Desenvolvimento de aplicativo móvel em Flutter para divulgação de animais achados e perdidos

Dâmaris Sbrizza[1], Guilherme Vaz Pereira[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Farroupilha
95.174-274 – Farroupilha – RS – Brasil

damaris.sb@outlook.com, guilherme.pereira@farroupilha.ifrs.edu.br

Abstract*: According to research made by IBGE in 2019, 39.4 million Brazilian homes have the presence of at least one dog or one cat. Pets are many times considered family members, causing suffering when they are lost. This paper describes the development of a cross-platform mobile app for posting lost or found animals. The app user may receive notifications about animals found in the area and exchange messages with other users. A review of the most used mobile app development approaches as well as cross-platform frameworks is provided, along two scenarios describing the app's different uses.*

Resumo*: De acordo com pesquisa do IBGE de 2019, 39,4 milhões de domicílios brasileiros têm a presença de ao menos um cão ou um gato. Animais de estimação são muitas vezes considerados como membros da família, causando sofrimento quando são perdidos. O presente artigo descreve o desenvolvimento de um aplicativo móvel multiplataforma para o cadastro de animais perdidos ou encontrados na rua. O usuário, além de receber notificações ao se aproximar de uma animal cadastrado no sistema, pode manter contato com outros usuários pelo aplicativo. São revistas as abordagens e frameworks de desenvolvimento de aplicativos móveis mais utilizados, bem como apresentados dois casos de uso do sistema.*

1. Introdução

Os primeiros países a legislarem sobre proteção aos animais foram a França e a Grã-Bretanha. O primeiro, através do Código Penal de 1791, vedando envenenamentos e atentados a cães de guarda. O segundo, em 1822 proibindo maus tratos a animais pertencentes a terceiros. No mesmo ano, nesse país, foi criada a RSPCA (*Royal Society for the Prevention of Cruelty to Animals*), associação que promove o bem-estar dos animais.

No âmbito internacional, a Declaração Universal dos Direitos dos Animais é proclamada em 1978, pela UNESCO em Bruxelas, Bélgica, sendo subscrita, inclusive, pelo Brasil. Em 1987, o Conselho da Europa assina a Convenção Europeia para a Proteção dos Animais de Companhia, que reconhece a obrigação moral do homem de respeitar todas as criaturas vivas e afirma haver "laços particulares existentes entre o homem e os animais de companhia" [Santana, Macgregor, Souza, Oliveira 2004].

A partir da década de 1990, as autoridades deram início a uma série de políticas com o objetivo de conter a disseminação de doenças, bem como reduzir os acidentes causados por animais de rua ou com envolvimento desses.

Pesquisa realizada pelo Instituto Brasileiro de Geografia e Estatística em 2019[1], indica uma parcela relevante de domicílios com animais de estimação, cachorros e gatos. Grande parte dessas pessoas considera seus animais como membros de família.

Em redes sociais, tais como *Facebook*, há diversos canais dedicados tanto à divulgação de animais perdidos como dos animais errantes encontrados. Em Caxias do Sul, são exemplos o grupo *Achados e Perdidos - Animais Caxias do Sul*, com mais de 16.000 membros e *Cadê Meu Bicho Caxias-RS*, com mais de 10.000 membros. As informações divulgadas nessas redes, apesar de terem um alcance amplo, são muito dispersas, uma vez que podem ser relativas a animais encontrados em locais muito afastados do usuário. Um aplicativo móvel tem a possibilidade de centralizar as informações, além de observar a distância atual entre o animal e o usuário e fornecer filtros tanto de características quanto da localização dos animais, tornando a busca mais eficiente.

Tendo em vista as necessidades de apoiar o poder público no controle da população de animais de rua e auxiliar as pessoas a encontrarem seus animais de estimação perdidos, este trabalho propõe um aplicativo *mobile* multiplataforma.

Tal aplicação, desenvolvida com o *framework Flutter*, oferece funcionalidades relacionadas ao cadastro e localização de animais e interação entre os usuários. Por exemplo, é possível que um usuário notifique outro ao avistar um animal relacionado a uma postagem e receba notificações acerca da proximidade de um animal.

O presente artigo está organizado nas seguintes seções: a seção 2 revisa a legislação de proteção animal existente e lista as principais abordagens no desenvolvimento de aplicativos móveis, bem como os *frameworks* multiplataforma mais populares. A seção 3 relaciona as tecnologias e ferramentas utilizadas na construção da aplicação, além de descrever as principais funcionalidades do sistema. Na seção 4 são apresentados dois cenários de uso da aplicação e, por fim, a seção 5 contém as considerações finais sobre o trabalho e sugestões para trabalhos futuros.

2. Referencial Teórico

A Constituição da República Federativa do Brasil de 1988, Art. 255, parágrafo 1º, inciso VII, incumbe ao poder público "promover a educação ambiental em todos os níveis de ensino e a conscientização pública para a preservação do meio ambiente" [Brasil 1988].

A Lei estadual nº 15.363[2] de 5 de novembro de 2019, consolida a legislação relativa à Proteção aos Animais no Estado do Rio Grande do Sul. Segundo o Art. 2º, parágrafo 1º, é vedado "ofender ou agredir fisicamente os animais, sujeitando-os a

[1] Disponível em https://biblioteca.ibge.gov.br/visualizacao/livros/liv101748.pdf.

[2] Disponível em https://ww3.al.rs.gov.br/legis/M010/M0100099.asp?Hid_Tipo=TEXTO&Hid_TodasNormas=65763&hTexto=&Hid_IDNorma=65763

qualquer tipo de experiência capaz de causar sofrimento ou dano, bem como as que criem condições inaceitáveis de existência" [Rio Grande do Sul 2019].

Em Caxias do Sul, a Lei municipal nº 8.542[3] de 7 de agosto de 2020 estabelece o Código Municipal de Proteção aos Animais, determinando as sanções e penalidades administrativas para aqueles que praticarem maus-tratos aos animais e cria o Fundo de Proteção Animal. O Art. 4, inciso V define o que é guarda responsável:

> [...] o conjunto de compromissos assumidos pela pessoa natural ou jurídica - guardiã ou responsável - ao adquirir, adotar ou utilizar um animal, que consiste no atendimento das necessidades físicas, psicológicas, ambientais e de saúde do animal e na prevenção de riscos que este possa causar à comunidade ou ao ambiente, tais como os de potencial agressão, transmissão de doenças ou danos a terceiros. [Caxias do Sul 2020].

Baseado no 6º Relatório do Comitê de Especialistas em Raiva da Organização Mundial de Saúde (OMS) de 1973, teve início um período de captura e extermínio, numa tentativa de controle da superpopulação de animais de rua. Ao longo do tempo, além de não ter sido observada uma redução expressiva na população de animais de rua, foi constatada a utilização de métodos não humanitários de captura, confinamento e extermínio de cães e gatos.

Levando em consideração o fracasso dessa experiência, a OMS elaborou o 8º Relatório do Comitê de Especialistas em Raiva, que recomendou uma série de medidas visando prevenir a superpopulação de animais de rua e o consequente abandono - dentre elas, a educação ambiental voltada à posse responsável [Santana, Macgregor, Souza, Oliveira 2004].

De acordo com a Pesquisa Nacional de Saúde realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE) em 2019, foi estimado que cerca de 46,1 % dos domicílios possuem ao menos um cão e 19,3% possuem ao menos um gato, totalizando 39,4 milhões de domicílios. No contexto da posse responsável, os animais de estimação surgem como membros familiares não-humanos:

> O animal não humano assume a condição de filho na vida dos consortes familiares, ao acordar cumprimentar com primeiro "bom dia"; colocar a refeição; banhar; educar; preocupar-se quando ficam doentes; levar ao veterinário; proporcionar conforto; dar atenção; brincar; levar para passear; o medo de perder aquele ser extremamente dependente. Ser pai ou mãe vai muito além do ser ou não ser imposto pelo ordenamento jurídico, é sentir-se, todos os dias, responsável e amar incondicionalmente aquele ser, independentemente de ser uma pessoa ou um animal. [Belchior and Dias 2020].

A Figura 1 demonstra posts encontrados em um grupo do *Facebook* de divulgação de animais encontrados e perdidos. Além de evidenciar a preocupação com o bem-estar dos animais em questão, pode-se observar o tipo de informações divulgadas, como fotos e local em que foram vistos pela última vez.

---

[3] Disponível em http://hamurabi.camaracaxias.rs.gov.br/Hamurabi-faces/externo/exibicao.jsf?leiId=25835&from=resultados

Figura 1. Posts em rede social sobre animais perdidos e encontrados. Fonte: *Facebook*.

A seguir são revisados tópicos relativos às questões tecnológicas envolvidas no desenvolvimento deste trabalho.

2.1. Abordagens no desenvolvimento de aplicativos móveis

O desenvolvimento de aplicativos móveis para os sistemas operacionais Android e iOS é dificultado tanto pela necessidade de haver uma cobertura de plataformas, em parte, incompatíveis, quanto pela fragmentação de dispositivos nas mais variadas configurações, entre elas, PCs, tablets, relógios e até mesmo vestuário, cada qual com suas capabilidades. O desenvolvimento de aplicativos para a plataforma Android em específico também não é uniforme, devido ao grande número de customizações específicas dos fornecedores [Rieger and Majchrzak 2019].

Neste sentido, o suporte multiplataforma permite que desenvolvedores mantenham apenas uma base de código para servir a uma série de plataformas diferentes. Ao passo que essa abordagem oferece generalizações suficientes para atender a várias plataformas, ela também deve permitir que os desenvolvedores possam aproveitar vantagens específicas dos dispositivos [Rieger and Majchrzak 2019].

Aplicativos móveis são desenvolvidos seguindo uma de três abordagens: aplicativos nativos, aplicativos *web* ou aplicativos híbridos [Jabangwe, Edison, Duc 2018] [Heitkötter, Hanschke, Majchrzak 2012]. No desenvolvimento de aplicativos nativos, o desenvolvedor trabalha apenas com o SDK (*Software Development Kit*) e *frameworks* específicos do sistema operacional móvel. Caso seja necessário suporte a várias plataformas, elas devem ser desenvolvidas separadamente.

Aplicativos *web* (*web apps*) são implementados utilizando HTML, CSS e *JavaScript*, e utilizam o *browser* como seu ambiente *runtime*, aproveitando o suporte a *browsers* já existente nas plataformas móveis. Esses aplicativos, contudo, não têm acesso a funcionalidades específicas do *hardware* dos dispositivos, como, por exemplo, câmera e sensores [Heitkötter, Hanschke, Majchrzak 2012].

A abordagem híbrida surge, então, como uma forma de resolver a falta de acesso a esses recursos nativos. Fazendo uso de um motor de renderização *web* encapsulado em um motor nativo como *runtime*, aplicativos híbridos devem ser instalados no dispositivo, o que não é necessário com aplicativos *web*. O código-fonte, além de ser comum ao código de *web apps*, também tem acesso a APIs (*Application Programming Interfaces*) de funcionalidades específicas do dispositivo [Heitkötter, Hanschke, Majchrzak 2012].

2.2. Frameworks multiplataforma

A seguir é feita uma breve apresentação de alguns frameworks multiplataforma utilizados atualmente, dentre eles *React Native*, *Ionic, Xamarin* e *Flutter*.

Tabela 1. Comparativo de *frameworks* multiplataforma.

	React Native	**Ionic**	**Xamarin**	**Flutter**
Desenvolvido por	Meta	Drifty	Microsoft	Google
Criado em	2015	2013	2011	2018
Código aberto	Sim	Sim	Sim	Sim
Linguagem	JavaScript	JavaScript	C#	Dart

O *React Native*[4] é um *framework* de código aberto criado pela Meta (anteriormente Facebook) em 2015. Baseado na biblioteca *React*[5], permite a criação de aplicativos para as plataformas *Android* e *iOS* utilizando a linguagem *JavaScript*. Em tempo de execução, o framework cria as versões nativas dos componentes escritos em *React*, fazendo com que a interface de usuário fique muito semelhante à interface nativa.

O framework de desenvolvimento *Ionic*[6] foi lançado em 2013. Inicialmente construído com base em *AngularJS* e *Apache Cordova*, atualmente é baseado em *Web Components*, permitindo o uso de tecnologias web como *Angular*, *React* e *Vue* para a construção de aplicativos móveis híbridos na linguagem *JavaScript*.

O *Xamarin*[7] é uma plataforma de código aberto, criada em 2011 e adquirida pela Microsoft em 2016, que complementa a plataforma de desenvolvimento *.NET* com bibliotecas e ferramentas para construção de aplicativos para *Android*, *iOS*, *tvOS*, *watchOS*, *macOS* e *Windows*. Aplicativos escritos para a plataforma utilizam a linguagem *C#*, o que facilita o compartilhamento de código entre todo o ecossistema .NET.

O *framework* selecionado para a realização do projeto foi o *Flutter*, por ser uma tecnologia de vanguarda, pela facilidade de desenvolvimento, incluindo a configuração inicial do ambiente e a compilação, pela performance em relação ao desenvolvimento

[4] https://reactnative.dev/.

[5] https://reactjs.org/.

[6] https://ionic.io/.

[7] https://dotnet.microsoft.com/en-us/apps/xamarin.

com *frameworks* nativos [Nawrocki 2021] e por sua crescente popularidade na comunidade de desenvolvimento [JetBrains 2021].

O *Flutter*[8] é um *framework* criado em 2018 e mantido pelo Google, empresa que também desenvolve o SDK nativo para *Android*. Permite o desenvolvimento de aplicativos móveis para *Android* e *iOS*, para *web* e para desktops *Windows*, *macOS* e *Linux*. Os aplicativos são desenvolvidos utilizando a linguagem *Dart*[9] e fazem uso de componentes fundamentais chamados *widgets*. O framework disponibiliza as bibliotecas *Material* e *Cupertino*, que implementam controles das linguagens de design para *Android* e *iOS*, respectivamente. Dentre suas funcionalidades mais características está o *hot reload*, que possibilita a visualização das mudanças de código assim que são salvas, sem a necessidade de recompilar o código.

Ele é composto por três camadas de arquitetura principais: o *Framework*, desenvolvido em *Dart*[10], composto por classes responsáveis por animações, gestos e abstrações de *layout* etc.; a *Engine*, desenvolvida em C++, que fornece implementações de baixo nível da API principal do *Flutter*, incluindo gráficos, entrada e saída de arquivos e de rede; e o *Embedder*, escrito na linguagem apropriada do sistema operacional em questão, que fornece um ponto de entrada no sistema operacional e permite a incorporação de um código escrito em *Flutter* a aplicações existentes [Flutter 2022].

O *Flutter* usa um paradigma declarativo, isto é, a interface é reconstruída do início com base no estado da aplicação, sem ser necessário um comando que force a atualização de suas partes. O estado, em linhas gerais, pode ser descrito com tudo o que existe na memória enquanto a aplicação é executada, ou ainda, o conjunto de valores de uma aplicação em um determinado momento. Tendo em vista a complexidade da aplicação, tanto atual quanto futura, é importante escolher uma técnica de gerenciamento de estado apropriada.

3. Desenvolvimento

O sistema desenvolvido consiste em um aplicativo móvel multiplataforma em que usuários têm acesso a um cadastro de animais perdidos ou avistados nas ruas, permitindo que eles gerenciem postagens de animais, enviem mensagens aos autores dos *posts* e recebam notificações quando estiverem próximos à localização de algum animal registrado.

Nesta seção são descritas as tecnologias e ferramentas aplicadas no desenvolvimento do aplicativo, bem como suas funcionalidades.

3.1. Gerenciamento de estado em Flutter

Os *widgets* são as unidades base no desenvolvimento em *Flutter*. Eles formam uma hierarquia baseada em composição, em que cada *widget* pertence a um *widget* "pai" e recebe dele um contexto, como ilustrado na Figura 2. O *widget root* é o que contém a

[8] https://flutter.dev/.

[9] https://dart.dev/.

[10] https://dart.dev/.

aplicação, sendo geralmente a widget *MaterialApp* ou *CupertinoApp*, de acordo com o sistema operacional do dispositivo. O widget *Scaffold* representa a estrutura base de *layout* de uma tela. Ele contém o widget *AppBar*, que apresenta o título da página (*widget Text*) na barra superior e também o widget *Center* como corpo principal da tela. O corpo contém uma coluna (*widget Column*) que possui três *widgets* "filho": *Text*, *SizedBox* e *ElevatedButton*, que representam um texto, um espaço vertical em branco e um botão, respectivamente. Todos os widgets possuem um método *build()*, responsável por determinar a representação visual do elemento.

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Meu Aplicativo'),
        ),
        body: Center(
          child: Column(
            children: [
              const Text('Hello World'),
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () => print('Clique'),
                child: const Text('Botão'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Figura 2: Código demonstrando a hierarquia de *widgets* em uma classe *Flutter*
Fonte: Flutter Architectural Overview[11].

Os *widgets* são divididos em dois tipos de classe: as imutáveis, que não possuem propriedades que mudam ao longo do tempo, chamadas *stateless*, e as que têm suas características alteradas (através de interações com o usuário, por exemplo), chamadas *stateful*. Estas últimas armazenam seu estado em uma classe separada, que é herdada da classe *State*. O estado é alterado através do método *setState()*, sendo que o método *build()* é executado sempre que o estado é alterado [Flutter 2022].

Este método de gerenciamento de estado se torna ineficiente em aplicações complexas, pois é necessário controlar as alterações e atualizar várias partes da interface. Isso inclui responder a ações do usuário, controlar dados entre telas diferentes,

[11] https://docs.flutter.dev/resources/architectural-overview

alterar dados em um local da aplicação enquanto lida com a resposta em outros locais que leem esses dados [Arshad 2021]. Dentre as diversas técnicas desenvolvidas para contornar esse problema destacam-se *Provider* e *BLoC*.

Provider[12] é o pacote de gerenciamento de estado recomendado pelos desenvolvedores do *Flutter* e amplamente utilizado para o gerenciamento de estado de aplicações. Ele se utiliza de três conceitos: *ChangeNotifier*, *ChangeNotifierProvider* e *Consumer*. *ChangeNotifier* é uma classe base do *Flutter* que funciona de maneira semelhante ao padrão de projeto *Observer* [Gamma, Helm, Johnson, Johnson, Vlissides 1995], ou seja, quando um objeto do tipo *ChangeNotifier* altera seu estado, ele notifica o *Consumer* que utiliza um objeto *ChangeNotifierProvider* para monitorar as alterações no estado do modelo em questão [Flutter 2022].

A biblioteca *Bloc*[13] (*Business Logic Components*), inclui a biblioteca *Provider* em sua implementação e dá um passo além ao separar as regras de negócio da interface de usuário. Ele é baseado em *streams*, ou fluxos, que são sequências de dados disponibilizados ao longo do tempo.

Um fluxo, chamado *bloc*, recebe um evento da interface e devolve um estado para ela. O *bloc* é composto por três tipos de classe: as que representam o estado, as que representam o evento e as que representam as regras de negócio [Arshad 2021]. A Figura 3 ilustra o funcionamento do *bloc*, onde a interface de usuário envia um evento para o bloc, que faz uma requisição a uma base de dados, e devolve um estado.

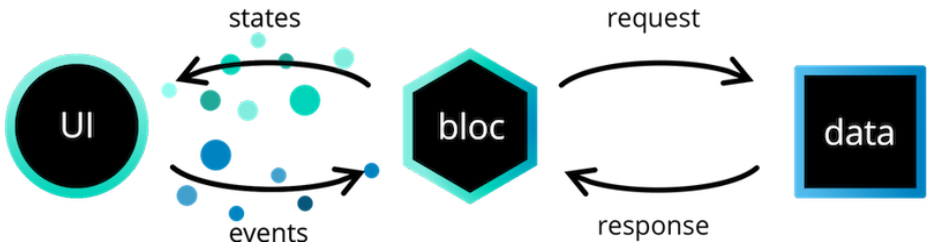

Figura 3: Funcionamento do Bloc [Bloc 2022].

3.2. Ferramentas

O sistema foi desenvolvido com o *framework Flutter* na linguagem *Dart*, com o auxílio da IDE (*Integrated Development Environment*) *Visual Studio Code*[14], editor de código-fonte leve e gratuito. Ele oferece suporte a vários *frameworks* e linguagens de programação e possui integração com ferramentas que facilitam a criação, depuração e compilação do código.

Dentre suas extensões utilizadas no projeto, destacam-se *Dart* e *Flutter*, que auxiliam na edição, refatoração, execução e *reload* de aplicativos construídos em *Flutter*, além da extensão *bloc*, que traz suporte à biblioteca *Bloc*.

---

[12] https://pub.dev/packages/provider.

[13] https://bloclibrary.dev/.

[14] https://code.visualstudio.com/.

A aplicação foi desenvolvida e testada primeiramente em *Android*, sendo necessária a instalação da *IDE Android Studio*[15] para que as dependências do sistema operacional fossem adicionadas ao projeto.

Para o controle de versão foi utilizado o *Git*[16], sistema de versionamento de código aberto, cujo repositório remoto foi hospedado na plataforma de hospedagem para desenvolvimento de *software, GitHub*[17]. A criação dos diagramas UML foi desenvolvida com o auxílio do software online e gratuito *diagrams.net*[18].

O banco de dados utilizado foi o *Cloud Firestore*[19]. É um sistema de banco de dados não relacional (*NoSQL*) flexível e escalável para desenvolvimento *mobile*, *web* e de servidor que mantém os dados das aplicações cliente sincronizados em tempo real. Para ser utilizado, é necessário ter uma conta Google e fazer o registro da aplicação através do console do *Firebase*[20].

3.3. Funcionalidades da Aplicação

O sistema possui dois atores: Usuário Visitante e Usuário Autenticado. O primeiro tem acesso apenas à lista de animais e à página de detalhe da postagem. O usuário autenticado, além de manter seus dados de autenticação e dados de contato, gerencia seu cadastro de animais e pode enviar mensagens para os autores de outras postagens, assim como habilitar e configurar o recebimento de notificações de proximidade dos animais cadastrados.

A Figura 4 ilustra o diagrama de casos de uso das principais funcionalidades do sistema. A seguir, uma breve descrição dos principais casos de uso.

- **Listar animais**: o Usuário, sendo Visitante ou Autenticado, visualiza a relação de animais cadastrados, ordenados pela postagem mais recente. Em cada postagem constam informações sobre o animal, entre elas: nome ou tipo de animal, sexo, idade, porte, data e local onde foi visto pela última vez, foto e status.
- **Ver detalhe do animal**: A partir da lista de postagens, o Usuário seleciona o animal de seu interesse e tem acesso à tela de detalhe. Nela são exibidas todas as informações disponíveis do animal: nome, foto, tipo do animal, sexo, cor, idade, porte, raça, data e mapa da última localização e dados de contato do autor da postagem. Se o Usuário for autenticado, será exibido também um botão para enviar mensagem ao autor.
- **Manter Cadastro de Usuário**: o Usuário Autenticado pode alterar seus dados de autenticação (e-mail e senha) e de contato (nome, número de telefone e número do WhatsApp).
- **Manter Cadastro de Animais**: o Usuário Autenticado pode adicionar, alterar e remover postagens de animais. Ao incluir um animal, os status disponíveis são apenas “Perdido” ou “Avistado”. Se o Usuário Autenticado adicionou um animal

[15] https://developer.android.com/studio/.

[16] https://www.git-scm.com/.

[17] https://github.com/damarissbrizza/pet3.

[18] https://www.diagrams.net.

[19] https://firebase.google.com/products/firestore/.

[20] https://console.firebase.google.com.

com o status “Perdido”, posteriormente poderá alterá-lo para “Encontrado”. Se selecionado inicialmente o status “Avistado”, poderá modificá-lo para “Recolhido”. A seleção da localização pode ser feita manualmente através de um mapa ou solicitando a localização atual do dispositivo.

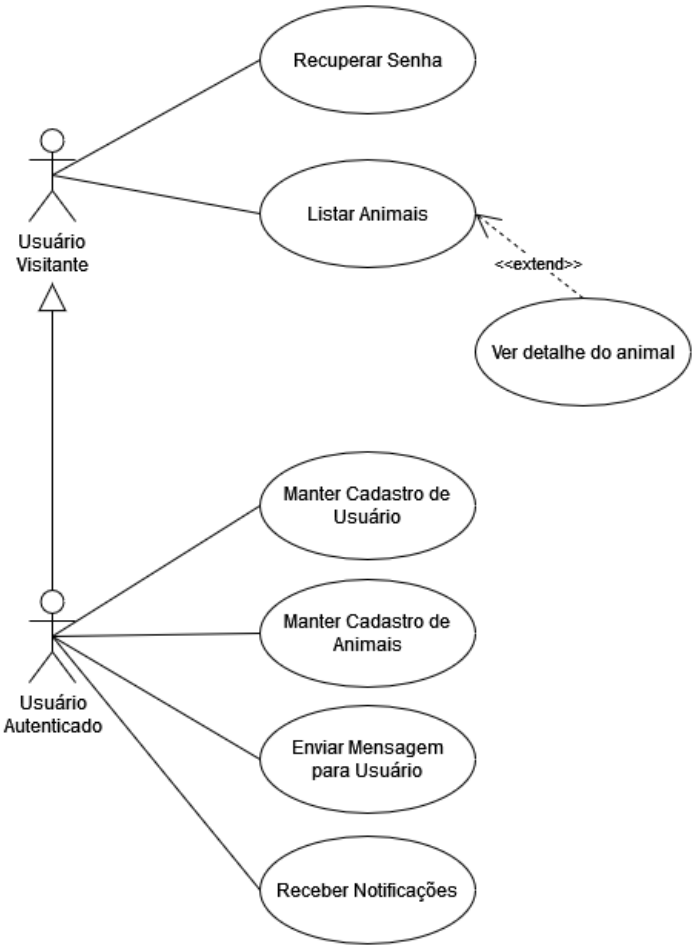

Figura 4: Diagrama UML de casos de uso da aplicação. Fonte: Autor.

- **Enviar Mensagem para Usuário**: através da tela de detalhe de um animal, o Usuário Autenticado entra na tela do chat e envia mensagens para o autor da postagem. A relação de chats do Usuário Autenticado é acessada através do menu do aplicativo e exibe, para cada chat, a imagem do animal, o nome do usuário que enviou a última mensagem e o texto da última mensagem.
- **Receber Notificações**: o Usuário Autenticado pode habilitar as notificações de localização de animais através de uma tela de configurações e selecionar o raio da distância máxima entre ele e a última localização do animal. Quando o dispositivo estiver posicionado dentro da área configurada, ele receberá uma notificação. Ao tocar na notificação, o aplicativo o levará até a tela de detalhe do animal, onde consta o ponto no mapa onde o animal foi visto pela última vez.

A Figura 5 demonstra como os dados foram organizados no *Cloud Firestore*. As coleções principais são *Animal* e *User*. Tomou-se a decisão de armazenar as informações de tipo, raça e cor do animal em suas próprias coleções, visando facilitar o

acréscimo de novos documentos. Pode-se observar que um chat é composto de apenas dois usuários e é relativo a um animal específico.

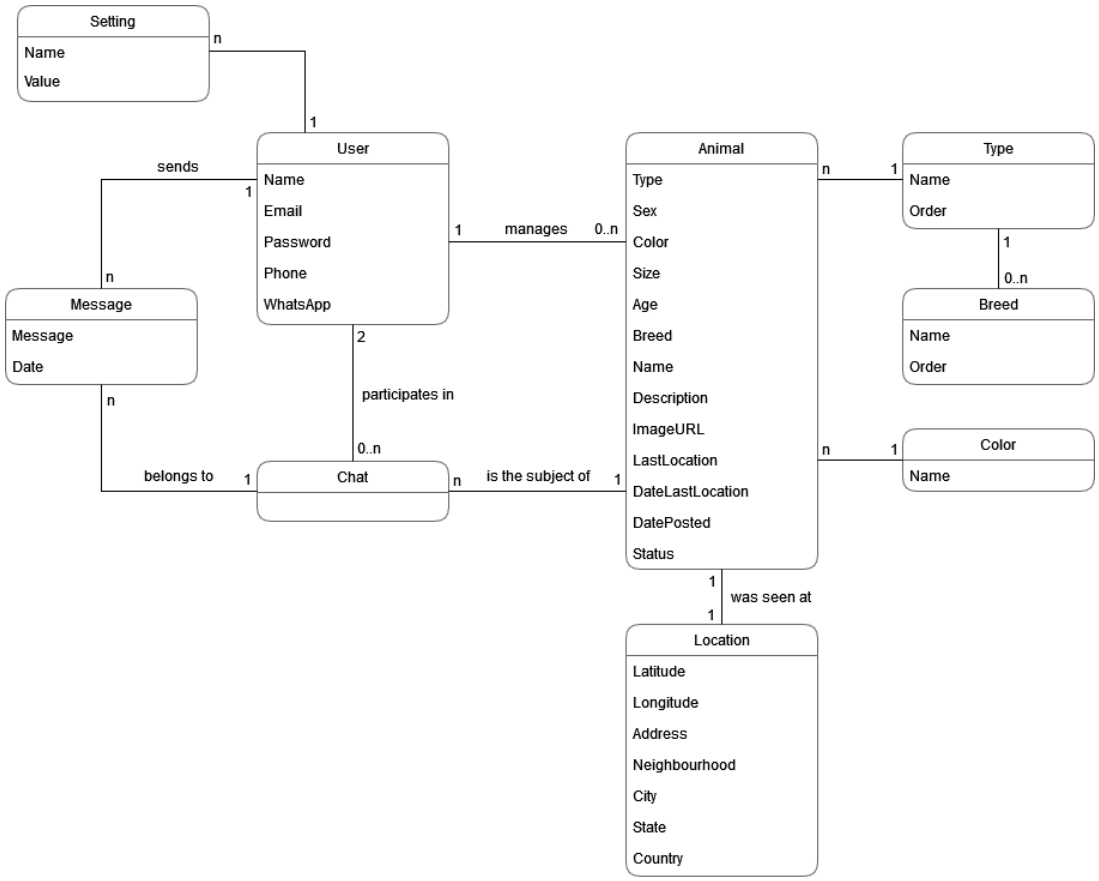

Figura 5: Diagrama demonstrando a organização das informações no banco de dados *Cloud Firestore*. Fonte: Autor.

4. Resultados

Nesta seção são descritos dois cenários de uso do aplicativo de modo a apresentar as principais funcionalidades do sistema.

4.1. Cenário de Uso 1

Uma pessoa sai para passear com seu cachorro em um dia de feriado. Ao passar por um parque, ela ouve o som de fogos de artifício e vê seu cachorro correr assustado para longe. Após passar cerca de uma hora a sua procura, ela volta ao local onde estava quando o cão fugiu, abre o aplicativo e faz *login* com seu e-mail e senha já cadastrados (Figura 6).

Após entrar, o usuário acessa o menu do aplicativo, toca na opção *Meus Animais* e na tela seguinte, pressiona o botão *Adicionar*. Na tela *Cadastro de Animal* (Figura 8), ele informa os dados do cachorro, adiciona uma foto da galeria de seu *smartphone* e após selecionar sua localização atual, faz a inclusão da postagem.

Após algumas horas, a pessoa visualiza na tela *Lista Geral de Animais* (Figura 7) uma postagem de outro usuário sobre um cachorro parecido com o seu. Na tela de

Detalhe do Animal (Figura 9), ela toca no botão *Enviar Mensagem* e inicia a comunicação com o autor da postagem (Figura 10).

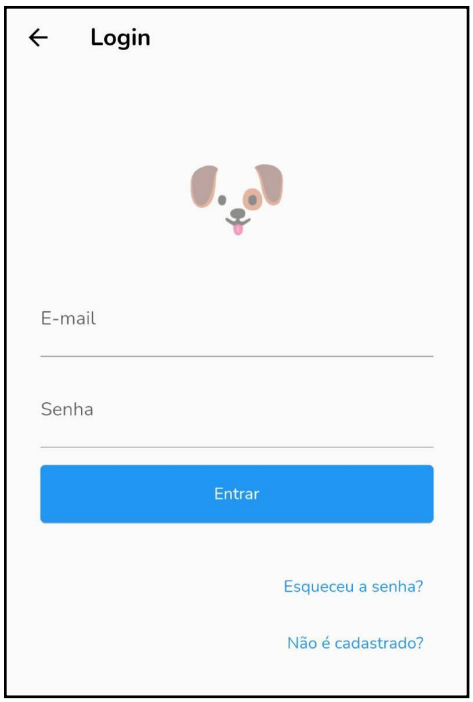

Figura 6: Tela *Login*.

Figura 7: Tela *Lista Geral de Animais*

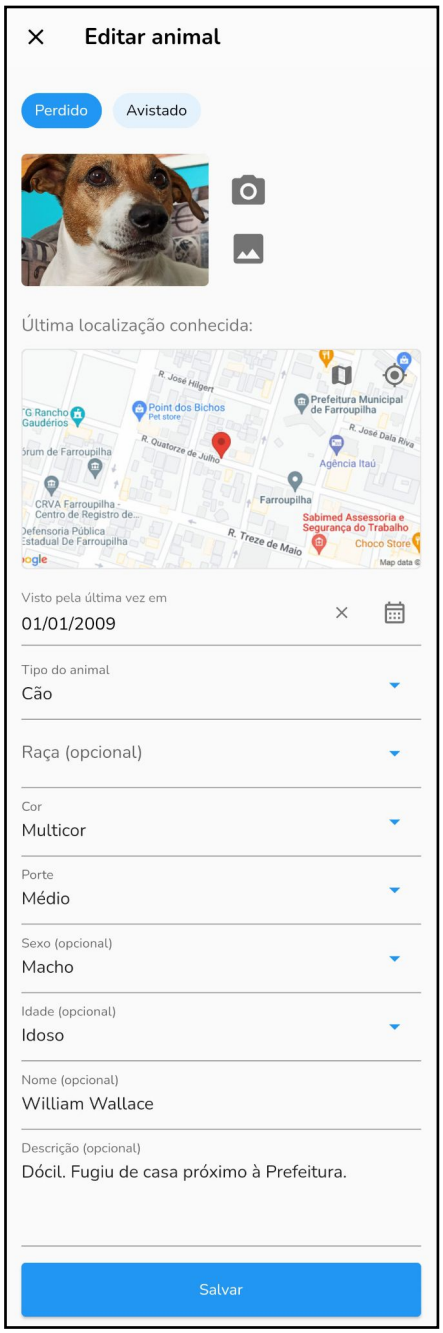

Figura 8: Tela *Cadastro de Animal*.

Figura 9: Tela *Detalhe do Animal*.

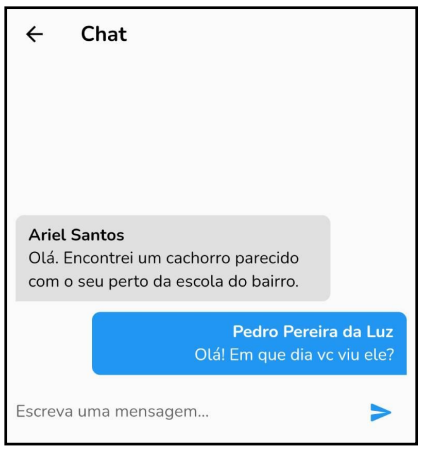

Figura 10: Tela *Chat*.

4.2. Cenário de Uso 2

Uma pessoa tem o costume de fazer postagens de animais em redes sociais quando os vê perambulando sozinhos pela rua e, quando possível, os recolhe. Muitas vezes ela descobre que passou perto de um animal perdido, mas não o viu. Ela gostaria de ser avisada assim que estiver próxima a um animal que esteja perdido ou que necessite de cuidados. Ela, então, faz download do aplicativo e realiza seu cadastro (Figura 11).

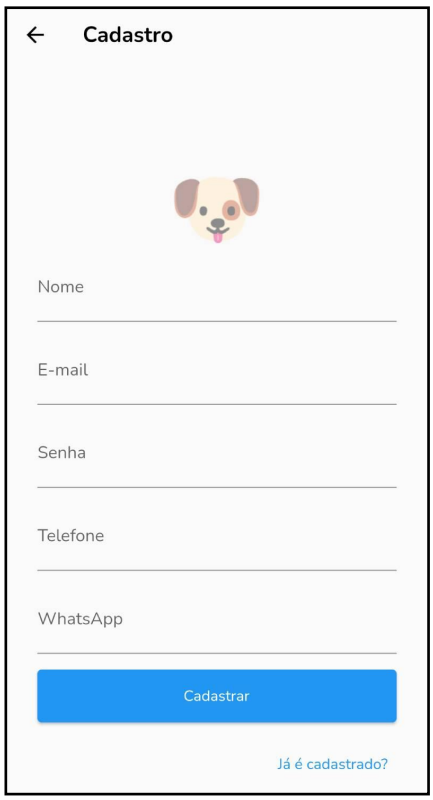

Figura 11: Tela *Cadastro de Usuário*.

Através do menu do aplicativo, o usuário, agora já autenticado, acessa a tela de configurações (Figura 12) e habilita o envio de notificações. Ele também configura a

distância entre o animal e o dispositivo a partir da qual o aplicativo envia notificações sobre ele.

No dia seguinte, ao retornar para casa, recebe a notificação de que um animal está próximo de sua localização (Figura 13). A pessoa toca na notificação e é levada à tela do detalhe do animal. Ela verifica que o local é próximo de sua residência e se dirige até lá, encontrando o animal nas proximidades.

Figura 12: Tela *Configurações*.

Figura 13: Notificação do aplicativo no *smartphone* do usuário.

5. Considerações finais

A preocupação do poder público com o bem-estar animal é relativamente recente. Apesar de normas visando a proteção dos animais terem surgido no início do século XIX no âmbito internacional e de a responsabilidade governamental estar explícita na Constituição brasileira de 1988, leis regionais específicas que definem modelos concretos de proteção animal foram estabelecidas apenas na última década.

Este trabalho propôs um aplicativo para dispositivos móveis que tem por objetivo, além de facilitar e agilizar a retirada de animais das ruas, reunir o animal de estimação, considerado ente querido, a sua família com rapidez e eficiência.

A abordagem de desenvolvimento multiplataforma, considerada híbrida, permite a redução do tempo de desenvolvimento bem como atingir um maior público-alvo. O

framework Flutter foi escolhido para o desenvolvimento após considerar as vantagens e desvantagens das opções.

O aplicativo desenvolvido permite que usuários cadastrem animais encontrados nas ruas ou perdidos, entrem em contato com outros usuários pelo aplicativo e recebam notificações quando um animal cadastrado estiver próximo do usuário.

Foram descritos dois cenários de uso: o primeiro, em que uma pessoa perde seu animal de estimação e utiliza o aplicativo para localizá-lo, e o segundo de uma pessoa que recolhe animais de rua e deseja receber notificações quando está próxima de algum animal cadastrado..

As funcionalidades mais importantes para se obter um MVP (*minimum viable product*) foram desenvolvidas com sucesso nesse trabalho. Durante o desenvolvimento foram identificadas funcionalidades com potencial para agregarem valor à aplicação, tais como: a configuração da exibição dos dados de contato do usuário da tela de detalhe do animal; enviar notificação quando o usuário recebe uma mensagem pelo aplicativo; adicionar filtros na lista geral de animais, incluindo tipo de animal, porte e data da postagem; configuração das características dos animais sobre os quais deseja receber notificação; moderação das mensagens e postagens, podendo ser em forma de denúncia; divulgação das postagens em outras mídias sociais, aumentando o alcance da informação e, por fim, a implementação de um módulo de adoções, que também pode ser utilizado por associações protetoras de animais.

6. Referências

Arshad, Waleed. (2021) “Managing State in Flutter Pragmatically: Discover how to adopt the best state management approach for scaling your Flutter app”, Packt Publishing Ltd.

Belchior G. P. N. and Dias M. R. M. S. (2020) “Os Animais de Estimação como Membros do Agrupamento Familiar”, Revista Brasileira de Direito Animal, Salvador, 15(3), https://periodicos.ufba.br/index.php/RBDA/article/view/38788. Acesso em: 9 nov. 2022.

Brasil, Constituição (1988), Constituição da República Federativa do Brasil. Capítulo VI - Do Meio Ambiente. Art. 225.

Caxias do Sul, Lei nº 8.542, de 7 de agosto de 2020. Estabelece, no âmbito do Município de Caxias do Sul, o Código Municipal de Proteção aos Animais, determinando as sanções e penalidades administrativas para aqueles que praticarem maus-tratos aos animais, cria o Fundo de Proteção Animal e dá outras providências. Caxias do Sul, Câmara Municipal, 2020.

Flutter (2022). https://flutter.dev/. Acesso em: 10 nov. 2022.

Gamma, E., Helm, R., Johnson, R., Johnson, R. E., & Vlissides, J. (1995) “Design patterns: elements of reusable object-oriented software”. Pearson Deutschland GmbH.

Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2012) “Evaluating cross-platform development approaches for mobile applications”, In: International Conference on

Web Information Systems and Technologies, Springer, Berlin, Heidelberg, p. 120-138.

IBGE, B. (2020) “Pesquisa nacional de saúde: 2019; informações sobre domicílios, acesso e utilização dos serviços de saúde: Brasil, grandes regiões e unidades da federação/IBGE”, Coordenação de Trabalho e Rendimento, Rio de Janeiro, IBGE. https://biblioteca.ibge.gov.br/visualizacao/livros/liv101748.pdf. Acesso em: 9 nov. 2022.

Jabangwe, R., Edison, H., & Duc, A. N. (2018) “Software engineering process models for mobile app development: A systematic literature review”, Journal of Systems and Software, 145, p. 98-111.

JetBrains (2021), “The State of Developer Ecosystem in 2021”, https://www.jetbrains.com/lp/devecosystem-2021/.

Nawrocki, P., Wrona, K., Marczak, M., & Sniezynski, B. (2021) “A comparison of native and cross-platform frameworks for mobile applications”, Computer, 54(3), p. 18-27.

Rieger, C., & Majchrzak, T. A. (2019) “Towards the definitive evaluation framework for cross-platform app development approaches”, Journal of Systems and Software, 153, p. 175-199.

Rio Grande do Sul, Lei nº 15.363 de 5 de novembro de 2019. Consolida a legislação relativa à Proteção aos Animais no Estado do Rio Grande do Sul, Porto Alegre, Assembleia Legislativa, 2019.

Ripenapps (2018) “Cross-Platform App Development: Trends, Tactics & Tools”, Medium, https://ripenapps.medium.com/cross-platform-app-development-trends-tactics-tools-d05f78bc657c. Acesso em: 10 nov. 2022.

Santana, L. R., Macgregor, E., Souza, M. F. D. A., & Oliveira, T. P. (2004) “Posse responsável e dignidade dos animais”, In: Congresso Internacional de Direito Ambiental, 8, São Paulo, Instituto O Direito por um Planeta Verde, p. 533-552.

“The Six Best Cross-Platform App Development Frameworks”, Kotlin, https://kotlinlang.org/docs/cross-platform-frameworks.html. Acesso em: 10 nov. 2022.