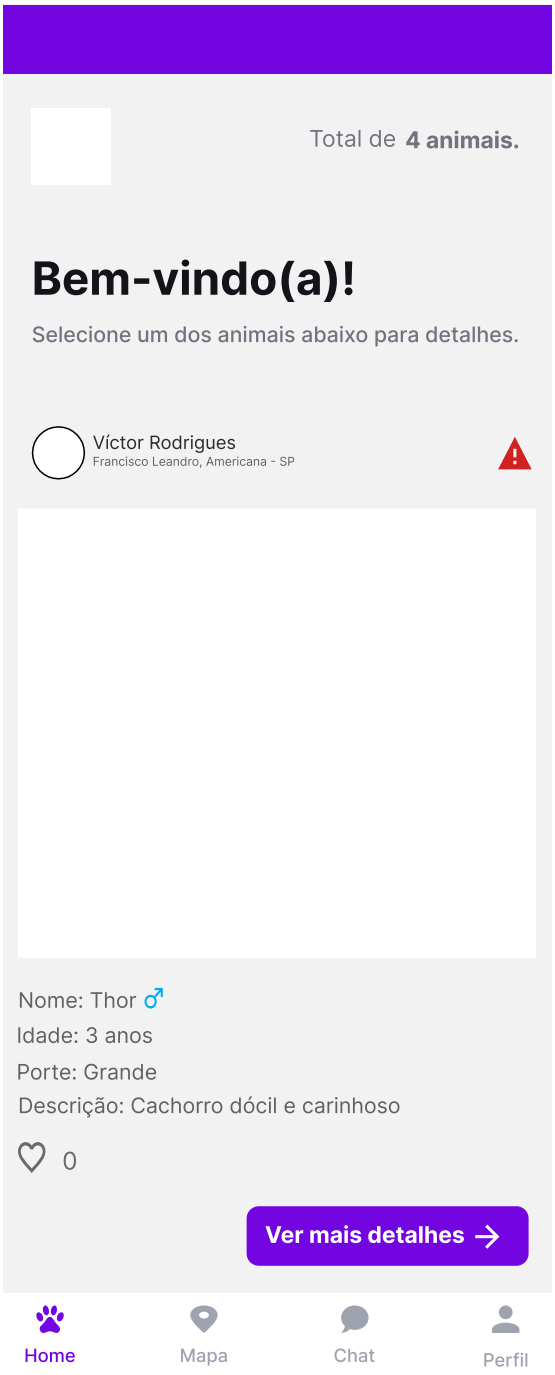
- **Campo nome completo:** Para preenchimento do nome completo do usuário.
- **Campo usuário:** Para preenchimento do usuário.
- **Campo e-mail:** Para preenchimento do e-mail do usuário.
- **Campo senha e confirme sua senha:** Para preenchimento da senha única.

- **Campo *WhatsApp*:** Para preenchimento do número de telefone para contato via *WhatsApp* do usuário.
- **Campo cidade:** Para preenchimento da cidade do usuário.
- **Campo UF:** Para preenchimento do estado (UF) do usuário.
- **Campo rua:** Para preenchimento da rua do usuário.
- **Campo número:** Para preenchimento do número da casa do usuário.
- **Campo bairro:** Para preenchimento do bairro do usuário.
- **Campo cep:** Para preenchimento do CEP do usuário.
- **Campo latitude:** Preenchido automaticamente através da localização do smartphone do usuário.
- **Campo altitude:** Preenchido automaticamente através da localização do smartphone do usuário.
- **Botão Voltar:** Para redirecionar o usuário para o passo anterior ou para a tela de login caso seja o primeiro passo.
- **Botão Avançar:** Para redirecionar o usuário para o passo posterior ou finalizar o cadastro caso seja o último passo.

4.3.3 Tela Home

A Figura 23 apresenta a *Home* do sistema, na qual é apresentada uma lista de animais para adoção, priorizando a proximidade ao usuário, trazendo informações básicas como nome, sexo, idade, descrição e um carrossel contendo as fotos de cada animal. Além disso, contam também com um botão para reportar um *post* para os administradores e outro para redirecioná-los para a tela de detalhes do animal.

Figura 23 - Tela *Home*.



Fonte: Elaborado pelos autores.

A tela *Home* apresentada na Figura 23 é composta por:

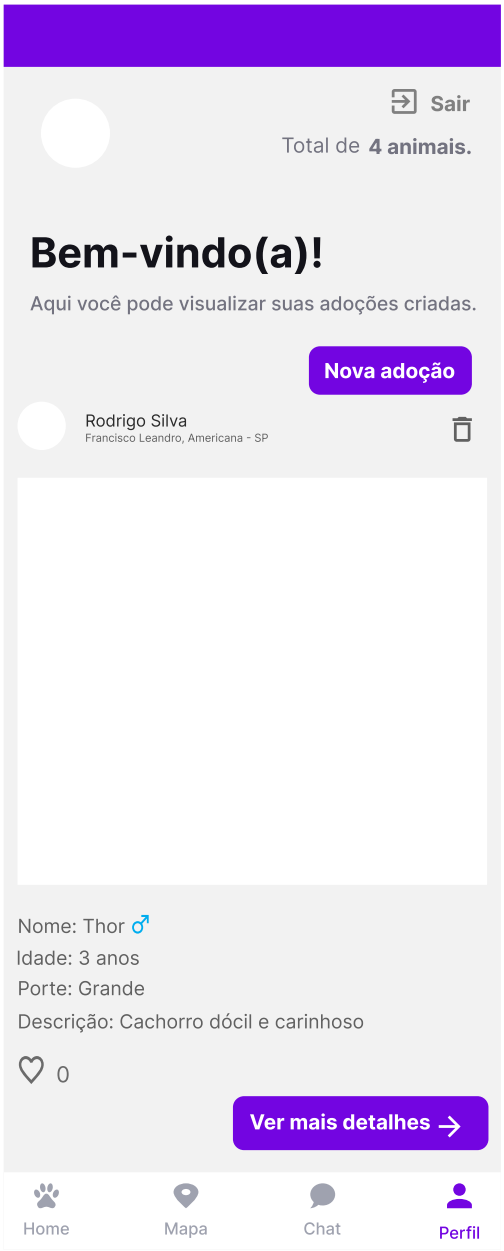
- **Texto Total de animais:** Apresenta o total de animais encontrados para adoção.

- **Lista de *post*:** Uma lista com os animais encontrados, cada uma contendo nome do criador da adoção, endereço, carrossel de fotos, nome, idade, porte e descrição do animal, botão para *report*, botão de curtida e botão para visualizar mais detalhes sobre o animal.
- **Barra de navegação:** Para acesso às outras partes do aplicativo.

4.3.4 Tela de Perfil

A Figura 24 apresenta o perfil do usuário, na qual é apresentada uma lista de animais para adoção criadas pelo mesmo, trazendo informações básicas como nome, sexo, idade, descrição e um carrossel contendo as fotos de cada animal. Além disso, contam também com um botão para exclusão da adoção no sistema e outro para redirecioná-lo para a tela de detalhes do animal. A tela traz também uma contagem no canto superior direito indicando o total de adoções criadas, um botão para realizar o *logout* da aplicação e outro para a criação de uma nova adoção.

Figura 24 - Tela de Perfil.



Fonte: Elaborado pelos autores.

A tela de perfil apresentada na Figura 24 é composta por:

- **Texto Total de animais:** apresenta o total de adoções criados pelo usuário.
- **Botão *Logout*:** Para logout do usuário na aplicação.
- **Lista de *post*:** Uma lista com os animais encontrados, cada uma contendo nome do criador da adoção, endereço, carrossel de fotos, nome, idade, porte e

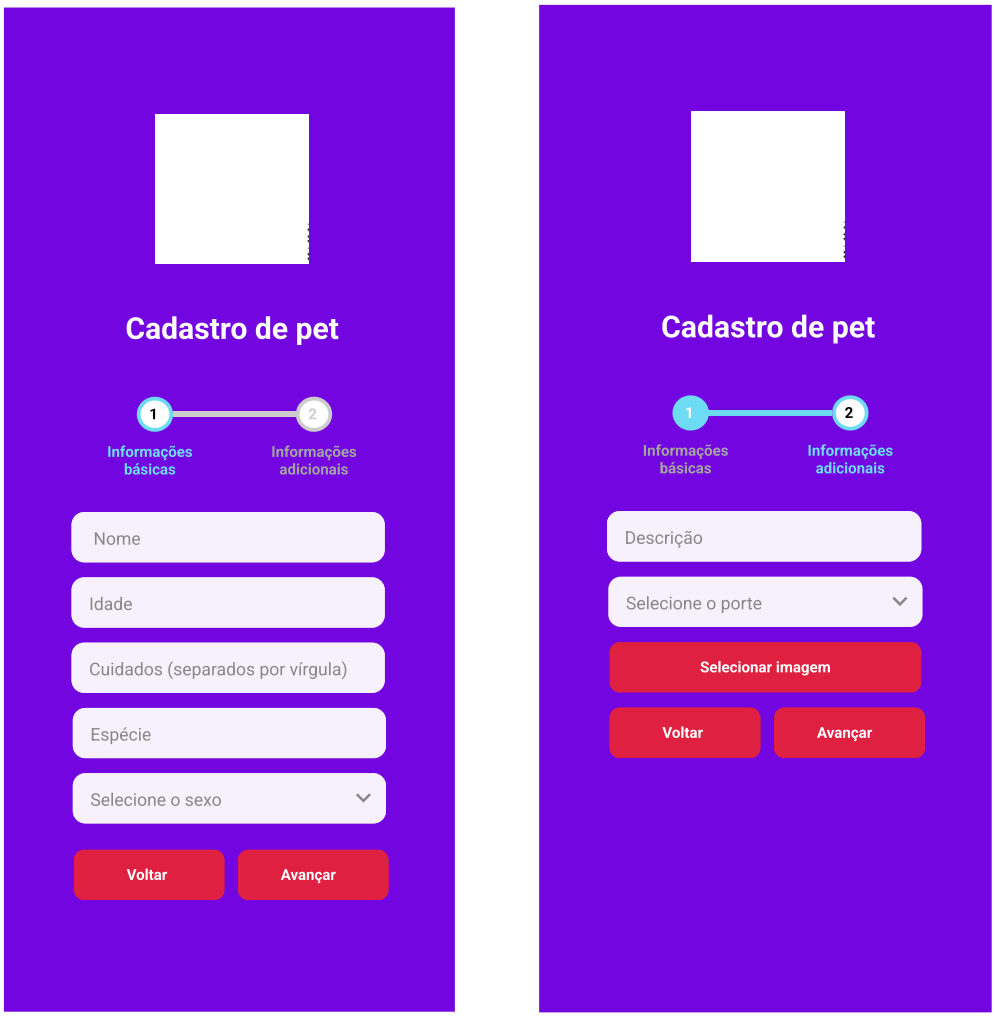
descrição do animal, botão para exclusão, botão de curtida e botão para visualizar mais detalhes sobre o animal.

- **Barra de navegação:** Para acesso às outras partes do aplicativo.

4.3.5 Tela de cadastro de animal

A Figura 25 apresenta o cadastro de uma nova adoção no sistema, na qual são apresentados vários campos separados em passos (Informações básicas e Informações Adicionais) sendo eles: Nome, idade, cuidados veterinários, espécie, sexo, descrição, porte e fotos.

Figura 25 - Tela de cadastro de adoção.



Fonte: Elaborado pelos autores.

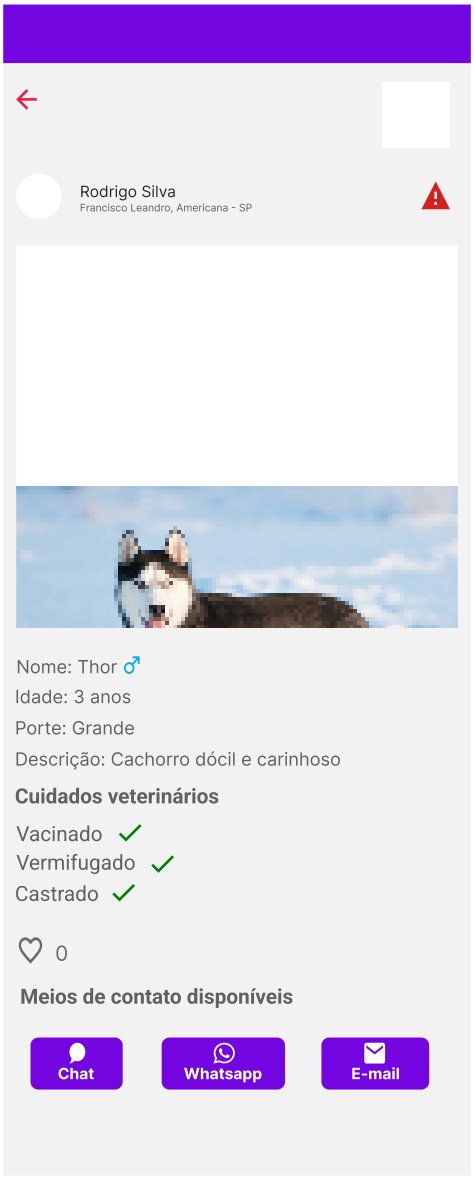
A tela de cadastro de animal apresentada na Figura 25 é composta por:

- **Campo nome:** Para preenchimento do nome do animal.
- **Campo idade:** Para preenchimento da idade do animal.
- **Campo cuidados:** Para preenchimento dos cuidados veterinários do animal.
- **Campo espécie:** Para seleção da espécie do animal.
- **Campo sexo:** Para seleção do sexo do animal.
- **Campo descrição:** Para preenchimento da descrição do animal
- **Campo porte:** Para seleção do porte do animal.
- **Botão Selecionar Imagem:** Para que o usuário possa inserir uma ou várias imagens atreladas à adoção.
- **Botão Voltar:** Para redirecionar o usuário para o passo anterior ou para a tela de login caso seja o primeiro passo.
- **Botão Avançar:** Para redirecionar o usuário para o passo posterior ou finalizar o cadastro caso seja o último passo.

4.3.6 Tela de detalhes da adoção

A Figura 26 apresenta os detalhes da adoção, na qual é apresentado todas as informações sobre. Além disso, também conta novamente com o botão de *report* e a lista de meios de contato disponíveis para se comunicar com o doador.

Figura 26 - Tela de detalhes da adoção.



Fonte: Elaborado pelos autores.

A tela de detalhes da adoção apresentada na Figura 26 é composta por:

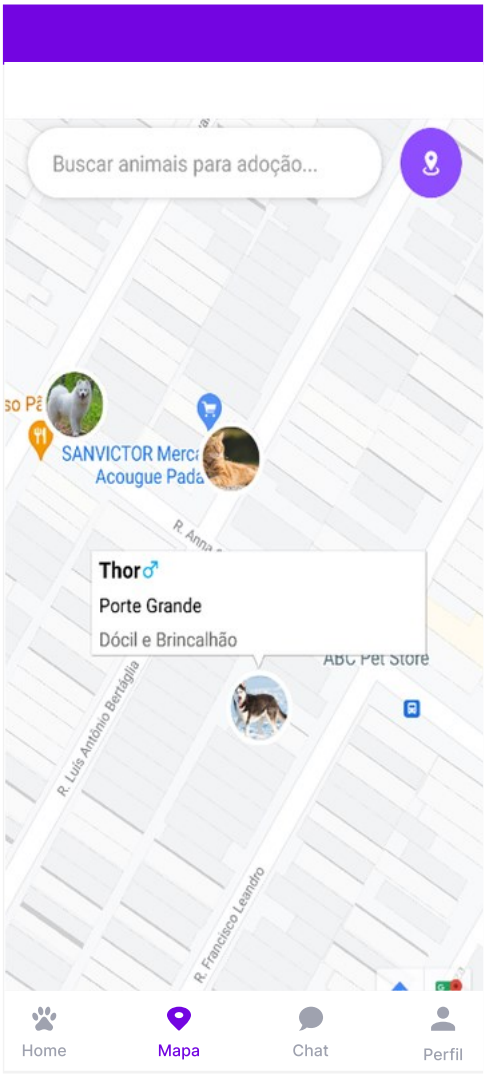
- **Card de informações:** Apresenta todas as informações do animal como nome, idade, porte, descrição, fotos e cuidados veterinários.
- **Botão de reportar:** Para reportar a adoção para os administradores.
- **Botão Voltar:** Para redirecioná-lo para a página anterior.
- **Botão Chat:** Para redirecioná-lo ao chat para contato com o doador.

- **Botão *WhatsApp*:** Para redirecioná-lo para o aplicativo *WhatsApp* direto no chat com o doador com uma mensagem pré-escrita.
- **Botão *E-mail*:** Para redirecioná-lo para o aplicativo de *e-mails* direto na criação de um novo *e-mail*, com destinatário e corpo do *e-mail* pré-escritos.

4.3.7 Tela do Mapa

A Figura 27 apresenta o mapa do aplicativo, no qual é mostrado um mapa com a localização atual do usuário contendo uma foto de cada animal encontrado num raio de 10 quilômetros em suas respectivas coordenadas. Também possui um campo de busca para filtragem de animais caso o usuário desejar.

Figura 27 - Tela do Mapa.



Fonte: Elaborado pelos autores.

A tela do mapa apresentada na Figura 27 é composta por:

- **Mapa:** Um mapa na localização atual do usuário
- ***Card* de animal:** Um *card* contendo a foto dos animais encontrados na região
- **Barra de navegação:** Para acesso às outras partes do aplicativo

5 CONCLUSÃO

Este trabalho teve como objetivo final, desenvolver uma aplicação para as plataformas *Android* e *iOS* que conectasse pessoas e instituições que estão doando animais de estimação com pessoas que desejam adotar um animal. Utilizamos as linguagens de programação *Java e JavaScript* com auxílio de seus respectivos *Frameworks Spring Boot e React Native*, juntamente com o Banco de Dados *MySQL*.

Não houve dificuldades expressivas no desenvolvimento desse Projeto, uma vez que a equipe já possuía os conhecimentos necessários que foram adquiridos durante o curso e em trabalho profissional.

O sistema não foi desenvolvido por completo devido aos integrantes da equipe já trabalharem efetivamente na área realizando muitas horas extras, inclusive aos fins de semana, juntamente com trabalhos de diversas matérias da faculdade, o que ocasionou na falta de tempo para o desenvolvimento prático completo do sistema. Mas o código atual se encontra no *GitHub* dos integrantes da equipe.

Por fim, o protótipo foi desenvolvido com sucesso, excedendo as expectativas da equipe. Espera-se que a versão final do Projeto seja desenvolvida e disponibilizada nas plataformas na *App Store* e *Play Store* para *download* público e de código aberto.

O Projeto também permitiu a equipe a experiência no desenvolvimento, bem como o aprendizado no trabalho em equipe. Além disso, também agregou diversos conhecimentos técnicos a respeito das tecnologias utilizadas e engenharia de *software*.

REFERÊNCIAS

A. O´BRIEN, James. **Sistemas De Informação e as decisões gerencias na era da Internet**. 9. ed. São Paulo: Saraiva, 2004. ISBN 9788502032767.

AUGUSTO, Gustavo. **Java**: tudo o que você precisa saber para começar. 13 maio 2021. Disponível em: <https://www.zup.com.br/blog/java/>. Acesso em: 13 jun. 2022.

BANCO de Dados MySQL. Disponível em: <https://www.mysql.com/>. Acesso em: 13 jun. 2022.

C. LAUDON, Kenneth; P. LAUDON, Jane. **Sistemas de informação gerenciais**: administrando a empresa digital. 5. ed. São Paulo: Pearson Prentice Hall, 2005. 584 p. ISBN 9788587918390.

GONÇALVES, Leandro Salenave. **Sistema de informação**. v. 1, p. 05-12, 2012.

IBM CLOUD EDUCATION. **Application programming interface (API)**. 19 ago. 2020. Disponível em: <https://www.ibm.com/cloud/learn/api/>. Acesso em: 13 jun. 2022.

LEMOS, Simone. **Cresce o número de adoções e de abandono de animais na pandemia**. 17 jun. 2021. Disponível em: <https://jornal.usp.br/atualidades/cresce-o-numero-de-adocoes-e-de-abandono-de-animais-na-pandemia/>. Acesso em: 13 jun. 2022.

MARTUCCI, Mariana. **Abandono de animais aumentou cerca de 60% durante a pandemia**. 27 dez. 2021. Disponível em: <https://exame.com/bussola/abandono-de-animais-aumentou-cerca-de-60-durante-a-pandemia/>. Acesso em: 13 jun. 2022.

MICROSOFT. **Documentation for Visual Studio Code**. 3 nov. 2021. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 13 jun. 2022.

O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML. DEVMEDIA. 2012. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 25 jun. 2022.

ROSSALLI, Bárbara. **Spring Boot**: como começar. 1 jun. 2021. Disponível em: <https://www.zup.com.br/blog/spring-boot>. Acesso em: 13 jun. 2022.

SAMOYLOV, Nick. **Introduction to programming**. Disponível em: <https://www.safaribooksonline.com/library/view/introduction-toprogramming/9781788839129/4738e05d-c2cb-4541-ad0f-cf5b61d2d62b.xhtml>. Acesso em: 13 jun. 2022.

SOMMERVILLE, Ian. **Software engineering**. 8. ed. Harlow, England: Addison-Wesley, 2007. 840 p. ISBN 0321313798.

SOMMERVILLE, Ian. **Software engineering**. 9. ed. Boston: Pearson, 2011. ISBN 9780137035151.