UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALINE MARIEL NICHELLE

APLICATIVO MÓVEL PARA GERENCIAMENTO DE DADOS DE ANIMAIS DE ESTIMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2018

ALINE MARIEL NICHELLE

APLICATIVO MÓVEL PARA GERENCIAMENTO DE DADOS DE ANIMAIS DE ESTIMAÇÃO

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Anderson Luiz Fernandes
Co-Orientador: Prof. Robison Cris Brito

PATO BRANCO
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

APLICATIVO MÓVEL PARA GERENCIAMENTO DE DADOS DE ANIMAIS DE ESTIMAÇÃO

POR

ALINE MARIEL NICHELLE

Este trabalho de conclusão de curso foi apresentado no dia 28 de junho de 2018, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Prof. MSc Anderson Luiz Fernandes
Orientador

Prof. MSc Robison Cris Brito
Coorientador

Prof. Esp. João Guilherme Brasil Pichetti

Profª Mª Andreia Scariot Beulke

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª Drª Beatriz Terezinha Borsoi
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

Pouco conhecimento faz com que as criaturas se sintam orgulhosas. Muito conhecimento, que se sintam humildes. É assim que as espigas sem grãos erguem desdenhosamente a cabeça para o céu, enquanto que as cheias a baixam para a terra, sua mãe.

Leonardo da Vinci

RESUMO

NICHELLE, Aline Mariel. Aplicativo móvel para gerenciamento de dados de animais de estimação. 2018. 57 f Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

A convivência da humanidade com animais de estimação tem se intensificado ao longo dos últimos anos, tendo em vista a necessidade das pessoas de companhia. Concomitante a este crescimento, nota-se o aumento considerável da utilização de celulares e aplicativos móveis disponíveis no mercado. Assim, este trabalho, tem como objetivo auxiliar os donos no cuidado do animal de estimação por meio de aplicativos para celular para o gerenciamento de dados como: dados do animal e do seu dono, vacinações, consultas e exames. Para o alcance do objetivo procedeu-se uma análise sobre o mercado de animal de estimação e a utilização de aparelhos celulares juntamente com a utilização de aplicativos móveis. Com base nessas prerrogativas, estre trabalho propôs o desenvolvimento de um aplicativo na plataforma Android (sistema operacional da Google). Para isso, optou-se pela utilização do Android Studio para a execução do projeto, escolha dos ícones de aplicação e configuração do layout. Para testes práticos, foi utilizado celular compatível com a versão criada 4.0.3. A ferramenta Android Studio demonstrou ter um bom desempenho para o desenvolvimento do aplicativo e para a implementação das notificações, as quais ajudará o usuário no gerenciamento dos dados do animal.

Palavras-chave: Animal de estimação; Android; Gerenciamento de dados.

ABSTRACT

NICHELLE, Aline Mariel. Aplicativo móvel para gerenciamento de dados de animais de estimação. 2018. 57 f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

The interaction between Mankind and Pets has been intensifying within the last few years, regarding the friendship needed by man from any companion. With this growth, it is notable the considerable rise of mobile phone app's available on the current Market. Thus, this work aims to assist the pet owner through important data like: Owner and pet information, vaccinations, medical check-ups and exams. To reach the objective, have been made several analysis of the current mobile phone pet app's Market. Based on this prerogatives, has been developed on Android Platform (Google's operational system) the App. The Android Studio was used to execute, choose icons and configurate the Layout's App. Regarding practice tests, was used a compatible 4.0.3 cell phone Android version. The Android Studio tool demonstrated good performance along with the implementation of notifications, an application that assists the user in managing the animal's data.

Keywords: Pet; Android; Data management.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

ADT	Android Development Toolkit
API	Interfaces de Programação de Aplicações
UML	Linguagem de Modelagem Unificada
XML	Linguagem Extensível de Marcação Genérica
IDE	Ambiente de Desenvolvimento Integrado
JDK	Kit de Desenvolvimento Java
SDK	Pacote de Desenvolvimento de Software
WTP	Plataforma de Ferramentas da Web
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
JSON	Javascrip Object Notation

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, o objetivo geral e os específicos, a justificativa e a estrutura do trabalho.

1.1 CONSIDERAÇÕES INICIAIS

A convivência entre seres humanos e animais de estimação intensifica-se a cada ano. Conforme pesquisa constata-se que há um aumento significativo de cães e gatos nas casas dos brasileiros. Cerca de 44,3% dos lares possuem ao menos um cão e 17,7% têm pelo menos um gato. A estimativa é que a população de cães e gatos no Brasil, seja de 74,3 milhões. (Instituto Brasileiro de Geografia e Estatística, IBGE, 2015).

Para Knoploch (2015), números como estes mostram a probabilidade de no Brasil possuir mais cachorros do que crianças. Conforme pesquisa realizada, o país tinha aproximadamente 43 milhões de crianças de 0 a 14 anos. (Pesquisa Nacional por Amostra de Domicílios, PNAD, 2015).

A partir destes dados, pode-se entender que as pessoas, famílias, estão tendo animais de estimação como companheiros.

Outro aumento considerável nos últimos anos é o do uso de celular pelos brasileiros. Em 2014, o Brasil estava em 6º lugar no mercado mundial de smartphone. Já no segundo trimestre de 2015, teve um aumento 23,5% comparado com o semestre anterior, cerca de 72 milhões de smartphone com acesso à Internet. (OPUS Software, 2014). Assim, pode-se afirmar que o uso de celulares e Internet, tornou-se essencial para a vida de muitas pessoas, pois torna a vida das pessoas mais práticas e móveis. Isso porque com o uso de smartphones é possível realizar desde uma ligação à realizar transações internacionais, sem precisar usar um computador para realizar essas atividades. E é esta praticidade e comodidade que torna seu uso essencial na vida de muitas pessoas.

A partir deste crescimento, tanto do companheirismo de um animal de estimação como da intensificação do uso de celular e suas funcionalidades, este trabalho propôs o desenvolvimento de um aplicativo móvel utilizando a plataforma Android, que possibilite ao dono de um cão, por exemplo, o gerenciamento histórico de seu animal.

Para isso, o aplicativo possui funcionalidades como o cadastro de um ou mais animais de estimação, seu nome, idade, gênero, raça, agendamentos, geração de lista de tratamentos e de exames. O sistema, ainda, conta com o gerador de QrCode, que auxilia o médico veterinário na visualização dos exames do animal.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um aplicativo utilizando a plataforma Android para gerenciamento de dados dos animais de estimação.

1.2.2 Objetivos Específicos

Por meio do sistema será possível:

- Notificar os usuários da data em que o animal tenha algum agendamento..
- Armazenar dados do histórico do animal, últimas consultas, tratamento e exames.
- Utilizar *QrCode* para obter informações de exame do animal durante a consulta.
- Permitir que usuários de diferentes redes tenham acesso às informações dos animais.

1.3 JUSTIFICATIVA

O aumento significativo de animais de estimação inseridos nos lares brasileiros é de grande relevância. De acordo com a Pesquisa Nacional da Saúde (PNS) e o IBGE realizada em 2015, de cada 9 domicílios 5 possuem cães ou gatos. O animal possui cuidados particulares, como por exemplo: ir ao médico veterinário, tomar vacinas regularmente, banho, tosa, alimentação balanceada, entre outros. Estas são algumas atividades que seus donos necessitam gerenciar para que se tenha um animal saudável em casa.

Concomitante a este aumento, observa-se o crescimento da utilização de aplicativos nos celulares, seja para conversar, compartilhar fotos ou gerenciamento pessoal e profissional. Cerca de 58%, em 2015 comparado ao ano anterior, de acordo com o *site Mobile Time*, e em média 20 aplicativos, de acordo com pesquisa realizada por Millward Brown, encomendada pela *Mobile Marketing Association* em dezembro e 2015.

O aplicativo desenvolvido como resultado deste trabalho, permite gerenciar os dados do animal, possibilitando um auxílio mais efetivo, e maior controle das atividades a serem realizadas para que o animal tenha uma vida de qualidade por meio de cuidados técnicos e domésticos.

Para a construção deste aplicativo, foram utilizadas diferentes tecnologias, pois a aplicação por ser acessada por aparelhos celulares contando com a tecnologia do *QrCode*. Esta tecnologia apresenta um código de barras bidimensional no qual trará informações do animal de estimação. E estas informações podem ser visualizadas por um médico veterinário para que possa acompanhar os cuidados do animal.

O sincronismo para que mais de um usuário receba as informações do mesmo animal de estimação, é realizado por meio de um *web service*, que permite a aplicações de diferentes linguagens enviem informações através de uma linguagem universal, como o *Xtesnbible Markup Language* (XML). Para a persistência dos dados e para que os mesmos sejam sincronizados nas diversas aplicações, foi utilizado o *PostgreSQL*. Os dados do animal, agendamentos, exames e tratamentos, são inseridos no Android Studio, ferramenta utilizada para o desenvolvimento do aplicativo proposto.

1.4 ESTRUTURA DO TRABALHO

A estrutura do trabalho contém uma relação dos capítulos. No primeiro capítulo foi apresentada a ideia geral do trabalho, com seus conceitos, objetivo geral e específicos e a justificativa. O capítulo dois é composto pelo referencial teórico com um relato da tecnologia utilizada para o desenvolvimento do trabalho proposto, o sistema Android. No capítulo três apresenta os materiais e o método utilizado para a realização deste trabalho. No capítulo quatro é mencionado os resultados obtidos e na sequência é realizada a conclusão do trabalho mencionado no capítulo cinco.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o contexto da convivência dos animais com as pessoas, assim como os fundamentos das tecnologias utilizadas, as quais oferecem um embasamento para a proposta deste trabalho, que tem como referência, o gerenciamento de dados de animais de estimação por meio de aplicativo móvel.

2.1 SURGIMENTO E O CONVÍVIO COM ANIMAIS DE ESTIMAÇÃO

Adestrar um animal é saber adaptar-se, ensinar, treinar para que o convívio entre seres humanos e animal seja o mais amistoso possível. E esta convivência teve início a mais de 30 mil anos, de acordo com Caetano (2010, apud Berzins, 2000, p.14), “estudos apontam para a relação homem-animal na pré-história, onde foram encontrados sítios arqueológicos em que o animal doméstico era enterrado em posição de destaque ao lado do seu provável dono”.

Com o aumento deste convívio, o homem percebeu que os animais poderiam se tornar fonte de proteção de suas habitações, sendo posteriormente utilizadas como itens de vestuário e locomoção.

Caetano (2010 apud Starling, Thomas e Guidi 2005, p. 14), relatam:

> Há milhões de anos o Homem primitivo já dividia seu território com os cães selvagens. Naquela época os cães permaneciam à frente da caverna, pela oferta de carne fresca, caçada pelos homens. Essa relação possibilitava ao ser humano uma segurança territorial contra qualquer invasor.

Essa segurança que o animal oferecia para o homem era uma forma de recompensa, pois o ser humano lhe dava comida em troca desta proteção. Aos poucos a dependência de uma pelo outro foi criando laços mais fortes e a convivência foi tornando-se afetuosa.

De acordo ainda com Martins, a distância que existia entre seres humanos e animal, aos poucos foi se estreitando e o convívio tornou-se pacífico a ponto dos animais seguirem os seres humanos na caça.

Durante a Idade de Ferro e Bronze, outro animal teve seu destaque, o cavalo. Tinha grande serventia nas atividades pastoris, acompanhou a evolução da sociedade e

posteriormente foi domesticado. Ele era utilizado como meio de transporte rápido e confiável até o surgimento da máquina a vapor (Caetano, 2010 apud LEVINE, 1999).

Mas os animais de pequeno porte é que conquistaram os homens que aos poucos foi trazendo cachorros, gatos para mais próximo, e hoje estão nos lares, fazendo parte da família.

Em uma pesquisa realizada pela Universidade de Cambridge, em 2002 e divulgada em abril de 2015 por Jocelaine Santos no site Sempre Família, mostrou que pessoas que adquirem animais de estimação têm melhoras, na autoestima, bem-estar, contribui para uma vida mais saudável, estreitam relações familiares, as crianças são mais felizes e diminui a sensações de solidão. Esses são os resultados mais relevantes mencionados na pesquisa. A partir disso, percebemos quão importante os animais de estimação são importantes na vida de seus donos, pois estes são alguns benefícios que os mesmos oferecem.

Com os benefícios apresentados, cada vez mais pessoas estão adquirindo animais de estimação e tendo cuidados significativos com os mesmos, pois os mesmos benefícios que eles nos oferecem, faz com que seus donos também os devolvem em forma de carinho, atenção e cuidados.

2.2 PLATAFORMA DE DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

Na atual sociedade em que estamos inseridos, a utilização dos dispositivos móveis, vem conquistando cada vez mais adeptos. Neste contexto, provocada pela relação homem e tecnologia, sentiu-se a necessidade de preservar e transmitir informações, conhecimento. De acordo com Fernandes (2010), “as comunicações móveis fazem parte do dia-a-dia de quase toda a gente, fator que tornou os dispositivos móveis cada vez mais complexos e poderosos”. A partir dessa intensificação, nota-se o crescimento do mercado desta ferramenta juntamente com o seu uso.

E neste contexto, a tecnologia Android é copiosamente utilizada. A ferramenta Android é desenvolvida para dispositivos móveis, celulares (smartphones, IOS), tablets e demais sistemas. Em plataforma Linux, oferecendo flexibilidade e fácil utilização, Brito (2015). O Android pertence à empresa Google, mas o sistema é desenvolvido por um consórcio de empresas chamado *Open Handset Alliance* criada em 2007 onde outras empresas do setor uniram-se a Google para tornar a plataforma mais fortalecida no mercado, Brito (2015).

A plataforma Android possui a seguinte arquitetura que é dividida em quatro camadas Eggea (2013):

A primeira camada *Applications* é composta de lista de aplicações padrões como: navegador, calendário, mapas, programa de SMS, gerenciador de contatos entre outros. Esta camada utiliza linguagem Java em seu desenvolvimento.

Na segunda camada *Application Framework* são bibliotecas responsáveis por funções específicas, como a reutilização de código. Na sequência são apresentados os principais componentes que fazem parte dessa camada:

- Componentes gráficos, listas, grids, botões, caixas de texto, podem ser reutilizados para construir uma nova aplicação;
- Provedores de conteúdo que permitem aplicações acessarem ou compartilhares informação com outras;
- Administrar memórias que o aplicativo não esteja utilizando, liberando assim para as demais;
- Manter as notificações para que, todas as aplicações móveis mostrem ao usuário a mesma mensagem, reutilização de código.

A terceira camada é subdivida em *Libraries* (bibliotecas) e *Android Runtime* (ambiente de execução). Esse é composto pelas bibliotecas padrão do Android e pela máquina virtual Dalvik. A *Libraries* é formada por um conjunto de bibliotecas C/C++, utilizadas em diversos componentes e possuindo funções particulares. Entre as bibliotecas, as principais são:

- Bibliotecas de Mídias: suportam execução e gravação da maioria dos formatos de imagens, áudio e vídeo;
- Gerenciador de Superfície: mantem acesso ao display do dispositivo e camadas de gráficos 2D e 3D de múltiplas aplicações;
- *LibWebCore*: navegador web utilizado na plataforma Android;
- 3D *libraries*: biblioteca baseada em *OpenGL* ES 1.0 APIs, a qual utiliza aceleração 3D no hardware para renderização de modelos gráficos;
- *FreeType*: renderização em formatos bitmaps e vetoriais;
- *SQLite*: gerencia acesso ao banco de dados relacional disponível para todas as aplicações.

O ambiente de execução é composto pela máquina virtual Dalvik. Todas as aplicações em Android podem ser executadas no próprio processo. Esta máquina foi criada para otimizar e suportar múltiplas máquinas virtuais.

A quarta e última camada é composta pelo Linux Kernel que, nas versões 2.6 e 3.0, é responsável pelas tarefas de gerenciamento de memória, gerenciamento de processos, acesso a redes, entre outros.

A Figura 1 apresentada a seguir demonstra a arquitetura da plataforma Android.

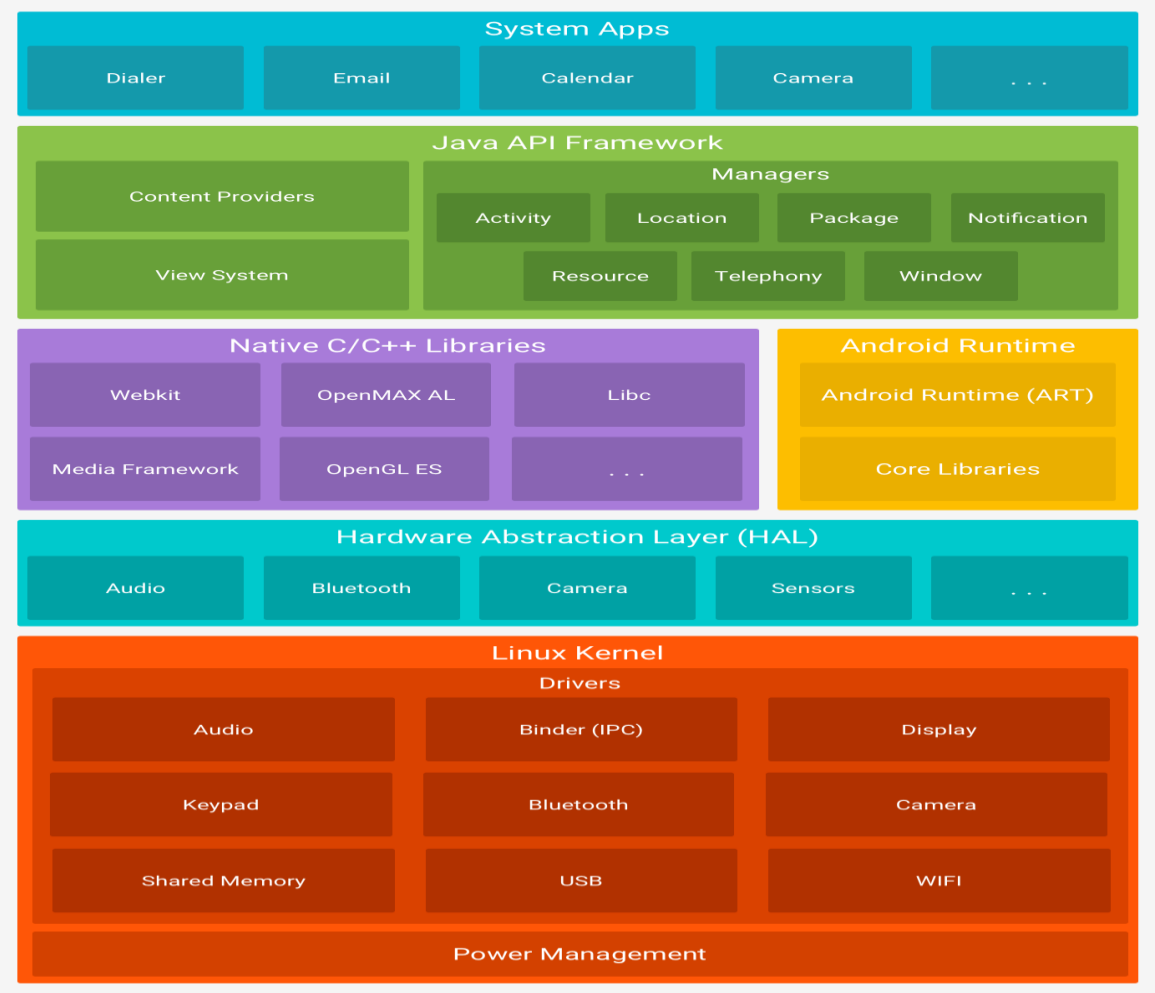

Figura 1 - Arquitetura da plataforma Android.
Fonte: Developers, 2018.

O Android Studio possui a linguagem Java como linguagem de programação para a execução de aplicativos. Paralelo a essa linguagem de programação, para o desenvolvimento da interface visual, gráfica, do aplicativo, são utilizados arquivos XML, o que simplifica o processo de desenvolvimento, Brito (2015).

2.2.1 Plataforma de Desenvolvimento

O pacote utilizado pela Android para a execução dos aplicativos é o Android Studio que foi lançada pela Google. Esta estrutura utilizada traz consigo recursos pré-definidos, incluindo o Android SDK e o kit de Desenvolvimento Java Padrão, JDK, de acordo com Brito (2017). Este software facilita na construção de aplicativos móveis através de suas funcionalidades fazendo com que aplicação posteriores possam reutilizar o código já escrito para outras aplicações.

Para o desenvolvimento de um aplicativo, é utilizado o Android SDK, um software disponível para Windows, Linux e MacOS, apresenta um emulador que recria a tela de um celular, ferramentas, uma API completa com linguagem Java e com as demais classes para o desenvolvimento das aplicações.

Com o *plug-in*, é possível simular a tela de um celular e ver como os componentes do aplicativo estão distribuídos na tela, bem como, depurar o código-fonte, caso necessário. Outra forma de analisar a organização dos componentes é através do celular, pois ao conectá-lo na porta USB do computador, tem-se uma visão melhor e real de como será o funcionamento do aplicativo, o que é mais produtivo para os desenvolvedores, de acordo com Lecheta, (2013).

2.2.2 Componentes do aplicativo

Para cada aplicativo criado, haverá quatro componentes. Cada um deles tem uma finalidade única e possui seu ciclo de vida. Os quatro componentes são:

- Atividades: é a tela de interação com o usuário. A subclasse em que é implementada é na *Activity.*
- Serviços: não possui interface com o usuário. Ele é um componente executado em segundo plano para realizar operações longas ou trabalhos remotos. Os serviços são implementados na subclasse *Services.*

- Provedores de conteúdo: gerencia o armazenamento e compartilhamento de dados do sistema em um bando de dados. A subclasse em que é implementado é *ContentProvider*.
- Receptores de transmissão: são componentes que respondem a ações do sistema. Exemplo: comunicar que alguns dados foram alterados e estão disponíveis para uso. Esta subclasse é implementada pelo *BroadcastReceiver*.

A Figura 2 mostra a representação dos componentes para a criação de um aplicativo.

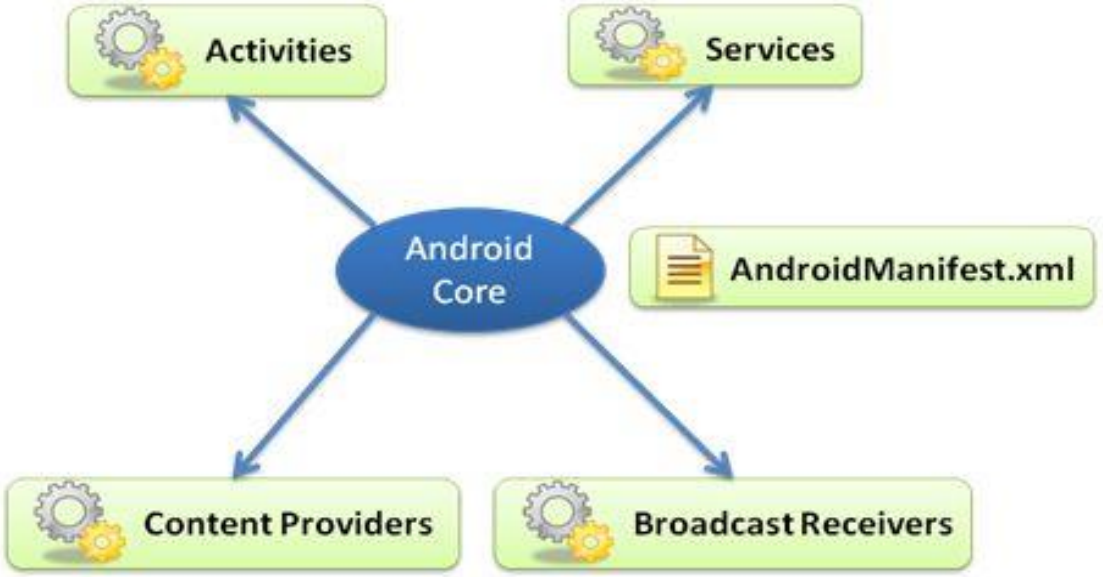

Figura 2 - Componentes para a criação de aplicativos.
Fonte: Tosin, 2011.

Cada um dos componentes ilustrados na Figura 2, proporcionam um melhor desempenho, permitindo que o usuário crie um aplicativo de forma a suprir inúmeras necessidades.

3 MATERIAIS E MÉTODO

A ênfase deste capítulo está em reportar o que e como será feito para alcançar o objetivo do trabalho. Este capítulo está subdividido, inicialmente, em duas seções, sendo uma para os materiais e outra para o método.

3.1 MATERIAIS

Nesta seção são elencadas as ferramentas utilizadas para o desenvolvimento do trabalho.

Tabela 1 - Materiais utilizados para a realização do trabalho proposto.
Fonte: Autoria própria.

Ferramenta / Tecnologia	**Versão**	**Finalidade**
Moqups	2.4.85	Criação da interface para o usuário.
Visual Paradigm	14.0	Modelagem baseada em UML.
Android Studio	3.0	Modelagem do aplicativo.
Eclipse Java EE IDE for Web Developers.	Oxygen	Construção do *web services*.
PostgreSQL	9.6.2	Banco de Dados.
PgAdmin III	1.22.1	Sistema de Gerenciamento de Banco de Dados.
Spring	1.5.8 RELEASE	Construção do *web services*.
Java		Construção do *web services*.
Material Design		Construção de design visual interativo.

Qr Code Generator	2.4.0	Construção de Qr Code.
Notificação		Construção de alertas para o usuário.
Firebase		Autenticação do responsável do animal através da conta do *Gmail.*

3.1.1 Moqups

O Moqups, é uma ferramenta utilizada para a criação da interface, de como será distribuído os componentes na tela, apresentando um design próximo ao resultado final, Silva (2015).

Foi utilizada a versão 2.4.85 para a criação inicial das telas do aplicativo para a visualização mais concreta do produto final. Sendo uma ferramenta com linguagem HTML5 possui diversas opções pré-formatadas para a utilização, Larrossa.

3.1.2 Visual Paradigm

A ferramenta Visual Paradigm fornece aos programadores uma melhor visualização e manipulação dos diagramas pertinentes a projetos, facilitando assim um melhor entendimento do que está sendo proposto.

Uma das tecnologias utilizadas para a realização do trabalho proposto, foi a Visual Paradigm, versão 14.0, que é voltada para *Unified Modeling Language* (UML) que tem como objetivo a estruturação do trabalho por meio de diagramas, por exemplo: entidade-relacionamento, caso de uso, classes e de sequência, Araújo (2007).

3.1.3 Android Studio

O Android Studio é um ambiente de desenvolvimento integrado, *IDE*. Ferramenta lançada pela Google e utilizada para a criação de aplicativos móveis.

A versão utilizada foi a 2.3 para a construção, implementação do aplicativo final. A linguagem Java foi utilizada para a programação. Paralelo a essa linguagem de programação, para o desenvolvimento da interface visual, gráfica, do aplicativo, foi utilizado a linguagem XML, para simplificar o desenvolvimento, Brito (2015).

3.1.4 Eclipse

O Eclipse é um IDE de código livre utilizado para desenvolvimento de software na linguagem Java, mas é também um dispositivo com suporte a muitas linguagens, tais como, C/C++. A aplicação disponibiliza funcionamento em vários ambientes

Com conhecimento em Java, esta ferramenta utiliza esta linguagem em seu desenvolvimento. E conta com um conjunto de serviços que facilitam na implementação de um *web service*. O conjunto de padrões de *plug-in* inclui também Ferramentas de Desenvolvimento Java (JDT) na sua IDE, de acordo com Chris Aniszczyk e David Gallardo (2012).

Para Fernanda B. Faria, et al.(2010), alguns dos componentes do Eclipse são: *Workspace,* que é gerenciador de arquivos e projetos do usuário; *SWT (The Standard Widget Toolkit)* e *JFace*: são os gerenciadores de interface com os usuários; *WorkBench*: suporte para a interface gráfica; *Team*: controla as versões e histórico dos recursos; *Debug*: suporte e depuração de programas; *Help:* recurso de ajuda para os usuários e *Update*: que controla as atualizações disponíveis.

A plataforma do Eclipse disponibiliza pacotes para o desenvolvimento, são eles: Eclipse JDT, que é a base dos *plug-ins* na linguagem java, o Eclipse SDK, é o pacote de distribuição da IDE java, o Eclipse WTP, é utilizado para o desenvolvimento da linguagem web, e o compilador do JDT, que é seu próprio compilador java.

3.1.5 *PostgreSQL*

PostgreSQL é um banco de dados de código aberto e trabalha com o banco objeto-relacional. Possui grande confiabilidade e persistência em seus dados. Este software está disponível para os principais sistemas operacionais como: Linux, UNIX, macOS, Windows entre outros, Dionísio (2015).

A programação pode ser realizada em C, C++, Java entre outras. Além de sua linguagem fácil de programar, o *PostgreSQL* possui uma biblioteca com centenas de funções já pré-definidas. De acordo com o site do *PostgreSQL*, "Os disparadores e os procedimentos armazenados podem ser escritos em C e carregados no banco de dados como uma biblioteca, permitindo uma grande flexibilidade na extensão de suas capacidades". Esta estrutura permite que cada programador crie e personalize seus dados.

Como possui o código-fonte disponível, esta licença oferece a liberdade de usar, modificar e distribuir de acordo com sua preferência, Biazus (2003). As modificações realizadas são somente alteradas no seu sistema ou para quem forem disponibilizadas. Esta plataforma de desenvolvimento permite que se produza software internos, web ou comerciais que exigem um Sistema Gerenciador de Banco de Dados Relacional, SGBDR, com capacidade maior que os demais.

3.1.6 Spring

É um framework criado por Rod Johnson por volta de 2002. Possui código aberto, que tem como intuito simplificar a programação na linguagem Java, o que possibilitou a construção de aplicações que anteriormente era realizada somente em *Enterprise JavaBeans*, Gentil (2012).

Esta ferramenta, tem como um de suas principais características, proporcionar para os desenvolvedores um maior nível de produtividade. Possuindo desta forma somente funcionalidades necessárias, diminuindo assim a complexidade do desenvolvimento do sistema, Dias (2016).

3.1.7 Java

O Java foi criado em 1992 pela antiga *Sun Microsystem*, hoje pertencente a *Oracle*. A linguagem deriva do C e C++. Porém para programar nesta linguagem é mais fácil, pois a programação é orientada a objeto, onde todos os elementos inseridos são objetos. Desta forma faz com que o nível de linguagem seja de alto nível, Teixeira (2012).

Com este intuito, os desenvolvedores tinham como objetivo era criar uma linguagem simples, orientada ao objeto e que qualquer pessoa pudesse utiliza-la. De acordo com Pereira 2009, “a linguagem deveria possuir arquitetura neutra e portável, de forma que pudesse ser utilizada em diversos sistemas operacionais [...]”. Isto fez com que esta linguagem se tornasse a mais utilizada em 2015 e 2016 de acordo com o site CanalTech (2016).

3.1.8 *Web Service*

Esta tecnologia será utilizada para a integração dos sistemas e na comunicação entre diferentes aplicações. Os componentes existentes em um *Web Service*, permitem que as aplicações, que possuem uma linguagem particular, recebam e enviem dados que são traduzidas para a linguagem universal, o XML e o Json, de acordo com Erl (2009).

O *Web Service* garante ao usuário receber e verificar informações de forma transparente, segura e sem complicações. Esta ferramenta possui vantagens consideráveis para sua utilização: simplicidade na implementação, segurança de informações e redução de custo, pois para várias aplicações, pode-se reutilizar códigos, Pinto (2016).

3.1.9 Material Design

Lançado pela Google em 2014, o Material Design, é inspirado em materiais ou objetos do mundo real, como sombra, luzes, cores, formas e animações. Essas interações em terceira dimensão, faz com que as informações sejam apresentadas frente umas às outras, com mais animação, por meio de seus comandos de toque na tela, para o usuário, de acordo com Cordeiro (2017).

Com a utilização do Material Design, o Android fornece novos elementos que podem ser utilizados na criação do uma aplicação, são eles, temas, widgets, como *RecyclerView,*

CardView, Pallete Toolbar, sombras e animações personalizadas. A primeira versão do Android com suporte para o Material Design é o Lollipop, versão 5.0.

De acordo com Neves (2015), a principal finalidade é proporcionar ao usuário a combinação de formas geométricas que são vistas diariamente, dando assim uma apresentação funcional a aplicação.

3.1.10 *QrCode*

Esta tecnologia consiste em um código de barras em 2 dimensões, vertical e horizontal, diferente do código de barras tradicional que trabalha somente com o horizontal, Xavier (2014). Ele é constituído de códigos e caracteres decodificados em uma imagem, permitindo grande armazenamento de dados.

Para apresentar as informações codificadas, basta tirar uma foto com a câmera do celular e possuir um programa para a leitura do código bidimensional. Logo após, será apresentada as informações contidas no *QrCode.*

3.1.11 Notificações

As notificações são um alerta, em forma de mensagem ou imagem, para informar o usuário sobre algo que ele precisa ser lembrado em um determinado tempo. O uso dela, torna a aplicação mais inteligente permitindo uma maior interação com o usuário, Ogliari e Brito (2013).

Uma das vantagens é que, as notificações são geradas independente da aplicação, se está aberta ou não, permitindo assim, que o usuário tenha melhor desempenho com as atividades que estão sendo realizadas no momento em que a notificação é gerada. Ela permite que o alerta seja visualizado primeiramente como um ícone na área de notificação. Caso o usuário queira visualizar ela por completo, basta ele clicar na notificação a qualquer momento, Developers (2018).

3.2 MÉTODO

As estratégias utilizadas para definir o sistema são baseadas nas fases propostas por Pressman (1995) para o modelo de ciclo de vida clássico que são: engenharia de sistemas, análise, projeto, codificação, testes e manutenção. A seguir fases definidas para este trabalho:

O primeiro passo para a construção deste projeto, foi o levantamento de requisitos, no qual foi definido o problema e realizada uma análise de sistema semelhante existente no mercado, MyPets. Foi realizado, também, o levantamento de requisitos necessários, para o sistema, tendo como ponto fundamental o que o sistema deve suprir. Posteriormente foi elencado os requisitos funcionais e não-funcionais necessários para que o aplicativo possua segurança e desempenho nas suas funcionalidades. Nesta fase foi construído o esboço das telas do aplicativo com a ferramenta Moqups.

Na sequência, foi realizada a análise, foi identificado conceito e metodologia da UML seus diagramas de caso de uso, classes e interação com descrição minuciosa dos requisitos elaborados na fase anterior. Também foi realizada uma análise do software Android Studio, utilizado para a criação do aplicativo proposto.

O próximo passo foi a realização do projeto. Nesta fase, os diagramas realizados na fase anterior tiveram como finalidade a composição do projeto no sistema desenvolvido. Foi definido, também, os pontos mais relevantes e as funcionalidades de destaque para a composição do sistema.

Em seguida foi realizada a etapa de codificação. Depois de todos os levantamentos e análises realizadas, iniciou-se a fase de codificação do software para o sistema operacional Android. As linguagens Java e XML disponíveis no programa facilitaram no desenvolvimento.

Na sequência, foi criado o *Web Service* que contém os métodos de salvar, editar, excluir e listar os animais, agendamentos, tratamentos e exames. Este *Web Service* tem a conexão com o banco de dados externo, no qual é salvo todas as informações cadastradas no aplicativo. O *Web Service* terá a função de mostrar para o/os usuário(s) as informações já cadastradas.

Por fim, foi realizado a etapa de testes, que a cada nova implementação e codificação no programa, eram realizados testes de validação e integridade das informações no banco de dados para identificar possíveis erros, para assim dar prosseguimento no desenvolvimento do aplicativo.

4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos com o trabalho. Uma breve descrição do aplicativo proposto, com suas funcionalidades. Está disponível neste capítulo, a modelagem do sistema e os diagramas para o desenvolvimento do mesmo, as telas de apresentação do aplicativo juntamente com parte do código-fonte com objetivo de exemplificar as tecnologias utilizadas para sua implementação.

4.1 ESCOPO DO SISTEMA

O aplicativo de gerenciamento de dados do animal de estimação deve gerenciar todos os processos para o bem-estar do animal, desde banho e tosa, até vacinação e medicamentos. O acesso do responsável do animal será realizado por meio de um *login*, que deverá ser a conta do *e-mail* pertencente ao *Gmail*. Esta autenticação é feita pelo *Firebase*, o qual valida o *e-mail* do responsável retornando o *token* de validação. Para que mais de um usuário possua as mesmas informações, basta que dois ou mais aplicativos acessem com a mesma conta de *e-mail*. Para ter acesso às informações bem como o *login* é necessário que o usuário possua acesso à internet, pois o servidor e o banco de dados da aplicação é somente externa.

O objetivo do sistema é cadastrar o animal com os dados como: nome, raça, sexo, data de nascimento. Posterior a isso, o cronograma de agendamentos de banho, tosa e demais higienes pessoais, bem como de vacinação e medicamentos. Para cada agendamento será cadastrado uma data específica. Quando o dia agendado chegar, o sistema emitirá um aviso, pela manhã, para o dono informando qual atividade deve ser realizada, se é banho ou vacinação, por exemplo. No aplicativo haverá um campo para o arquivamento de exames. Este arquivamento de dados será por foto. A outra funcionalidade é o cadastramento de tratamento do animal, no qual será informado qual medicamento, se é gotas, comprimido ou injeção, data de início e final do tratamento. Durante este intervalo, o usuário receberá um alerta lembrando deste evento.

O sistema terá a opção do *QrCode* que terá como finalidade apresentar para o médico veterinário os últimos exames feitos pelo animal. Esta funcionalidade será disponível por meio de um banco de dados remoto, para armazenamento externo do aplicativo, utilizando o

PostgreSQL. Ele terá uma vinculação através de um *web service*, por meio do Eclipse JEE Oxygen no qual será apresentada para o médico veterinário no momento da leitura do *QrCode*.

4.2 MODELAGEM DO SISTEMA

A definição de requisitos tem como principal objetivo elencar as funcionalidades que o sistema irá produzir. Estas funções ocorrem com a entrada e saída de dados.

A seguir são elencados os requisitos funcionais e não-funcionais do aplicativo, juntamente com seus atores responsáveis por cada tarefa. As referências cruzadas são verificadas de acordo com os requisitos vinculados.

Na Tabela 2 apresenta os requisitos funcionais do sistema.

Tabela 2 - Requisitos Funcionais.
Fonte: Autoria própria.

Nome	Atores	Descrição	Referências cruzadas
F01 – Manter responsável	Responsável pelo animal	O responsável do animal deve acessar e informar seu e-mail no sistema.	
F02 – Manter animal	Responsável pelo animal	O responsável do animal deve cadastrar o animal de estimação informando os dados solicitados.	F01
F03 – Manter agendamentos	Responsável pelo animal	O responsável do animal deve cadastrar o agendamento para cada animal que ele tenha cadastrado no aplicativo.	F01, F02
F04 – Manter exames médicos	Responsável pelo animal	O responsável do animal deve registrar uma foto dos exames médicos realizados para o animal.	F01, F02
F05 – Manter tratamentos médicos	Responsável pelo animal	Cadastrar tratamentos médicos informando os dados solicitados.	F01, F02
F06 – Consultar dados do animal	Responsável pelo animal	Poderá consultar as informações do animal podendo ser alteradas.	F01, F02

F07 – Consultar agendamentos	Responsável pelo animal	O responsável do animal pode consultar os agendamentos do seu animal.	F01, F02, F03
F08 – Gerar QrCode dos exames	Médico veterinário	O médico veterinário poderá visualizar os exames do animal por meio do QrCode.	F01, F02, F03, F04, F05
F09 – Emitir alertas	Sistema	O sistema deve emitir notificações para informar o usuário, que o animal possui tarefas a serem cumpridas	F01

Os requisitos não funcionais mostram o comportamento do aplicativo no momento em que o usuário está usando o mesmo. Para Silva Filho, 2008, são aqueles atributos que estão ligados aos atributos de qualidade. São importantes, pois são utilizados como critérios na forma de implementação, nas escolhas de alternativas de projeto e no estilo da apresentação do projeto proposto.

A Tabela 3 mostra os requisitos não funcionais.

Tabela 3 - Requisitos Não-Funcionais.
Fonte: Autoria própria.

Nome	Descrição
NF01	Obter resposta o mais rápido possível
NF02	Disponibilidade online
NF03	A utilização de login será somente no primeiro acesso.

4.3 MODELAGEM DE CASO DE USO

De acordo com Bezerra (2007), este modelo representa as funcionalidades do sistema, bem como a interação com elementos externos ao sistema. Para Jacobson (2006), é um documento que descreve a sequência de eventos que um ator utilizará em um sistema para realizar de forma completa um processo.

Segundo Bezerra (2007), este modelo é importante, pois através dele é direcionado tarefas para o desenvolvimento do sistema ou software. Este modelo relata uma sequência completa de interação entre o sistema e os agentes externos, mostrando somente o que será apresentado e não qual método será utilizado para chegar ao mesmo.

O diagrama de caso de uso apresentado na Figura 3 contém as funcionalidades do sistema. O responsável pelo animal tem a função de cadastrar todas as informações pertinentes ao animal de estimação.

As listas de raça e tipo do animal e a lista de agendamento, já possuem valores pré-definidos, no qual o responsável pelo animal poderá somente selecionar as opções. Nos demais campos, o mesmo deve informar os dados.

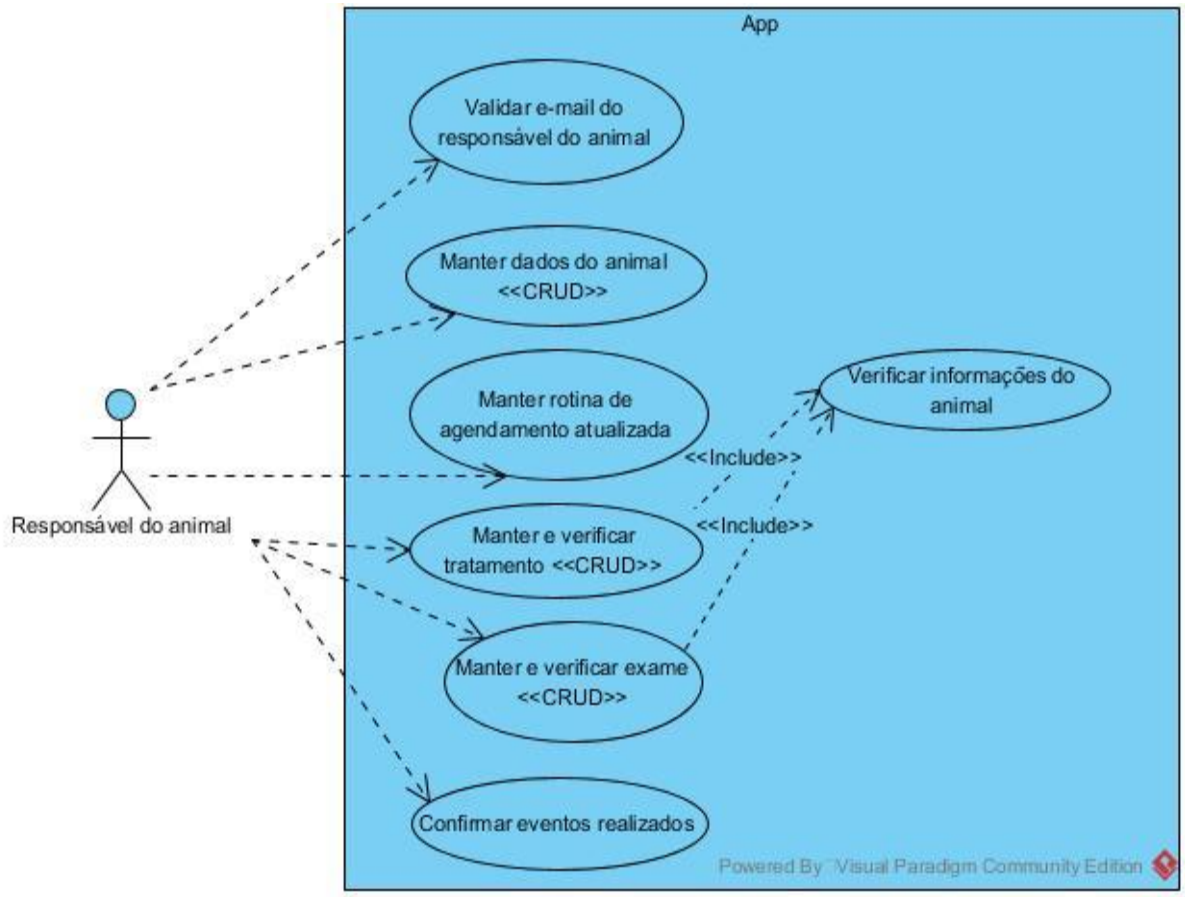

Figura 3 - Diagrama de Caso de Uso do aplicativo desenvolvido.
Fonte: Autoria própria.

Na sequência são detalhados os casos de uso juntamente com seu respectivo fluxo principal, variantes e exceções. No Quadro 1 será detalhado o caso de uso para manter dados do responsável pelo animal, o qual é realizado através da autenticação pelo Firebase o qual recebe a autorização de acesso.

Caso de Uso: Manter e-mail do responsável do animal. O sistema deve permitir o *login* do usuário através de sua conta no Gmail.
Atores: Responsável do animal
Pré-condições: O usuário deve possuir conta no Gmail

Pós-condições: Após *login* efetuado, poderá inserir dados do animal de estimação.
Fluxo Principal: 1.[IN] O usuário informa dados . 2.[OUT] O sistema mostra tela de cadastro de animal.
Tratamento de Exceções: 1a. Usuário não possui conta no Gmail. 1a.1 Inserir dados para efetivar cadastro. 1a.2 Retorna ao passo 2

Quadro 1 – Expansão do caso de uso: manter dados do responsável pelo animal.
Fonte: Autoria própria.

O Quadro 2 apresenta o detalhamento do caso de uso de manter os dados do animal, nome, data de nascimento, tipo do animal, raça sexo e a foto. Caso o responsável não preencha todos os campos, surge na tela uma mensagem informando o mesmo que necessita preencher os dados faltantes. Logo após o preenchimento destes campos restantes, o sistema salva as informações e mostra a tela de listagem do animal.

Caso de Uso: Manter dados do animal. O sistema deve permitir que o usuário cadastre o ou os animais de estimação.
Atores: Responsável do animal.
Pré-condições: O responsável do animal deve possuir dados necessários para o cadastro do animal.
Pós-condições: Após cadastro, poderá inserir dados do animal de estimação.
Fluxo Principal: 1.[IN] O responsável do animal informa os dados do animal. 2.[OUT] O sistema apresenta tela para inserir os dados. 3. [IN] O responsável do animal insere os dados. 4. [OUT] O sistema salva dados inseridos.
Tratamento de Exceções: 1a. Responsável do animal não cadastrou dados do animal. 1a.1 Responsável do animal deverá informar os dados necessários. 1a.2 Retorna ao passo 4

Quadro 2 – Expansão do caso de uso: manter dados do animal.
Fonte: Autoria própria.

No Quadro 3 está descrito o caso de uso manter a rotina do animal atualizada. O responsável pelo animal pode cadastrar todos os agendamentos necessários do animal, consulta, banho, tosa, vacina e informar a data em que este compromisso é estipulado. Este dia agendado, emitirá para o responsável um alerta avisando que o animal possui uma programação para aquela data.

Caso de Uso: Manter rotina do animal atualizada. O sistema deve permitir que o responsável do animal atualize rotina, agendamentos, do animal de estimação.
Atores: Responsável do animal.
Pré-condições: O responsável do animal deve atualizar rotina do animal.

Pós-condições: Após cadastro, poderá acompanhar e alterar rotina do animal.
Fluxo Principal: 1.[IN] O responsável do animal possui os agendamentos necessários e pertinentes do animal de estimação. 2.[OUT] O sistema apresenta tela para inserir agendamentos. 3. [IN] O responsável do animal insere agendamentos. 4. [OUT] O sistema apresenta a tela dos agendamentos. 5. [OUT] O sistema salva dados inseridos.
Tratamento de Exceções: 1a. Responsável do animal não cadastrou dados do animal. 1a.1 Responsável do animal deverá informar os dados necessários. 1a.2 Retorna ao passo 2 3a. Responsável do animal não insere dados de agendamento do animal de estimação. 3a.1 Retorna ao passo 5.

Quadro 3 – Expansão do caso de uso: manter rotina do animal atualizada.
Fonte: Autoria própria.

4.4 MODELAGEM DO DIAGRAMA DE CLASSES

O diagrama de classes é encontrado com maior frequência na modelagem de sistemas orientados a objetos, pois através dele torna-se a visualização mais clara do produto final. Tem-se também, mais evidência das especificações e da documentação de modelos estruturais que compõem a construção do sistema (Booch; Rumbaugh; Jacobson, 2000).

Para Bezerra (2007, p. 112), “uma classe é composta de atributos e de operações. Os atributos correspondem à descrição dos dados armazenados pelos objetos de uma classe. (...) As operações correspondem à descrição das ações que os objetos de uma classe sabem realizar”. Portanto, esta modelagem apresenta um panorama geral, com suas classes nomes, atributos, operações, relacionamentos, do que é necessário para a criação e implementação do sistema, tornando de forma mais concisa as reais necessidades.

A Figura 4 ilustra o diagrama de classes do aplicativo proposto. Cada classe possui seus atributos e operações necessárias para melhor desempenho da aplicação. O relacionamento entre elas, ocorre entre a classe Animal com as demais, pois esta classe é a principal e a de Agendamento, Exame e Tratamento, dependem do animal para serem preenchidas. Por este motivo o relacionamento estre elas ocorrem sempre com a classe Animal, podendo cada animal possuir zero ou muitos agendamentos, exames e tratamentos.

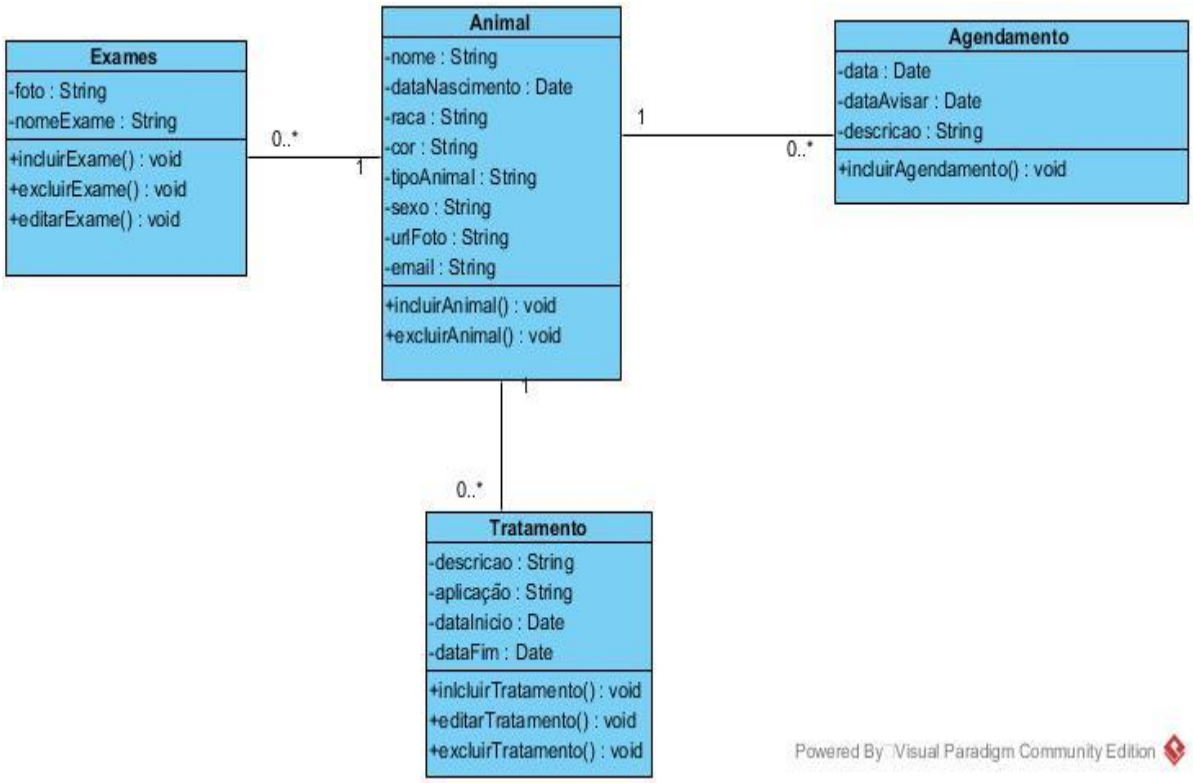

Figura 4 - Diagrama de Classes referente ao aplicativo proposto.
Fonte: Autoria própria.

4.5 MODELAGEM DO DIAGRAMA DE SEQUÊNCIA

Para Bezerra (2007), esta modelagem apresenta interações entre os objetos na ordem em que os processos irão acontecendo. Concomitante a isto, esta modelagem permite a apresentação de mensagens durante a atividade.

Na Figura 5 é apresentado o diagrama de sequência onde o responsável pelo animal realiza o *login* através de sua conta no Gmail. Após a autenticação do responsável, ele poderá realizar o cadastro do animal, o qual é necessário preencher no sistema os dados como: nome, data de nascimento, sexo, tipo do animal, raça e foto do animal. Feito este cadastro, poderá realizar um agendamento, que necessita das seguintes informações: descrição do agendamento, este já vem definido da lista de agendamento, e dia em que será o evento, pois neste dia, o sistema irá gerar para o responsável uma notificação.

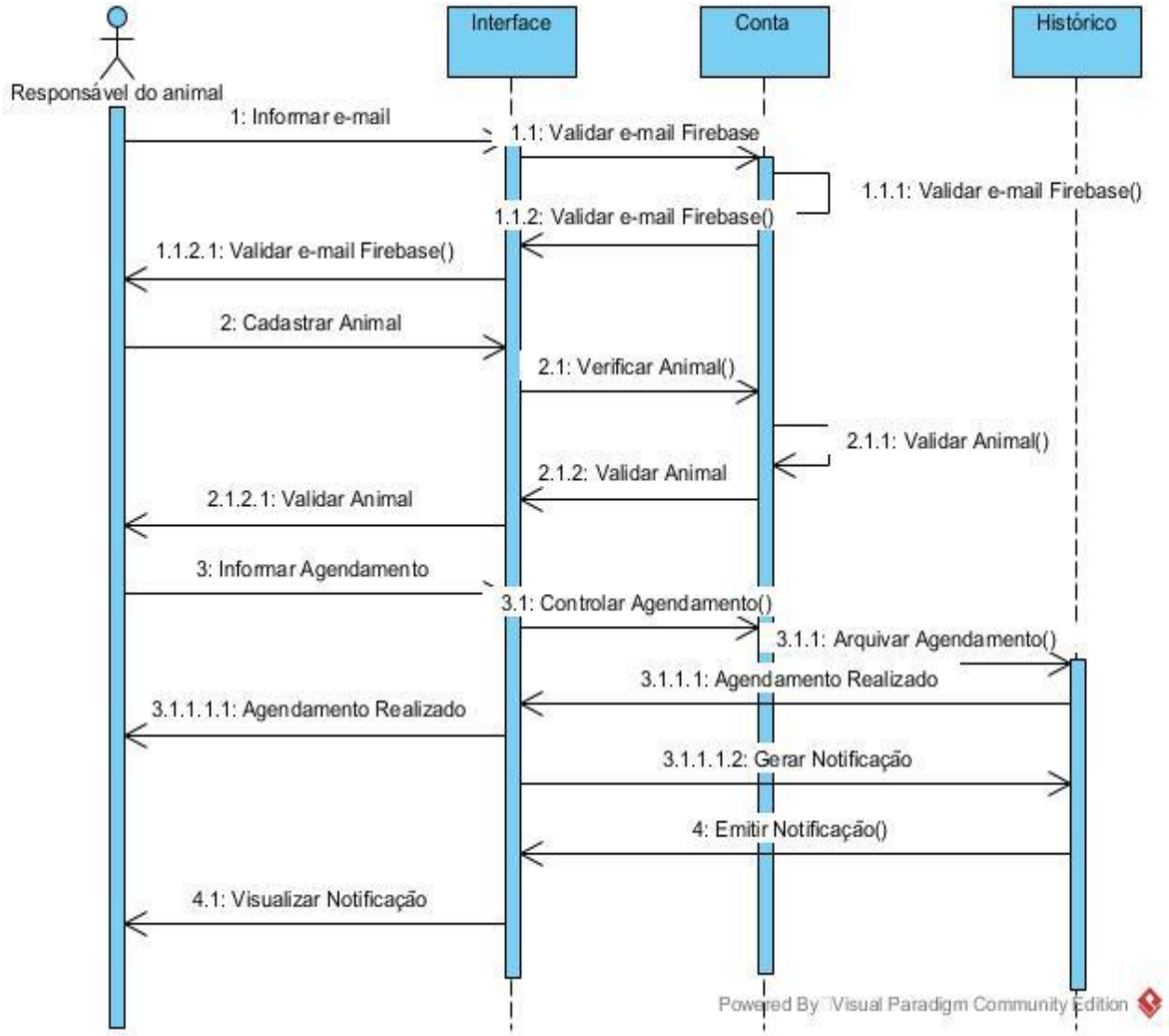

Figura 5 - Diagrama de Sequência da aplicação desenvolvida.
Fonte: Autoria própria.

O diagrama acima mostra os objetos trocando interações, informações, e o comportamento que cada uma possui. Este desempenho é ressaltado nas mensagens que cada um recebe e envia para cada objeto.

4.6 MODELAGEM DO DIAGRAMA DE ENTIDADE RELACIONAMENTO

É um modelo conceitual utilizado para descrever as entidades envolvidos em um domínio de negócios, com seus atributos e como elas se relacionam entre si.

Ele representa de forma abstrata como a estrutura se configurará para o banco de dados.

Na Figura 6 será apresentado o diagrama de Entidade-Relacionamento do aplicativo desenvolvido durante este trabalho, o qual apresenta as classes juntamente com seus atributos e suas interações. O banco de dados utilizado é o PostgreSQL.

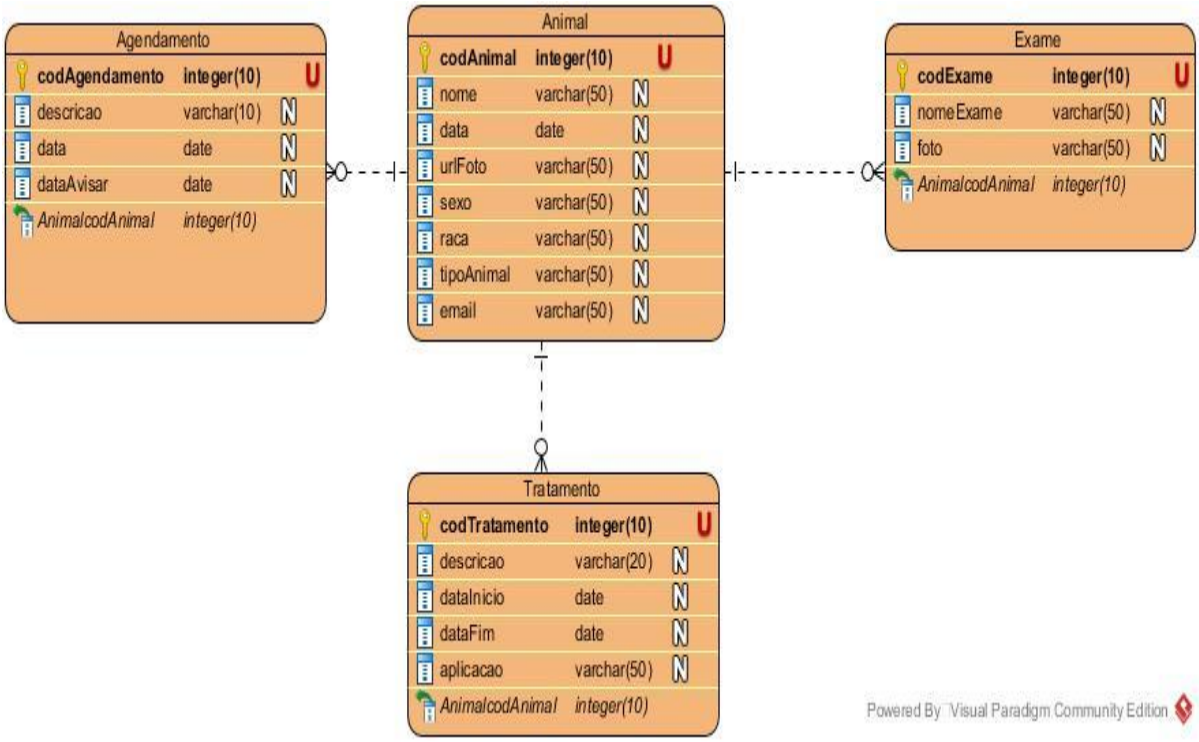

Figura 6 - Diagrama de Entidade-Relacionamento do aplicativo desenvolvido.
Fonte: Autoria própria.

O diagrama de entidade-relacionamento é utilizado para que tenha uma melhor visualização da modelagem, bem como o desenvolvimento da base de dados, o que evita alterações durante a análise, Rodrigues (2014).

4.7 APRESENTAÇÃO DO SISTEMA

Nesta seção são apresentadas as funcionalidades e os recursos que compõem o sistema por meio das telas. A apresentação do aplicativo é na forma de texto, juntamente com a telas de maior relevância. Estas telas são uma cópia das telas do aplicativo desenvolvido.

A Figura 7.a e 7.b apresentam as telas iniciais do aplicativo. Esta tela da Figura 7.a apresenta o um botão para o *login* do responsável pelo animal por meio da sua conta do

Gmail, no qual o sistema solicita ao usuário que informe seu e-mail. Caso não possua, poderá solicitar a criação de uma conta.

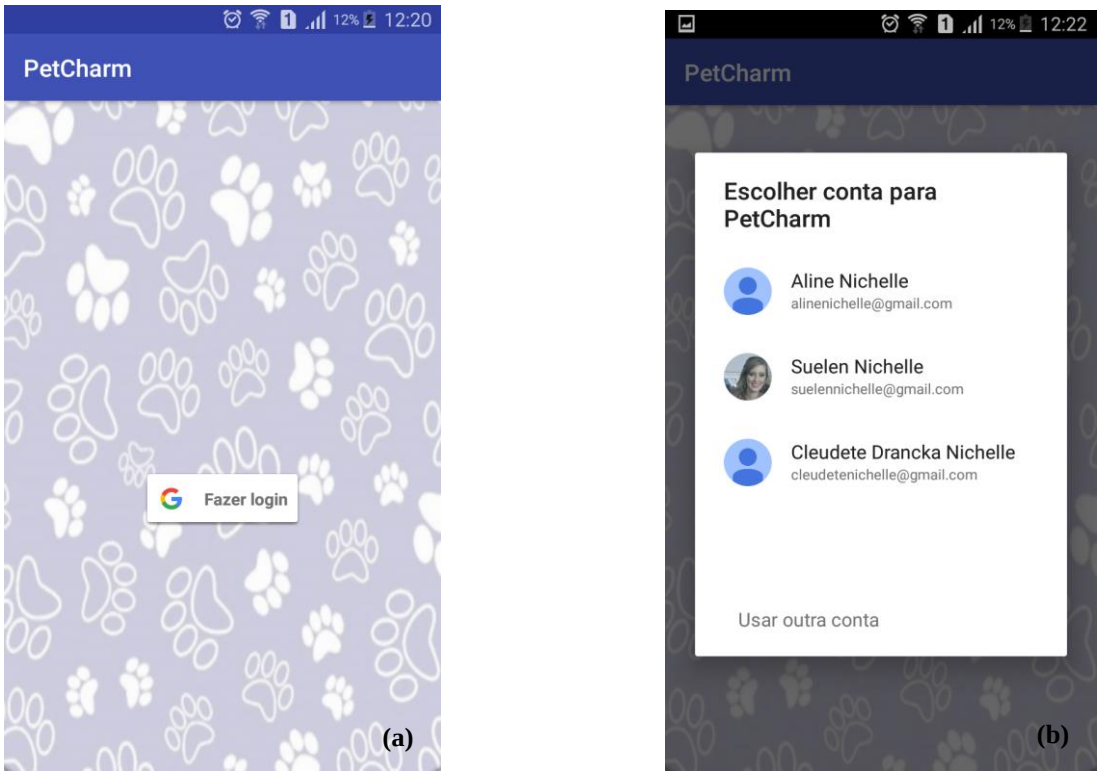

Figura 7 - a. Tela inicial b. Tela de login.

A Figura 8.a, demonstra a lista dos animais cadastrados no sistema. Esta tela é apresentada para o responsável do animal logo após a validação do e-mail. A opção acessar, tem como funcionalidade mostrar para o usuário a tela de menu, que será apresentada mais abaixo. Já a opção de excluir, é para deletar o animal da listagem apresentada.

A Figura 8.b apresenta o cadastro do animal. Esta tela é acessada por meio do botão que se encontra na parte inferior direita da tela do celular, representado pelo ícone “+”. Para o cadastro devem ser informados os dados do animal, imagem, nome, data de nascimento, sexo, espécie e sua raça.

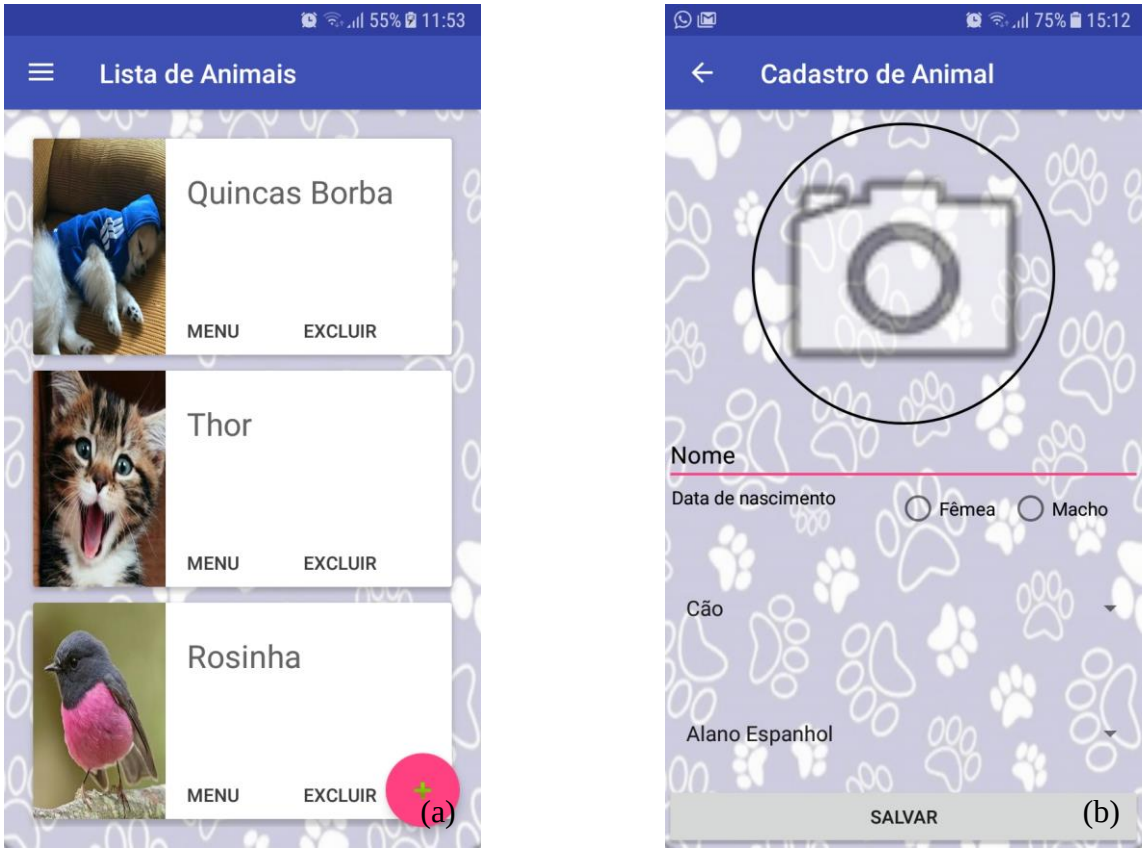

Figura 8 – a. Listagem de animais. b. Tela de cadastro do animal.

No botão de menu, que consta na lista de animais, o usuário tem a opção de cadastrar alguns dados para gerenciamento do animal, como: agendamento, exame e tratamento, conforme mostra a Figura 9.a.

Ao clicar no botão de agendamento que consta na tela de menu, irá mostrar para o usuário uma nova tela contendo uma lista de agendamento possíveis para o animal conforme é apresentado na Figura 9.b.

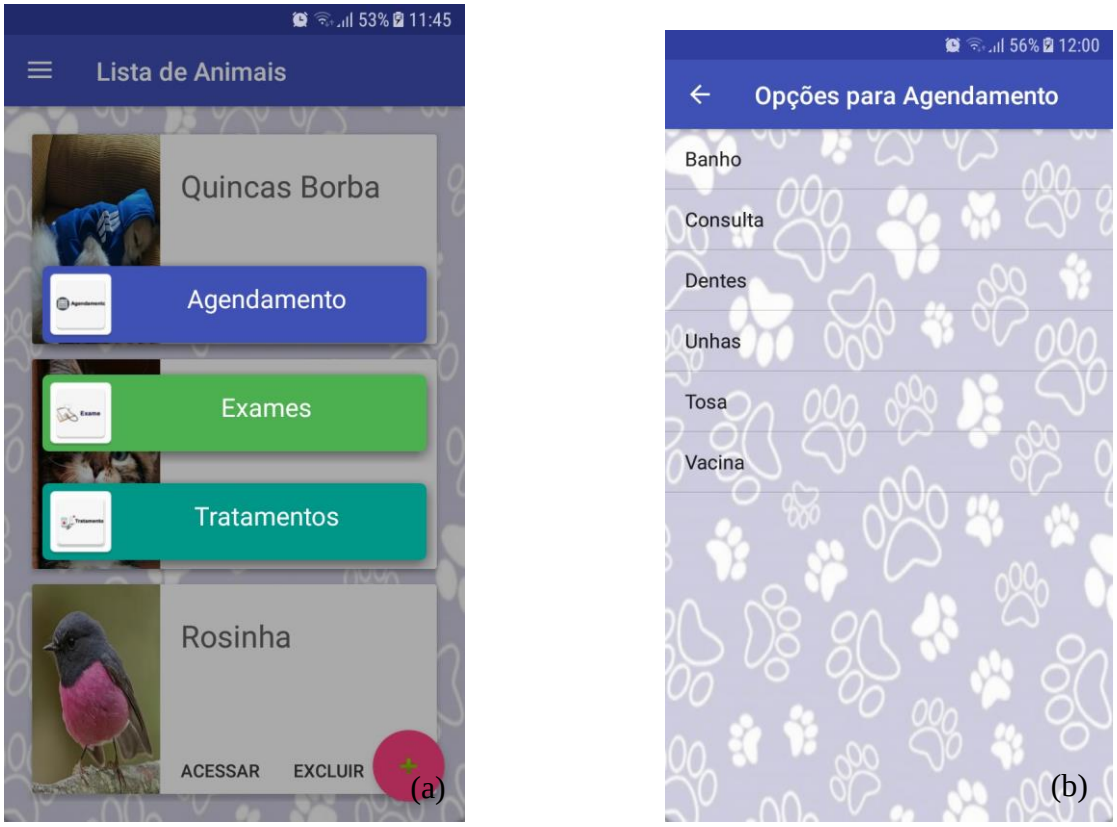

Figura 9 – a. Tela de menu. b. Tela de lista de opções para o agendamento.

Após clicar em um dos itens da lista, o mesmo será carregado na próxima tela, a de descrição do agendamento. Nela consta: a descrição do agendamento, a data atual, a data na qual o evento será realizado e o botão de salvar, conforme mostra a Figura 10.a.

Na tela lista de exame, como mostra a Figura 10.b, será gerada uma listagem, com os exames salvos pelo usuário. Nesta tela costa o nome do exame, a imagem, os botões de editar e excluir o item da lista. Quando clicado no editar, é carregado novamente as informações na tela de descrição de exame. Já o botão excluir, elimina o exame da lista, bem como do banco de dados. O botão com o ícone “+” para adicionar exame.

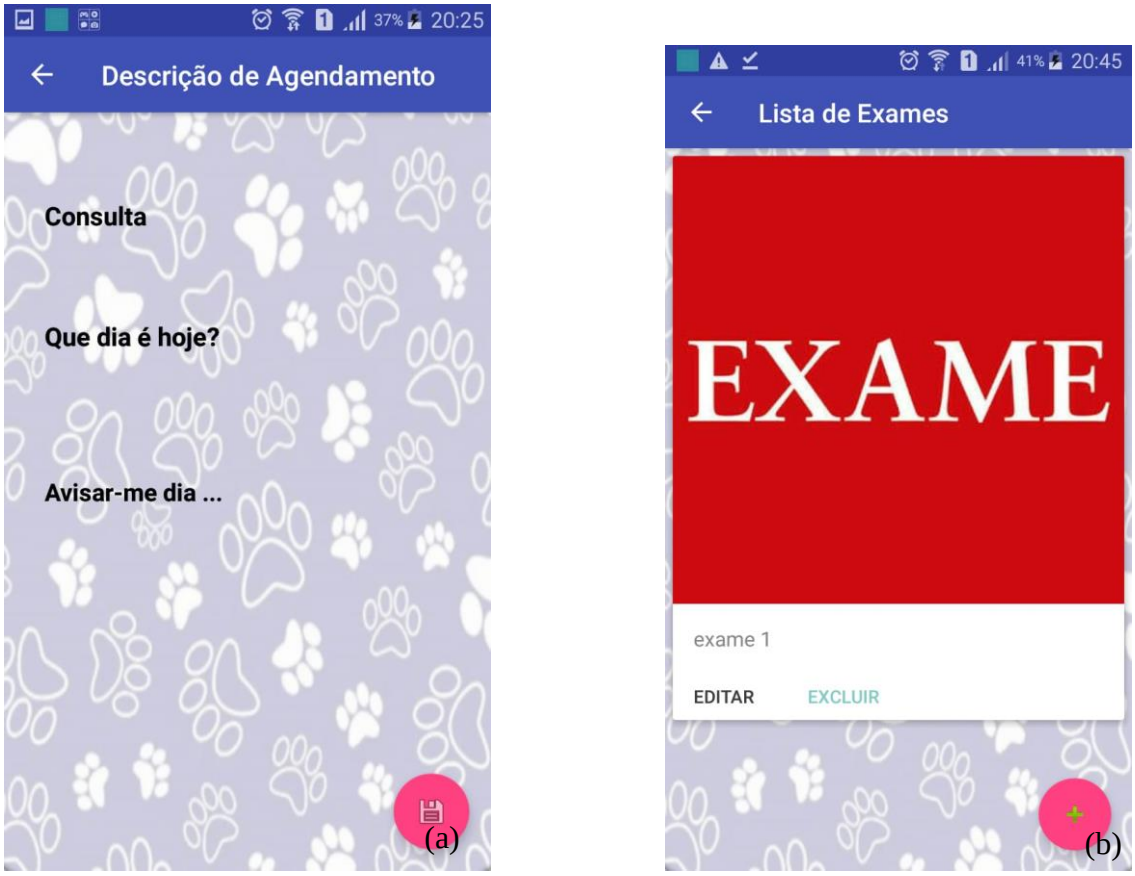

Figura 10 – a. Descrição do agendamento. b. Listagem dos exames.

Quando clicado no botão de editar da lista de exames, ele vai retornar para a tela de descrição do exame, mas com os dados carregados da tela anterior. Dando para o usuário um botão para gerar um QrCode. Este código de barras de bidimensional, possui o caminho de onde esta foto está salva, como mostra a Figura 11.a.

Na Figura 11.b, consta a tela de lista de tratamento. Nessa tela é apresentada a listagem dos tratamentos cadastrados para o animal selecionado. Para cada tratamento cadastrado, a lista apresenta o botão de editar e excluir. O editar, quando clicado, abrirá a tela de descrição de tratamento, mas com os campos já preenchidos. O botão excluir, deleta o tratamento da lista e do banco de dados. Na parte inferior da tela consta o ícone “+” para adicionar um novo tratamento.

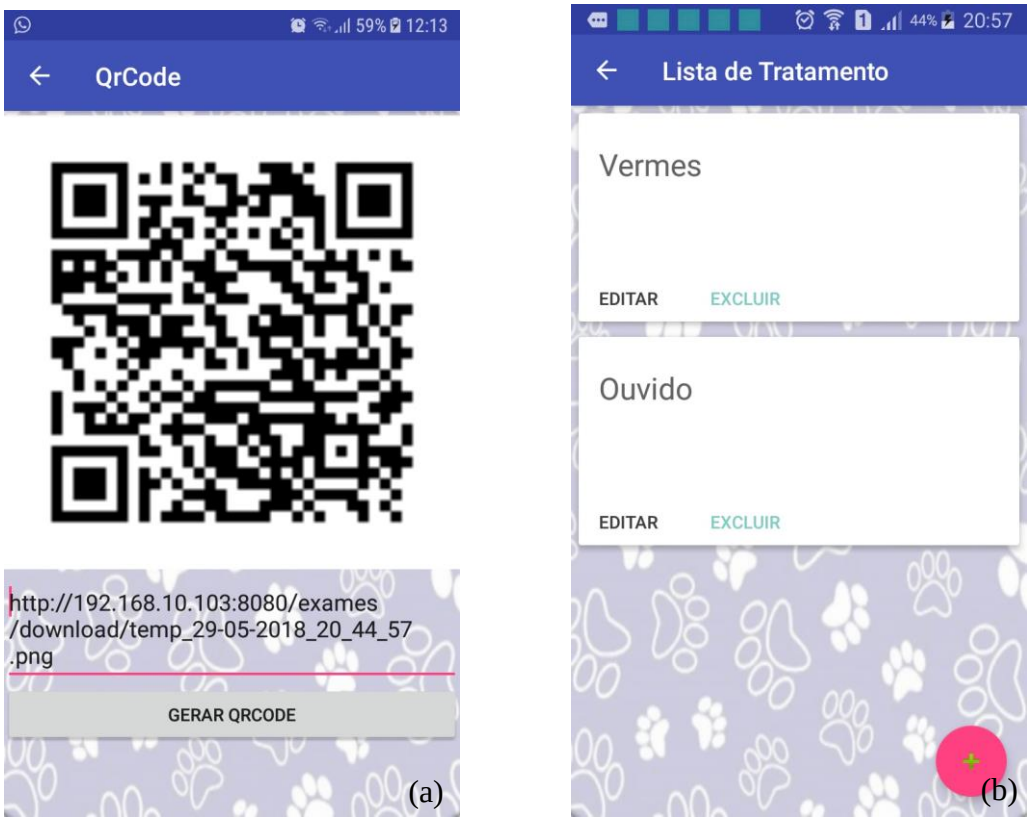

Figura 11 – a. QrCode gerado a partir da tela de exame. b. Tela de lista de tratamento.

Quando clicado no botão com o ícone “+” o usuário será redirecionado para a tela de descrição de tratamento que consta os seguintes campos: nome do medicamento, forma de aplicação, data de início, data de fim do tratamento e o botão salvar, na parte inferior da tela, como mostra a Figura 12.a.

O aplicativo desenvolvido tem a função de gerar uma notificação para o usuário, de acordo com a data informada no agendamento e no tratamento. As notificações são geradas e informadas para o usuário de acordo com a data informada pelo usuário no agendamento. Já no tratamento, as notificações são geradas durante o intervalo das datas, conforme mostra a Figura 12.b.

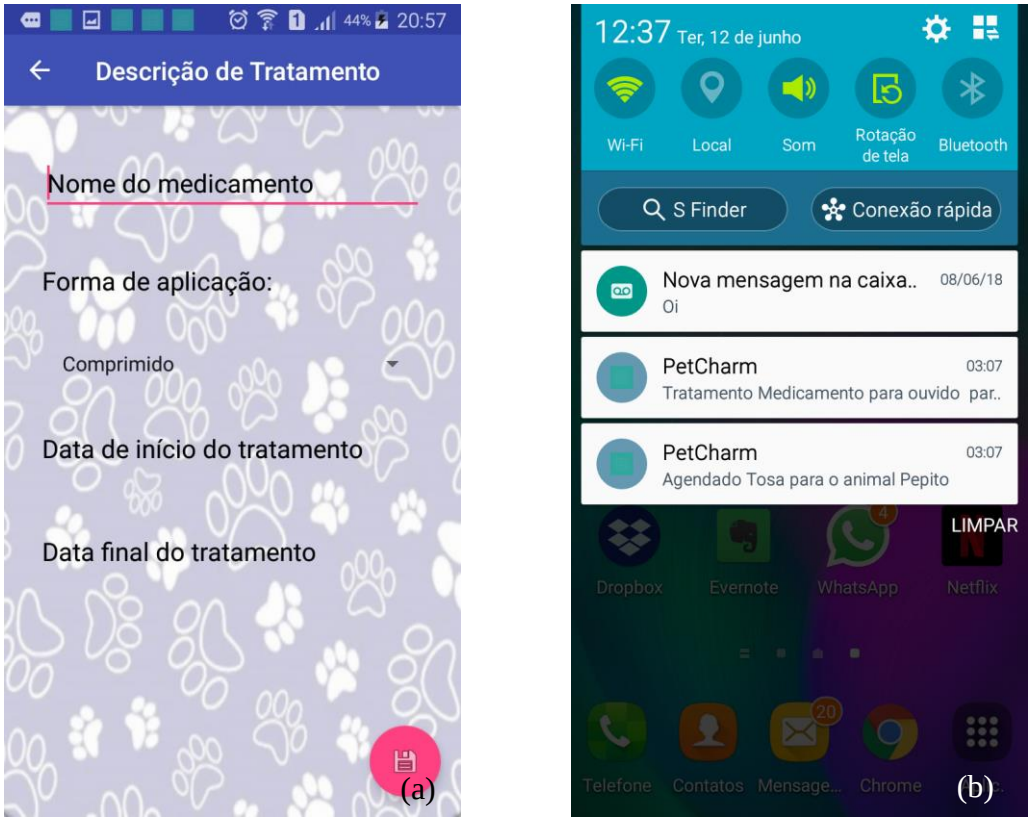

Figura 12 – a. Tela de descrição de tratamento. b. Notificações de Agendamento e Tratamento.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Será apresentado a seguir, uma listagem contendo partes da implementação do sistema, que tem como objetivo apresentar uma parte da codificação que foi desenvolvida.

A Listagem 1 mostra parte da codificação do método *onAuthStateChanged onActivityResult* e *firebaseAuthWithGoogle*, apresentada na Figura 1, a qual autentica o responsável do animal por meio do e-mail. Para monitorar o estado de autenticação do usuário, cria-se o variável mAuthListener, que traz consigo o método para verificar o estado da autenticação, apresentado no método que inicia na linha 3 até a 9. Já o método *firebaseAuthWithGoogle*, apresentado a partir da linha 34, tem como funcionalidade de, se o usuário está conectado, ele recebe um *token* no *GoogleSignInAccount* que troca este *token* por uma credencial do Firebase para confirmar a autenticação. Quando bem-sucedido este processo, pode ser usado o método *getCurrentUser* para coletar informações do usuário, neste caso, o e-mail.

```
mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth)
{
        if (firebaseAuth.getCurrentUser() != null) {
            startActivity(new Intent(CadastroActivity.this,
ListarActivity.class));
        }
    }
@Override
public void onActivityResult(int requestCode, int resultCode, Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task =
GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            GoogleSignInAccount account =
task.getResult(ApiException.class);
            if (task.isSuccessful()) {
                firebaseAuthWithGoogle(account);
            } else {
                Toast.makeText(CadastroActivity.this, "Autenticação
errado", Toast.LENGTH_SHORT).show();
            }
        } catch (ApiException e) {
            e.printStackTrace();
        }

    }
}

private void firebaseAuthWithGoogle(GoogleSignInAccount account) {
    AuthCredential credential =
GoogleAuthProvider.getCredential(account.getIdToken(), null);
    mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task)
{
                    if (task.isSuccessful()) {
                        mAuth.getCurrentUser();
                    } else {
                        Toast.makeText(CadastroActivity.this,
"Authentication failed.",
                                Toast.LENGTH_SHORT).show();
                    }
                }
            });
}
```

Listagem 1 – Métodos para autenticar usuário através do e-mail.

A Listagem 2 apresenta a codificação do método da função *onClickSalvarAnimal* do botão salvar da classe de Animal, esta classe contém as funcionalidades de formulário de cadastro do animal, conforme a Figura 14. Quando clicado no botão Salvar, o aplicativo vai

validar os campos, como exemplo a linha 2, se o campo nome estiver vazio, o sistema informará o usuário que deve informar o nome, conforme a linha 3. Na linha 7, apresenta a codificação para salvar a imagem do animal, se a fotografia for diferente de nulo, ela será setada no campo UrlFoto. Foi criado o método uploadFoto o qual enviará para o *AsyncTask* do animal a fotografia para que seja salva no banco. Com todos os campos preenchidos, ao clicar em Salvar, serão mandadas todas as informações para o *AnimalAsynTask*, o qual salvará no banco os dados do animal.

```
1 public void onClickSalvarAnimal(View view) {
2    if (etNome.getText().toString().equals("")) {
3        etNome.setError("Informe o nome!");
4        etNome.requestFocus();
5        return;
6    }
```

```
7    if (fotografia != null)
8       animal.setUrlFoto(uploadFoto());
9    else if (fotoAnimal != null & fotoAnimal != "")
10        animal.setUrlFoto(fotoAnimal);
11
12    animal.setNome(etNome.getText().toString());
13
14    String msg = "";
15    if (rbFem.isChecked()) {
16        msg = "Fêmea";
17    } else if (rbMac.isChecked()) {
18        msg = "Macho";
19    }
20    animal.setSexo(msg);
21
22    String strData = tvDataNasc.getText().toString();
23    if (!"".equals(strData)) {
24        String[] partes = strData.split("/");
25        if (partes.length >= 3) {
26            Calendar data = new
27 GregorianCalendar(Integer.parseInt(partes[2]),
28                    Integer.parseInt(partes[1]) - 1,
29 Integer.parseInt(partes[0]));
30            animal.setData(data);
31        }
32    }
33
34    animal.setTipoAnimal(spAnimal.getSelectedItem().toString());
35    animal.setRaca(spRaca.getSelectedItem().toString());
36    animal.setEmail(account.getEmail());
37
38
39    new AnimalAsyncTask(this, new Command() {
40        @Override
41        public void execute(Object params) {
42            Toast.makeText(CadastrarAnimalActivity.this, "Sucesso!!!",
43 Toast.LENGTH_LONG).show();
```

```
44 setResult(PetCharmUtils.ATUALIZAR_LISTA);
45             finish();
46         }
47     }).execute("salvar", animal);
48 }
```

Listagem 2 - Método da função onClickSalvarAnimal da classe Animal.

A Listagem 3 apresenta parte do código desenvolvido do método de listarAnimais na lista de animais, apresentado na Figura 13. Esta codificação é responsável pela criação dos cartões nos quais apresentam-se os animais cadastrados, bom como com suas funcionalidades, acessar o menu ou excluir o animal. Neste método eu instancio o animal no objeto e crio os cartões para cada animal, a partir da linha 13, onde crio com a imagem, o nome do animal e os botões de menu e excluir para cada animal o cartão. Este é um formato utilizando o Material Design.

```
1 private void listarAnimais() {
2     new AnimalAsyncTask(this, new Command() {
3         @Override
4         public void execute(Object params) {
5             if (params != null && params instanceof List) {
6                 List lista = (List) params;
7                 if (lista.size() > 0) {
8                     mListView.getAdapter().clearAll();
9 }
10                    for (Object obj : lista) {
11                        if (obj instanceof Animal) {
12                            Animal animal = (Animal) obj;
13                            Card card = new
14 Card.Builder(getBaseContext())
15                                    .withProvider(new CardProvider())
16
17.setLayout(R.layout.material_basic_image_buttons_card_layout)
18                                    .setTitle(animal.getNome())
19                                    .setTitleGravity(Gravity.START)
20                                    .setDrawable(animal.getUrlFoto())
21                                    .setDrawableConfiguration(new
22 CardProvider.OnImageConfigListener() {
23                                        @Override
24                                        public void
25 onImageConfigure(@NonNull RequestCreator requestCreator) {
26                                            requestCreator.fit();
27                                        }
28                                    })
29                                    .addAction(R.id.left_text_button,
30 new TextViewAction(getBaseContext())
31                                            .setText("Menu")
32
33 .setTextResourceColor(R.color.black_button)
34                                            .setListener(new
35 OnActionClickListener() {
36                                                @Override
37                                                public void
38 onActionClicked(View view, Card card) {
39                                                    Intent i = new
40 Intent(ListarActivity.this, MenuActivity.class);
```

```
41
42 i.putExtra("animal", (Animal) card.getTag());
43
44 startActivityForResult(i, REQUEST_CODE);
45                                                    }
46                                                 }))
47                                        .addAction(R.id.right_text_button,
48 new TextViewAction(getBaseContext())
49                                                .setText("Excluir")
50
51 .setTextResourceColor(R.color.black_button)
52                                                .setListener(new
53 OnActionClickListener() {
54                                                    @Override
55                                                    public void
56 onActionClicked(View view, Card card) {
57
58 deletarAnimal(card);
59                                                    }
60                                                }))
61                                        .endConfig()
62                                        .setTag(animal)
63                                        .build();
64                                mListView.getAdapter().add(card);
65                            }
66                        }
67                    }
68                }
69            }
70    }).execute("listar", account.getEmail());
71 }
```

Listagem 3 – Método de listarAnimais apresentado na tela Lista de Animais.

A Listagem 4, exibe o método NotificarAgendamento. Ele é responsável para gerar uma alerta para o responsável do animal informando que o animal tem um evento naquela data específica. Verifica-se a data atual, através da variável calendarAtual, e a data que o usuário informou na aplicação, variável *calendar* que recupera através do getDataAvisar, se as datas forem iguais, conforme mostrado da linha 6 a linha 11, é construído a notificação da linha 13 a 16 criando a mensagem que é apresentada para o responsável. Quando o responsável clica nesta notificação, ele é redirecionado para a aplicação.

```
1 private void NotificarAgendamento(Context context, Agendamento
2 agendamento) {
3    Calendar calendarAtual = Calendar.getInstance();
4    Calendar calendar = Calendar.getInstance();
5    calendar.setTime(agendamento.getDataAvisar().getTime());
6    if (calendar.get(Calendar.DAY_OF_MONTH) ==
7 calendarAtual.get(Calendar.DAY_OF_MONTH)
8            && calendar.get(Calendar.MONTH) ==
9 calendarAtual.get(Calendar.MONTH)
10            && calendar.get(Calendar.YEAR) ==
11 calendarAtual.get(Calendar.YEAR)) {
```

```
12
13          String mensagem = String.format("Agendado %s para o animal
14 %s", agendamento.getDescricao(), agendamento.getAnimal().getNome());
15          NotificationCompat.Builder mBuilder = criarBuilder(context,
16 mensagem);
17
18          Intent resultIntent = new Intent(context,
19 CadastroActivity.class);
20
21          TaskStackBuilder stackBuilder =
22 TaskStackBuilder.create(context);
23          stackBuilder.addParentStack(CadastroActivity.class);
24          stackBuilder.addNextIntent(resultIntent);
25          PendingIntent resultPendingIntent =
26                  stackBuilder.getPendingIntent(
27                          0, PendingIntent.FLAG_UPDATE_CURRENT
28                  );
29          mBuilder.setContentIntent(resultPendingIntent);
30          NotificationManager mNotificationManager =
31                  (NotificationManager)
32 context.getSystemService(Context.NOTIFICATION_SERVICE);
33
34          mNotificationManager.notify(novoIdentificadorAleatorio(),
35 mBuilder.build());
36      }
37 }
```

Listagem 4 – Método de NotificarAgendamento.

Na Listagem 5, será apresentado a função para fazer o upload da imagem no servidor e seu armazenamento. Nas linhas 3 e 4 ele recebe o nome e a imagem com o formato de *MultipartFile* que é utilizado para arquivos grande e que são enviados em partes. Na linha 5 inicia a validação do conteúdo, no qual obtém os bytes da imagem. Pega diretório da instalação no diretório *tmpfiles*, conforme linha 10 e 11. Logo após cria um arquivo para a gravação dos bytes da imagem, e grava este conteúdo no arquivo criado, linhas 15 a 23. A partir da linha 24, são tratadas as exceções de falhas de *upload* da imagem.

```
1 @RequestMapping(value = "/uploadImagem", method = RequestMethod.POST)
2     public @ResponseBody
3     String upload(@RequestParam("nome") String nome, @RequestParam("imagem")
4 MultipartFile imagem) {
5           if (!imagem.isEmpty()) {
6                 try {
7                       byte[] bytes = imagem.getBytes();
8
9                       String rootPath = System.getProperty("user.dir");
10                      File dir = new File(rootPath + File.separator +
11 "tmpFiles");
12 if (!dir.exists())
13                            dir.mkdirs();
14
15                      File serverFile = new File(dir.getAbsolutePath() +
16 File.separator + nome);
```

```
17                          BufferedOutputStream stream = new
18 BufferedOutputStream(
19                                      new FileOutputStream(serverFile));
20                          stream.write(bytes);
21                          stream.close();
22
23                          return nome;
24                  } catch (Exception e) {
25                          return "You failed to upload " + nome + " => " +
26 e.getMessage();
27                  }
28              } else {
29                      return "You failed to upload " + nome + " because the file
30 was empty.";
31              }
32      }
```

Listagem 5 – Função para upload da imagem no servidor.

4.5 TESTES

Os testes foram realizados de forma informal, de acordo com o desenvolvimento no aplicativo. Foi realizado testes individualizados para validação de campos, bem como sua persistência no banco de dados.

Novos testes foram realizados para garantir que a imagem estava sendo salva no banco de dados.

Após a inserção do *login* por e-mail, foi realizado testes com mais de um dispositivo para a verificação e a integridade dos dados, onde foi cadastrado um novo animal e de um novo tratamento. Logo após verificando o outro celular para confirmar se as informações contidas nos dois aparelhos estavam sincronizadas de acordo com o e-mail cadastrado para ambos.

Ao realizar a codificação do QrCode, foi baixado um aplicativo para a leitura deste código de barra bidimensional. Foram cadastrados alguns exames e logo após gerado o QrCode. A partir disso foi feita a leitura deste código de barra e verificado, após a leitura, a decodificação gerando a imagem do exame ao qual estava sendo gerado o QrCode. Esta funcionalidade tem como finalidade auxiliar o médico veterinário, para que o mesmo tenha informações dos resultados de exames. O médico deverá ter em mãos somente um dispositivo que possua um aplicativo para leitura de QrCode.

Com a codificação das notificações, tanto para o agendamento, como para o tratamento, foram cadastrados novos eventos para confirmar a geração de notificações para o usuário, estas ocorrem pela manhã.

A realização dos testes, foram realizadas em um celular, no qual foi instalado o aplicativo, para cada dado salvo, era verificado sua integridade no bando de dados. Para acessar o banco de dados externo, foi criado um *web service.*

A partir da integração de todas as funcionalidades do sistema proposto, foram realizados novos testes garantindo assim a integridade das informações bem como os alertas por ele gerado. Para a realização dos testes, foram utilizados dados de animal de estimação e criados algumas informações para inserir no aplicativo, para assim obter os dados esperados, verificando assim, a funcionalidade esperada perante este trabalho.

5 CONCLUSÃO

Com o intuito da construção de um aplicativo para gerenciar os dados de animais de estimação para dispositivos móveis. Este trabalho abordou o uso de aplicativos e dispositivos móveis, celular, *tablets* e o crescimento de animais de estimação.

Para a realização deste trabalho foram elencadas as atividades necessárias para a construção de um aplicativo que contém as seguintes funcionalidades: incluir agendamentos, tratamentos e exames, emitir notificação informando ao usuário que possui pendência de seu animal, gerar QrCode de exames. Estas informações serão consultadas por meio de um *QrCode* para que possam auxiliar o médico veterinário.

Para a realização da modelagem, foi utilizada a UML, por meio do programa Visual Paradigm para a construção de diagramas. As modelagens deste software foram explanadas ao longo do trabalho para conceituar as tecnologias e as ferramentas utilizadas no auxilio no processo de desenvolvimento do aplicativo através dos diagramas apresentados.

O desenvolvimento do aplicativo foi desenvolvido com o software Android Studio, utilizando linguagem Java e o emulador que recria a tela de um celular ou *tablet*, o SDK. Essas ferramentas auxiliaram no desenvolvimento da proposta deste trabalho. Esta ferramenta tem como principal funcionalidade auxiliar os proprietários no cuidado com seus animais de estimação, realizando e gerenciamento de dados do animal.

Como trabalhos futuros, visando complementar o que foi implementado neste trabalho, podem ser adicionadas novas funcionalidades, como a inclusão do *QrCode* na coleira do animal, para que o médico veterinário ao decodificar o QrCode, tenha as informações do animal.

6 REFERÊNCIAS

Linux Open Souce Specialists. Disponível em: <https://www.4linux.com.br/o-que-e-jboss> Acesso em: 13/06/2017.

Linux Open Souce Specialists. Disponível em: <https://www.4linux.com.br/o-que-e-postgresql> Acesso em: 28/06/2017.

ABEMD, Associação Brasileira de Marketing de Dados. Disponível em: <http://abemd.org.br/noticias/mma-mobile-report-2015-internet-e-meio-mais-indispensavel-para-brasileiros> Acesso em: 20/06/2017.

ANISZCZYK, CHRIS; GALLARDO, DAVID. **Introdução à plataforma Eclipse.** Disponível em: <https://www.ibm.com/developerworks/br/library/os-eclipse-platform/> Acesso em: 10/06/2017.

ARAÚJO, Marco Antônio, Modelagem de dados com a Visual Paradigm - Do modelo de classes à criação do banco de dados. Outubro, 2007.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML.** Rio de Janeiro: Elsevier, 2007

BIAZUS, Diogo de Oliveira. **Introdução e Histórico.** Disponível em: <https://wiki.postgresql.org/wiki/Introdu%C3%A7%C3%A3o_e_Hist%C3%B3rico> Acesso em: 11/06/2017.

BRITO, Robison Cris. **Android para iniciantes com Eclipse Passo a passo.** Rio de Janeiro: Editora Ciência Moderna Ltda, 2015.

BRITO, Robison Cris. **Android com Android Studio passo a passo.** Rio de Janeiro: Editora Ciência Moderna Ltda, 2017.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário.** Rio de Janeiro: Elsevier, 2000.

CAETANO, Elaine Cristina Salvaro. **As Contribuições Da TAA – Terapia Assistida por Animais à Psicologia.** Junho, 2010, p. 14-19.

CanalTech; **Java é a linguagem de programação mais utilizada no mundo.** Disponível em: <https://canaltech.com.br/noticia/programacao/java-e-a-linguagem-de-programacao-mais-utilizada-no-mundo-55819/> Acesso em 12/06/2017.

CORDEIRO, Fillipe. **Introdução ao Material Design.** Disponível em: <https://www.androidpro.com.br/blog/design-layout/android-material-design-introducao/> Acesso em: 10/05/2018.

DEVELOPERS. **Arquitetura da Plataforma.** Disponível em: <https://developer.android.com/guide/platform/?hl=pt-br> Acesso em: 03/07/2018.

DEVELOPERS. **Notificações.** Disponível em: <https://developer.android.com/guide/topics/ui/notifiers/notifications?hl=pt-br> Acesso em: 13/06/2018.

DIAS, Emílio. **Como criar um Web Services RESTful com Spring Boot.** Disponível em: <http://blog.algaworks.com/como-criar-web-services-restful-com-spring-boot/> Acesso em: 02/05/2018

DIGITAL, Convergência. Brasileiro tem, em média, 20 aplicativos por smartphone. Disponível em: <http://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?UserActiveTemplate=site&infoid=41595&sid=17#.WNAlKfnyvDf> Acesso em: 19/03/2017.

DIONISIO, Edson José. **PostgreSQL Tutorial.** Disponível em: <https://www.devmedia.com.br/postgresql-tutorial/33025> Acesso em: 12/05/2018.

EGGEA, Rodrigo Fagundes. **Aplicação Android Utilizando Sistema de Localização Geográfica para determinação de Pontos Turísticos na cidade de Curitiba.** 2013, p. 16-19.

EIS, Diego; **O básico: O que é HTML?** Disponível em: < https://tableless.com.br/o-que-html-basico/> Acesso em 13/06/2017.

ERL,Thomas. **Web Services.** Disponível em: <https://www.devmedia.com.br/web-services/2873> Acesso em: 11/06/2018.

FARIA, Fernanda B.; LIMA, Priscila da S. N.; DIAS, Luiz G.; SILVA, Andrea A.; COSTA Mayara P. da.; BITTAR, Thiago J.. **Evolução e Principais Características do IDE Eclipse.** Disponível em : <http://www.enacomp.com.br/2010/cd/artigos/completos/enacomp2010_23.pdf> Acesso em 10/06/2017.

FERNANDES, Vitor Hugo Correia. **Transposição de aplicações Desktop para plataformas móveis.** Out/2010, p.1-4.

GENTIL, Efraim. **Introdução ao Spring Framework.** Disponível em: <https://www.devmedia.com.br/introducao-ao-spring-framework/26212> Acesso em: 02/05/2018

Getting Started. Disponível em: < http://tableless.github.io/iniciantes/manual/css/> Acesso em: 12/06/2017.

IBGE, Instituto Brasileiro de Geografia e Estatística. **Panorama da saúde brasileira um múltiplos aspectos.** Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/imprensa/ppts/00000021940606122015061802940 64.pdf> Acesso em: 16/03/2017

IBGE, Instituto Brasileiro de Geografia e Estatística. **Plano Nacional de Saúde.** Disponível em: <http://portalarquivos2.saude.gov.br/images/pdf/2015/agosto/24/PNS-Volume-2-completo.pdf> Acesso em: 16/03/2017

IBOPE <http://www.ibopeinteligencia.com/noticias-e-pesquisas/pesquisa-traca-o-perfil-dos-proprietarios-de-pets-no-brasil/> Acesso em 08/06/2017

KNOPLOCH, Carol. **Brasil tem mais cachorros de estimação do que crianças, diz pesquisa do IBGE.** Disponível em: <http://oglobo.globo.com/sociedade/saude/brasil-tem-mais-cachorros-de-estimacao-do-que-criancas-diz-pesquisa-do-ibge-16325739> Acesso em: 17/03/2017

LARMAN, Craig. **Utilizando UML e Padrões.** Porto Alegre: Bookman, 2007.

LARROSSA, Luciano. **Moqups** Disponível em: < http://www.apptuts.com.br/review/web/moqups/> Acesso em: 27/06/2017.

LECHETA, Ricardo R.. **Google Android Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** São Paulo: Novatec Editora, 2013.

LEONHARDT, Caio César; BOUVIÉR, Wilian. **Desenvolvimento de Aplicação Móvel com estudo se caso para Clínica Estética.** 2012, p.1-6.

MANDEL, Arnaldo; SIMON, Imre; LYRA, Jorge L. de. **Informação: computação e comunicação.** Disponível em: < https://www.ime.usp.br/~is/infousp/imre/imre.htm> Acesso em: 27/05/2017.

MARTINS, Mauricio de Freitas. **A história dos animais de estimação.** Disponível em: <http://www.artigos.com/artigos/20204-a-historia-dos-animais-de-estimacao> Acesso em: 11/05/2017.

NAPOL, Igor. Brasil tem o mercado mais competitivo para aplicativos móveis. Disponível em:<https://www.tecmundo.com.br/apps/105145-brasil-tem-mercado-competitivo-aplicativos-moveis.htm> Acesso em 09/06/2017

NEVES, Thiago Tadeu. **Material Design: Como aprimorar a qualidade de interface em Android.** Disponível em: <https://www.devmedia.com.br/material-design-como-aprimorar-a-qualidade-de-interface-em-android/33773> Acesso em: 16/05/2018

NOBREGA, Pablo Bruno de Moura. **Explorando o WildFly 8.** Disponível em: < http://www.devmedia.com.br/explorando-o-wildfly-8/30713> Acesso em: 13/06/2017**.**

OGLIARI, Ricardo da Silva; BRITO, Robison Cris. **Android Do Básico ao Avançado.** Rio de Janeiro: Editora Ciência Moderna Ltda, 2014.

OGLIARI, Ricardo; BRITO, Robison Cris. **Trabalhando com notificações no Android – Revista Mobile Magazine 47.** Disponível em: < https://www.devmedia.com.br/trabalhando-com-notificacoes-no-android-revista-mobile-magazine-47/27637> Acesso em: 13/06/2018.

OLIVEIRA, Eric C M. **JBoss: Um Servidor Open-Source Java de Sucesso**. Disponível em: <http://www.linhadecodigo.com.br/artigo/1135/jboss-um-servidor-open-source-java-de-sucesso.aspx> Acesso em: 13/06/2017.

PEREIRA, Ana Paula; **O que é Java?** Disponível em: <https://www.tecmundo.com.br/programacao/2710-o-que-e-java-.htm> Acesso em 12/06/2017.

PEREIRA, Julio Cezar. **Servidor de aplicações JBOSS.pdf**. 2003. p. 30 – 40.

PINTO, Pedro. **O que são Web Services? Para que servem?** Disponível em: < https://pplware.sapo.pt/tutoriais/o-que-so-web-services-para-que-servem/> Acesso em 11/06/2018.

PNAD, Pesquisa Nacional por Amostra de Domicílios. **Síntese de indicadores 2015.** Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/livros/liv98887.pdf> Acesso em: 20/03/2017.

PRESSMAN, Roger S. **Engenharia de Software.** São Paulo, *Pearson Makron Books*, 1995.

SANTOS, Jocelaine. **Cinco benefícios que animais de estimação trazem a família.** Disponível em: <https://www.semprefamilia.com.br/cinco-beneficios-que-animais-de-estimacao-trazem-a-familia/> Acesso em: 12/06/2017

SILVA, Marcelo. **Mockup: para que serve?** Disponível em: <http://blog.creativecopias.com.br/mockup-para-que-serve/> Acesso em: 03/07/2018.

SILVA, Rafael Felix da. **O pequeno notável.** Disponível em: <http://www.devmedia.com.br/sqlite-o-pequeno-notavel/7249> Acesso em: 09/06/2017

SILVA FILHO, Antônio Mendes da. **Requisitos Não Funcionais.** Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525> Acesso em: 20/05/2017

SOBRINHO, Cristiana. **Android (Sistema Operativo).** Disponível em: < http://knoow.net/ciencinformtelec/informatica/android-sistema-operativo/> Acesso em: 05/06/2017

SOFTWARE, Opus. **Estatísticas de uso de celular no Brasil.** Disponível em: <http://www.opus-software.com.br/estatisticas-uso-celular-brasil/> Acesso em: 16/03/2017

SQLite. Disponível em: <http://www.sqlite.org/about.html>Acesso em: 05/06/2017

TAVARAYAMA, Rodrigo; SILVA, Regina Célia Marques Freitas; MARTINS, José Roberto. **A Sociedade da Informação: Possibilidades e Desafios.** Abr/2012, p 255-259.

TEIXEIRA, Marcelo Mendonça; LIMA JÚNIOR, Joel Alves de. **Cidadania digital: uma proposta de dispositivo móvel para o monitoramento das cidades.** Dez/2013, p.3-4.

TEIXEIRA, Maria. **Programação Orientada a Objetos – Linguagem Java.** 2012. p. 1-10.

TIME, Mobile. **Uso dos aplicativos móveis cresceu 58% em 2015.** Disponível em: <http://www.mobiletime.com.br/05/01/2016/pesquisa-uso-dos-aplicativos-moveis-cresceu-58-em-2015/425202/news.aspx?noticiario=TT> Acesso em: 19/03/2017

TOSIN, Carlos. **Conhecendo o Android.** Disponível em: < http://www.dicas-l.com.br/arquivo/conhecendo_o_android.php#.Wx9SB4pKjIU> Acesso em: 12/06/2017

XAVIER, Fox. **Qr Code: entenda o que é e como funciona o código.** Disponível em: <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2011/03/um-pequeno-guia-sobre-o-qr-code-uso-e-funcionamento.html> Acesso em: 12/06/2018