UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

THEODORO LOUREIRO MOTA

Flockr: aplicativo móvel multiplataforma KMM para tutores de *pets* que combina rede social com gestão de saúde animal

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Marcelo Soares Pimenta

Porto Alegre
2021

RESUMO

Este trabalho consiste no desenvolvimento e análise do Flockr, aplicativo multiplataforma voltado para tutores de animais de estimação. O objetivo do aplicativo é fornecer ao usuário uma rede social com foco em *pets* associada a um conjunto de ferramentas destinadas a auxiliar no cuidado da saúde do animal. Para o desenvolvimento do aplicativo foi empregado o modelo cliente-servidor, utilizando Kotlin Multiplatform Mobile em conjunto com desenvolvimento nativo Android e iOS para os clientes e Node.js em conjunto com o serviço Parse Server para o servidor.

Palavras-chave: Aplicativo móvel. Android. iOS. Multiplatforma. Kotlin. KMM.

ABSTRACT

This work presents the development and analysis of Flockr, a multiplatform application intended for pet owners. The purpose of the application is to provide the user with a social network focused on pets associated with a set of tools designed to assist in the care of the animal's health. The client-server model was used in the development of the application, using Kotlin Multiplatform Mobile in conjunction with native Android and iOS development for clients and Node.js in conjunction with the Parse Server service for the server-side.

Keywords: Mobile application. Android. iOS. Multiplatform. Kotlin. KMM.

LISTA DE FIGURAS

Figura 2.1 Arquitetura do sistema Android. 15
Figura 2.2 Janela principal do Android Studio. 16
Figura 2.3 Ciclo de vida de uma Atividade. 18
Figura 2.4 Figura demonstrando a interação das partes em um MVP 20
Figura 2.5 Figura demonstrando a estrutura básica de um projeto KMM 21

Figura 3.1 Perfil do Flockr no Instagram 23
Figura 3.2 Tela principal do aplicativo DogCha! 24
Figura 3.3 Tela inicial do MedSafe mostrando as próximas doses a serem administradas. 25
Figura 3.4 Plano de saúde planejado pelo aplicativo Petable. 26

Figura 4.1 Protótipo final da tela de abertura do Flockr. 32
Figura 4.2 Protótipo final da edição de perfil. 33
Figura 4.3 Interface padrão do *Android Studio* demonstrando a edição de um arquivo gradle. 34
Figura 4.4 Processo de comunicação entre clientes e servidor 35
Figura 4.5 Figura demonstrando a interação das camadas em um MVP com KMM 36
Figura 4.6 Arquitetura interna da biblioteca 37
Figura 4.7 Exemplo de uma Bottom Navigation View 42
Figura 4.8 Tela contendo o *feed* de publicações do Flockr 43
Figura 4.9 Comparação entre os dois tipos de perfis, o próprio e o de terceiros. 44
Figura 4.10 Tela principal da gestão de saúde do animal 45
Figura 4.11 Cadastro de vacinação 45
Figura 4.12 Registro de eventos de abertura de tela durante o período beta do Flockr. 46
Figura 4.13 Fluxo de login e cadastro de conta no Flockr. 47
Figura 4.14 Fluxo de cadastro de *pet*. 48
Figura 4.15 Ferramenta para recortar a foto de perfil. 49
Figura 4.16 Tela contendo o *feed* de publicações selecionas para o usuário. 50
Figura 4.17 Telas demonstrando as opções de busca de perfil. 51
Figura 4.18 Telas de notificações. 52
Figura 4.19 Fluxo de publicação de conteúdo. 53

Figura 5.1 Gráfico de distribuição de idade por sexo. 55
Figura 5.2 Gráfico com o nível de escolaridade dos participantes. 56
Figura 5.3 Gráfico com o nível de experiência dos participantes na utilização de redes sociais em smartphones. 56
Figura 5.4 Gráfico contendo informação sobre o nível de dificuldade que os usuários encontraram ao realizar o cadastro no aplicativo. 57
Figura 5.5 Gráfico contendo informação sobre o método utilizado para cadastro no aplicativo. 57
Figura 5.6 Gráfico contendo informação sobre o nível de dificuldade que os usuários encontraram ao realizar o cadastro do *pet* no aplicativo. 58
Figura 5.7 Gráfico contendo informação sobre a quantidade de *pets* cadastrados por usuário. 58
Figura 5.8 Gráfico contendo informação sobre publicações na rede social. 59
Figura 5.9 Gráfico contendo informação se a ferramenta de edição de imagem foi ou não utilizada durante as publicações. 59
Figura 5.10 Gráfico contendo informação se a ferramenta de edição de imagem foi ou não utilizada durante as publicações. 60
Figura 5.11 Gráfico contendo informação se a ferramenta de gestão de saúde foi utilizada. 60
Figura 5.12 Gráfico contendo informação se os usuários tiveram dificuldades ao utilizar a ferramenta de edição de saúde. 61
Figura 5.13 Gráfico contendo informação se os usuários encontraram falhas no aplicativo durante o período de uso. 61
Figura 5.14 Gráfico contendo informação se os usuários encontraram dificuldades na utilização do aplicativo. 62

LISTA DE TABELAS

Tabela 3.1 Tabela com comparativo de maturidade e número de downloads dos aplicativos relacionados. 27
Tabela 3.2 Tabela com comparativo de funcionalidades dos aplicativos analisados. 28

Tabela 4.1 Tabela com *user stories* criados para o Flockr 31
Tabela 4.2 Tabela com os principais *endpoints* criados para o Flockr. 40

Tabela 5.1 Tabela demonstrando os blocos de questões utilizadas na avaliação do Flockr. 54

LISTA DE ABREVIATURAS E SIGLAS

KMM	Kotlin Multiplatform Mobile
SDK	Software Development Kit
API	Application Programming Interface
MVP	Model-View-Presenter
App	Application
UI	User Interface

SUMÁRIO

LISTA DE FIGURAS **5**
LISTA DE TABELAS **7**
1 INTRODUÇÃO **11**
2 FUNDAMENTOS E CONCEITOS RELACIONADOS **13**
2.1 Android **13**
2.1.1 Arquitetura 13
2.1.1.1 *Application Framework* 13
2.1.1.2 *Binder IPC* 14
2.1.1.3 Android *system services* 14
2.1.1.4 HAL 14
2.1.1.5 *Kernel*do Linux 14
2.1.2 Android Studio 15
2.1.3 Componentes Básicos 16
2.1.3.1 Atividades 16
2.1.3.2 Fragmentos 18
2.1.3.3 *Intents* 19
2.1.4 Linguagem Kotlin 19
2.2 Padrão MVP **19**
2.3 Kotlin Multiplatform Mobile **20**
3 TRABALHOS RELACIONADOS **22**
3.1 Instagram **22**
3.2 DogCha! **23**
3.3 MediSafe **24**
3.4 Petable **25**
3.5 Comparativo de características **26**
3.5.1 Maturidade 26
3.5.2 Quantidade de instalações 27
3.6 Comparativo de funcionalidades **27**
3.6.1 Rede Social 28
3.6.2 Gestão de saúde 28
4 DESENVOLVIMENTO DO APLICATIVO: FLOCKR **30**
4.1 Modelagem das funcionalidades **30**
4.1.1 *User stories* 30
4.1.2 Prototipação 31
4.2 Ambiente de desenvolvimento **33**
4.3 Arquitetura **34**
4.3.1 Cliente-Servidor 34
4.3.2 Padrão MVP + KMM 35
4.4 Biblioteca KMM **36**
4.4.1 Distribuição 38
4.5 Backend **38**
4.5.1 Parse Server 38
4.5.2 *Endpoints* 39
4.5.3 Infraestrutura e *Deploy* 39
4.6 Funcionalidades **41**
4.6.1 *Login* e registro de usuários 41
4.6.2 Navegação 41
4.6.3 Rede Social 42

4.6.4 Perfil 43
4.6.5 Controle de saúde animal 44
4.7 Análise de dados 46
4.8 Telas 46
4.8.1 *Login* 47
4.8.2 Cadastro de perfil 47
4.8.3 *Feed* 49
4.8.4 Busca 50
4.8.5 Notificações 51
4.8.6 Criação de conteúdo 52
5 AVALIAÇÃO 54
5.1 Perfil dos participantes 54
5.2 Análise dos resultados 56
6 CONCLUSÃO 63
6.1 Trabalhos futuros 63
REFERÊNCIAS 65

1 INTRODUÇÃO

Com o objetivo de aproximar a prática profissional aos conhecimentos adquiridos ao longo da formação enquanto cientista da computação, a elaboração deste trabalho de conclusão de curso consiste em uma análise e apresentação de um produto desenvolvido pela empresa Mocka (MOCKA, 2021), onde o autor ocupa o cargo de desenvolvedor Android e líder de desenvolvimento há 3 anos. A utilização da plataforma KMM, assim como a atualização constante de metodologias e conceitos, foi fruto de um processo de pesquisa interna, que se mostrou transversal à teoria acadêmica e ao exercício profissional.

No ano de 2020 a crise econômica no Brasil afetou diversos setores da economia que apresentaram perdas de até 4%, entretanto, a chamada indústria *pet* obteve um crescimento de 13,5% em relação ao ano de 2019, apresentando o faturamento acima dos 40 bilhões de reais (EXAME, 2021). De acordo com o presidente-executivo do Instituto Pet Brasil, os motivos desse crescimento são a profissionalização e a inovação presentes no mercado *pet* brasileiro, hoje o terceiro maior do mundo. A população de animais domesticados no território brasileiro é de mais de 139 milhões (ABINPET, 2021), o que demonstra o potencial do setor na economia brasileira.

Buscando integrar o mercado *pet*, a empresa Mocka (MOCKA, 2021), durante os anos 2020 e 2021, desenvolveu o Flockr, aplicativo para smartphones Android e iOS. Utilizando como referência um protótipo de mesmo nome idealizado no ano de 2014 pelo CEO da empresa, o app oferece a tutores de animais de estimação uma rede social voltada para *pets* aliada a um conjunto de ferramentas para gestão de saúde animal. A rede social cumpre o papel de atrair e manter o usuário engajado no aplicativo enquanto a seção de gestão de saúde entrega valor ao usuário, fornecendo diversas ferramentas necessárias para manter o animal saudável, tais como: controle de vacinação, controle de peso do animal, controle de consultas veterinárias, agendamento de banho e tosa, entre outros.

Este trabalho tem como objetivo principal detalhar o desenvolvimento do aplicativo Flockr, demonstrando todas as etapas desde a prototipação até a publicação e avaliação do app, atribuindo maior enfoque à versão para Android. Como objetivo secundário busca-se detalhar o uso do framework Kotlin Multi Platform Mobile e avaliar os benefícios que sua utilização trouxe para o projeto do aplicativo Flockr. O autor integrou ativamente todas as etapas do desenvolvimento da versão apresentada, sua contribuição foi parcial em etapas pontuais, que serão sinalizadas ao longo do trabalho.

O presente trabalho foi desenvolvido tendo como destinatários principais desen-

volvedores mobile que visam implementar aplicativos em iOS e Android. A composição das etapas e organização do texto são, portanto, direcionadas às necessidades dos leitores alvo. A monografia está estruturada em seis capítulos, no capítulo 2 serão desenvolvidas a fundamentação teórica e a conceituação, o Capítulo 3 apresenta quatro aplicativos *mobile* relacionados às temáticas abordadas pelo Flockr: redes sociais e gestão de saúde. O capítulo 4 aprofunda o desenvolvimento do aplicativo Flockr detalhando a modelagem de funcionalidades, ambiente de desenvolvimento, arquitetura, uso do KMM, desenvolvimento do *backend*, funcionalidades, coleta de dados e suas principais telas. O capítulo 5 explora a avaliação do Flockr através de um questionário aplicado a usuários. Por fim, o capítulo 6 apresenta a conclusão associada a uma seção que explora as intenções de trabalhos futuros para o aplicativo.

2 FUNDAMENTOS E CONCEITOS RELACIONADOS

Neste capítulo serão apresentados os conceitos e tecnologias fundamentais para a compreensão do trabalho realizado.

2.1 Android

Android é um sistema operacional de código aberto, baseado em uma versão modificada do Linux, criado e desenvolvido pela empresa Android Inc. O sistema foi desenvolvido com o intuito de proporcionar uma melhor experiência de usuário em câmeras digitais, entretanto, com o crescimento do mercado mobile, a companhia reposicionou o projeto para atender a essa demanda. Em julho de 2005 a Android Inc foi comprada e incorporada pela Google e, em 2008, o sistema operacional foi disponibilizado para o público através do *smartphone HTC Dream*.

2.1.1 Arquitetura

Conforme demonstrado na figura 2.1 a arquitetura do sistema Android é uma pilha dividida em cinco camadas contendo seis elementos principais. A seguir será discutido o objetivo e responsabilidade de cada uma das camadas.

2.1.1.1 Application Framework

É a camada superior da pilha e tem como objetivo fornecer uma base de funcionalidades acessível ao usuário final. Ela consiste nos aplicativos disponibilizados junto ao sistema operacional, tais como, email, navegador, calendário, mapa e câmera. Estes aplicativos não têm privilégios especiais no sistema operacional se comparados a aplicativos desenvolvidos por terceiros. Esta camada também pode ser utilizada por desenvolvedores para fornecer funcionalidades específicas, tais como, reprodução de músicas, envio de mensagens, visualização de imagens, dentre outros.

2.1.1.2 Binder IPC

O *Binder Inter-Process Communication* (IPC) tem como objetivo permitir que os aplicativos cruzem os limites dos seus processos e chamem os códigos de serviços do sistema Android. Desta forma, APIs de alto nível podem se comunicar com serviços do sistema operacional. A nível de aplicação essa comunicação ocorre de forma transparente para o desenvolvedor.

2.1.1.3 Android system services

Android *system services* são componentes modulares disponibilizados pelo sistema operacional, possuem funcionalidades expostas por APIs que possibilitam que o desenvolvedor possa se comunicar com serviços do sistema e acessar o hardware subjacente. Conforme demonstrado na figura 2.1, o Android possui dois grupos de serviço, os de sistema (como Window Manager Notification Manager) e os de mídia (serviços envolvidos na reprodução e gravação de mídia).

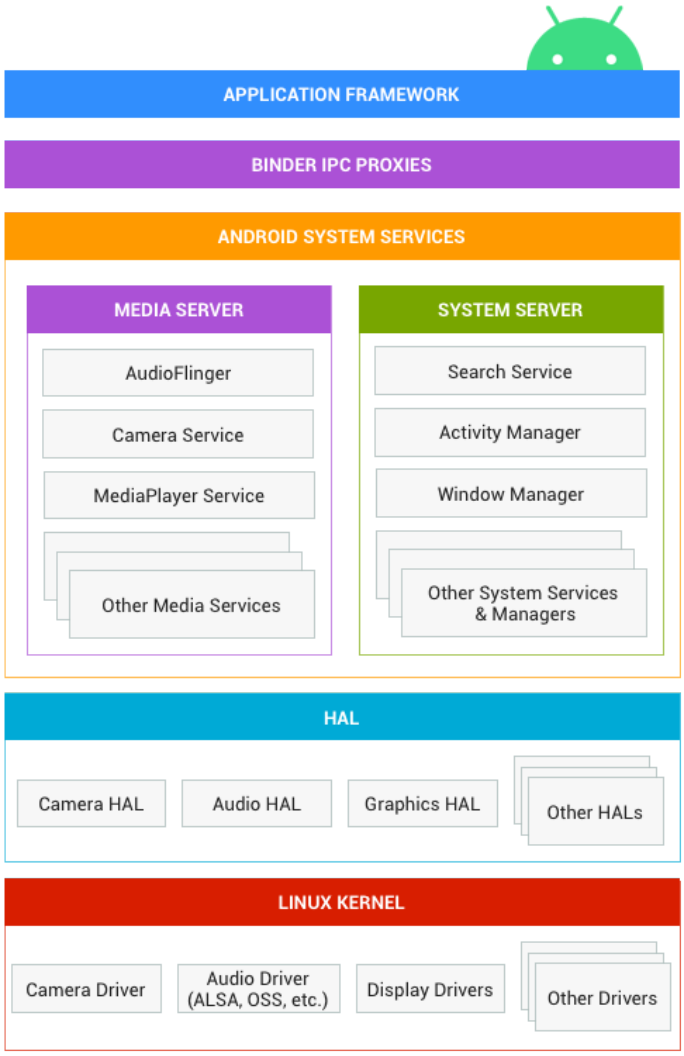
2.1.1.4 HAL

A camada HAL (*Hardware abstraction layer*) é composta por um conjunto de interfaces padrão (chamadas de HALs) que são disponibilizadas para os fornecedores de hardware, desta forma as fabricantes de *smartphones* podem usar o componente que desejarem na construção do seu dispositivo, contanto que o mesmo implemente uma HAL. As implementações de HAL são empacotadas em módulos e carregadas pelo sistema Android no momento apropriado.

2.1.1.5 Kernel*do Linux*

A camada mais baixa da pilha é uma versão modificada do *kernel* Linux, responsável pelas funcionalidades básicas do sistema operacional, como gerenciamento de memória e processos. Entre as modificações feitas podemos citar o *Low Memory Killer* (um sistema de gerenciamento de memória que é mais agressivo na sua preservação), o *wakelocks* (um serviço do sistema que controla gerenciamento de energia), o *driver* do Binder IPC – vide subseção 2.1.1.2–, entre outras modificações importantes para uma plataforma *mobile*.

Figura 2.1 – Arquitetura do sistema Android.



Fonte: https://source.android.com/devices/architecture. Acessado em: 23/04/2021

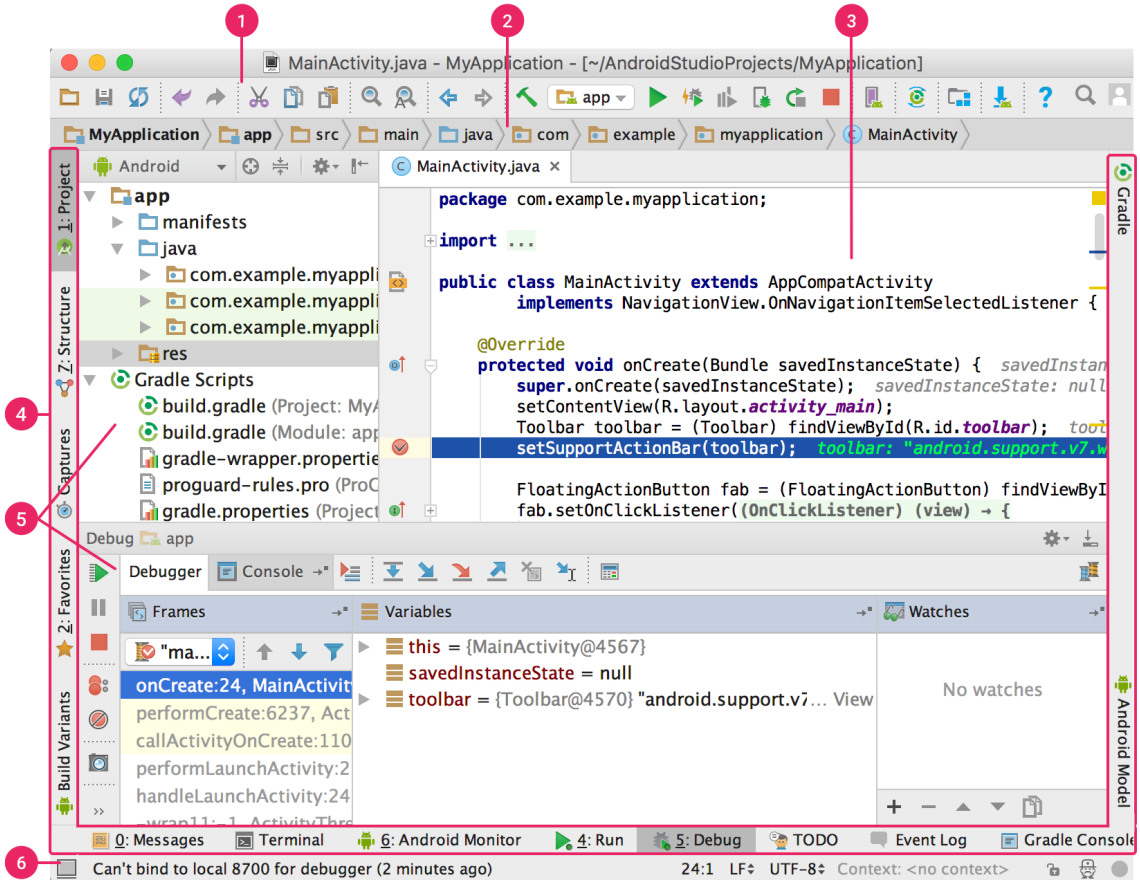
2.1.2 Android Studio

Para o desenvolvimento do trabalho foi utilizado o Android Studio que é a IDE oficial para o desenvolvimento de aplicativos Android e é baseado no *InteliJ IDEA* (ANDROID DEVELOPERS, 2021a). Além do editor e das ferramentas incluídas no *InteliJ IDEA*, o Android Studio oferece uma série de funcionalidades e recursos voltados para o desenvolvimento de aplicativos, tais como: emulador, editor de layout, sistema de compilação baseado em Gradle, integração com sistema Git, *frameworks* para testes, dentre outros.

Na figura 2.2 podemos observar que a janela principal do Android Studio oferece,

além do editor de texto (ponto 3), ferramentas de depuração de código (pontos 5 e 6), de edição de texto (ponto 1) e de navegação no projeto (pontos 2 e 4).

Figura 2.2 – Janela principal do Android Studio.



Fonte: https://developer.android.com/studio/intro. Acessado em: 23/04/2021

2.1.3 Componentes Básicos

A seguir serão apresentados os principais componentes utilizados durante o desenvolvimento de um aplicativo Android.

2.1.3.1 Atividades

Atividades são componentes pertencentes ao *framework* do Android que fornecem a janela necessária para desenhar a interface do usuário e interceptar suas interações com a mesma. O modelo de aplicação da plataforma utiliza uma pilha de Atividades, onde apenas a que está no topo da pilha se encontra em execução, as demais ficam interrompidas para economizar processamento, se a atividade do topo da pilha for destruída a que se

encontra abaixo retoma a execução do momento em que foi interrompida. Caso a pilha fique vazia o aplicativo é finalizado.

As Atividades possuem um ciclo de vida com onze estados que podem ser observados na figura 2.3. Ao ocorrer uma mudança de estado, as Atividades, disponibilizam esta informação ao desenvolvedor através de *callbacks*. Os principais estados do ciclo de vida de uma atividade são:

onCreate: Estado atingido na criação da atividade, pode ser utilizado pelo desenvolvedor para instanciar objetos e bibliotecas.

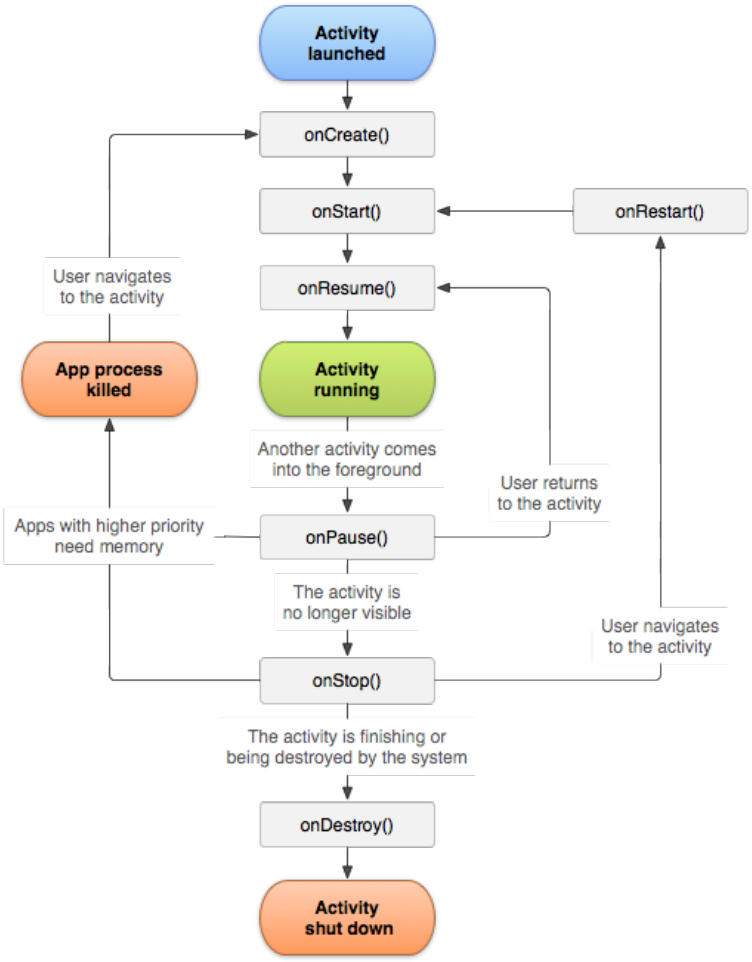
onResume: Estado atingido sempre que a atividade chega no topo da pilha e está visível na tela do dispositivo.

onPause: A Atividade não se encontra mais no topo da pilha entretanto, ainda é visível para o usuário, portanto, permanece anexada ao *WindowsManager*. A Atividade que se encontra neste estado pode ser eliminada pelo sistema para liberar memória.

onStop: Estado atingido sempre que a atividade não está mais visível, pois se encontra totalmente encoberta por outra atividade ou aplicativo. Atividades neste estado também podem ser eliminadas pelo sistema operacional para liberar recursos.

onDestroy: Estado atingido momentos antes da atividade ser excluída pelo sistema operacional e pode ser utilizada para salvar dados que não devem ser perdidos, como por exemplo, informações inseridas pelo usuário.

Figura 2.3 – Ciclo de vida de uma Atividade.



Fonte: https://developer.android.com/guide/components/activities/activity-lifecycle. Acessado em: 23/04/2021

2.1.3.2 Fragmentos

Um Fragmento é um componente modular e reutilizável, que contém seus próprios *UI* e ciclo de vida, e que tem como objetivo permitir a criação de interfaces mais complexas e flexíveis. O Fragmento deve ser hospedado em uma Atividade, que pode conter diversos fragmentos de uma única vez. O ciclo de vida do fragmento é, portanto, diretamente ligado ao da atividade (e.g. quando uma atividade é pausada todos os fragmentos hospedados nela também são) (ANDROID DEVELOPERS, 2021c).

2.1.3.3 Intents

Intents são objetos de mensagens utilizados para solicitar uma ação de outro componente do aplicativo, ou em alguns casos de outro aplicativo que expõem uma funcionalidade (e.g. um *browser* pode receber *Intents* contendo *links*). Embora existam diversos casos de usos para o emprego de *Intents* o mais comum deles é iniciar uma nova Atividade (ANDROID DEVELOPERS, 2021d).

2.1.4 Linguagem Kotlin

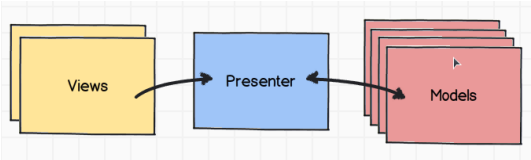
Desenvolvida pela JetBrains em 2011, Kotlin é uma linguagem de programação multiplataforma, orientada a objetos, funcional e estaticamente tipada, que pode ser compilada para *Java Virtual Machine* e código nativo via LLVM. Em fevereiro de 2012 a JetBrains tornou a linguagem *open source* sob a licença Apache 2.0, fato que acelerou o desenvolvimento e adoção da mesma (JETBRAINS, 2021a). Em 2017, foi anunciado pela Google que Kotlin havia se tornado a linguagem oficial do sistema Android no lugar de Java.

Apesar de possuir uma sintaxe mais concisa e um pouco diferente da linguagem Java, Kotlin é projetada para ter uma interoperabilidade total com código Java, permitindo que um mesmo projeto misture arquivos de ambas. No momento do desenvolvimento deste trabalho a linguagem se encontra na versão 1.4.32 e recebe atualizações mensalmente.

2.2 Padrão MVP

Model-view-presenter é uma derivação do modelo MVC (Model-view-controler) projetado para facilitar testes unitários automatizados e permitir uma maior separação de interesses na lógica de apresentação. No MVP o *model* é o modelo de dados que será exibido ou alterado, a *view* é uma interface que guia os comandos do usuário à camada *presenter* para que ela possa atuar sobre o *model*. Por fim, o *presenter* é responsável pela parte lógica, recuperando os dados dos repositórios, editando os modelos, e os formatando conforme necessário para exibi-los na *view* – interação demonstrada na figura 2.4. O padrão também permite a reutilização da parte lógica em diferentes *UIs*, contanto que implementem a *view* necessária para comunicação com o *presenter*.

Figura 2.4 – Figura demonstrando a interação das partes em um MVP



Fonte: Autor

Esta arquitetura foi amplamente adotada, principalmente em projetos Java, após a publicação do artigo *The Taligent Programming Model for C++ and Java*, onde Potel (1996) detalha a utilização e implementação do padrão MVP na *Taligent Inc*,uma subsidiária da IBM.

2.3 Kotlin Multiplatform Mobile

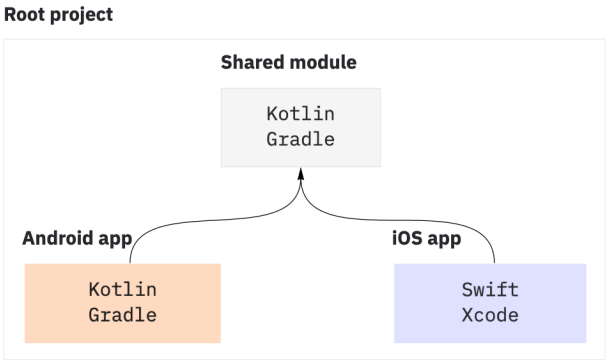
O *Kotlin Multiplatform Mobile* é uma tecnologia que permite centralizar a parte lógica de aplicativos Android e iOS em uma única base de código (Kotlin Foundation, 2021), sendo que apenas a parte específica de cada plataforma precisa ser implementada de forma nativa, como a UI. Desenvolvido pela JetBrains, criadora da linguagem Kotlin, a tecnologia permite fácil integração em projetos já existentes, visto que, ela não exige a migração completa da parte lógica das aplicações, permitindo assim a sua implementação em etapas.

A estrutura básica de um projeto em KMM –vide figura 2.5 – é composta por três componentes: o projeto do aplicativo Android, o projeto do aplicativo iOS e o módulo compartilhado.

Para uso em aplicativos iOS, o módulo compartilhado é compilado em um *framework* - uma espécie de diretório hierárquico com recursos compartilhados usados nas plataformas Apple. Este se conecta ao projeto Xcode, que irá gerar o aplicativo iOS. Por ser compilado utilizando *Kotlin/Native* resultando em binários nativos, o *framework* não apresenta nenhum *overhead* em relação ao projeto XCode.

Para integração no projeto Android existem duas opções: compilar o módulo compartilhado em uma biblioteca *aar* e importa-la no projeto, ou incluir o módulo compartilhado diretamente no projeto Android, desta forma ambos são compilados simultaneamente. O *Kotlin Compiler* transforma o módulo compartilhado em *Java Byte Code*, não apresentando perda de desempenho durante sua execução.

Figura 2.5 – Figura demonstrando a estrutura básica de um projeto KMM



Fonte: https://kotlinlang.org/docs/mobile/discover-kmm-project.html. Acessado: 24/04/2021

3 TRABALHOS RELACIONADOS

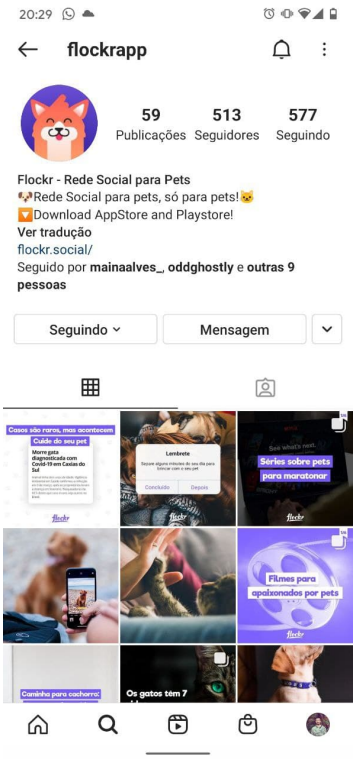
O Flockr é um aplicativo destinado a tutores de *pets* que associa o conceito de rede social com gestão de saúde animal, portanto, neste capítulo serão apresentados os principais aplicativos para smartphones relacionados a estas temáticas. Para cada um deles será feita uma análise de funcionalidades e qualidades e por fim uma comparação entre suas diversas características. O estudo apresentado neste capítulo foi o ponto de partida para o desenvolvimento do Flockr, aplicativo foco do trabalho que será apresentado no capítulo 4.

3.1 Instagram

Lançado em março de 2010 para Android e iOS o aplicativo obteve um crescimento exponencial atingindo em apenas 2 anos a marca de 100 milhões de usuários ativos. A popularidade fez com que fosse adquirido pelo Facebook por US$ 1 bilhão de dólares (UPBIN, 2021). O *app* além de contar com uma rede social para compartilhamento de fotos ainda apresenta ferramentas avançadas de edição de imagens e video.

O principal diferencial do Instagram é seu feed infinito de fotos que tem como objetivo capturar a atenção do usuário por um longo período de tempo (MOLLA; WAGNER, 2021). Quando não existem mais fotos novas para os perfis que o usuário segue, o aplicativo sugere novos conteúdos de acordo com o padrão de utilização do usuário. Além disso, o aplicativo permite a criação e gerenciamento de diversos perfis, cada um deles com seu próprio conteúdo – vide Figura 3.1. Para interação entre os usuários é possível comentar em fotos assim como enviar mensagens privadas.

Figura 3.1 – Perfil do Flockr no Instagram



Fonte: Autor

O aplicativo é monetizado por meio de anúncios feitos através da plataforma de anúncios do Facebook, além disso ele permite a criação de perfis comerciais onde o usuário pode criar uma espécie de e-commerce.

Dentre os aplicativos analisados ele é o mais bem sucedido, tanto em popularidade quanto em rentabilidade.

3.2 DogCha!

DogCha! é uma rede social voltada para donos de *pets* que conta com aplicativos disponíveis para Android e iOS. De acordo com Daniel Lorenzo, criador do app, o DogCha! possui mais de 2 mil usuários ativos e tem como objetivo atingir a marca de 20 mil usuários durante o ano de 2021.

Contando com uma interface e um conjunto de funcionalidades bastante inspiradas no Instagram – vide Figura 3.2 –, o aplicativo oferece ao usuário um *feed* infinito de publicações, uma ferramenta para compartilhar fotos e vídeos, lista de amigos, chat privado e comentários nas publicações. É possível fazer o cadastro e gerenciamento de

diversos perfis contendo fotos e informações básicas do animal. Por fim, o app apresenta um mapa com pontos de interesse para donos de *pet*, tais como: veterinário, pet shops, parques, centros de adoção, entre outros.

Figura 3.2 – Tela principal do aplicativo DogCha!

Fonte: Autor

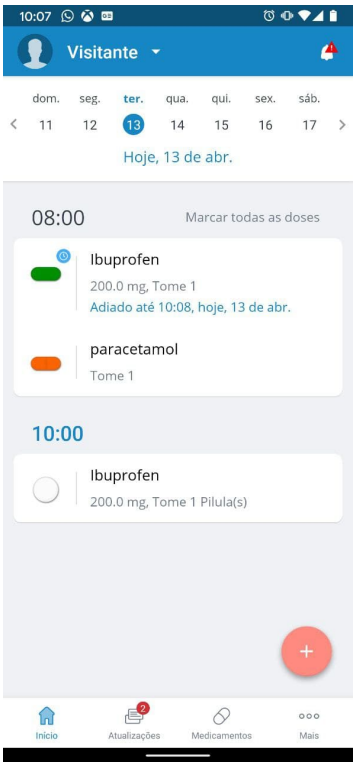
3.3 MediSafe

O MediSafe (MEDISAFE, 2021) é um aplicativo lançado em 2012 disponível para Android e iOS, que tem como objetivo utilizar a tecnologia mobile para eliminar as dificuldades no gerenciamento de prescrições médicas. Para tal, o usuário necessita cadastrar suas prescrições médicas no app. O cadastro de medicações pode ser feito de forma simplificada, sendo informado apenas o nome da medicação e a dosagem, ou de forma detalhada, permitindo a inserção de mais informações como datas, intervalos da medicação, fotos e alarmes personalizados. Após feito o cadastro o aplicativo dispara alarmes e notificações para garantir que o usuário tome o remédio no momento certo.

Para manter o controle das medicações existe um histórico e um calendário – vide Figura 3.3 – que pode ser compartilhado e gerenciado por outros usuários do aplicativo.

O app ainda oferece uma ferramenta para consulta de bula de remédios e permite conectividade com o *Google Fit* no *Android* e *Apple Health* no iOS.

Figura 3.3 – Tela inicial do MedSafe mostrando as próximas doses a serem administradas.



Fonte: Autor

3.4 Petable

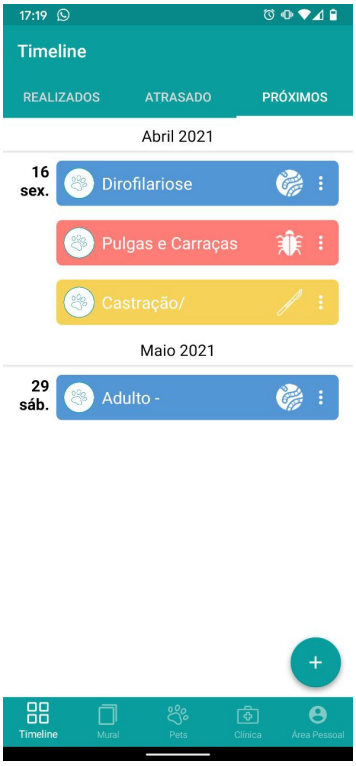
Petable é um aplicativo português, disponível para Android e iOS, que tem como objetivo ajudar o usuário a cuidar da saúde de seu animal de estimação. Oferecendo suporte a diversos tipos de animais, entretanto, o foco maior do aplicativo é em cães e gatos. A interface é bastante simples e amigável, muitas vezes apresentando telas com instruções de uso, facilitando a sua utilização.

Focado na medicina preventiva, oferece um conjunto de ferramentas projetadas por veterinários para manter o animal saudável, sendo a principal delas o plano de saúde do animal –vide figura 3.4 –, que projeta todas as vacinas e remédios que o animal necessita para evitar doenças típicas da raça e da espécie. O usuário tem a opção de cadastrar quantos animais desejar, cada um com seu próprio plano de saúde, além disso também é possível cadastrar a clínica veterinária de preferência, facilitando o contato em caso de

emergência. Para manter o usuário engajado o aplicativo conta com um blog interno que é atualizado constantemente com dicas escritas por veterinários.

O aplicativo também conta com um sistema de pontos, que são obtidos conforme o usuário cumpre o plano de saúde do animal. Esses pontos podem ser gastos em campanhas para causas animais, tais como: Patinhas Online, ABEAC e APATA.

Figura 3.4 – Plano de saúde planejado pelo aplicativo Petable.



Fonte: Autor

3.5 Comparativo de características

Nesta seção será desenvolvido o comparativo de características dos aplicativos Instagram, DogCha!, Petable e MediSafe.

3.5.1 Maturidade

A maturidade de um aplicativo é definida pelo tempo desde o lançamento de sua primeira versão até a data da escrita deste trabalho, quanto maior o tempo desde o lançamento do aplicativo maior a maturidade do mesmo. As informações utilizadas nesta

seção foram obtidas através do site AppTrace (2021), empresa dedicada à coleta e análise de dados de aplicações publicadas na *PlayStore* e *AppStore*.

Dentre os aplicativos analisados neste trabalho se destacam com maior maturidade, como podemos perceber pela tabela 3.1, o Instagram e MediSafe, publicados pela primeira vez em 2012 e 2013, seguidos pelo Petable que teve sua primeira versão lançada em 2014. O DogCha!, inicialmente publicado em 2019, é o menos maduro dos aplicativos analisados.

	Instagram	**DogCha!**	**MediSafe**	**Petable**
Maturidade (anos)	9	2	8	7
№ de downloads (estimativa)	$1bi$–$2bi$	$10mil$–$50mil$	$1mi$–$5mi$	$10mil$–$50mil$
Sistema operacional	Android e iOS	Android e iOS	Android e iOS	Android e iOS

Tabela 3.1 – Tabela com comparativo de maturidade e número de downloads dos aplicativos relacionados.

3.5.2 Quantidade de instalações

A *PlayStore* não expõem publicamente o número de downloads das aplicações, no entanto, disponibiliza uma faixa de instalações na qual o *app* se enquadra (e.g. entre 10 e 50 mil instalações). A *AppStore* não fornece esse tipo de informação, portanto nessa seção levaremos em conta apenas dados fornecidos pela Google para dispositivos Android.

O Instagram é o aplicativo com maior número de downloads dentre os analisados contendo mais de 1 bilhão de downloads. O MediSafe aparece em segundo, estando na casa do milhão de downloads. Entre o DogCha! e o Petable não podemos afirmar qual tem maior número de downloads, pois ambos encontram-se na mesma faixa, entre 10 mil e 50 mil downloads.

3.6 Comparativo de funcionalidades

Nesta seção é feito o comparativo das funcionalidades dos aplicativos analisados separando-as em duas categorias: Funcionalidades voltadas para redes sociais e funcionalidades voltadas para gestão de saúde.

	Instagram	**DogCha!**	**MediSafe**	**Petable**
Gerenciamento de múltiplos perfis	presente	ausente	ausente	presente
Compartilhamento de conteúdo	presente	presente	parcial	parcial
Ferramenta de edição de fotos e vídeos	presente	ausente	ausente	ausente
Conectividade com outros usuários	presente	presente	ausente	ausente
Ferramenta para gerenciar saúde	ausente	ausente	presente	presente
Filtro de conteúdo	parcial	presente	presente	ausente
Busca de conteúdo	presente	presente	ausente	ausente

Tabela 3.2 – Tabela com comparativo de funcionalidades dos aplicativos analisados.

3.6.1 Rede Social

O Instagram e o DogCha! apresentam capacidade de compartilhar fotos e vídeos, seguir amigos e ser seguido, visualizar o *feed* de postagens, curtir postagens, buscar conteúdo, enviar mensagens via chat privado e comentar em postagens. Apesar das semelhanças, a criação e gerenciamento de múltiplos perfis em uma mesma conta só é possível no Instagram. No *feed* de postagens o DogCha! apresenta uma ferramenta de filtragem onde o usuário pode escolher o tipo de postagem que deseja ver.

O MediSafe e o Petable não apresentam funcionalidades características de rede social, entretanto, ambos permitem compartilhar conquistas e conteúdo em redes sociais como Facebook e Instagram.

3.6.2 Gestão de saúde

O Petable e o MediSafe são aplicativos voltados para gestão de saúde, sendo o primeiro voltado para saúde de animal e o segundo para saúde de humanos, mesmo assim compartilham diversas funcionalidades como cadastro de remédios, calendário de medicações, lista de medicações atrasadas e alarmes para notificações. Dentre os dois apenas o Petable permite a criação e gerenciamento de múltiplos perfis, entretanto, o MediSafe permite que seja compartilhado com outro usuário o calendário de remédios para que o controle seja feito por mais de uma pessoa. O MediSafe conta com um cadastro e gerenciamento de medicações mais detalhado permitindo cadastro de alarmes recorrentes e intermitentes, tipos de comprimidos e fotos da prescrição.

As demais aplicações analisadas, DogCha! e Instagram, não apresentam ferramentas de auxílio gerenciamento de saúde, mesmo que, no Instagram seja possível en-

contrar com facilidade conteúdo voltado à saúde.

4 DESENVOLVIMENTO DO APLICATIVO: FLOCKR

Neste capítulo serão detalhados a concepção, desenvolvimento e implementação do Flockr, um aplicativo destinado a tutores de *pets* que combina funcionalidades de uma rede social com gestão de saúde para animais. A rede social teve como inspiração o Instagram e tem como objetivo atrair usuários para o aplicativo e mantê-los engajados, enquanto a gestão de saúde entrega ferramentas necessárias a todo tutor de animais de estimação, como: controle de vacinação e vermífugo, agenda de banho e tosa, registro de consultas veterinárias, entre outros.

A seguir, serão analisadas as *user stories* criadas, a prototipação, o ambiente de desenvolvimento, a arquitetura utilizada, a construção do *backend* e as funcionalidades do aplicativo. Ao final do capítulo serão apresentadas capturas de tela das principais funcionalidades da aplicação.

Apesar de o aplicativo ter sido desenvolvido como produto da empresa Mocka (MOCKA, 2021), o autor participou integralmente de todos os processos apresentados a seguir, com exceção da prototipação, *design* e *backend*, onde sua contribuição foi o acompanhamento e gerenciamento do desenvolvimento.

4.1 Modelagem das funcionalidades

Nesta seção serão detalhadas as duas principais técnicas de modelagem de funcionalidades utilizadas no desenvolvimento do Flockr: a criação de *user stories* e a prototipação telas e de funções.

4.1.1 *User stories*

Após a análise dos aplicativos relacionados e de entrevistas informais com usuários de redes sociais e donos de animais de estimação, foi criado um conjunto de *user stories* associadas a estes temas. Segundo Cohn (2004) *user stories* são curtas narrativas de usuários fictícios descrevendo uma funcionalidade que desejam na aplicação (e.g desejo poder comprar ração pelo aplicativo). Todas as *user stories* a seguir são escritas do ponto de vista do usuário e para cada uma delas foi atribuída uma prioridade de desenvolvimento –vide tabela 4.1.

Prioridade	**como ...**	**quero...**	**para...**
Alta	usuário	me cadastrar no sistema	conseguir acessar o aplicativo.
Alta	usuário	poder me logar no sistema	conseguir acessar o aplicativo.
Alta	usuário	me logar no sistema através de outras redes sociais	para acessar o sistema com mais facilidade.
Alta	usuário	cadastrar meu cachorro	poder postar fotos e vídeos dele.
Média	usuário	cadastrar todos meus animais de estimação	ser notificado de vaciná-los na data correta.
Alta	usuário	seguir meus amigos	acompanhar as fotos e atividades deles.
Alta	usuário	acessar e editar meu perfil	manter as minhas informações atualizadas.
Alta	usuário	deletar o perfil do meu *pet*	remover todas as informações do sistema.
Alta	usuário	acessar perfis de outros *pets*	ver fotos do mesmo.
Alta	usuário	comentar na publicação de outros usuários	expressar minha opinião sobre o conteúdo.
Média	usuário	curtir fotos e vídeos de outros usuários	demonstrar que gostei do conteúdo.
Alta	usuário	publicar fotos e vídeos	que outros usuários possam acompanhar meu *pet*.
Baixa	usuário	receber notificações e vídeos	saber quando tem conteúdo novo.
Baixa	usuário	enviar mensagens privadas	me comunicar com outros usuários.
Baixa	usuário	editar fotos com filtros antes de publicá-las	deixar a foto mais bonita.
Alta	usuário	deletar publicações	poder remover publicações erradas.
Média	usuário	cadastrar o peso do meu *pet*	manter histórico do seu peso.

Tabela 4.1 – Tabela com *user stories* criados para o Flockr

4.1.2 Prototipação

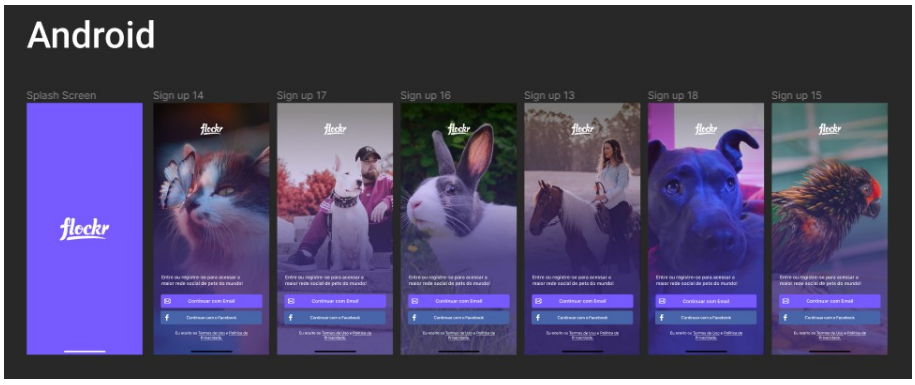
Durante a etapa de prototipação do Flockr a equipe de desenvolvimento trabalhou em conjunto com a equipe de *design* e, utilizando as *user stories* criadas anteriormente, foi

desenvolvido um protótipo inicial contendo apenas as funcionalidades de alta prioridade.

A prototipação das telas e da navegação da aplicação ocorreu em três etapas, a primeira mais simples foi um esboço das telas em um quadro negro, o que facilitou alterações e também proporcionou que todos os membros do time participassem do processo. Para a segunda etapa a equipe de *design* utilizou a ferramenta online Figma (2016) que tem como objetivo facilitar a prototipação de aplicações mobile, permitindo ao usuário projetar as telas desenhadas diretamente em um smartphone. A terceira e última etapa consistiu na elaboração, a partir das telas criadas na etapa dois, de um protótipo navegável através de um celular Android ou iOS. Para tanto, foi utilizada uma ferramenta interna ao Figma chamada de *Figma mirror*.

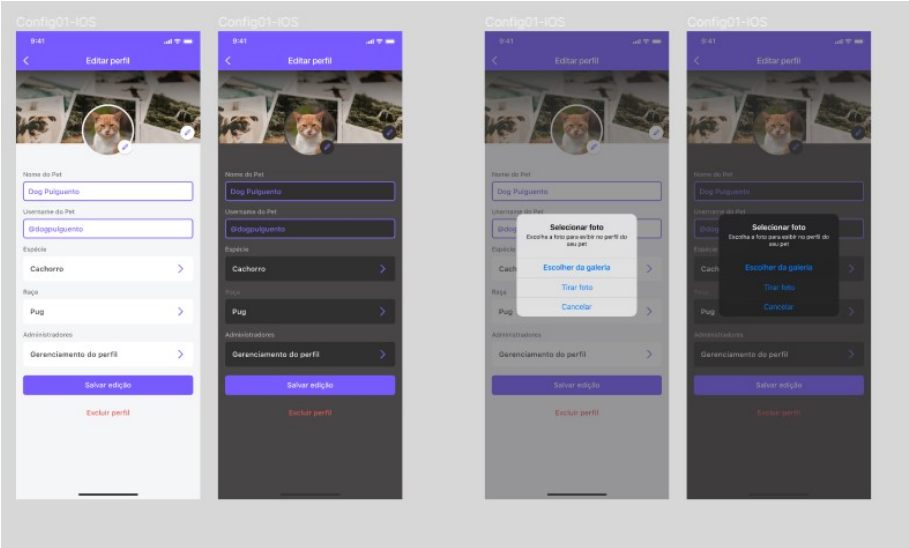
A partir do resultado obtido na etapa três – que em parte pode ser observado nas figuras 4.1 e 4.2 – a equipe de design realizou testes com usuários para identificar possíveis problemas de navegação e corrigi-los antes de dar início à próxima etapa de desenvolvimento.

Figura 4.1 – Protótipo final da tela de abertura do Flockr.



Fonte: Autor

Figura 4.2 – Protótipo final da edição de perfil.



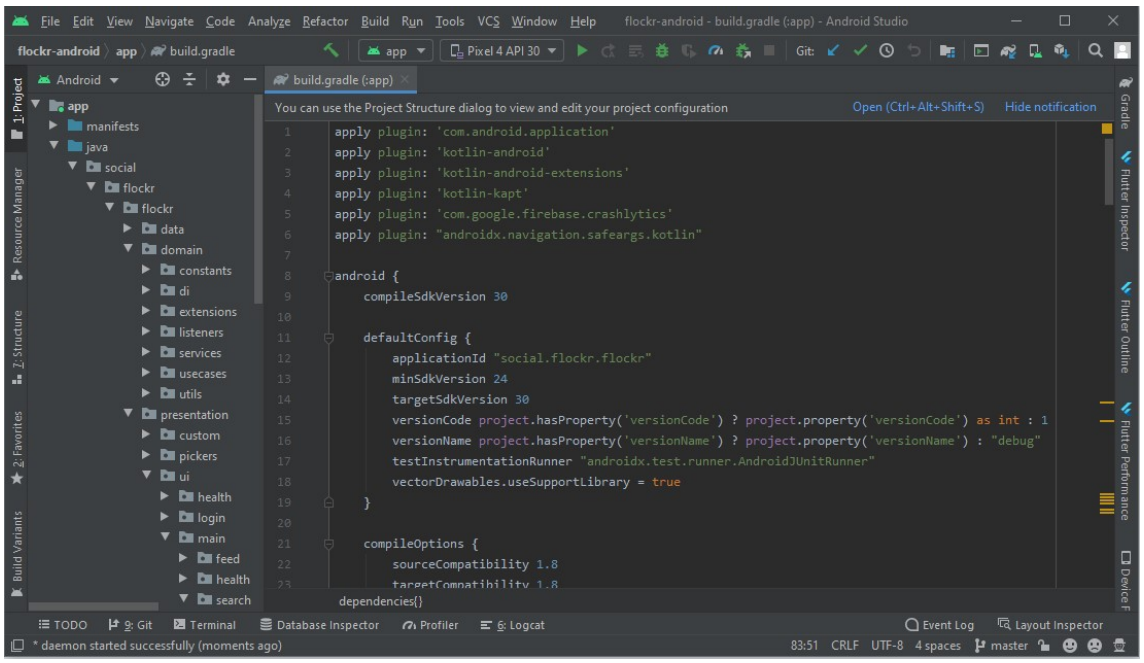
Fonte: Autor

4.2 Ambiente de desenvolvimento

Para o desenvolvimento do Flockr na plataforma Android foi utilizado o *Android Studio* – IDE oficial fornecida pela Google e lançada em 2014 – em conjunto com a linguagem *Kotlin*, que em 2017 se tornou a linguagem oficial da plataforma (MILLER, 2021). Baseado no software *InteliJ IDEA* desenvolvido pela *JetBrains*, o Android Studio oferece um conjunto de ferramentas necessárias para desenvolver uma aplicação Android tais como, compilação baseada em Gradle, refatoração voltada para *framework* Android, editor de layout e ferramentas de *Lint*, que indicam problemas de compatibilidade de versão e de performance.

Para compilação do *app* foi utilizado o sistema *Gradle* – já incluso no Android Studio – por facilitar o gerenciamento de dependências e da configuração do projeto. Dentro do arquivo de configuração do gradle –vide figura 4.3 – é possível escrever o código em Kotlin ou Groovy permitindo a automatização de processos que de outra forma teriam de ser feitos manualmente (e.g. antes da cada compilação atualizar todas as bibliotecas e rodar todos os testes unitários).

Figura 4.3 – Interface padrão do *Android Studio* demonstrando a edição de um arquivo gradle.



Fonte: Autor

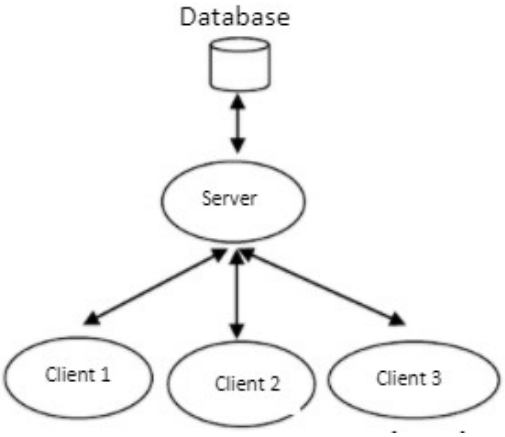
4.3 Arquitetura

Nesta seção será detalhada a arquitetura utilizada para o desenvolvimento do Flockr, com enfoque inicial no modelo cliente-servidor, seguido pelo padrão MVP + KMM. Também serão analisados o desenvolvimento da biblioteca utilizada pela aplicação e a distribuição do Flockr.

4.3.1 Cliente-Servidor

De acordo com Oluwatosin (2014) o modelo cliente-servidor é uma estrutura distribuída de aplicação onde os recursos e serviços são disponibilizados por um ou mais servidores e, através da internet, um ou mais clientes podem acessar estes recursos – vide figura 4.4.

Figura 4.4 – Processo de comunicação entre clientes e servidor



Fonte: Oluwatosin (2014)

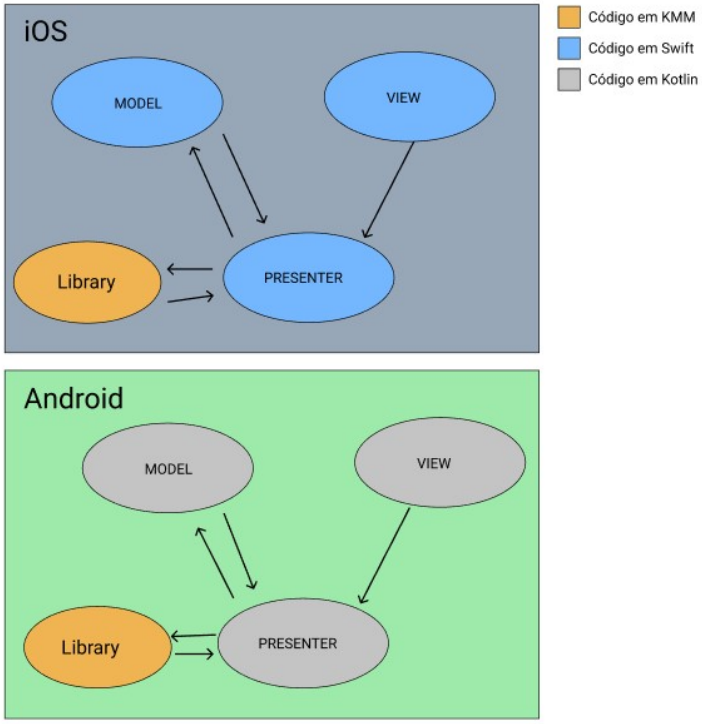
Neste trabalho cada smartphone executando o Flockr assume o papel de cliente e, através de uma *API* – detalhada na seção 4.5 – se conecta ao servidor que é responsável por fornecer os serviços necessários para o funcionamento do app.

4.3.2 Padrão MVP + KMM

No desenvolvimento do Flockr foi adotado o padrão MVP devido à maior separação de interesses na lógica de apresentação. Entretanto, esta lógica precisa ser reescrita para todas as plataformas em que o app será desenvolvido – no caso do Flockr essas plataformas são *Android* e *iOS*.

Com intuito de evitar retrabalho e unificar a parte lógica da aplicação em uma única base de código, foi feita uma modificação ao padrão MVP (Kotlin Foundation, 2021), na qual foi desenvolvida uma biblioteca em *KMM* contendo toda a parte lógica do aplicativo – detalhada na subseção 4.4.

Figura 4.5 – Figura demonstrando a interação das camadas em um MVP com KMM



Fonte: Autor

Conforme demonstrado na figura 4.5, após integrar a biblioteca ao projeto obtemos a seguinte arquitetura: *model*, *view* e *presenter* mantém com as mesmas atribuições do modelo tradicional MVP, entretanto, o *presenter* repassa todas as chamadas possíveis para biblioteca, ou seja, todas chamadas que não sejam específicas da plataforma. Como resultado foi possível isolar na biblioteca todas as requisições para o servidor, assim como a camada de autenticação e *cache*.

4.4 Biblioteca KMM

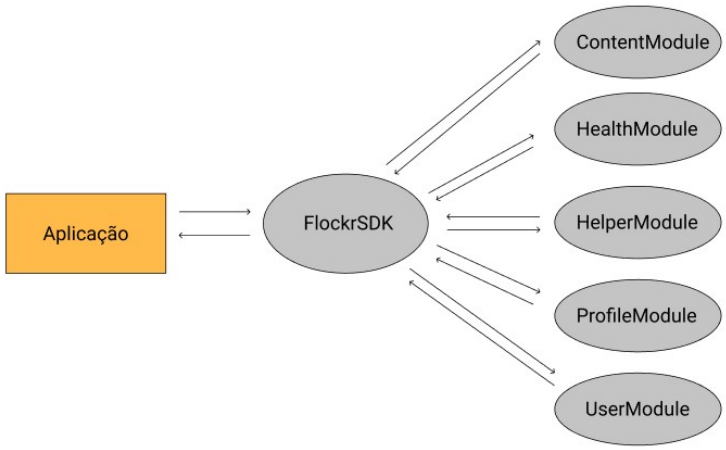
A primeira etapa no desenvolvimento da biblioteca foi identificar tudo o que poderia ser implementado utilizando o *framework KMM* e o que deveria ser implementado de forma nativa. Tomando este critério como base, foi estabelecido que toda a parte de comunicação com o servidor, serialização e desserialização de modelos e banco de dados ficariam na biblioteca, o restante seria implementado de forma nativa.

A partir da definição do objetivo, foi criado um projeto separado apenas para a biblioteca, utilizando o *Android Studio*. Dois plugins foram utilizados no desenvolvimento,

o *ktor* (JETBRAINS, 2021b) para chamadas REST e o *sqldelight* (SQUARE, 2021) para o banco de dados local.

A arquitetura interna da biblioteca ficou definida de forma bastante simples –vide figura 4.6–: Uma classe de entrada chamada *FlockrSDK* que contém todas as chamadas públicas da biblioteca, uma classe de configuração da biblioteca e cinco módulos contendo chamadas para *API*.

Figura 4.6 – Arquitetura interna da biblioteca



Fonte: Autor

Cada módulo possui a capacidade de fazer requisições ao servidor e armazenar as respostas em um banco de dados local, assim, caso o usuário repita a mesma chamada com frequência o módulo busca a informação diretamente do banco local sem a necessidade de repetir a chamada para o servidor. Caso o cliente não tenha internet disponível o módulo utiliza a mesma estratégia, buscando a informação diretamente no banco local. A biblioteca ainda conta com uma classe de configuração onde o usuário necessita informar as chaves de produção e desenvolvimento.

Para distribuição e utilização da biblioteca foi escolhida a forma recomendada pela Kotlin Foundation (2021). Este processo é constituído pela compilação para o formato Java bytecode para Android e binário nativo para iOS, gerando respectivamente uma arquivo *.aar* e um *.framework*. A versão final da biblioteca conta com 68 funções públicas e 37 *models* e mais de mil linhas de código.

4.4.1 Distribuição

O Flockr foi distribuído através das lojas oficiais das plataformas, *AppStore* (iOS) e *PlayStore* (Android). Para facilitar o processo de publicação, que costuma ser bastante demorado e burocrático, foi utilizado o software *Fastlane* (GOOGLE, 2021b), que possibilita a automatização de todo o processo de publicação, executando sequencialmente a compilação do projeto, o *upload* dos arquivos para a loja, a publicação do app e a atualização de textos e imagens.

Para testar o Flockr antes de torná-lo disponível para todo o público, o aplicativo foi inicialmente publicado em versão beta (onde apenas usuários selecionados podem realizar o download do app) e após dois meses de teste foi liberado para produção, permitindo o download para todos os usuários. Durante o período de *beta testing* são coletadas informações de *bugs* e *feedbacks* dos usuários, posteriormente corrigidos antes do envio do app para produção (GOOGLE, 2021c). Até o momento do desenvolvimento deste trabalho, o ciclo de *beta testing* é repetido a cada nova versão enviada para loja, porém com menor duração.

4.5 Backend

Nesta seção será realizada uma breve descrição do desenvolvimento do *backend* conferindo maior ênfase ao Parse Server e aos *endpoints* criados para a aplicação. A participação do autor nessa etapa do desenvolvimento consistiu na concepção, teste e modelagem dos *endpoints*.

4.5.1 Parse Server

O Parse Server é um *backend open-source* que pode ser executado em qualquer infraestrutura com suporte à *node.js* e *mongoDB*. Até o ano de 2017 o Parse Server era vendido como um BaaS (*Backend as a Service*) onde o desenvolvedor podia criar um *backend* escalável voltado para dispositivos *mobile* sem se preocupar com a necessidade de infraestrutura ou gerenciamento de servidores. Ao final de 2017 o Parse foi descontinuado e transformado em *open-source*, permitindo que os usuários continuassem utilizando todas as ferramentas criadas pela plataforma, contanto que o fizessem em sua sua própria

infraestrutura.

Como um *BaaS*, o Parse oferece uma camada de autenticação baseada em OAuth (OAUTH, 2021) dando suporte à autenticação com Google, Facebook, Twitter, Instagram, entre outros. Um ferramenta essencial para um *backend mobile* é o envio de *push notifications*, que também é oferecida pelo Parse Server e conta com uma implementação bastante simples permitindo o disparo de notificações através de gatilhos adicionados à inserções e remoções no banco de dados. As *features* apresentadas acima, e o fato do Parse Server ser *open source*, compõe os principais motivos para a decisão da equipe de desenvolvimento do Flockr utilizá-lo como *backend*.

4.5.2 *Endpoints*

A arquitetura utilizada para o desenvolvimento do Flockr, conforme explorado na seção 4.3.1, ocorre por meio dos *endpoints*, terminais de comunicação fornecidos do servidor para o cliente. Através dos *endpoints* o cliente pode realizar requisições para o servidor, tais como, executar rotinas, serviços ou apenas adquirir informações disponibilizadas pelo mesmo.

A API do Flockr contém um total de 54 *endpoints* que foram desenvolvidos por meio da ferramenta *cloud code*, fornecida pelo Parse Server para executar funções e rotinas no *server side*. Na tabela 4.2 serão listados os principais *endpoints* da aplicação.

4.5.3 Infraestrutura e *Deploy*

Toda a infraestrutura do aplicativo Flockr foi desenvolvida em um servidor hospedado pela empresa Digital Ocean que oferece planos escaláveis a um baixo custo e alta qualidade. Os serviços contratados são: uma máquina hospedada em Nova Iorque que conta com sistema operacional Ubuntu 20.04, 2GB de processador intel, 50GB de SSD NVMe, 5TB de transferência mensal e um serviço de backup automático.

Durante o desenvolvimento do *backend*, por ter sido avaliada como uma alternativa mais ágil e flexível, foi utilizada uma instância local do Parse Server versionada em um repositório git. Com o intuito de efetuar o *deploy* para a Digital Ocean é necessário acessar o servidor do Flockr via SSH e clonar o repositório git contendo a implementação. Após o processo de obtenção dos arquivos deve-se executar o serviço do Parse Server com

Endpoint	Tipo	Descrição
/users	POST	Cria uma nova conta no sistema
/login	GET	Loga o usuário no sistema
/logout	POST	Desconecta o usuário no sistema
/users/userObjectId	PUT	Edita informações da conta no sistema
/users/me	GET	Recupera dados da conta salvos no sistema
/requestPasswordReset	POST	Caso o usuário não lembre da senha esse endpoint inicia o processo de recuperação
/classes/Profile	POST	Cadastra um novo *pet* na conta do usuário
/classes/Profile/id	PUT	Edita o perfil do *pet* em uso
/classes/Profile/id	GET	Recupera informações do perfil do *pet* desejado
/classes/Profile/id	DELETE	Apaga o *pet* selecionado da conta do usuário
/classes/Post	GET	Lista todas as publicações na rede social levando em conta quem o perfil segue
/classes/Post/id	GET	Recupera todas as informações da publicação desejada
/classes/Post	POST	Cria uma nova publicação com o *pet* selecionado
/classes/Post/id	PUT	Permite editar publicação desejada
/classes/Post/id	DELETE	Apaga do sistema a publicação desejada
/files/file	POST	Permite o envio de arquivos para o sistema
/files/id	DELETE	Apaga do sistema o arquivo desejado
/classes/Comment	POST	Cria comentário em publicação
/classes/Comment/id	DELETE	Apaga comentário desejado
/classes/Comment/id	GET	Lista todos os comentários feitos na publicação desejada
/classes/Specie	GET	Lista todas as espécies de animais disponíveis para cadastro
/classes/Breed/id	GET	Lista todas as raças da espécie informada
/classes/Follow/id	POST	Permite seguir outros perfis
/classes/Follow/id	DELETE	Permite deixar de seguir um perfil
/classes/Follow	GET	Lista todos os perfis que o *pet* selecionado segue
/classes/Event	POST	Permite o cadastro de eventos relacionados a saúde do *pet*, tais como, vacinação, vermifugo, banho, entre outros
/classes/Event	PUT	Permite editar o evento informado
/classes/Event	GET	Lista todos os eventos cadastrados para o *pet* selecionado
/classes/Event/id	DELETE	Apaga do sistema o evento desejado

Tabela 4.2 – Tabela com os principais *endpoints* criados para o Flockr.

o comando *node start*.

4.6 Funcionalidades

Nesta seção será realizada uma análise das principais funcionalidades do aplicativo e uma breve descrição de sua implementação na plataforma Android.

4.6.1 *Login* e registro de usuários

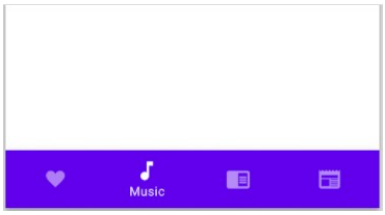
A fim de utilizar o aplicativo é necessário que o usuário cadastre uma conta nova ou efetue o login com uma já existente. O login pode ser feito de duas maneiras, através de um botão que utiliza uma conta já existente do Facebook, ou por meio de email e senha. Para criar a conta é necessário apenas informar o email ainda não cadastrado na plataforma e a senha desejada.

Os métodos referentes ao login e cadastro com email foram todos implementados na biblioteca detalhada na seção 4.4, ficando a cargo da parte nativa apenas transmitir os parâmetros necessários para a biblioteca. Efetuar o login com Facebook requer um passo adicional, a obtenção do token OAuth, que necessita de uma implementação nativa feita no *presenter* da tela de login. Esta implementação utiliza o SDK oficial do Facebook para obter o token necessário e, em posse do mesmo, repassa-o para a biblioteca finalizar o login no sistema do Flockr.

4.6.2 Navegação

A navegação principal do app ocorre através de uma barra inferior, chamada de *Bottom Navigation View*, contendo cinco ícones que, ao serem selecionados, alteram o fragmento principal. Esse tipo de navegação é recomendada pela Google nos padrões de desenvolvimento do *Material Design* que é uma diretriz de design desenvolvida pela companhia para dispositivos Android (GOOGLE, 2021a).

Figura 4.7 – Exemplo de uma Bottom Navigation View



Fonte: Google (2021a)

Iniciar um fluxo de navegação instanciando uma nova *Activity*, e dentro dela substituir os fragmentos conforme a interação do usuário com o aplicativo, é outro modo de navegação utilizado no Flockr. Esse método permite um maior controle da pilha de fragmentos, visto que pode-se desempilhar um a um ou todos de uma única vez ao finalizar a atividade.

4.6.3 Rede Social

A rede social do Flockr tem como principal funcionalidade o *feed* de fotos, onde o usuário pode ver todas as últimas publicações feitas pelos seus amigos –vide figura 4.8. Para obter as informações e conteúdos necessários para a *feature*, é invocado o método *getFeed* da a biblioteca descrita na seção 4.4, que efetua a comunicação, requisições e lógicas necessárias para obtenção das informações, retornando apenas o conteúdo já tratado para ser exibido na parte nativa da aplicação.

Figura 4.8 – Tela contendo o *feed* de publicações do Flockr

Fonte: Autor

Para exibição das publicações na lista foi utilizado um componente chamado *RecyclerView* (ANDROID DEVELOPERS, 2021b), que é responsável por exibir as *views* sequencialmente de forma eficiente aperfeiçoando a capacidade de resposta do app e reduzindo o consumo de energia. Para trabalhar com a *RecyclerView* é necessária a implementação de um *Adapter*, responsável pela lógica de exibição da lista.

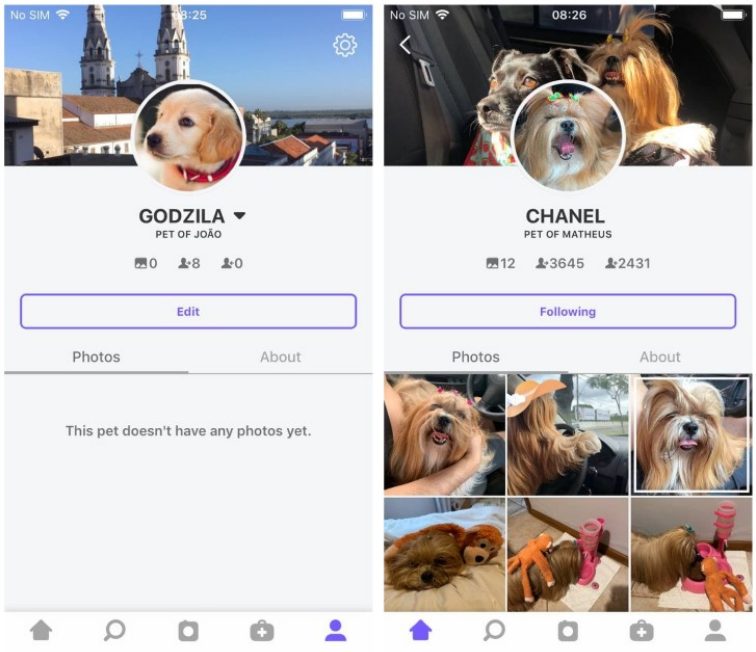
4.6.4 Perfil

No Flockr a tela de perfil de usuários pode ser acessada através de dois fluxos diferentes, liberando ou não a edição do conteúdo do mesmo. Ao acessar a tela de perfil pelo menu apresentado na subseção 4.6.2, é exibido o perfil do *pet* utilizado pelo próprio usuário, logo, a edição do conteúdo é habilitada. Já na tela de *feed*, ao clicar no nome de algum *pet*, a tela de perfil abre em outro modo, como um novo fluxo de navegação, permitindo assim apenas a visualização do conteúdo.

Para unificar esses dois modos de exibição em uma única implementação foi utilizado um fragmento contendo todo o conteúdo da tela, assim, podendo ser instanciado

onde for necessário. O controle de edição é feito a partir de uma checagem no campo *isOwner* contido no modelo *Profile*, caso o valor contido seja verdadeiro a edição é habilitada, caso contrário não.

Figura 4.9 – Comparação entre os dois tipos de perfis, o próprio e o de terceiros.



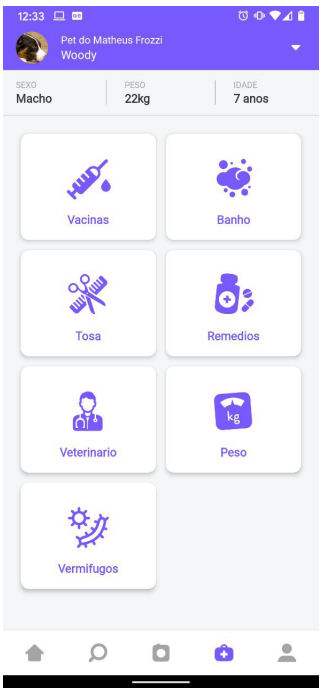
Fonte: Autor

4.6.5 Controle de saúde animal

O Flockr, além da função rede social, oferece também um controle completo da saúde do animal –vide figura 4.10 – permitindo o cadastro e gerenciamento de vacinas, banhos, tosas, remédios, vermífugos, consultas veterinárias e controle de peso. Para manipular um conjunto tão extenso de informações, todo o conteúdo cadastrado foi implementado como um objeto do tipo *Event* e dentro deste objeto o valor *type* informa a categoria à qual o evento pertence (e.g. evento de vacinação). A obtenção, cadastro, edição e deleção de eventos foi implementada utilizando a biblioteca citada na seção 4.4, portanto, estes métodos foram compartilhados entre as duas plataformas.

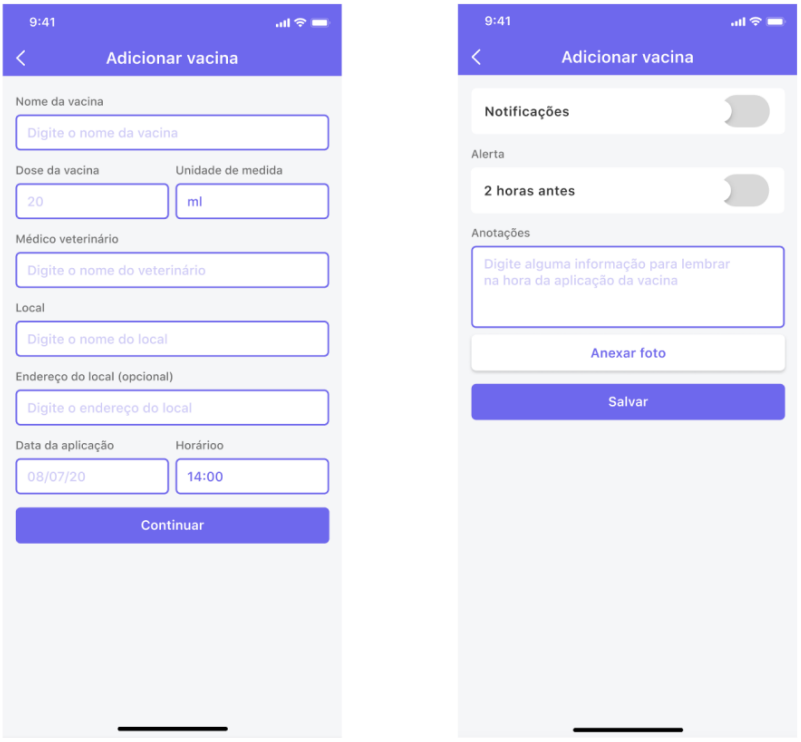
Durante o cadastro demonstrado na figura 4.11, o usuário deve incluir certas informações relativas ao evento, tais como: nome, informações específicas (no caso da vacina a dosagem), médico veterinário, local, foto e data. Após o cadastro inicial é oferecida ao usuário a opção de ativar uma notificação antes da data cadastrada.

Figura 4.10 – Tela principal da gestão de saúde do animal



Fonte: Autor

Figura 4.11 – Cadastro de vacinação

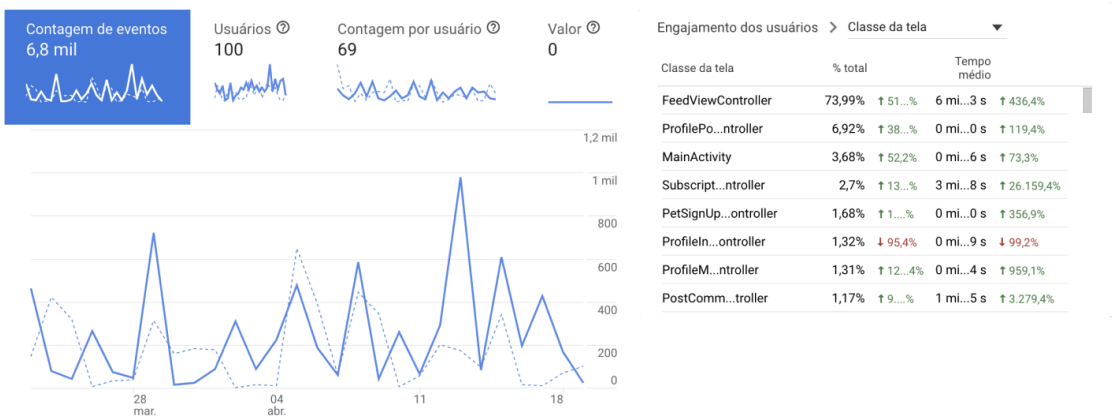


Fonte: Autor

4.7 Análise de dados

Para monitorar a interação dos usuários com o aplicativo foi utilizado o serviço *Google Analytics* que oferece uma ferramenta gratuita para coleta de dados de aplicações web e mobile. Para realizar o monitoramento do comportamento do usuário dentro do aplicativo foram cadastrados eventos de *log* em todas as telas e funcionalidades, assim, através de uma interface web disponibilizada pelo serviço, é possível avaliar quais funcionalidades e fluxos estão apresentando maior número de acessos –vide figura 4.12.

Figura 4.12 – Registro de eventos de abertura de tela durante o período beta do Flockr.



Fonte: Autor

Com base no comportamento dos usuários e com os dados coletados através do *Google Analytics*, será possível identificar problemas de usabilidade no app, assim como funcionalidades que não estão agradando o público base. Também através desses dados serão decididas quais das funcionalidades do Flockr serão ampliadas.

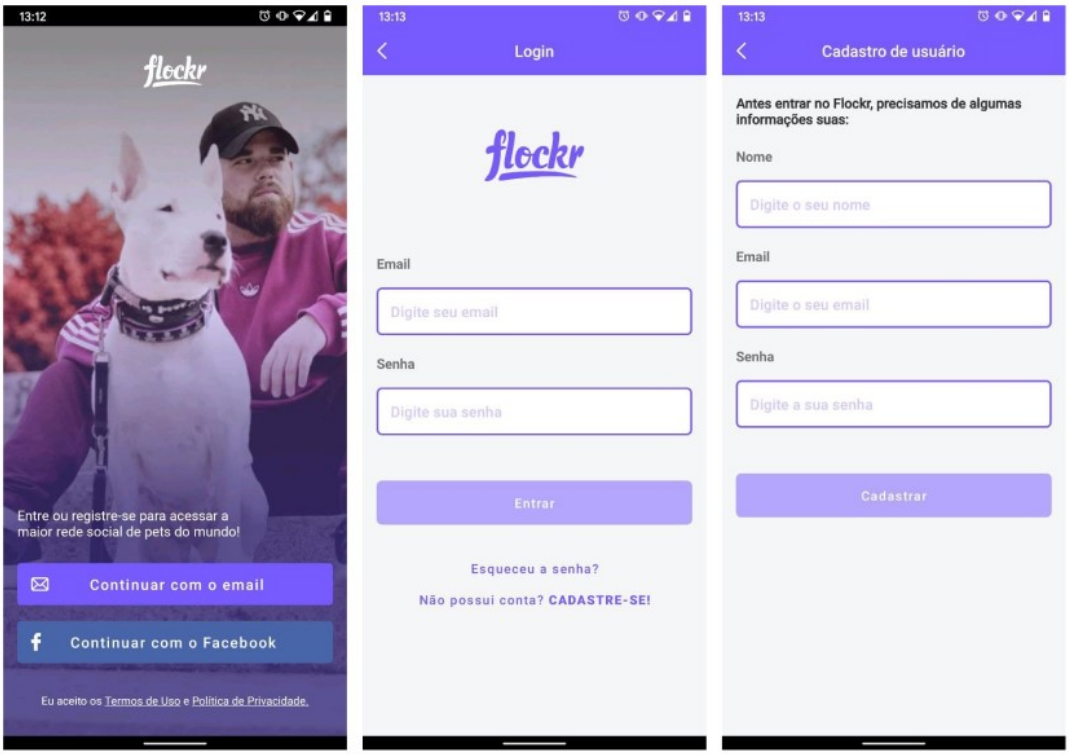
4.8 Telas

Esta seção apresenta as principais telas do Flockr, descrevendo a interação do usuário com as mesmas. Os dispositivos utilizados para captura de tela foram selecionados por se encontrarem disponíveis no momento do desenvolvimento e atenderem aos requisitos mínimos para a instalação do aplicativo, são eles: um Xiaomi Mi Mix3 com Android 11 e um Iphone6s com iOS 12.5.2.

4.8.1 *Login*

Ao entrar no aplicativo, caso o usuário não esteja logado, ele será encaminhado para o fluxo de login apresentado na figura 4.13. Nele existem duas possibilidades para acessar o sistema: através de uma conta do Facebook ou através de email e senha. Se necessário, ainda é possível cadastrar uma nova conta, informando nome, email e senha desejada. A navegação detalhada desta funcionalidade pode ser observada no vídeo disponível em https://youtu.be/L6QgZ5qKHF4.

Figura 4.13 – Fluxo de login e cadastro de conta no Flockr.



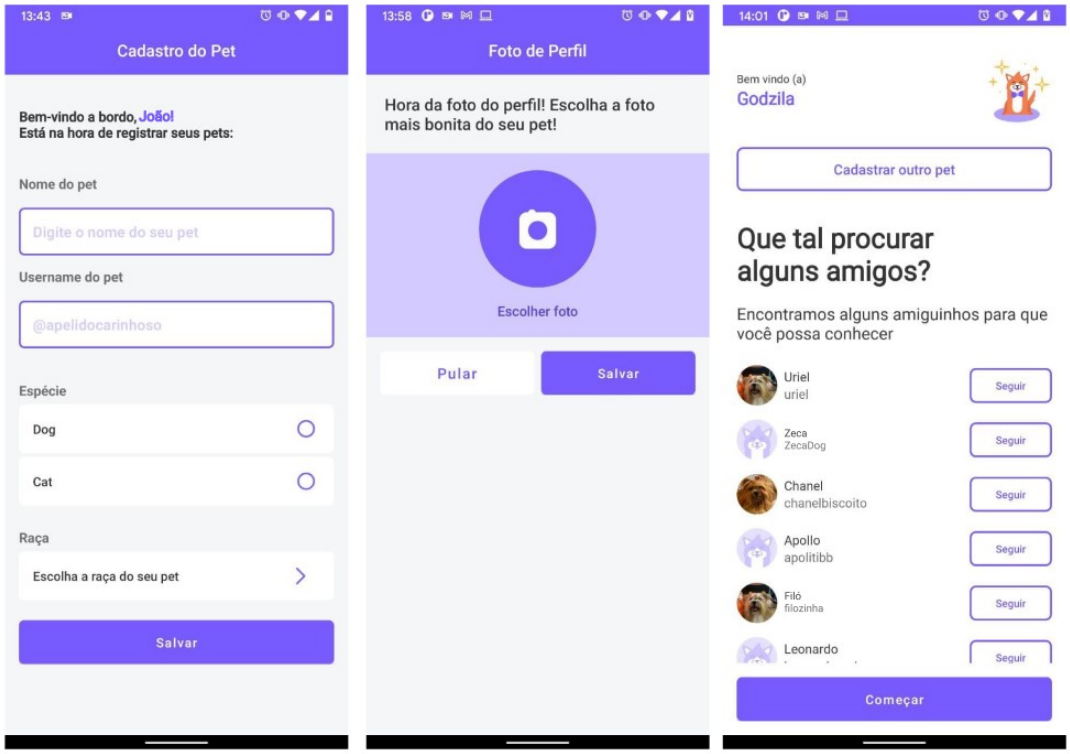
Fonte: Autor

4.8.2 Cadastro de perfil

Para obter acesso ao aplicativo, o usuário precisa ter ao menos um animal de estimação cadastrado, deste modo, o fluxo de cadastro de *pet* é iniciado logo após a criação de uma nova conta –vide figura 4.14–. Para realizar o registro do *pet* é necessário informar o nome, espécie e raça do animal e escolher o nome de usuário desejado. Também é oferecida uma ferramenta para incluir a foto de perfil do animal, nela é possível carregar

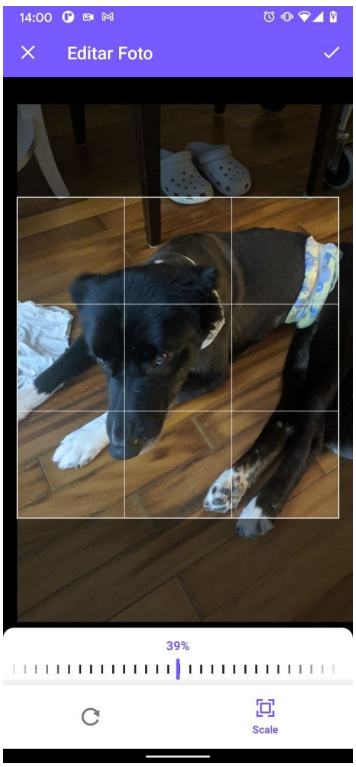
a foto da câmera ou da galeria do dispositivo e editá-la, rotacionado e cortando conforme apresentado na figura 4.15. Ao final do fluxo de cadastro é apresentada ao usuário uma lista contendo os perfis mais populares do Flockr, que podem ou não ser seguidos. A navegação detalhada desta funcionalidade pode ser observada no vídeo disponível em https://youtu.be/L6QgZ5qKHF4?t=33.

Figura 4.14 – Fluxo de cadastro de *pet*.



Fonte: Autor

Figura 4.15 – Ferramenta para recortar a foto de perfil.

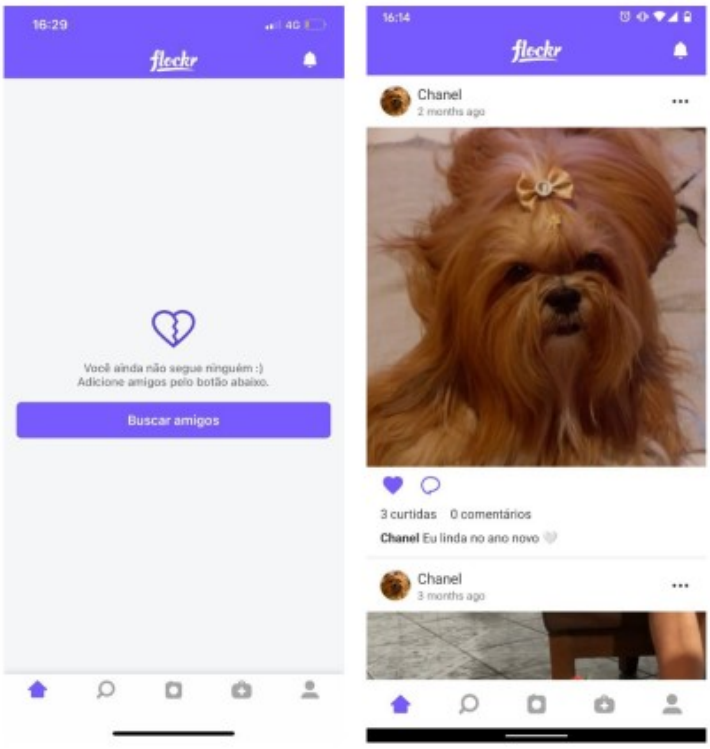


Fonte: Autor

4.8.3 *Feed*

Após logar no aplicativo o usuário é direcionado diretamente ao *feed* de publicações do Flockr, que é populado com conteúdo pertencente aos perfis seguidos pelo perfil do *pet* em uso no momento. Nesta tela é possível interagir com as publicações apresentadas através dos botões curtir e comentar. Além disso, existe a possibilidade de denunciar a publicação por conteúdo impróprio. Caso o *pet* não siga nenhum perfil é oferecido um botão de buscar amigos, que abre a tela de busca detalhada na subseção 4.8.4. Um vídeo com a utilização do *feed* está disponível em https://youtu.be/L6QgZ5qKHF4?t=85.

Figura 4.16 – Tela contendo o *feed* de publicações selecionas para o usuário.

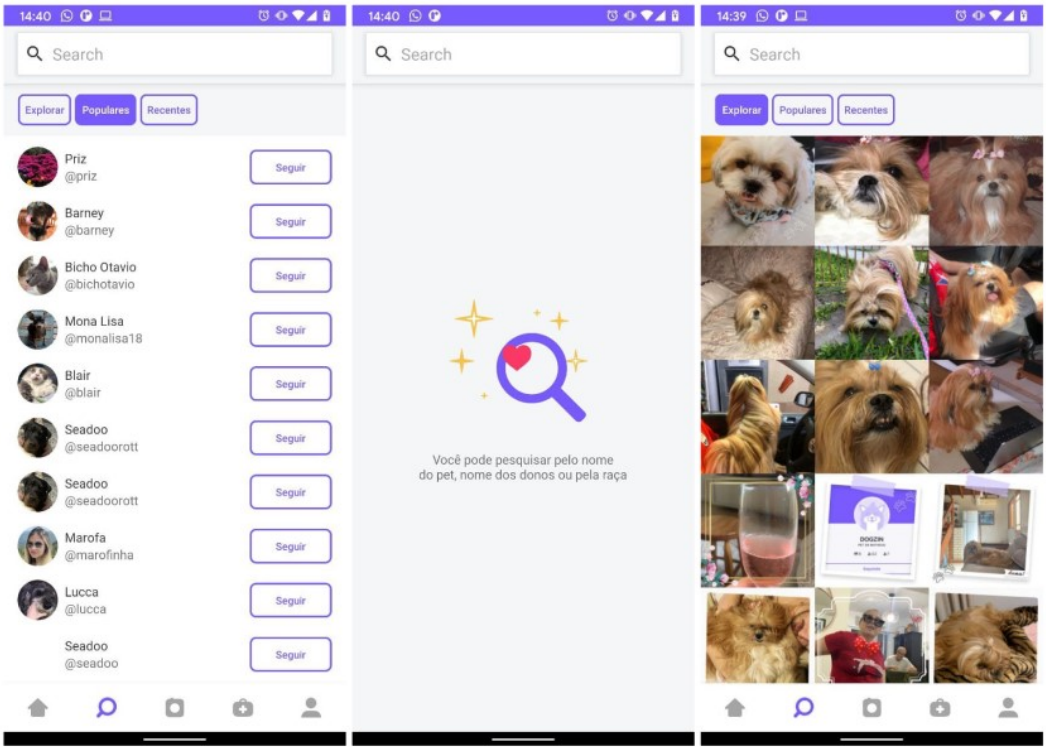


Fonte: Autor

4.8.4 Busca

O Flockr oferece ao usuário uma ferramenta completa de busca – figura 4.17– onde é possível encontrar perfis de diversas maneiras diferentes: buscando pelo nome do *pet* ou do dono, listando os perfis mais populares ou mais recentes e ou explorando através de fotos publicadas no app. A utilização da ferramenta de busca pode ser observada no vídeo disponível no *link* https://youtu.be/L6QgZ5qKHF4?t=135.

Figura 4.17 – Telas demonstrando as opções de busca de perfil.

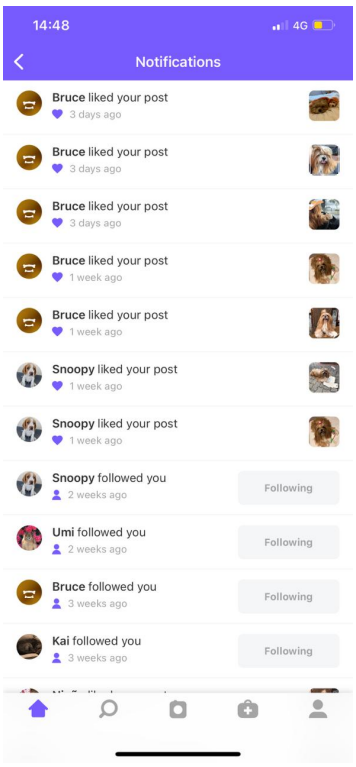


Fonte: Autor

4.8.5 Notificações

A seção de notificações –figura 4.18–, dentro da funcionalidade rede social, apresenta ao usuário um resumo dos últimos acontecimentos envolvendo o perfil em uso, tais como: fotos curtidas, perfis que o seguiram e eventos envolvendo a saúde do *pet*. Cada notificação possui uma ação específica, o direcionamento para a publicação que contém a foto curtida é realizado a partir de sua seleção no app, já a notificação que informa que o perfil foi seguido, apresenta um botão para seguir de volta.

Figura 4.18 – Telas de notificações.

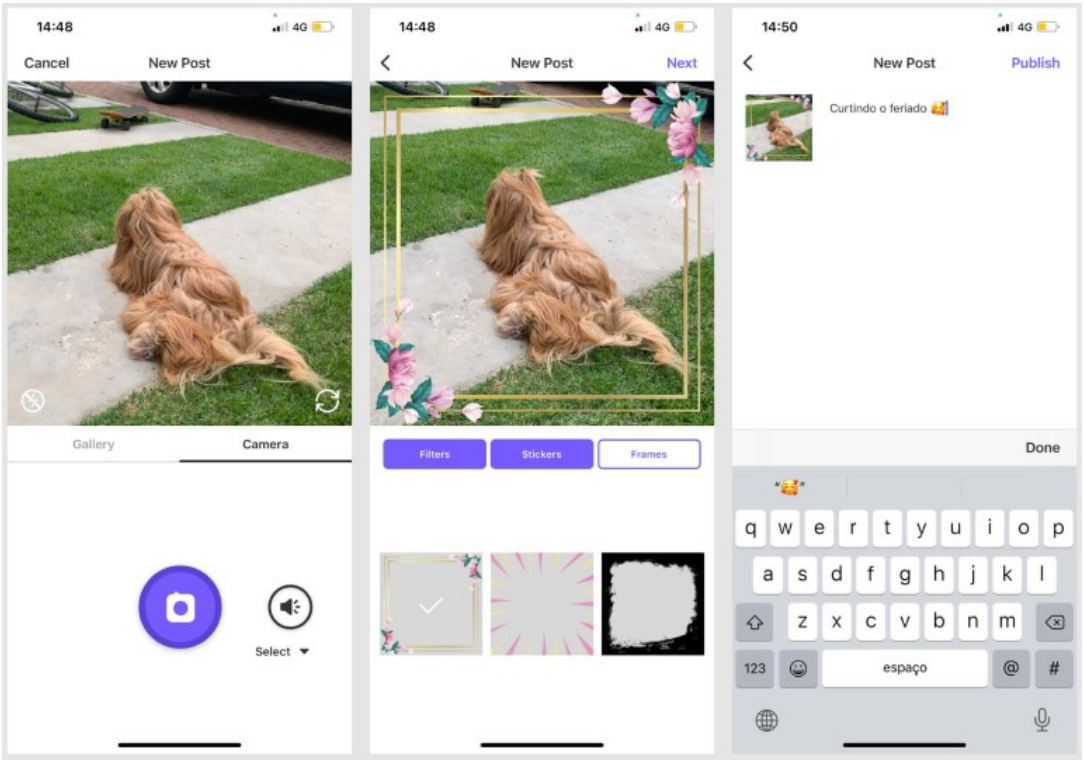


Fonte: Autor

4.8.6 Criação de conteúdo

O fluxo para publicação de conteúdo no Flockr é composto por três etapas: seleção da foto desejada, edição e elaboração de legenda para a mesma. Para obtenção da imagem o usuário pode capturá-la instantaneamente por meio da câmera do dispositivo, ou realizar o carregamento a partir da sua galeria de imagens. Com o intuito de deter a atenção dos animais de estimação durante a captura das imagens, foi desenvolvida uma funcionalidade adicional: por meio do aplicativo, é possível acionar diversos sons atrativos aos animais, facilitando o registro. Na edição é possível rotacionar a imagem, cortá-la, aplicar filtros e adesivos. Por fim, o usuário é apresentado a um campo de texto livre para escrever o que desejar sobre a publicação. O link *link* https://youtu.be/L6QgZ5qKHF4?t=177 contém um vídeo demonstrando o fluxo de publicação de conteúdo.

Figura 4.19 – Fluxo de publicação de conteúdo.



Fonte: Autor

5 AVALIAÇÃO

Neste capítulo será apresentada a avaliação do projeto Flockr de forma integral, levando em conta não apenas o desenvolvimento do software por meio das ferramentas já apresentadas, mas também a experiência do usuário no aplicativo.

A avaliação do aplicativo foi realizada através de um questionário online aplicado a usuários que utilizaram o Flocker durante o período de uma semana. Os usuários interagiram, durante o período de teste, com a versão para iOS disponível na App Store.

O questionário foi elaborado através da ferramenta Google Forms e é composto por quinze questões divididas em dois blocos de acordo com a tabela 5.1. O aplicativo tem como público alvo tutores de *pets*, independente de idade, gênero e escolaridade, portanto, com o intuito de atender à esta demanda, a partir da obtenção de um grupo heterogêneo de usuários, o questionário foi divulgado em 4 comunidades de WhatsApp compostas por grupos diversos, obtendo um universo amostral de doze participantes.

	Nºde questões	Tipo de questão
Bloco 1	1	Relativo ao sexo
	1	Relativo à idade
	1	Relativo ao nível de escolaridade
	1	Relativo ao uso de redes sociais
Bloco 2	4	Relativo ao cadastro no app
	3	Relativo ao uso da rede social
	2	Relativo ao uso da área de saúde
	2	Relativo ao uso geral do app

Tabela 5.1 – Tabela demonstrando os blocos de questões utilizadas na avaliação do Flockr.

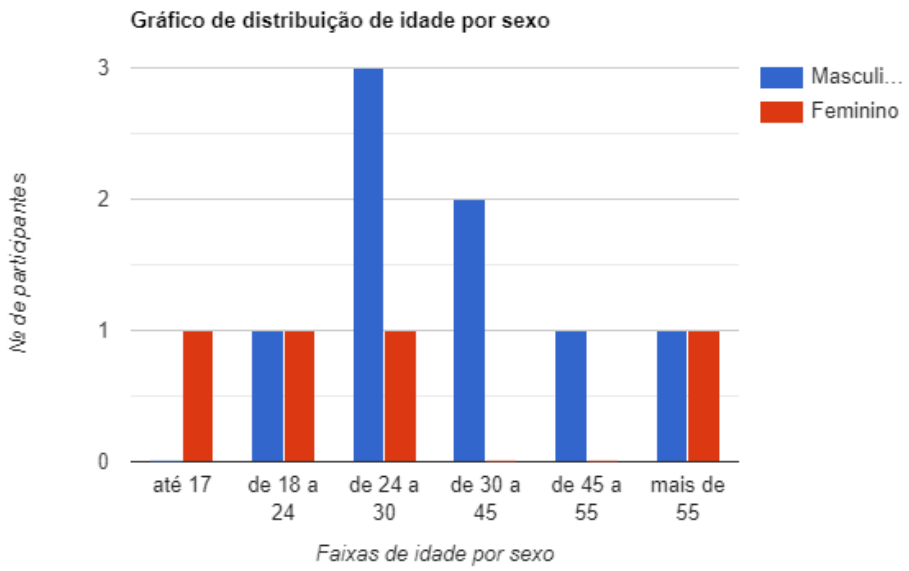
Nas seções seguintes será detalhado o resultado do questionário em duas etapas, na primeira será realizada a análise do bloco 1 e, na segunda, das afirmativas relacionadas ao bloco 2.

5.1 Perfil dos participantes

Nesta seção será realizada a análise das respostas do bloco 1, contendo as questões que contemplam a caracterização do usuário. O objetivo desta etapa foi coletar informações relevantes a respeito dos participantes para que fosse possível categorizá-los em grupos com características semelhantes, e assim, encontrar padrões de respostas para melhor entender sua interação com o aplicativo.

Ao avaliarmos o gráfico 5.1 podemos verificar que dos 12 participantes 8 são do sexo masculino, representando um total de 66,7%, enquanto 4 são do sexo feminino, representando 33,3%. Todos os participantes se identificaram com o gênero masculino ou feminino. A faixa etária entre 24 e 30 anos contém o maior número de participantes 33,3%, enquanto as faixas de 45 a 55 anos e maiores de 55 anos contém o menor número de representantes dentro da amostra, ambas representando 8,3%. As outras três faixas intermediárias correspondem, cada uma, a 16,7% da amostra.

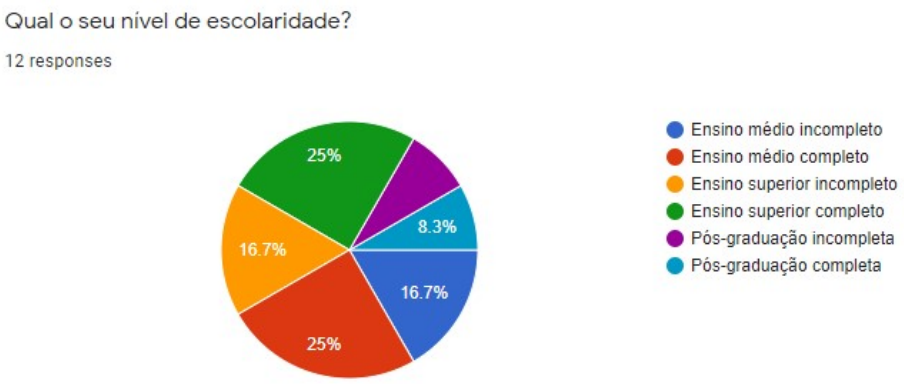
Figura 5.1 – Gráfico de distribuição de idade por sexo.



Fonte: Autor

O gráfico 5.2 apresenta a distribuição do nível de escolaridade dos participantes, onde podemos verificar que 5 participantes, representando 41,3% do total, possuem ao menos ensino superior completo, ao passo que apenas dois participantes, ou 16,7%, não possuem o ensino médio completo. O alto número de participantes com ensino superior pode estar relacionado ao fato de que nos grupos de WhatsApp, onde o questionário foi compartilhado, a maioria dos membros possuía graduação completa.

Figura 5.2 – Gráfico com o nível de escolaridade dos participantes.



Fonte: Google Forms

Por fim, o gráfico 5.3 apresenta a proporção de usuários com experiência na utilização de redes sociais em smartphones. Através dele podemos verificar que nenhum usuário declarou que não utiliza redes sociais no seu celular e apenas um usuário utiliza pouco. Da mesma forma, 50% dos participantes declararam que utilizam redes sociais todos os dias, apresentando familiaridade com as ferramentas ofertadas em redes sociais *mobile*. Estes dados serão relacionados com as respostas obtidas no bloco 2 na seção 5.2.

Figura 5.3 – Gráfico com o nível de experiência dos participantes na utilização de redes sociais em smartphones.

Fonte: Google Forms

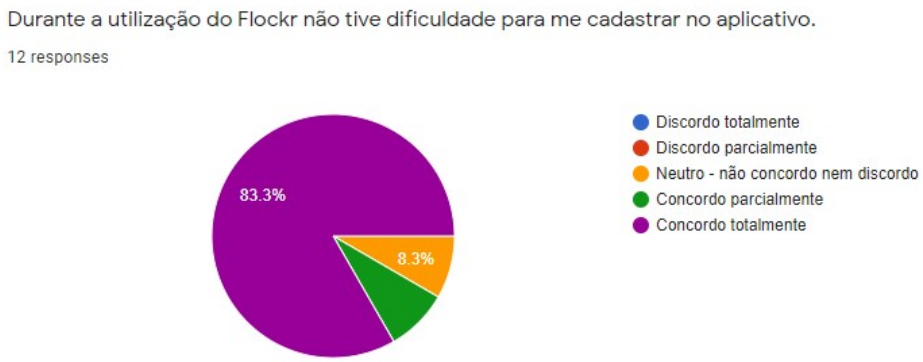
5.2 Análise dos resultados

Nesta seção será feita a análise das respostas referentes ao bloco 2, que tem como objetivo avaliar a navegação e experiência do usuário ao utilizar as funcionalidades do

Flockr. A fim de desenvolver melhor a análise, nesta etapa serão operadas relações entre as respostas obtidas nos dois blocos do questionário.

Os gráficos 5.4, 5.5, 5.6 e 5.7 apresentam informações sobre os cadastros de usuário e *pets* feitos no aplicativo e através deles podemos verificar que nenhum usuário declarou que encontrou dificuldades nos cadastros. Isso pode ser explicado pelo fato de que o fluxo de cadastro utilizado no Flockr foi baseado em outras redes sociais –vide capítulo 3– e de acordo com o gráfico 5.3 todos os participantes já utilizaram redes sociais em smartphones, facilitando assim o entendimento do processo de cadastro. Também é possível verificar que metade dos usuários efetuaram o login através do Facebook, um processo bastante simplificado para pessoas que já utilizam a rede social.

Figura 5.4 – Gráfico contendo informação sobre o nível de dificuldade que os usuários encontraram ao realizar o cadastro no aplicativo.



Fonte: Google Forms

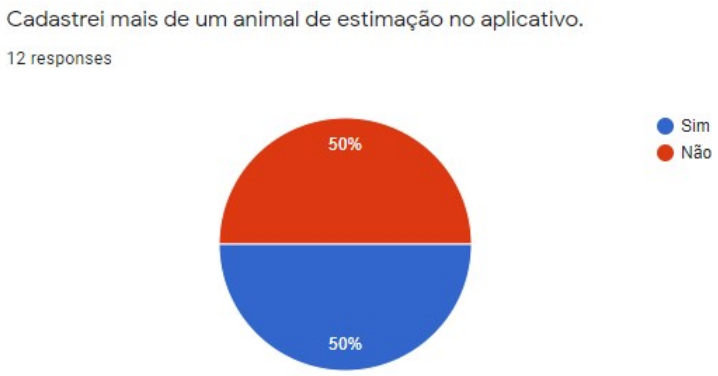
Figura 5.5 – Gráfico contendo informação sobre o método utilizado para cadastro no aplicativo.

Fonte: Google Forms

Figura 5.6 – Gráfico contendo informação sobre o nível de dificuldade que os usuários encontraram ao realizar o cadastro do *pet* no aplicativo.

Fonte: Google Forms

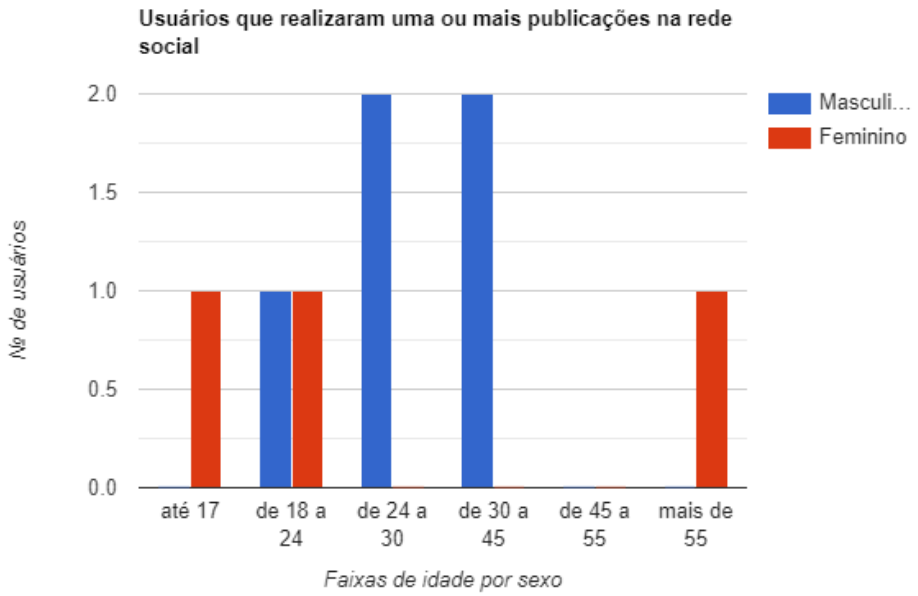
Figura 5.7 – Gráfico contendo informação sobre a quantidade de *pets* cadastrados por usuário.



Fonte: Google Forms

O gráfico 5.8 mostra que 8 usuários, ou 66,7% do total de participantes, realizaram publicações na rede social, sendo que em sua maioria possuíam menos de 30 anos. Apenas um usuário com mais de 45 anos realizou publicações e, de acordo com o gráfico 5.9, apenas 4 participantes utilizaram ferramentas de edição de imagens, como filtros e adesivos. Esses resultados demonstram que, dentre os participantes da pesquisa, as ferramentas de publicação nas redes sociais foram mais apelativas ao público mais jovem.

Figura 5.8 – Gráfico contendo informação sobre publicações na rede social.



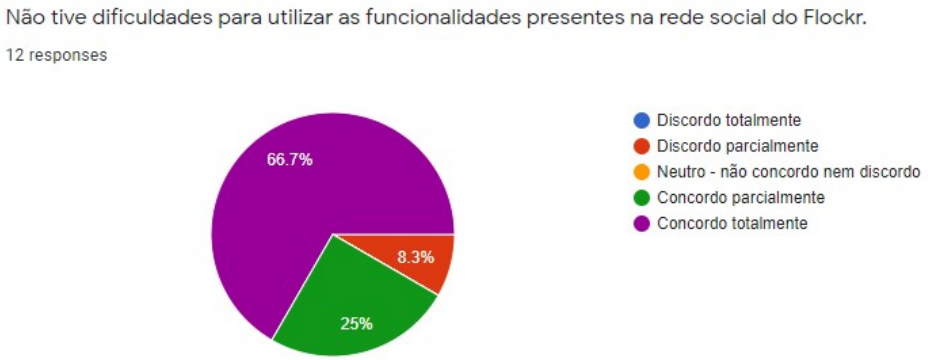
Fonte: Google Forms

Figura 5.9 – Gráfico contendo informação se a ferramenta de edição de imagem foi ou não utilizada durante as publicações.

Fonte: Google Forms

A partir do gráfico 5.10 identificou-se que apenas um usuário demonstrou alguma dificuldade na utilização da rede social presente no Flockr, este que já havia assinalado anteriormente – vide gráfico 5.3– que não utilizava redes sociais com frequência. Dentro do grupo que respondeu ao questionário, usuários de outras redes sociais não apresentaram dificuldades na utilização da rede social presente no aplicativo.

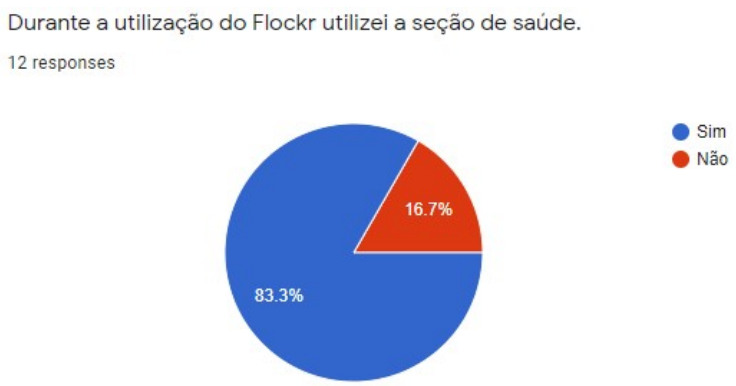
Figura 5.10 – Gráfico contendo informação se a ferramenta de edição de imagem foi ou não utilizada durante as publicações.



Fonte: Google Forms

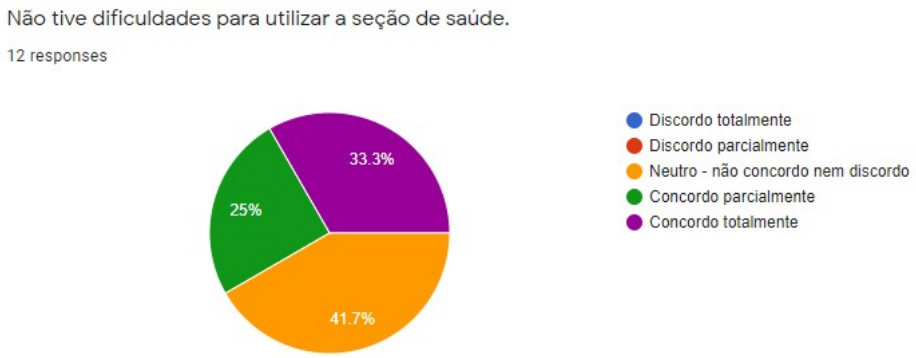
A análise dos gráficos 5.11 demonstra que apenas 2 usuários não utilizaram as ferramentas de gestão de saúde animal, portanto, pode-se concluir que, dentro da amostra, as ferramentas apresentadas no aplicativo se mostraram atrativas para tutores de *pets*. Já o gráfico 5.12 mostra que os usuários não tiveram dificuldades em utilizar as ferramentas da área de saúde animal, entretanto, 5 usuários permaneceram neutros perante a questão, sendo que dois deles não utilizaram as ferramentas.

Figura 5.11 – Gráfico contendo informação se a ferramenta de gestão de saúde foi utilizada.



Fonte: Google Forms

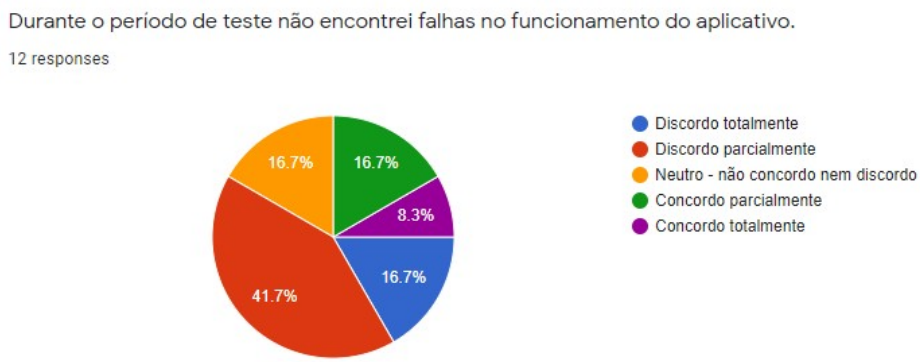
Figura 5.12 – Gráfico contendo informação se os usuários tiveram dificuldades ao utilizar a ferramenta de edição de saúde.



Fonte: Google Forms

A partir da análise dos gráficos 5.13 e 5.14 podemos identificar que mesmo que nenhum usuário tenha informado que apresentou dificuldades em utilizar a navegação e as ferramentas oferecidas no aplicativo, apenas 3 usuários, 25% da amostra, não encontraram falhas ou *bugs* na aplicação. Com essa informação é possível concluir que o aplicativo contém erros no código que estão gerando falhas ao usuário, portanto, se faz necessária uma investigação que possibilite correções em versões futuras.

Figura 5.13 – Gráfico contendo informação se os usuários encontraram falhas no aplicativo durante o período de uso.



Fonte: Google Forms

Figura 5.14 – Gráfico contendo informação se os usuários encontraram dificuldades na utilização do aplicativo.

Fonte: Google Forms

6 CONCLUSÃO

Este trabalho apresentou o desenvolvimento completo de um aplicativo *mobile* multiplataforma (Android e iOS) que foi desenvolvido como produto pela empresa Mocka (MOCKA, 2021), onde o autor ocupa o cargo de líder de desenvolvimento. No momento da elaboração deste trabalho, o Flockr está disponível na *AppStore* e na *PlayStore* na opção beta fechado. O Aplicativo ostenta a junção de uma rede social para *pets* com ferramentas de controle de saúde animal, onde a rede social cumpre o papel de atrair e manter o usuário engajado enquanto as ferramentas de saúde oferecem um diferencial único perante outros aplicativos concorrentes analisados no capítulo 3.

Durante a concepção do projeto, conforme apresentado na seção 4.3.2, foi tomada a decisão de construir a parte lógica da aplicação utilizando KMM. Esta tecnologia multiplataforma demonstrou cumprir todas as expectativas, uma vez que as bibliotecas geradas a partir dela não apresentaram perda de desempenho em relação aos métodos tradicionais, além de proporcionar uma maior agilidade no desenvolvimento.

A avaliação do Flockr demonstrou que o app cumpriu todos os requisitos planejados no início do projeto e que apresenta *bugs* que precisam ser corrigidos. Por meio da avaliação realizada também foi possível validar as *features* do app, visto que todas foram utilizadas ao menos uma vez pelos usuários que responderam ao questionário. No entanto, com o aumento da base de usuários do aplicativo, se faz necessário dar continuidade ao monitoramento constante do uso das funcionalidades, utilizando as ferramentas de coleta de dados citadas na seção 4.7.

A seguir serão descritas funcionalidades planejadas para o futuro do Flockr a fim de torná-lo mais atrativo para o usuário.

6.1 Trabalhos futuros

Em relação à rede social presente no Flockr, será implementado um sistema de sugestão de publicações, assim, quando não houver mais conteúdo novo para visualização, o *backend* irá retornar as publicações mais populares (não necessariamente publicadas por *pets* seguidos pelo usuário). Para o desenvolvimento desta funcionalidade serão apenas necessárias modificações na biblioteca em KMM e no *backend*.

Outra *feature* planejada para ser desenvolvida compete à área de saúde animal. Pretende-se possibilitar aos veterinários que realizarem o cadastro dos eventos de saúde,

assim o usuário receberia notificações dos mesmos mas não precisaria cadastrá-los. Para tanto, seria necessário criar um *website* de administração para que os veterinários gerenciem os eventos, bem como, a elaboração de uma ferramenta de permissão para que apenas o veterinário cadastrado pelo dono do *pet* possa editar os dados do animal.

Por fim, serão novamente corrigidos os *bugs* capturados nos relatórios obtidos através da ferramenta *Crashlytics* e, através do *Google Analytics*, será monitorada a utilização das funcionalidades do aplicativo. Caso alguma não esteja sendo utilizada, será realizado um estudo para identificar a causa e buscar o seu aperfeiçoamento.

REFERÊNCIAS

ABINPET. **Abinpet.org.br | Informações gerais do setor**. 2021. Available from Internet: <http://abinpet.org.br/infos_gerais/>.

ANDROID DEVELOPERS. **Conheça o Android Studio | Desenvolvedores Android**. 2021. Available from Internet: <https://developer.android.com/studio/intro>.

ANDROID DEVELOPERS. **Criar listas dinâmicas com o RecyclerView | Desenvolvedores Android**. 2021. Available from Internet: <https://developer.android.com/guide/topics/ui/layout/recyclerview>.

ANDROID DEVELOPERS. **Fragmentos | Desenvolvedores Android | Android Developers**. 2021. Available from Internet: <https://developer.android.com/guide/components/fragments>.

ANDROID DEVELOPERS. **Intents e filtros de intents | Desenvolvedores Android**. 2021. Available from Internet: <https://developer.android.com/guide/components/intents-filters>.

APPTRACE. **App statistics | App store intelligence | apptrace**. 2021. Available from Internet: <https://www.apptrace.com/>.

COHN, M. **User Stories Applied: For Agile Software Development (Addison Wesley Signature Series)**. [S.l.: s.n.], 2004.

EXAME. **Mercado sem crise: com alta de 13,5% em ano de pandemia, o setor pet crescerá mais em 2021 | Exame**. 2021. Available from Internet: <https://exame.com/bussola/mercado-sem-crise-com-alta-de-135-em-ano-de-pandemia-o-setor-pet-crescera-mais-em-2021/>.

FIGMA, I. **Figma**. 2016. Available from Internet: <https://www.figma.com/>.

GOOGLE. **Bottom navigation - Material Design**. 2021. Available from Internet: <https://material.io/components/bottom-navigation>.

GOOGLE. **fastlane - App automation done right**. 2021. Available from Internet: <https://fastlane.tools/>.

GOOGLE. **Firebase Crashlytics | A powerful Android and iOS crash reporting solution**. 2021. Available from Internet: <https://firebase.google.com/products/crashlytics>.

JETBRAINS. **JetBrains/kotlin: The Kotlin Programming Language**. 2021. Available from Internet: <https://github.com/JetBrains/kotlin>.

JETBRAINS. **Ktor: Build Asynchronous Servers and Clients in Kotlin | Ktor Framework**. 2021. Available from Internet: <https://ktor.io/>.

Kotlin Foundation. **Kotlin Multiplatform Mobile**. 2021. Available from Internet: <https://kotlinlang.org/lp/mobile/>.

MEDISAFE. **Medisafe - Medication management solutions**. 2021. Available from Internet: <https://www.medisafe.com/?lang=es>.

MILLER, P. **Google is adding Kotlin as an official programming language for Android development - The Verge**. 2021. Available from Internet: <https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017>.

MOCKA. **Mocka - Mobile & Web Development**. 2021. Available from Internet: <https://mocka.site/>.

MOLLA, R.; WAGNER, K. **People Spend Almost As Much Time on Instagram as They Do On Facebook**. 2021. Available from Internet: <https://www.vox.com/2018/6/25/17501224/instagram-facebook-snapchat-time-spent-growth-data>.

OAUTH. **OAuth 2.0 — OAuth**. 2021. Available from Internet: <https://oauth.net/2/>.

OLUWATOSIN, H. S. Client-Server Model. **IOSR Journal of Computer Engineering**, IOSR Journals, v. 16, n. 1, p. 57–71, 2014. ISSN 22788727.

POTEL, M. MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java. **Taligent Inc**, n. C, p. 1–14, 1996. ISSN <null>. Available from Internet: <http://metrology.googlecode.com/svn-history/r350/trunk/doc/ebooks/mvp.pdf>.

SQUARE. **SQLDelight**. 2021. Available from Internet: <https://cashapp.github.io/sqldelight/>.

UPBIN, B. **Facebook Buys Instagram For $1 Billion**. 2021. 1–4 p. Available from Internet: <http://www.forbes.com/sites/bruceupbin/2012/04/09/facebook-buys-instagram-for-1-billion-wheres-the-revenue/>.