DIEGO DA SILVA LOPES
GABRIEL WINICIUS CAIRES COELHO

Desenvolvimento do aplicativo Meupet para gestão de informações sobre animais domésticos

JI-PARANÁ
2022

DIEGO DA SILVA LOPES
GABRIEL WINICIUS CAIRES COELHO

Desenvolvimento do aplicativo Meupet para gestão de informações sobre animais domésticos

Projeto de Pesquisa apresentado à Banca Examinadora do Centro Universitário São Lucas Ji-Paraná, com requisito de aprovação para obtenção do Título de Bacharel em Sistema de informação.

Orientador: Prof. Esp. Romário Vitorino Ferreira

JI-PARANÁ

2022

Dados Internacionais de Catalogação na Publicação - CIP

L864d Lopes, Diego da Silva.

Desenvolvimento do aplicativo Meupet para gestão de informações sobre animais domésticos. / Diego da Silva Lopes; Gabriel Winicius Caires Coelho. – Ji-Paraná, 2022.
71 p.; il.

Projeto de Pesquisa (Bacharel em Sistemas de Informação) – Centro Universitário São Lucas Ji-Paraná, 2022.

Orientador: Prof. Esp. Romário Vitorino Ferreira.

1. Desenvolvimento Aplicação mobile. 2. Análise de requisito mobile. 3. React Native. 4. Carteira de vacinação digital MeuPet. I. Coelho, Gabriel Winicius Caires. II. Ferreira, Romário Vitorino. III. Título.

CDU 004.4:636.045

Ficha Catalográfica Elaborada pelo Bibliotecário Giordani Nunes da Silva CRB 11/1125

RESUMO

O presente trabalho especifica e descreve uma análise de requisitos do aplicativo MEUPET CARTEIRA DIGITAL voltado para plataforma mobile, com sistema operacional Android. Esse aplicativo busca facilitar o acesso rápido para as informações do animal de estimação eliminando aquela frase clássica não sei onde guardei. Para o desenvolvimento do projeto utiliza-se a metodologia de desenvolvimento cascata, e para a programação foi escolhido React Native por se um Framework multiplataforma podemos reutilizar o código tanto para Android, IOS, Windows e Web ele é baseado na linguagem JavaScript em conjunto ao Software Visual Studio Code da Microsoft. Também será utilizado Figma para criação e conceito da interface de usuário, também usamos *Android Studio* para criar um dispositivo virtual *Android* para criação do aplicativo, em sua base teórica apresentada utilizará a modelagem da aplicação por meio de diagramas UML, nos quais foram descritos os casos de uso e expandido, especificado os requisitos funcionais e não funcionais e, finalmente, diagrama de classes e diagrama de atividade.

Palavras Chaves: Desenvolvimento Aplicação mobile, Análise de requisito mobile, React Native, Carteira de vacinação digital MeuPet.

ABSTRACT

The present work specifies and describes a requirements analysis of the MEUPET CARTEIRA DIGITAL application aimed at the mobile platform, with the Android operating system. This application seeks to facilitate quick access to pet information by eliminating that classic phrase I don't know where I kept it. For the development of the project, the waterfall development methodology is used, and for the programming React Native was chosen because it is a cross-platform framework we can reuse the code for both Android, IOS, Windows and Web it is based on the JavaScript language together with the Software Visual Studio Code by Microsoft. Figma will also be used to create and concept the user interface, we also use Android Studio to create an Android virtual device to create the application, in its presented theoretical basis it will use the application modeling through UML diagrams, in which the cases were described of use and expanded, specifying the functional and non-functional requirements and, finally, class diagram and activity diagram.

Keywords: Mobile application development, Mobile requirement analysis, React Native, MeuPet digital vaccination card.

LISTA DE TABELAS

Tabela 1: Padrões de protocolo de vacinação 15
Tabela 2: Quadro dos tipos de vacinas obrigatórias 16
Tabela 3: Requisitos - Cadastrar Usuário 33
Tabela 4: Requisitos - Cadastrar Vacina..... 34
Tabela 5: Requisitos - Remédio Adicional 34
Tabela 6: Requisitos - Gerar Carteira 35
Tabela 7 – Ferramentas / Tecnologias 41

LISTA DE ILUSTRAÇÕES

Figura 1:Tipos de diagramas 18
Figura 2: Arquitetura do Android 21
Figura 3: Visual Studio Code 25
Figura 4: Figma 26
Figura 5: Diagrams.net 27
Figura 6: O processo de software 27
Figura 7: Casos de Uso 29
Figura 8: Diagrama de classe 36
Figura 9: Modelagem do banco de dados 36
Figura 10: Diagrama atividade Login 38
Figura 11: Diagrama de atividade cadastrar pet 38
Figura 12: Diagrama de atividade cadastrar vacina 39
Figura 13: Diagrama de atividade cadastrar remédio adicional 40
Figura 14: Diagrama de atividade gerar carteira 41
Figura 15: Imagem do site nodejs 43
Figura 16: CMD Node version 44
Figura 17: Instalação pacote CLI: 44
Figura 18: Criar projeto 45
Figura 19: Expo teamplate 45
Figura 20: Escolha de teamplate 46
Figura 21:Instalação via npm 46
Figura 22: project ready 47
Figura 23: Tela de login 48
Figura 24: Tela de carregamento 49
Figura 25: Tela home/principal 50
Figura 26: Tela de perfil usuário 51
Figura 27: Tela de cadastrar animal 52
Figura 28: Tela com animal estimação 53
Figura 29: Carteira de vacina 54
Figura 30: Tela de cadastro de vacina e lista de vacina 55

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas.
HAL	*Hardware Abstration Layer* (Camada de abstração de hardware).
DEX	*Dalvik Executable* (Máquina virtual de *Dalvik*).
AOT	*Ahead-of-time* (Compilação antes da execução).
JIT	*Just-in-time* (Compilador em tempo de execução).
API	*Application programming interface* (interface de programação de aplicações).
OpenGL ES	*OpenGL for Embedded Systems* (Subseção de API gráfica).
.APK	*Android Package* (Arquivo de Pacote padrão do Android.)
XML	*Extensible Markup Language* (Formato para a criação de documentos com dados organizados forma hierárquica).
SDK	*Software Development Kit* (Kit de desenvolvimento de software).
ACID	Acrônimo de Atomicidade, Consistência, Isolamento e Durabilidade.
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada).
OHA	*Open Handset Alliance*
NPM	*Node Package Manager (Gerenciador de pacotes do NodeJS)*

SUMÁRIO

INTRODUÇÃO 9
1.1. PROBLEMATIZAÇÃO 9
2. JUSTIFICATIVA 11
3. OBJETIVO GERAL 12
3.1. OBJETIVOS ESPECÍFICOS 12
4. REFERENCIAL TEÓRICO 13
4.1. RELAÇÃO ENTRE HOMEM E O ANIMAL DE ESTIMAÇÃO 13
4.2. VACINAÇÃO 14
4.3. A IMPORTÂNCIA DA VACINA 14
4.4. VACINAS OBRIGATORIAS 15
4.5. REFERENCIAL DE TECNOLOGIAS 17
4.5.1. UML 17
4.5.2. Sistemas Operacionais 19
4.5.3. Sistemas Operacionais Móveis 19
4.5.4. Android 19
4.5.5. JavaScript 24
4.5.6. NodeJs 24
4.5.7. React Native 24
4.5.8. Visual Studio Code 25
4.5.9. Figma 26
4.5.10. Diagrams.net/Draw.io 26
4.6. MODELO DE DESENVOLVIMENTO CASCATA 27
5. MATERIAIS E MÉTODOS 29
5.1. MODELAGEM DO SISTEMA 29
5.1.1. Casos De Uso 29
5.1.2. Documentação De Caso De Uso Expandido 30

5.1.2.1. Caso de Uso *Login* 30
5.1.2.2. Caso de Uso Pet 31
5.1.2.3. Caso de Uso Vacina 32
5.1.2.4. Caso de Uso Remédio Adicional 32
5.1.2.5. Caso de Uso Gerar Carteira 32
5.1.3. Requisitos Funcionais E Não Funcionais 33
5.1.3.1. Cadastrar Usuário 33
5.1.3.2. Cadastrar Vacinação 34
5.1.3.3. Remédio Adicional 34
5.1.3.4. Gerar Carteira 35
5.1.4. Diagramas De Classe 35
5.1.5. DIAGRAMA DE ATIVIDADE 37
5.1.5.1. Diagrama atividade Login 37
5.1.5.2. Diagrama de atividade cadastrar pet 38
5.1.5.3. Diagrama de atividade cadastrar vacina 39
5.1.5.4. Diagrama de atividade cadastrar remédio adicional 40
5.1.5.5. Diagrama de atividade gerar carteira 41
5.2. MATERIAIS 41
6. DESENVOLVIMENTO 43
6.1. Configuração do Projeto 43
7. FUNCIONALIDADE DO APLICATIVO 48
7.1. Tela login. 48
7.2. Tela de Carregamento 49
7.3. Tela Home 49
7.4. Tela de perfil usuário 51
7.5. Tela cadastrar animal de estimação 51
7.6. Tela com animal de estimação cadastrado 52

7.7. Tela da carteira de vacina.....53
7.8. Tela de cadastro de vacina e lista de vacina 55
8. CONSIDERAÇÕES FINAIS. 56
REFERÊNCIAS 57
9. Apêndice.....60
9.1. Código do Projeto.....60
9.1.1. Tela Inicial.....60
9.1.2. Tela de registro 61
9.1.3. Tela de carteira criação de carteira de vacinação 65
9.1.4. Tela de criação de prevenção 66
9.1.5. Tela de vacinas.....67
9.1.6. Tela de medicamento.....69
9.1.7. Tela de listar todos os animais de estimação 70

INTRODUÇÃO

Algo comum entre tutores de animal de estimação, é guardar informações importantes sobre as vacinas aplicadas, de certo modo temos conosco o cartão de vacinação que recebemos do veterinário, como existe intervalo de vacinação que pode ser de 3 meses, 6 meses a 12 meses, perceba que há um grande intervalo de tempo, o suficiente para poder esquecer onde guardou a carteira de vacinação, perdendo essa informação e quais medicamento foram aplicados.

A perda dos dados de vacinação pode acarretar problemas, como o atraso das vacinações dos animais, deixando os animais com uma janela de tempo sem proteção adequada das vacinas.

A criação de uma aplicação para armazenamento e gerenciamento dos dados de vacinação dos animais, pode ser uma ferramenta útil, permitindo o acesso ágil dos dados do animal.

1.1. PROBLEMATIZAÇÃO

A vivência entre seres humanos e animais de estimação intensifica-se a cada ano. De acordo com a pesquisa do G1 constata-se de que há no Brasil, 52 milhões de cães e 22 milhões de gatos em casas brasileiras. Cerca de 44,3% dos lares têm pelo menos um cachorro e 17,7% têm ao menos um gato. A estimativa é que a população de cães e gatos seja de 74,3 milhões no território brasileiro. (G1, 2015)

A partir desses dados, observa-se que as pessoas e famílias, estão tendo cada dia mais animais de estimação como companheiros.

Outro crescimento potencialmente considerável nos últimos anos é o uso de *Smartphone* pelos brasileiros. De acordo com Valente (2019), o Brasil fica em 5° lugar no ranking global de tempo despendido com esse aparelho (O dado é do relatório Estados de Serviços Móveis, elaborado pela consultoria especializada em dados sobre aplicativos para dispositivos móveis *App Annie*).

Com essa premissa, pode-se afirmar que o uso de celulares e consumo de serviços móveis, tornou-se essencial na vida de muitas pessoas. Os benefícios que essa tecnologia traz, é tornar a vida mais prática e móvel, pois o uso de smartphones permite realizar desde uma ligação a uma transação bancária, sem a necessidade de

um computador de mesa para execução de tais funcionalidades. Por causa dessa e outras aplicações, tornando essencial na rotina de muitas pessoas.

A partir dessa análise, o companheirismo de um animal de estimação e a intensificação do uso de smartphone, e suas vastas funcionalidades, faz com este trabalho tenha como plano o desenvolvimento de um aplicativo móvel utilizando a plataforma Android, de forma que venha possibilitar aos donos de animais de estimação, o gerenciamento do histórico de seu animal, o propósito de criação do aplicativo é trazer várias funcionalidades tais como, cadastrar um ou mais pets, registrar a carteira de vacinação de cada um, gerenciar histórico de vermicida e pesticida.

2. JUSTIFICATIVA

Os homens e os animais de estimação coexistem há gerações. De acordo com a reportagem do G1 baseado na pesquisa do IBGE, estima-se que existam mais de 73 milhões de animas de estimação no país, sendo eles 52 milhões de cães e 22 milhões de gatos.

Sendo assim, provavelmente já ocorreu com você, a perda de alguma documentação de seu animal de estimação. Partindo desde pressuposto, optou-se por desenvolver um aplicativo que possa auxiliar na organização dos dados animal de estimação.

Este aplicativo irá possibilitar a organização, armazenamento e a documentação do animal de estimação, tendo-a de forma rápida a palma da mão. O *app* irá possibilitar a administração de forma eficiente situação que possam ocasionar com seu animal, desde o controle de pragas ao dia para próxima dose de vermicida.

3. OBJETIVO GERAL

Desenvolver aplicativo para *Android* com intuito de armazenar informações dos animais de estimação.

3.1. OBJETIVOS ESPECÍFICOS

- Desenvolver a tela de cadastro do animal.
- Desenvolver a tela de carteira de vacinação
- Desenvolver a lista com um ou mais animais cadastrados.

4. REFERENCIAL TEÓRICO

4.1. RELAÇÃO ENTRE HOMEM E O ANIMAL DE ESTIMAÇÃO

Saber adestrar um animal é uma arte de adaptar-se, ensinar, treinar para que o convívio entre seres humanos seja o mais amigável possível. Esta convivência teve início a mais de 30 mil anos, de acordo com Caetano (2010), 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, de acordo com (Nichelle, 2018 apud Caetano, 2010).

Segundo Caetano, a definição entre a relação de homens e animais:

> 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. (CAETANO, 2010 apud Starling, Thomas e Guidi 2005, p. 14).

Essa segurança que o animal oferecia, era uma forma de recompensa, baseado em que o homem oferecia comida em troca de proteção, com o passar dos anos a dependência de um pelo outro acabou gerando um laço mais forte e cada vez mais até se tornar afetuoso.

Durante a Idade de Ferro e Bronze, outro animal teve seu destaque, o cavalo. O animal apresentava 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).

Com toda essa evolução homem e animal, e passar do tempo os animais que ficaram mais próximos e fazendo parte dos lares familiares foram os animais de pequeno porte, mais especificamente cães e gatos. De acordo com G1, divulgado em 02 junho 2015 pesquisa realizada pelo IBGE estima-se 52 milhões de cães e 22 milhões de gatos nas casas brasileiras, ou seja, existem mais de 73 milhões no território brasileiro.

Mais não é só afeto que esses animais trouxeram , em uma pesquisa realizada pela Universidade de Cambridge, em 2002 e divulgada em Abril de 2015 por Jocelaine Santos no jornal Gazeta do povo na categoria Sempre Família, aponta que pessoas que adotam animais de estimação tem melhoras em 5 pontos, são: convivência com animais de estimação faz bem, apontando para desenvolvimento na segurança e autoestima e sensação de bem-estar, eles contribuição para uma vida mais saudável apontando o cardiovascular, aproximação entre membros da família apontando estreita do laço familiar, crianças com são mais felizes apontando exercícios, valores como amizade respeito à vida e amor. Eles diminuem a sensação de solidão apontando para todas as idades de preferência na idade idosa.

Com esses benefícios cada vez mais pessoas adotam animais de estimação e tendo cuidados no padrão de bem-estar do animal, do mesmo modo que eles nos oferecem seus benefícios os donos têm o dever de devolver em forma carinhosa, atenção e cuidados necessários a eles.

4.2. VACINAÇÃO

O processo de vacinação em seres humanos é realizado para prevenir e fortalecer os anticorpos de um certo vírus, o mesmo acontece para os animais, pois compartilhamos o mesmo entendimento de proteção, e assim evitar contrair doença devido ao convívio do mesmo ambiente, de com Arca Brasil.

4.3. A IMPORTÂNCIA DA VACINA

Manter os animais protegido é o dever de seus tutores, pois precisam receber vacinação para estarem prevenidos de uma série de doenças que podem prejudicar o desenvolvimento e podendo até levá-los a óbito.

Para ser vacinado o animal deve estar saudável, sem febre ou diarreia e tem que estar com a aplicação de vermífugo em dia, caso não estiver atendendo esses princípios o organismo não pode receber a vacinação.

De acordo com o site Mundo animal (2021), quando se fala de um calendário de vacinação para os animais de estimação, é preciso entender que elas são feitas por ciclos, as principais logo nos primeiros dias de vida e depois reforços são

estabelecidos pelo veterinário e diante da região em que o animal vive - campanhas podem ser feitas pelos municípios.

Segundo a Arca Brasil (2022), o ciclo de vacinação é determinado por um protocolo receitado pelo veterinário. Porém, não existe um padrão de receituário pois isso depende de cada região, mas em si a orientação é a mesma. No geral os padrões de protocolo de vacinação mais utilizados entre eles são:

Tabela 1: Padrões de protocolo de vacinação

Idade	Descrição
Cães e gatos nunca vacinados	Cães ou gatos adultos que nunca foram vacinados ou filhotes que já passaram da época de vacinação. Recebem 2 doses de vacina múltipla no intervalo de 21 dias e 1 dose de vacina antirrábica.
Pela Idade - Gatos	• 60 dias – Múltipla Felina (Tríplice ou Quádrupla); • 90 dias – Múltipla Felina (Tríplice ou Quádrupla); • 120 dias – antirrábica
Pela Idade – Cães	• 45 dias – Múltipla Canina (V8 ou V10); • 75 dias – Múltipla Canina; • 105 dias – Múltipla Canina; • 135 dias – Antirrábica.

Fonte: Construído pelo Próprio autor informações baseada da Arca Brasil (2022)

4.4. VACINAS OBRIGATORIAS

Segundo a Arca Brasil (2022), existem algumas vacinas obrigatórias, como as, Múltiplas Canina (V8 ou V10) que protegem contra cinomose, parvovirose, coronavirose, hepatite infecciosa e leptospirose e a Antirrábica.

Esses dois são obrigatórios em qualquer protocolo, mas, também temos outras vacinas que acabam sendo importantes e essenciais são elas: Tosse dos canis, Giardíase e Leishmaniose Visceral Canina.

Tabela 2: Quadro dos tipos de vacinas obrigatórias

Vacinas	Descrição da Doença
Giardíase	Deve ser aplicada em cães a partir de 8 semanas de idade com duas doses com intervalo de 21 a 28 dias. Os cães adultos que nunca foram vacinados contra a giardíase deverão receber 2 doses da vacina. A proteção se dará após 15 dias da 2ª dose da vacina. O reforço é anual com apenas 1 dose.
Leishmaniose Visceral Canina	É aplicada em cães a partir de 4 meses de idade, saudáveis e soronegativos para Leishmaniose Visceral Canina. O protocolo completo deve ser feito com 3 doses, respeitando o intervalo de 21 dias entre cada dose (aplicação). A revacinação é anual, contada a partir da 1ª Dose
Traqueobronquite Infecciosa (“Tosse dos Canis”)	É causada por bactérias Bordetella em cães sadios, a partir de 8 semanas de idade, toma primeira dose já a segunda dose com intervalo de 2 a 4 semanas. A imunidade se inicia em 21 dias após a administração da segunda dose. O reforço é anual apenas 1 dose.

Fonte: Construído pelo Próprio autor informações baseada da Arca Brasil (2022)

Veja como são essenciais essas vacinas o tutor tem o dever e obrigação de administrar o histórico, o que mais adequado se não transformar esses dados em digital e por meio de gestão possamos cuidar melhor e mais eficiente.

4.5. REFERENCIAL DE TECNOLOGIAS

4.5.1. UML

Antes do surgimento da UML existia diversos métodos de modelagens gráficas de vários autores como: Shadler & Mellor, Coad & Yourdon, Grady Booch, Ivair Jacobson, James Rumbaugh e Fusion, em 1994 quando Rumbaugh se juntou ao Booch na *Rational*, com objetivo de unificar os métodos Booch[1] e OMT[2].

Desde então, no ano de 1995 saiu um esboço chamado *Unifed Process* (Processo Unificado) na mesma época Jacobson se associou à *Rational* e assim contribuiu para o projeto *Unifed Process* contribuindo com a incorporação do método OOSE[3], de acordo com (Martin,2011, p.30)

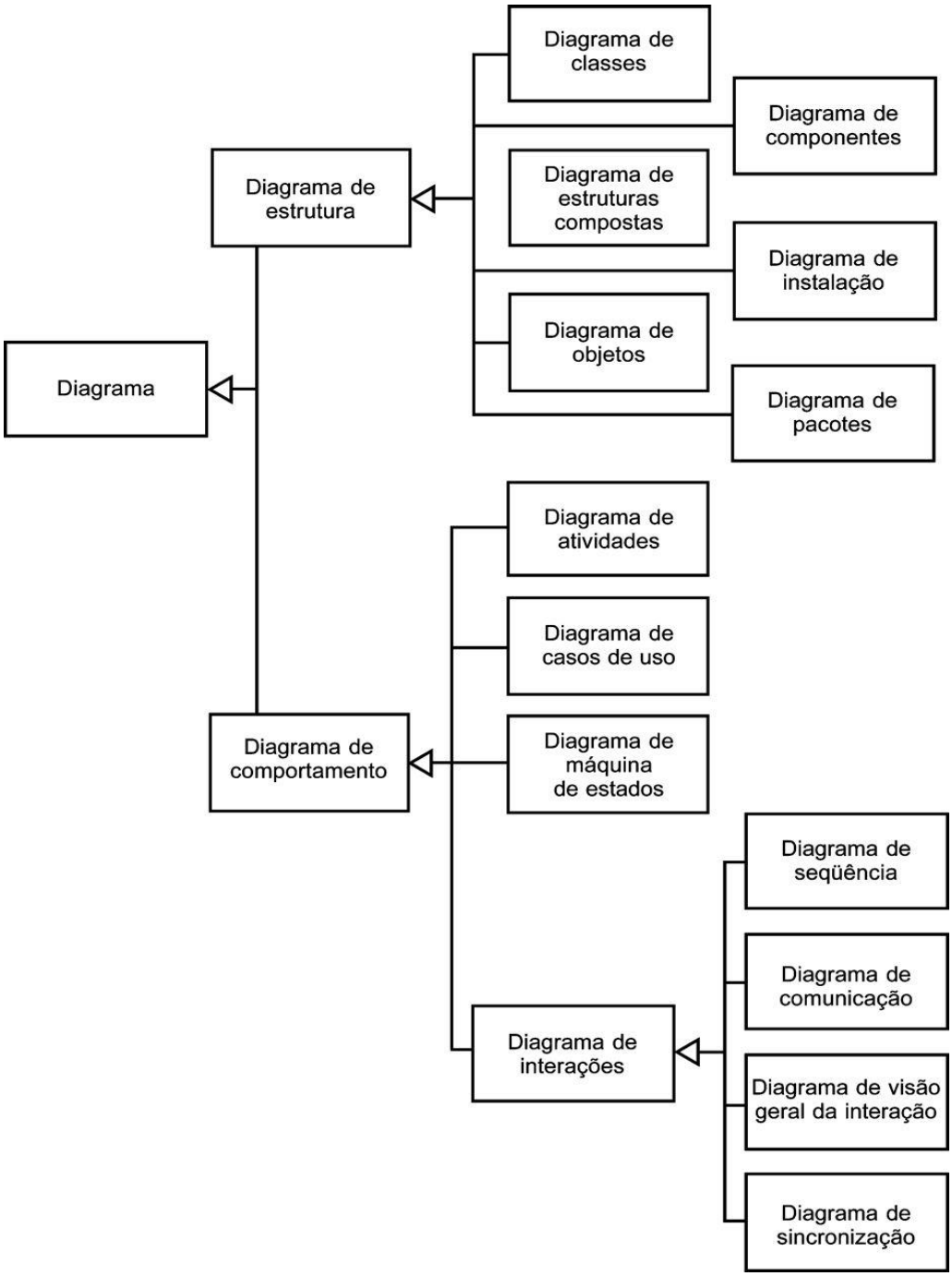
Assim surgiu a UML (*Unified Modeling Language*) Linguagem de Modelagem Unificada de forma genérica é uma linguagem de modelagem visual sua primeira versão oficial foi lançada em 1996, com proposito de ajudar a padronizar arquitetura projetos tornando fácil entendimento de toda a equipe.

A UML tem um peso significativo na produção de software criando diagramas visuais tornando o entendimento do projeto mais fácil para equipe, representando do abstrato para o visual sua lógica de como será arquitetura o projeto. Existe 13 tipos de diagramas conforme a Figura 1.

---

[1] Método *Boosh* (é um método para desenvolvimento de software orientado a objetos.)
[2] Método OMT (Object Modeling Techniquie)
[3] Método OOSE (Object Oriented Software Enginnering)

Figura 1:Tipos de diagramas



Fonte: (SAMEER, 2021)

Diagrama de estrutura: Esses diagramas são usados fazer estruturação do projeto. Por exemplo o diagrama de classe, vai criar as classes de acordo o documento de requisito, montando a suas estruturas que vai ser expandida no diagrama de comportamento.

Diagrama de comportamento: Esses diagramas trazem como vai para cada diagrama de classe criada mostrando todos os passos que deve fazer.

4.5.2. Sistemas Operacionais

De acordo com um dos principais autores da área sistema operacional é,

> Um sistema operacional é um programa que atua como intermediário entre o usuário e o hardware de um computador. O propósito de um sistema operacional é propiciar um ambiente no qual o usuário possa executar outros programas de forma conveniente, por esconder detalhes internos de funcionamento e eficiência, por procurar gerenciar de forma justa os recursos do sistema (Silberschatz, Galvin e Gagne, 2000, p.22].

Para Stallings (2012), um sistema Operacional (OS) usa recursos de hardware para proporcionar ao usuário um conjunto de serviços para realizar a sua utilização. O sistema operacional realiza o controle e gerenciamento de memória e periféricos.

4.5.3. Sistemas Operacionais Móveis

Um sistema operacional móvel é um programa feio para ser executado em dispositivos móveis como smartfones, tablets entre outros. Eles possuem características e funcionalidades de um sistema operacionais de computadores Desktop. São geralmente inicializados juntos com o dispositivo, provendo uma interface para iteração com o dispositivo, além de gerenciar recursos disponíveis em dispositivos moveis, como por exemplo, comunicação de rede, sensores diversos, wireless etc. (STEELE, 2020).

4.5.4. Android

O Android trata-se de um sistema operacional projetado para ser executado em dispositivos móveis (tablets, TVs, carros, vestíveis e smartphones). Desde seu lançamento em 2008, o sistema vem crescendo em popularidade, chegando a se tornar o líder de mercado, estando presente em 2,5 bilhões de dispositivos ativos (ANDROID, 2022).

Android é uma plataforma de código-aberto, ou seja, ele pode ser visualizado, modificado, aprimorado e redistribuído, tudo isso sem a necessidade de pagamento de nenhuma espécie de taxa, royalties ou outros custos. Contudo seu

desenvolvimento é regido pela *OPEN HANDSET ALLIANCE* (OHA), um grupo de empresas de tecnologia, que tem o *Google LCC* como integrante principal. Tem o objetivo de manter o Android com atualizações de segurança e suporte a novos hardwares que fabricantes possam empregar em novos aparelhos (OPEN HANDSET ALLIANCE, 2022).

O seu desenvolvimento foi iniciado pela empresa Android Inc., que posteriormente foi adquirida pelo Google em 2005. Inicialmente o Android foi desenvolvido para ser utilizado em câmeras digitais, mas a visão logo mudou para smartphones, graças ao seu mercado com potencial maior crescimento na época (TANENBAUM; BOS, 2016).

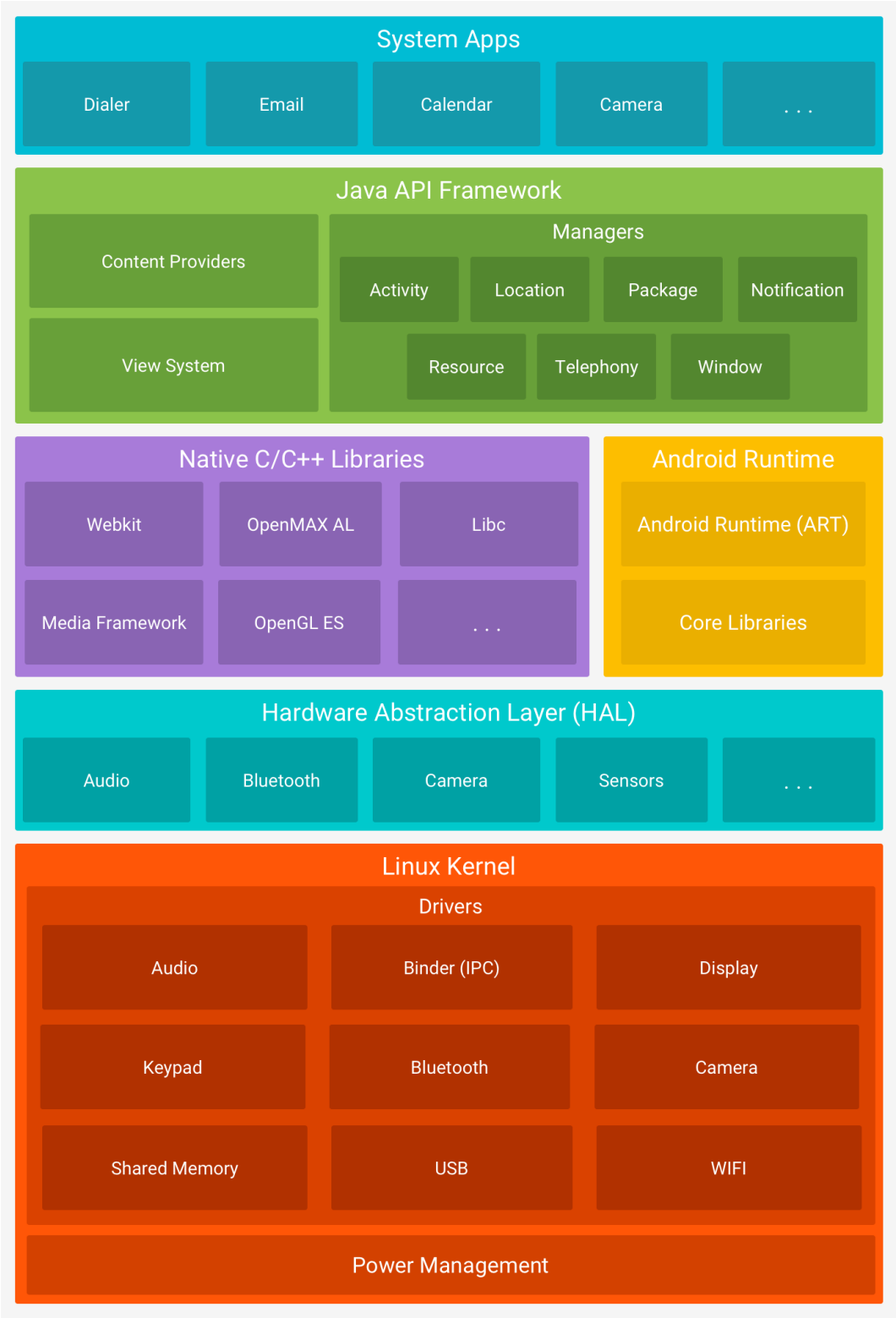
Em 2007 houve o lançamento do primeiro SDK (*Software Development Kit* – Kit de desenvolvimento de software) do Android. Já no ano seguinte foi lançado o primeiro dispositivo móvel comercial com a versão 1.0 o dispositivo Dream, com nome comercial de *T-Mobile G1*. Com o passar dos anos o sistema foi evoluindo com novas versões chegando até o lançamento da versão Android 12 (TANENBAUM; BOS, 2016).

De acordo com Tanenbaum (2015), o Android é um sistema de código aberto e baseada fortemente no *kernel* do Linux (núcleo do Linux), utilizando a maioria dos mecanismos presentes (processos, memória virtual, sistemas de arquivos...), adicionando extensões. Uma quantidade consideravelmente grande do sistema é escrita com linguagem de alto nível, o Java, e tende a seguir um projeto orientado a objetos. Já seu núcleo é composto por muitas bibliotecas de baixo nível que são escritos em C e C++.

Sua arquitetura é composta por várias camadas, cada uma dessas camadas tem características e propósitos próprios. Contudo, nem sempre essas camadas são totalmente separadas e por muitas vezes se infiltram dentro de outras camadas (GARGENTA, M.; NAKAMURA, 2014).

Está descrito abaixo na Figura 2 as camadas de componentes principais da plataforma Android, essas camadas estão detalhadas a seguir:

Figura 2: Arquitetura do Android



Fonte: (ANDROID DEVELOPERS, [s.d.])

Nos tópicos abaixo podem ser observado as principais camadas do Android segundo a documentação oficial do Android ((ANDROID DEVELOPERS, 2022):

- **Kernel do Linux**: Linux foi escolhido com base da pilha por conta da sua portabilidade, por se tratar de uma plataforma flexível e relativamente fácil de moldada para obter compatibilidade com várias arquiteturas de *hardware*, é altamente seguro, já que foi testado por décadas em diversos ambientes, contém recursos muitos uteis, por exemplo, o suporte para gerenciamento de memória e energia.
- **Camada de abstração de hardware (Hardware Abstration Layer - HAL)**: O Android foi projetado para que possa executar nos mais variados *hardwares*, contudo, não existe uma padronização de *drivers* disponíveis para acesso em cada dispositivo. Para que desenvolvedores de aplicativos tivessem que se preocupar com detalhes específicos de *hardware*, como por exemplo, o *drive* da placa gráfica do dispositivo. Existe uma camada que é responsável por fazer a abstração, fornecendo interfaces padrões que expõem os recursos de *hardware* do dispositivo por meio de módulos de bibliotecas, módulos que cara um dos quais implementa uma interface para um tipo específico de componente de hardware, um exemplo seria, a *interface* de interação com o hardware da câmera do dispositivo.
- **Camada de ambiente de tempo de execução (Android Runtime - ART)**: é o ambiente de tempo de execução usado por aplicativos e alguns serviços do próprio sistema no Android. Sua utilização foi iniciada após a versão 5.0 do Android, o ART veio para substituir a máquina virtual *Dalvik*. Foi projetado para trabalhar com o *Dalvik Executable* (DEX), que é um formato de *bytecode* projetado especialmente para oferecer consumo mínimo de memória. Alguns dos principais recursos do ART são a compilação *ahead-of-time* (AOT) e *just-in-time* (JIT), a coleta de lixo otimizada e um melhor suporte à depuração, geração de relatórios de erros e diagnósticos. A partir da versão 9 do Android, o AOT começou a otimizar arquivos compactador DEX do pacote de um aplicativo, convertendo-os para um código de máquina mais compacto,

assim possibilitando um tempo de inicialização dos aplicativos menor e um consumo menor de memória e espaço em disco.

- **Bibliotecas C/C++ nativas (Natives C/C++ Libraries)**: muitos serviços e componentes do Android, ART e o HAL, são criados a partir de códigos nativos, isso requer que as bibliotecas nativas sejam escritas em C e C++. Nessa camada se encontra APIs com a *OpenGL ES*, usada para renderização 3D, *Media Framework* faz o processamento de áudio e vídeo e a *Libc*, que se trata de uma implementação da biblioteca C padrão. Essas bibliotecas podem ser acessadas a partir das APIs da camada superior.
- **Java API Framework**: O conjunto de recursos do sistema operacional Android está disponível por meio de APIs escritas na linguagem Java, é a partir dela que a maioria dos desenvolvedores criam os aplicativos. Assim possibilitando que os desenvolvedores de aplicativos de terceiros possam ter acesso às mesmas APIs que os aplicativos do Android acessam. Dentro dessa camada pode ser encontrado bibliotecas para criação de interfaces, comunicação com outros aplicativos, gerenciamento de notificações, acesso aos dados de outros aplicativos, entre outras.
- **Aplicativos do sistema (System Apps)**: O Android já vem por padrão com um conjunto de aplicativos para correio eletrônico, SMS, calendário, contatos, câmera e outros. Esses aplicativos não têm prioridades especiais em relação de aplicativos de terceiros que o usuário possa instalar. Podendo ser substituído por outros aplicativos que o usuário deseja utilizar.

Os aplicativos Android tem a extensão .apk por convenção, que se refere a *Android Package*, ele contém tudo que forma o aplicativo. Entre os conteúdos importantes presentes no *APK*, está o código fonte do aplicativo, informações de assinatura de identificação de segurança do autor, recursos necessários para aplicativo como dados XML para *layout*, além do manifesto que contém o nome de pacote no estilo Java, para identificar unicamente a aplicação, a descrição de como a aplicação deve ser executada (TANENBAUM; BOS, 2016).

4.5.5. JavaScript

JavaScript® (às vezes abreviado para **JS**) é uma linguagem de programação leve, interpretada e baseada em objetos com funções de primeira classe, muito conhecida como a linguagem de script para páginas *WEB*, mas também utilizada em outros ambientes fora do navegador, como por exemplo o *Node.js (back-end)* e *React Native(mobile)* (FLANAGAN, 2013).

Javascript é uma linguagem de programação de alto nível, dinâmica, interpretada e não tipada, tornando-a bem conveniente para estilos de programação Orientados a objetos e funcionais. A sintaxe do JavaScript é derivada da linguagem Java, nas funções de primeira classe de *Scheme* e herança baseada em protótipo de *Self* (FLANAGAN, 2013).

4.5.6. NodeJs

De acordo com COUTO(2022), *NodeJS*, que é um ambiente que possibilita a execução de códigos *JavaScript* fora dos navegadores web. A utilizando do mesmo está vinculado ao *React-Native,* pois podemos usar arquivos *jsx*[4], ou seja, são funções JavaScript que em seu retorno temos elementos HTML, dessa forma podemos mesclar programação JavaScript e elemento marcadores de texto.

4.5.7. React Native

React Native é um framework de código aberto utilizado para o desenvolvimento de aplicativos híbridos usando React.js. Foi introduzido pelo *Facebook* em 2015 com o lema de “aprenda uma vez (React.js) e escreva em todo lugar (web e mobile)”. O React Native permite aos desenvolvedores reutilizar conhecimentos das tecnologias web para o desenvolvimento de aplicativos móveis. No entanto ele não se comporta como um site executado em uma *WebView*, o *Javascript* no React Native é renderizado como componentes nativos do sistema

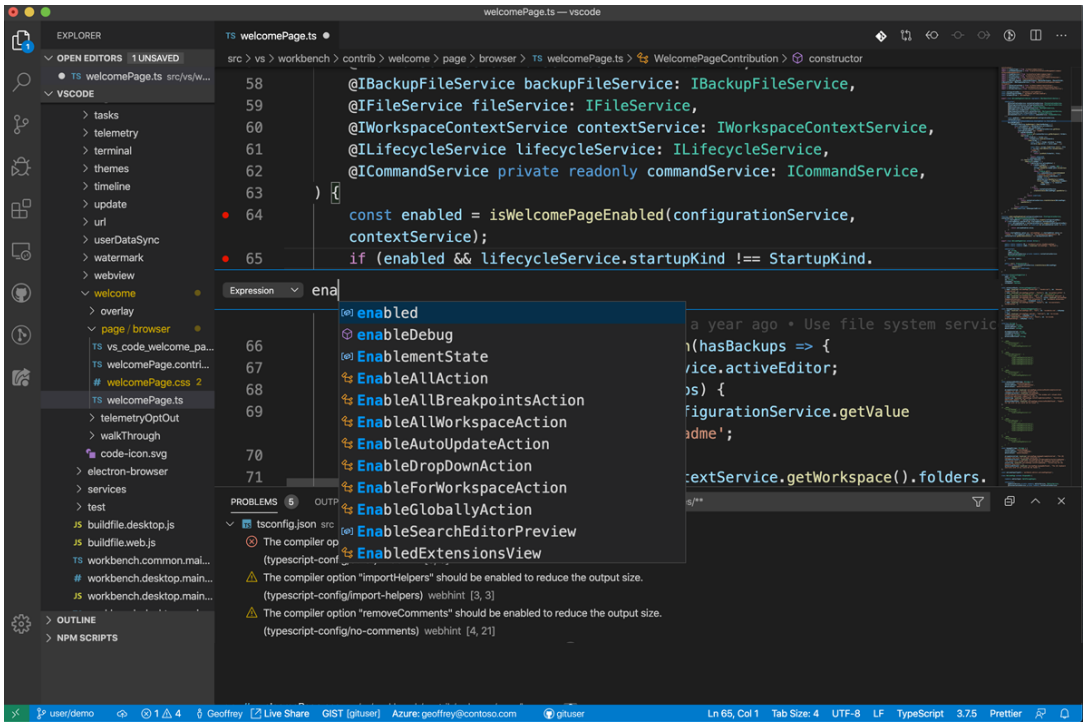
[4] O JSX significa JavaScript XML e nos permite escrever HTML no React, tornando fácil a sua adição. Com o JSX, podemos escrever os elementos HTML dentro do JavaScript e adicioná-los ao DOM sem utilizar métodos como createElement() ou appendChild().

utilizado, isso ocorre por meio de bibliotecas do *React Native* ou bibliotecas de terceiros (REACT NATIVE, 2022).

4.5.8. Visual Studio Code

O *Visual Studio Code* (Figura 5), é um editor de texto leve e poderoso disponível para plataformas Windows, MacOS e Linux. Ele já vem com suporte padrão para linguagem de programação como Javascript, TypeScript. Além de possuir um suporte a extensões, que possibilitam a integração com outras diversas Linguagens e idiomas (MICROSOFT, 2022).

Figura 3: Visual Studio Code



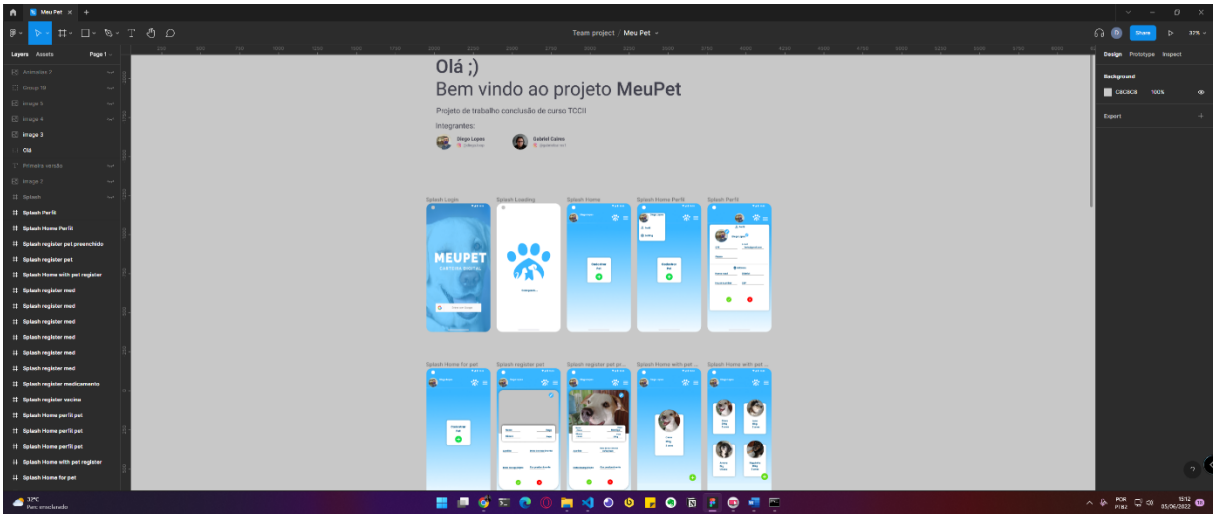
Fonte:(MICROSOFT, 2022)

Segundo (PINTO, 2017), o *Visual Studio Code* é um editor de texto que disponibiliza várias funcionalidades para programadores de forma simples e leve, tendo entre suas principais funcionalidades o suporte a diversas plataformas, controladores de versão (GIT) integrado e suporte para várias linguagens de programação.

4.5.9. Figma

Segundo SIRIUS INTERATIVA (2019), o Figma é uma ferramenta de design de interface onde todo trabalho é realizado através do navegador, logo ela é compatível com qualquer sistema operacional desktop. É multitarefa, ou seja, equipes podem explorar o mesmo projeto juntas observando alterações de outros integrantes em tempo real. Oferece ferramentas de versionamento nativo automático, biblioteca e componentes compartilhados.

Figura 4: Figma



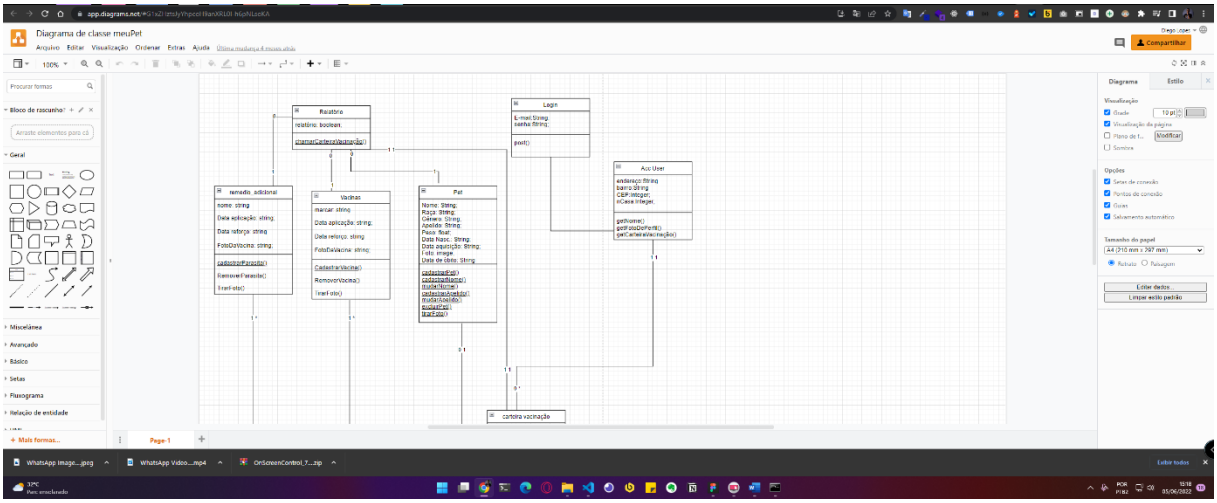
Fonte: Próprio autor, 2022

4.5.10. Diagrams.net/Draw.io

É uma ferramenta de criação de diagramas, fluxogramas, markups, esquemas etc. Ele pode ser utilizado diretamente no navegador, o que possibilita a edição compartilhada em tempo real com outros integrantes de uma equipe. O seu nome sofreu alterações com o passar do tempo iniciando com *Diagramly* passando para *Draw.IO* e por fim uma mudança para *Diagrams.net* (FERREIRA, 2015).

Possibilita a criação de projetos utilizando vários elementos, com por exemplo, Elementos típicos em fluxogramas e diagramas Relação-Entidades, componentes eletrônicos, Padrões de diagramas de UML, entre várias outras funções disponíveis. *Diagrams* tem código aberto, e pode ser modificado seguindo algumas diretrizes da comunidade.

Figura 5: Diagrams.net



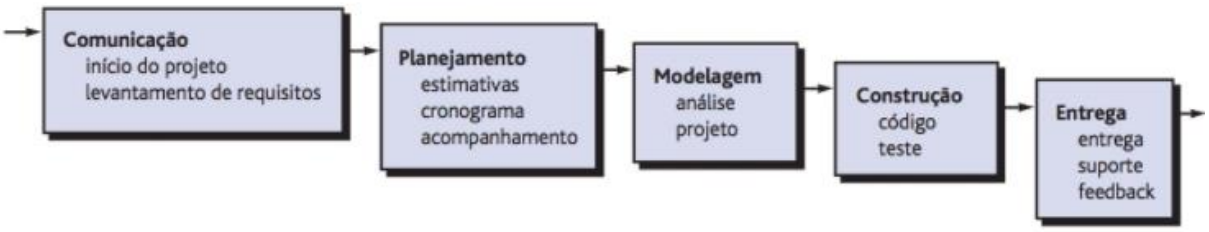
Fonte: Próprio autor, 2022.

4.6. MODELO DE DESENVOLVIMENTO CASCATA

Este modelo de processo subdivide um projeto com base nas atividades. Para construir *software* realizaremos certas atividades: análise dos requisitos, projeto, codificação e teste de acordo com (Martin,2011, p.40).

O modelo cascata por alguns chamado de ciclo de vida clássica, referente a uma abordagem sequencial e sistemática para os desenvolvedores de softwares, começando com especificação dos requisitos do cliente, avançando pelas fases de planejamento, logo após modelagem e disponibilização. De acordo com PRESSMAN (2016).

Figura 6: O processo de software



Fonte: PRESSMAN (2016)

Para Martin (2011), subdivide um projeto com base nas atividades, para a construção de um *software*, precisamos realizar certas atividades: análise dos requisitos, projeto, codificação e teste. Assim projetos que levaria um ano podem ter uma fase de análise de dois meses, seguida de uma fase projeto de quatro meses, logo após vira uma frase de codificação de três meses, e seguindo de uma fase de teste de mais três meses.

Dessa forma o modelo de processamento cascata faz uma previsão do tempo que irá gastar, faz uma estimativa do orçamento total do projeto, faz a validação e entrega o produto finalizado.

5. MATERIAIS E MÉTODOS

Para realização deste projeto, optou-se por utilizar o método de pesquisa exploratória e a modelo de desenvolvimento utilizada será cascata. Inicialmente, pretende-se fazer um levantamento dos dados mais importantes sobre o animal para armazenar.

Pretende-se utilizar a linguagem de programação (JavaScript) e biblioteca de criação de interface (React-Native) e o E*xpo* com suas API's que conversa com elementos nativos do sistema Android. A escolha levou em consideração o crescimento de sua utilização no desenvolvimento mobile pela comunidade, além de tornar possível fácil implementação em outras plataformas futuramente, decidimos utilizar o editor de código *Visual Studio Code*, pelo fato de existir vários plugins que ajuda na produção de codificação, e pelo fato de ser disponibilizado pela *Microsoft* que ajuda com uma forte usabilidade da comunidade de desenvolvedores.

A seguir apresentamos o planejamento completo da modelagem e ferramentas utilizadas para o desenvolvimento.

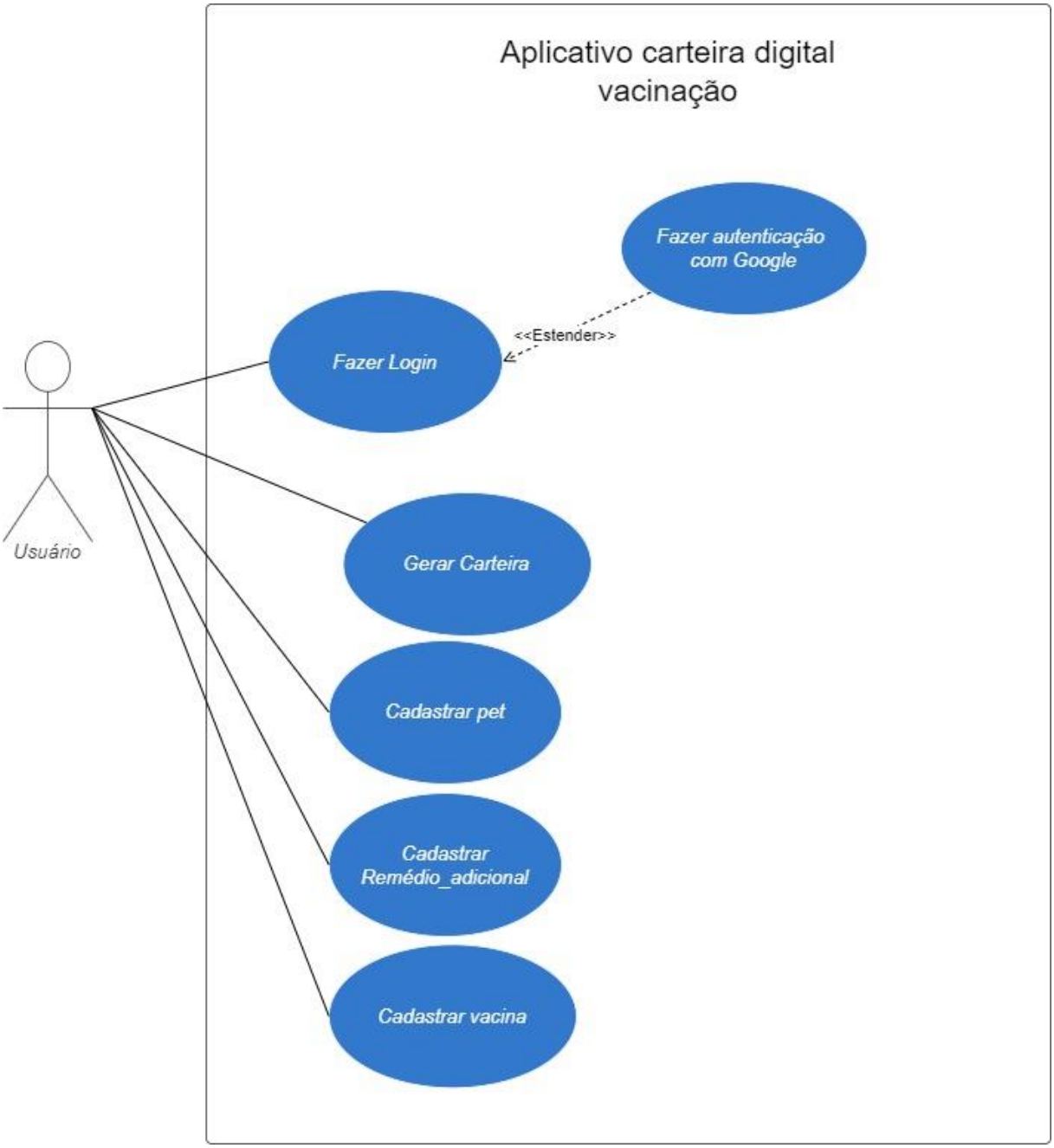
5.1. MODELAGEM DO SISTEMA

Esta seção é apresentada os diagramas de modelagem do aplicativo feito na ferramenta *Diagrams.net,* para o desenvolvimento do aplicativo mobile.

5.1.1. Casos De Uso

Na figura 7 logo abaixo é possível observar o digrama de Caso de Uso do aplicativo.

Figura 7: Casos de Uso



Fonte: Próprio autor, 2022

5.1.2. Documentação De Caso De Uso Expandido

5.1.2.1. Caso de Uso *Login*

Caso de Uso: Cadastrar Usuário

Descrição: O usuário deve cadastrar no aplicativo que ainda não consta na base de dados.

Ator Primário: Usuário

Atores Secundários: não tem.

Pré-condições: O usuário deve realizar o *login* no sistema

Fluxo Principal

1. O usuário ao momento de cadastrar pode vincular sua conta google.
2. O aplicativo vai carregar tela de *Loading* e abrirá na tela para completar o cadastro
3. O aplicativo solicita dados obrigatórios e não obrigatórios do Cliente, sendo eles Nome Completo, Data Nascimento, CPF, RG, Endereço, Sexo e Idade.
4. O aplicativo leva para o menu principal.
5. O aplicativo vai salvar no banco de dados local e depois fazer upload na conta vinculada.

5.1.2.2. Caso de Uso Pet

Caso de Uso: Cadastrar Pet.

Descrição: Cadastramento do Pet.

Ator Primário: Usuário.

Atores Secundários: Não tem.

Precondições: O usuário deve cadastrar o Pet.

Fluxo Principal

1. O usuário clica no botão +(mais) e cadastra.
2. O aplicativo vai pedir os dados no animal como: nome, idade, sexo, raça.
3. O usuário preenche os campos, o aplicativo vai pedir uma foto do animal.
4. O aplicativo vai salvar no banco de dados local e depois fazer upload na conta vinculada.

Fluxo Alternativo 1: Foto Pet

1. Na etapa 3 o usuário pode escolher tirar foto do pet ou deixar em branco.

5.1.2.3. Caso de Uso Vacina

Caso de Uso: Cadastrar vacina.
Descrição: Cadastramento da vacina.
Ator Primário: Usuário.
Atores Secundários: Não tem.
Precondições: O usuário deve cadastrar as vacinas.
Fluxo Principal

1. O usuário clica no perfil do pet.
2. O aplicativo vai expandir o perfil do animal.
3. Logo abaixo vai ter opções de cadastros como vacina, vermífugo.
4. O aplicativo vai salvar no banco de dados local e depois fazer upload na conta vinculada.

5.1.2.4. Caso de Uso Remédio Adicional

Caso de Uso: Cadastrar Remédio Adicional.
Descrição: Cadastramento de Remédio Adicional.
Ator Primário: Usuário.
Atores Secundários: Não tem.
Precondições: O usuário deve cadastrar as vacinas.
Fluxo Principal

1. O usuário clica no perfil do pet.
2. O aplicativo vai expandir o perfil do animal.
3. Logo abaixo vai ter opção de cadastrar remédio adicional.
4. O aplicativo vai salvar no banco de dados local e depois fazer upload na conta vinculada.

5.1.2.5. Caso de Uso Gerar Carteira

Caso de Uso: Gerar Carteira.
Descrição: Criando a carteira de vacinação.
Ator Primário: Usuário.
Atores Secundários: Não tem.

Precondições: O usuário tem a opção de gerar a carteira.

Fluxo Principal

1. O usuário após clicar no pet específico.
2. O aplicativo vai expandir para um ambiente mostrando um botão escrito gerar carteira.
3. Logo após gerado vai buscar os dados cadastrado da vacina e remédios adicionais.
4. O aplicativo vai salvar no banco de dados local e depois fazer upload na conta vinculada.

5.1.3. Requisitos Funcionais E Não Funcionais

5.1.3.1. Cadastrar Usuário

Tabela 3: Requisitos - Cadastrar Usuário

Requisito Funcional		
Nome: Cadastrar Usuário		Código: RF 01
Descrição: Cadastrar dados do usuário		
Estimativa de Esforço: 04h	Prioridade: Alta	
Requisito Não Funcional		
ID NF	Descrição	Categoria
1.1	As informações serão salvas vinculada na conta Google.	Especificação
1.2	Campo de preenchimento centralizado com destaque login e senha, logo abaixo o botão login, debaixo do mesmo, mais uma opção de vinculação a conta Google.	Interface
1.3	Todos os botões estarão centralizados na tela do aplicativo.	Usabilidade
1.4	Após cadastrar o usuário estará na tela principal clicando no perfil poderá cadastrar o restante dos dados faltantes.	Usabilidade

5.1.3.2. Cadastrar Vacinação

Tabela 4: Requisitos - Cadastrar Vacina

Requisito Funcional		
Nome: Cadastrar Vacinação		Código: RF 02
Descrição: Cadastrar dados da vacina		
Estimativa de Esforço: 04h	Prioridade: Alta	
Requisito Não Funcional		
ID NF	Descrição	Categoria
2.1	As informações serão salvas no dispositivo do usuário	Funcionalidade
2.2	Os preenchimentos dos dados serão feitos em modo lista, tais como nome do remédio, data vacinação, data reforço.	Interface
2.3	Terá dois botões em destaque são eles, salvar e não salvar.	Usabilidade

5.1.3.3. Remédio Adicional

Tabela 5: Requisitos - Remédio Adicional

Requisito Funcional		
Nome: Cadastrar Remédio Adicional		Código: RF 03
Descrição: Cadastrar Remédio Adicional		
Estimativa de Esforço: 04h	Prioridade: Alta	
Requisito Não Funcional		
ID NF	Descrição	Categoria
3.1	As informações serão salvas no dispositivo do usuário	Funcionalidade
3.2	Os preenchimentos dos dados serão feitos em modo lista, tais como nome do remédio, data aplicação, data reforço.	Interface

3.3	Terá dois botões em destaque são eles, salvar e não salvar.	Usabilidade

5.1.3.4. Gerar Carteira

Tabela 6: Requisitos - Gerar Carteira

Requisito Funcional		
Nome: Registrar Carteira		Código: RF 04
Descrição: Gerando Carteira de Vacinação		
Estimativa de Esforço: 03h	Prioridade: Alta	
Requisito Não Funcional		
ID NF	Descrição	Categoria
4.1	As informações serão no dispositivo do usuário	Funcionalidade
4.2	O gerenciamento da carteira terá um título escrito gerar carteira seguido logo abaixo um botão com símbolo de adição	Interface
4.3	Terá um botão em destaque para gerar a carteira.	Usabilidade

5.1.4. Diagramas De Classe

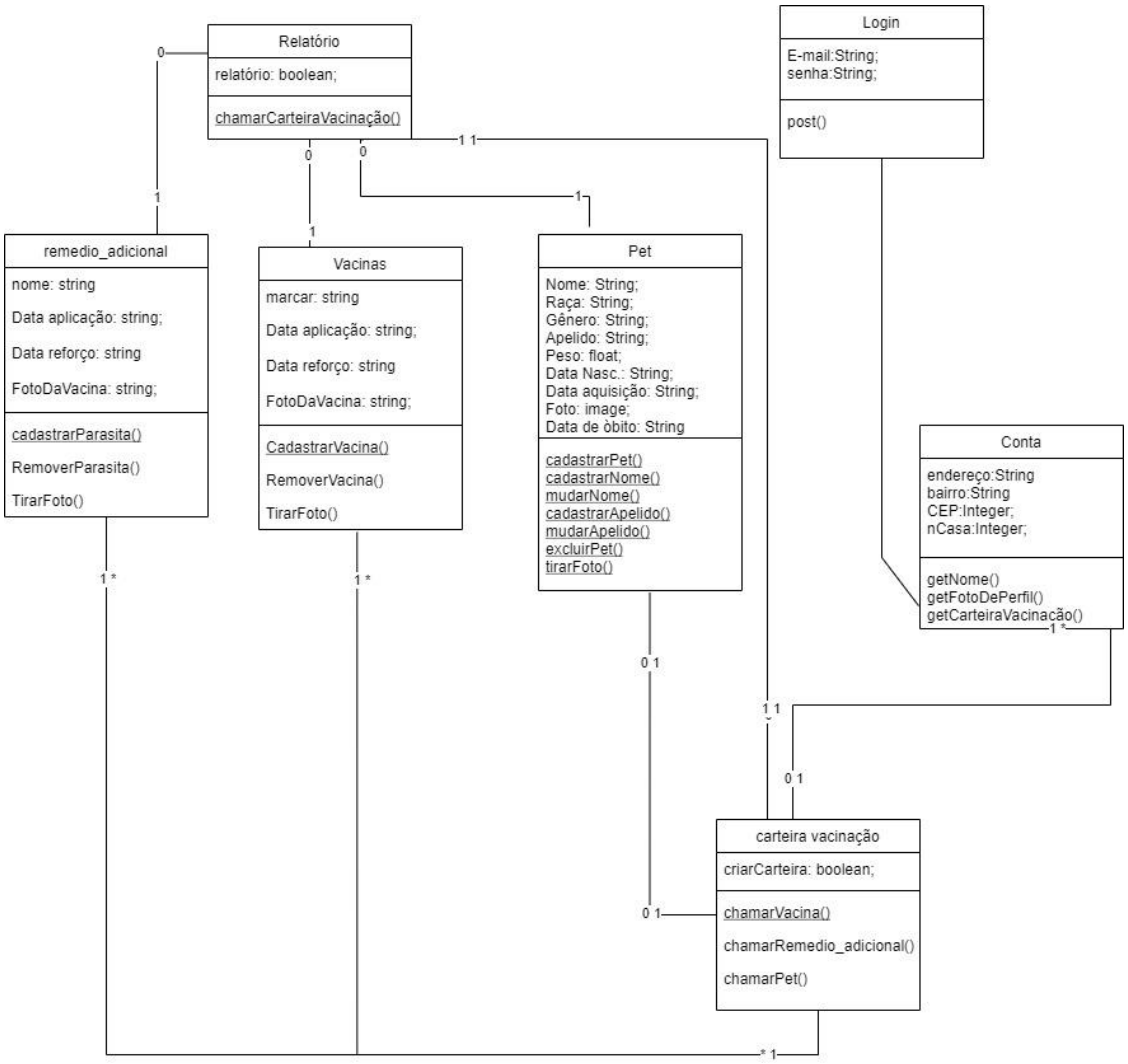
Para Guedes (2008), o diagrama de classe é, sem dúvidas, um dos principais diagramas da UML. É composto por classes que possuem atributos e métodos podendo haver relacionamentos com outras classes. O diagrama de classe visa demostrar como as classes desse diagrama se comunicam e expõem seus dados para as demais.

Segundo Silva (2007), o diagrama de classe corresponde ao um modelo gráfico de um programa orientado a objeto, descrevendo sua estrutura e os relacionamentos

entre suas classes36 de forma legível, modelo que pode ser diretamente implementado em uma linguagem de programação.

Logo abaixo na Figura 8 o diagrama de classe da aplicação.

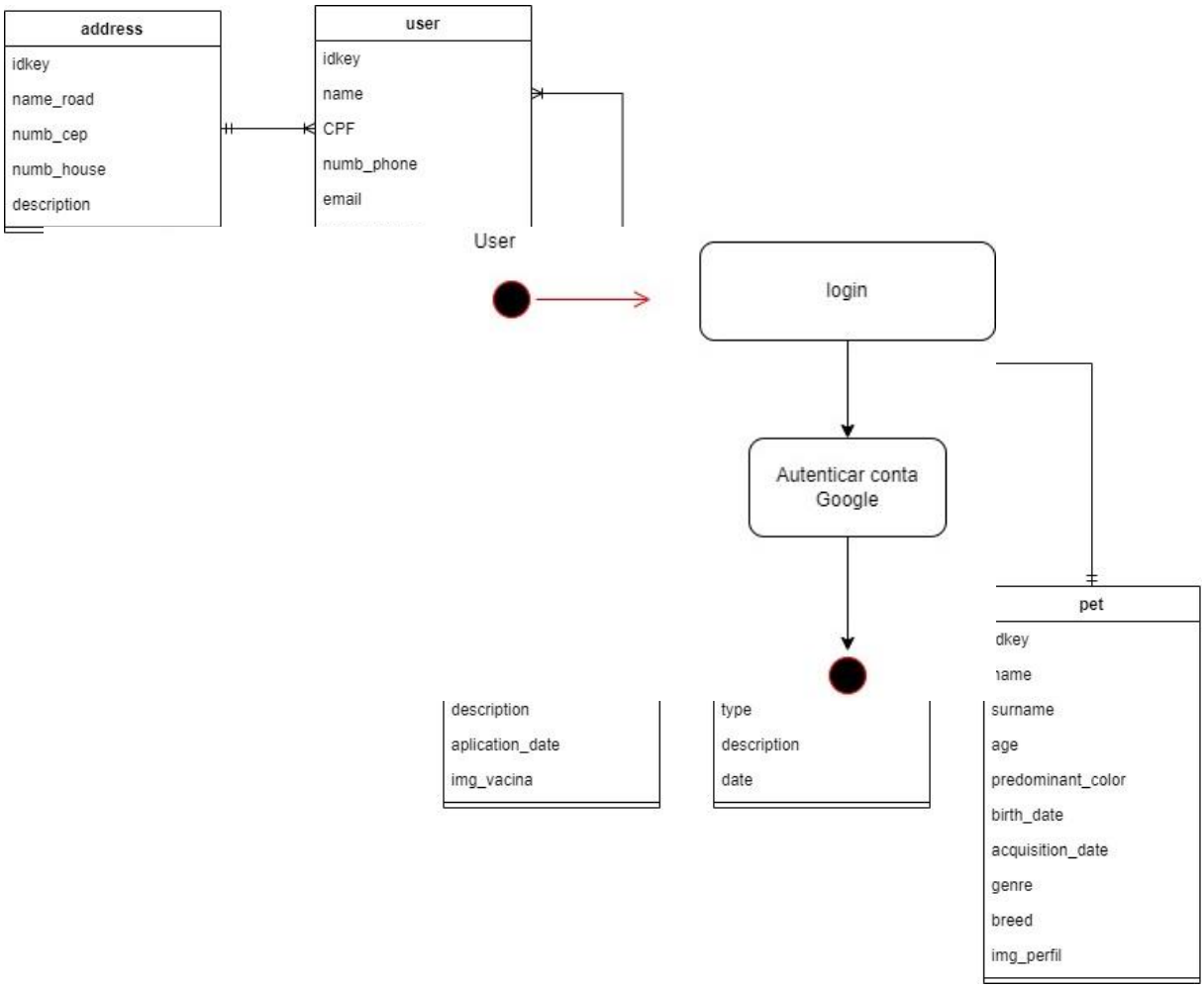
Figura 8: Diagrama de classe



Fonte: próprio autor, 2022.

Na figura 9 podemos ver a modelagem do banco de dados, a modelagem do banco é um jeito eficiente de compreendermos com clareza a relação das entidades.

Figura 9: Modelagem do banco de dados



Fonte: próprio autor, 2022.

5.1.5. DIAGRAMA DE ATIVIDADE

Diagrama de atividade é um gráfico de fluxo, nele mostra o fluxo de controle de uma atividade para outra, para serem empregados na modelagem de aspectos dinâmicos do sistema ou aplicativo.

5.1.5.1. Diagrama atividade Login

O fluxo de login é bem eficiente e simples, o usuário se autentica com a conta *Google* e pronto autenticado. Figura 10.

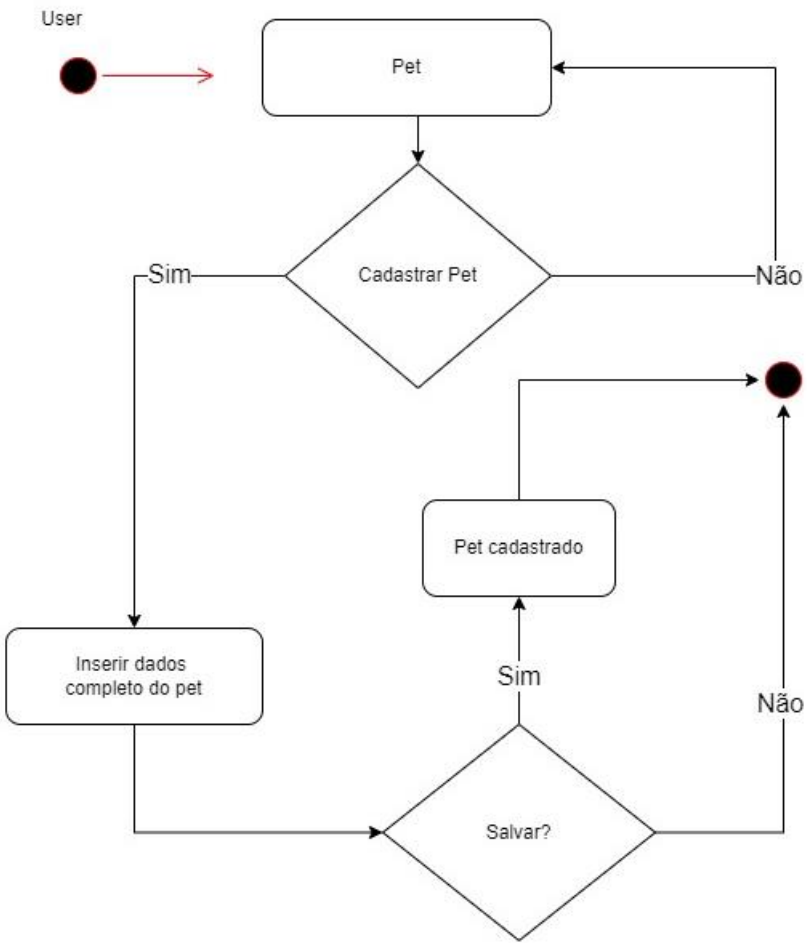
Figura 10: Diagrama atividade Login

Fonte: próprio autor, 2022.

5.1.5.2. Diagrama de atividade cadastrar pet

Diagrama de atividade cadastrar pet, nesse fluxo ocorre a seguinte sequência, cadastrar animal de estimação se sim, próximo fluxo, inserir os dados do animal, feito isso poderá salvar ou não, assim finalizando o fluxo, conforme na figura 11.

Figura 11: Diagrama de atividade cadastrar pet

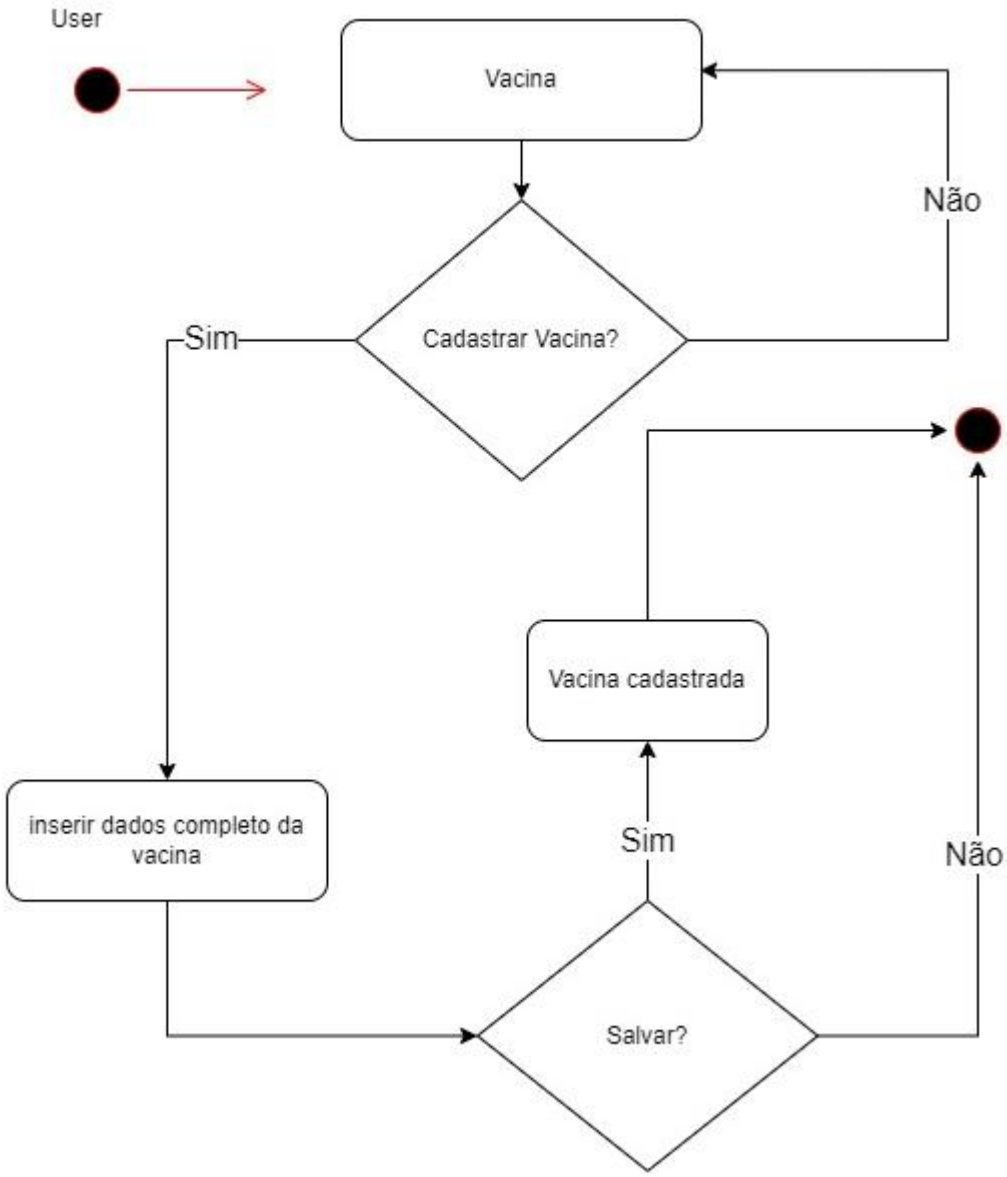


Fonte: próprio autor, 2022.

5.1.5.3. Diagrama de atividade cadastrar vacina

Diagrama de atividade cadastrar vacinas, o fluxo é bem parecido com a figura 12 terá de cadastrar vacina se sim irá para o próximo fluxo que é os dados da vacina e próximo fluxo é de salvar podendo salvar ou não, e encerrando o fluxo de cadastro de vacinas.

Figura 12: Diagrama de atividade cadastrar vacina

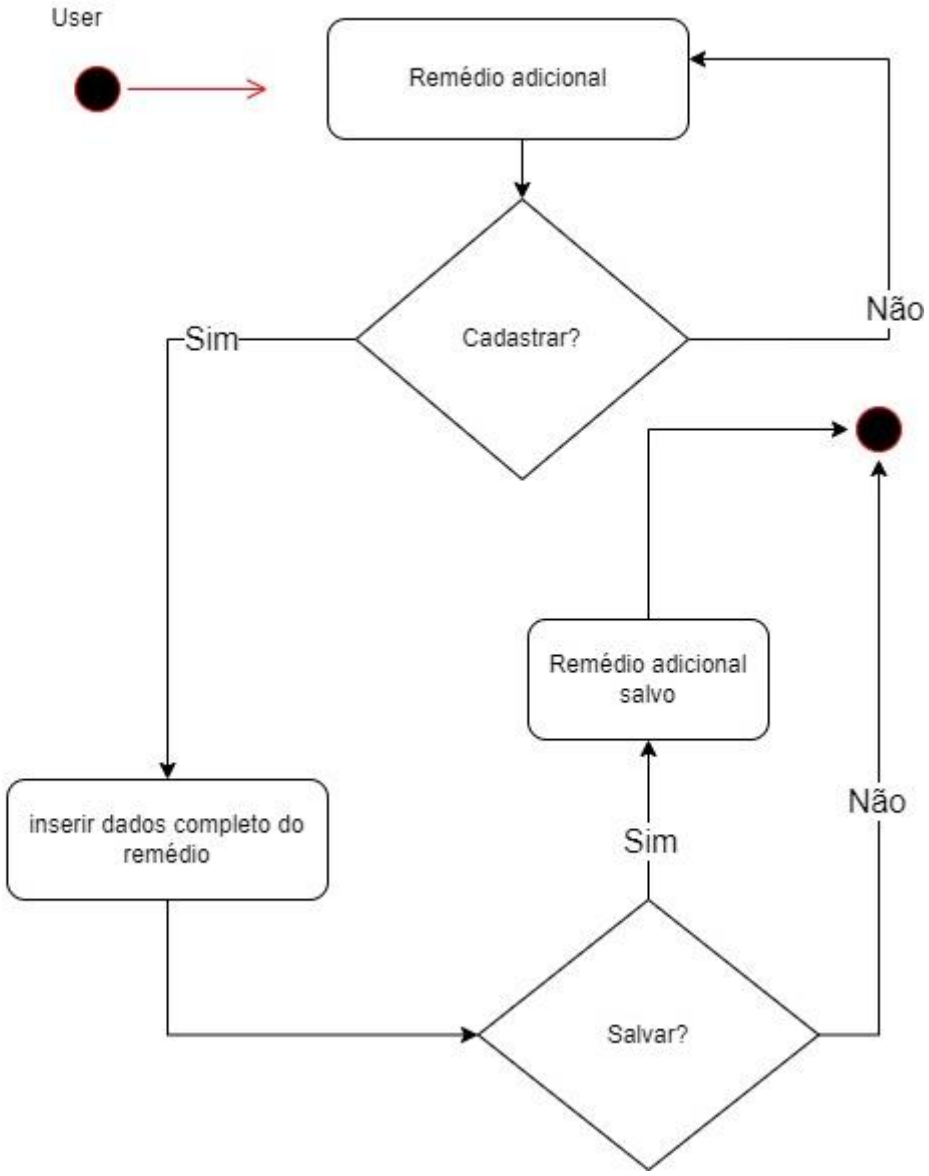


Fonte: próprio autor, 2022.

5.1.5.4. Diagrama de atividade cadastrar remédio adicional

No diagrama de cadastrar remédio o fluxo segue-se em cadastrar se sim ir para próximo passo informações do remédio feito isso ou não terá a opção de salvar ou não

Figura 13: Diagrama de atividade cadastrar remédio adicional

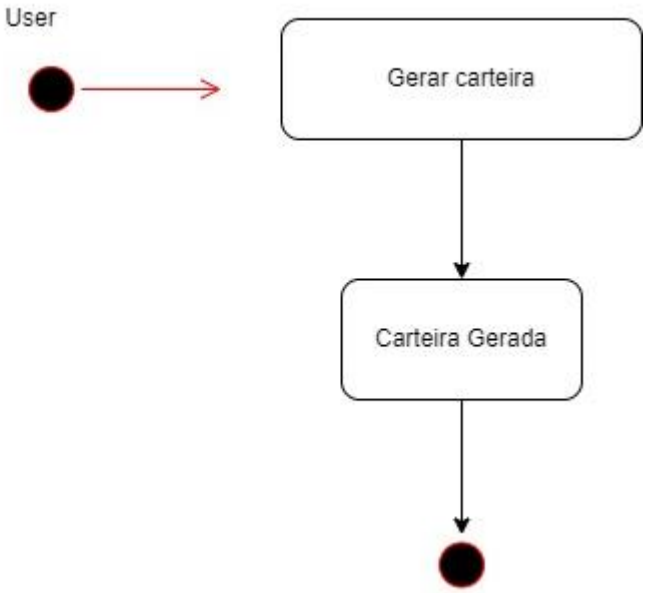


Fonte: próprio autor, 2022.

5.1.5.5. Diagrama de atividade gerar carteira

Diagrama de gerar carteira tem um fluxo bem direto que é gerar uma carteira no fluxo e sair logo em seguida, nesse caminho vai pegar toda informação do tutor e do pé.

Figura 14: Diagrama de atividade gerar carteira



Fonte: próprio autor, 2022.

5.2. MATERIAIS

Nesta seção vamos apresentar uma tabela com matérias usando nessa produção do aplicativo, está tabela contém ferramenta, versionamento da ferramenta e finalidade

Tabela 7 – Ferramentas / Tecnologias

Ferramenta / Tecnologia	Versão	Finalidade

Expo	V0.70	API’s que conversam nativamente com o sistema Android e seus componentes e *hardware* do celular.
Android-Sdk	2020.3	Permite utilizar emulador de Android.
React-Native	0.61	Ferramenta que permite criação de telas UI para Android.
NodeJs	18.12.1	Executar npm (9.1.2) e executar o projeto.
Diagrams.net	Não definida pelo desenvolvedor	Ferramenta utilizada para criação de diagramas
Figma	Não definida pelo desenvolvedor	Ferramenta utilizada para criação das telas
Visual Studio Code	1.67	Ferramenta editor código fonte
GitHub	Não definida pelo desenvolvedor	Versionamento do código

6. DESENVOLVIMENTO

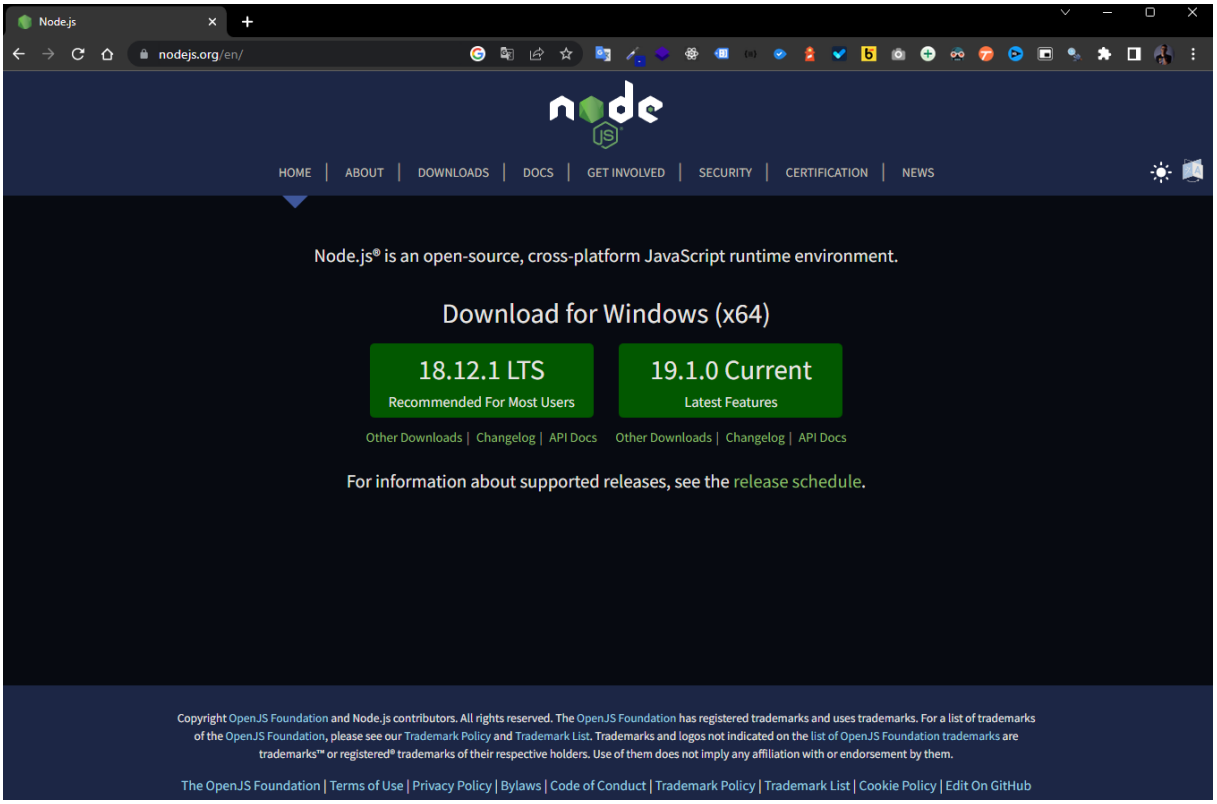
De acordo com os objetivos estabelecidos do presente trabalho, faz-se necessário o desenvolvimento das funcionalidades descritas nas seções anteriores. Para tanto iremos demonstrar como realizar a configuração do ambiente de desenvolvimento do projeto, interfaces de comunicação, e as funcionalidades e telas do aplicativo desenvolvido.

6.1. Configuração do Projeto

Para iniciarmos o projeto utilizando React-Native devemos ter instalado na nossa máquina o NodeJS para executar os comandos necessários. No site oficial do *NodeJS* pelo caminho da web: "*https://nodejs.org/en/*" existe duas opções para baixar,

Figura 15: Imagem do site nodejs

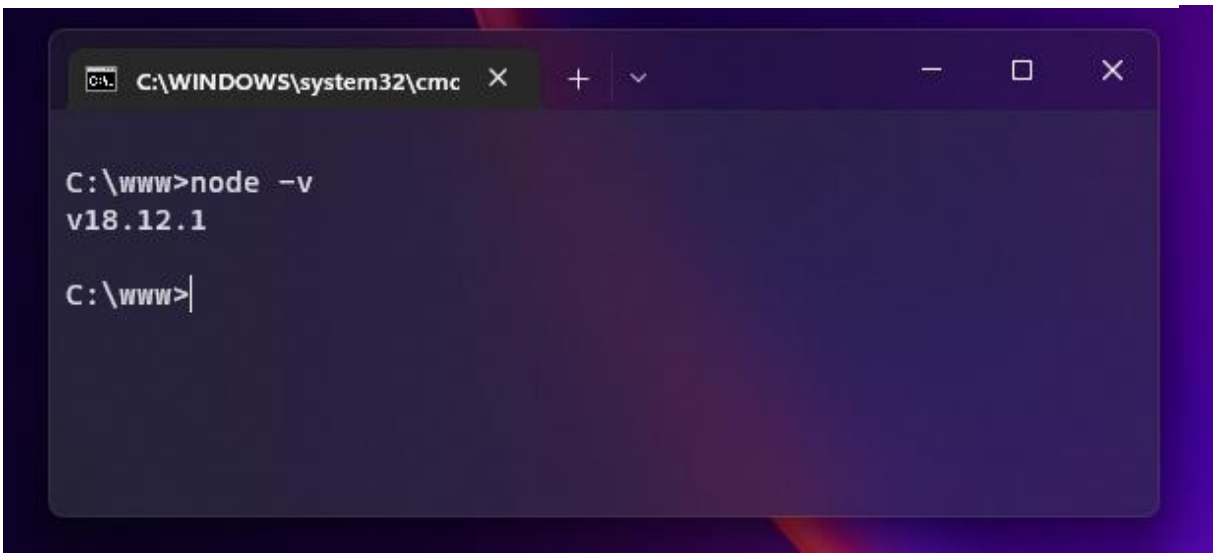
vamos utilizar a versão estável do programa que tem a sigla (LTS) significa versão estável ou seja bug corrigidos, conforme a figura



Fonte: próprio autor, 2022.

Feito isso, abre seu *Prompt* comando e certifica-se que a instalação do *NodeJS* está rodando com o comando “node -v”

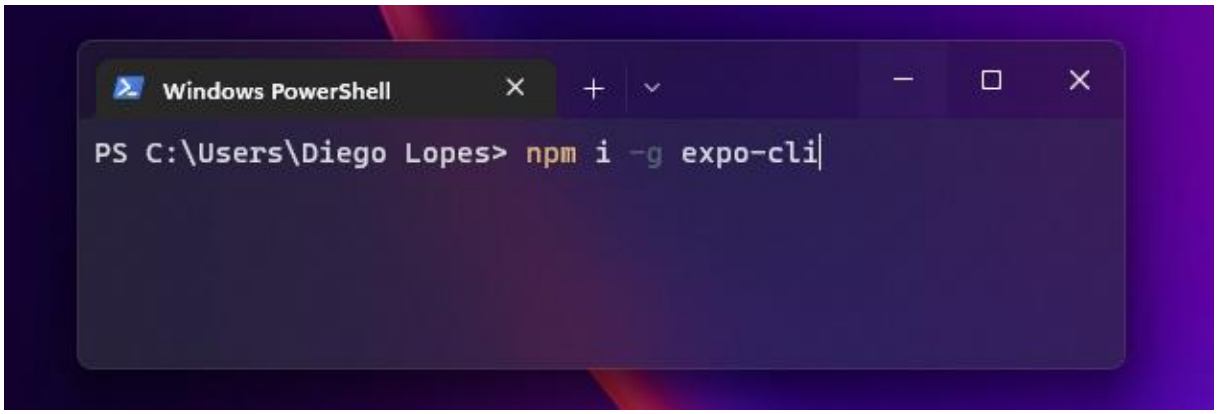
Figura 16: CMD Node version



Fonte: próprio autor, 2022.

Se aparecer a versão do node quer dizer que está funcionando corretamente. Abre o *Prompt* de comando CMD do *Windows* e digite:

Figura 17: Instalação pacote CLI:

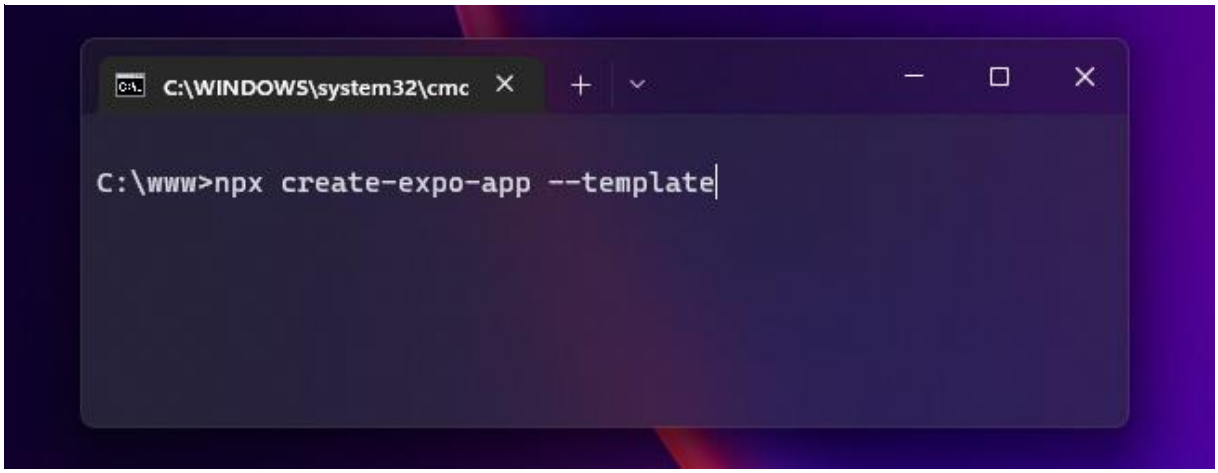


Fonte: próprio autor, 2022.

Npm é um gerenciador de pacotes nativo do *NodeJS,* i é abreviação de *Install* (instalar em português), -g é uma *Flag* indicando que é *Global Expo-cli* é o pacote que vamos instalar na nossa máquina.

Agora que está instalado, vamos criar nosso projeto, usamos o comando logo abaixo.

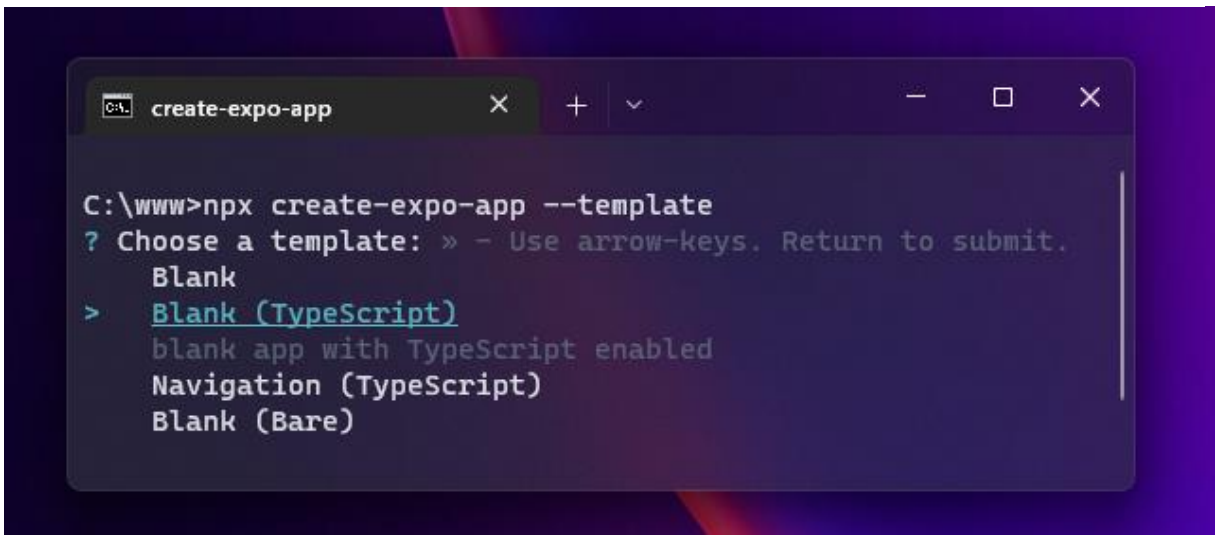
Figura 18: Criar projeto



Fonte: próprio autor, 2022.

Agora usaremos *NPX*, *npx* é usado para chamar o programa e executar, ou seja, estamos chamando *create-expo-app* para criar um projeto *expo,* em seguida usamos ao *Flag --template* para escolher o tipo de projeto modelo pré-definido.

Figura 19: Expo teamplate

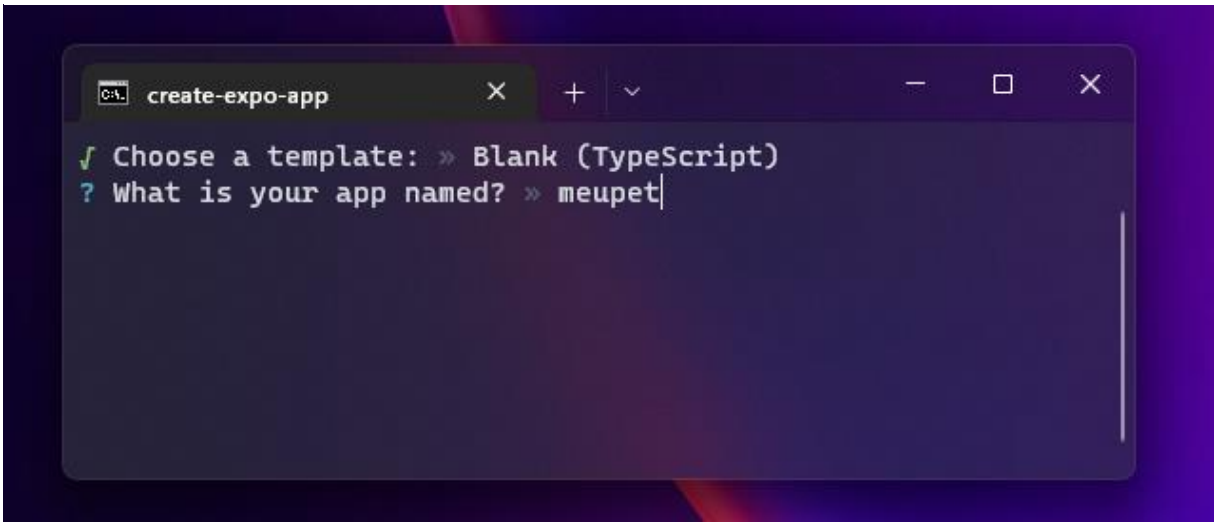


Fonte: próprio autor, 2022.

Vamos usar um modelo em B*lank* (em branco) só que com *TypeScript,* com esse *Superset,* conseguimos dar um tipo pré-definido para as variáveis assim

conseguimos evitar muitos erros, pois o próprio *JavaScript* é uma linguagem de baixa tipagem.

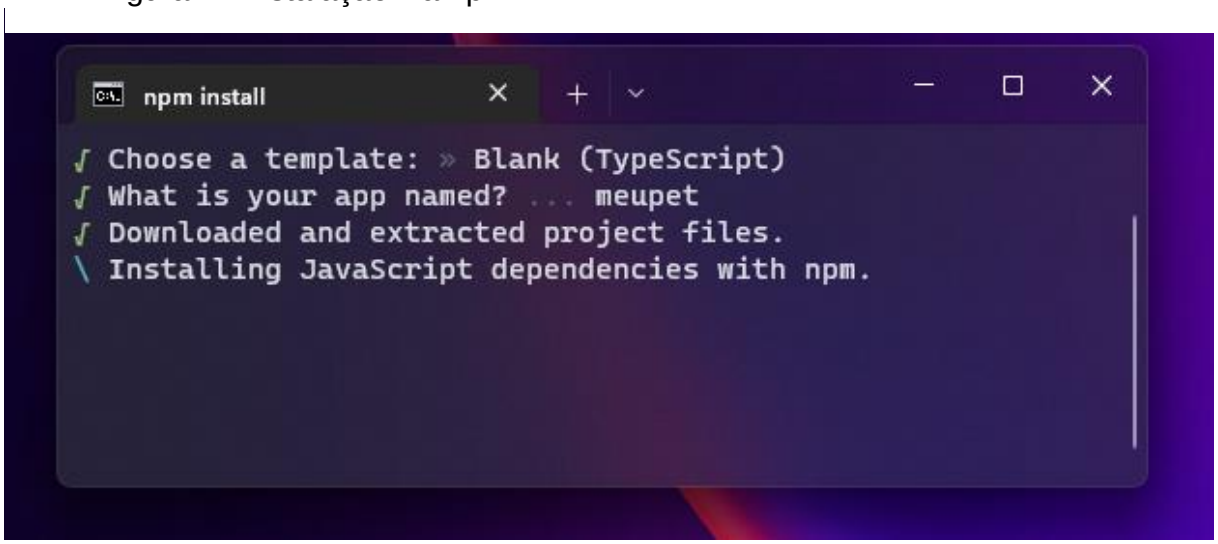
Figura 20: Escolha de teamplate



Fonte: próprio autor, 2022.

Agora colocamos o nome do projeto que chamamos de meupet, dê um *Enter* do teclado e vai começar a fazer o *Download* dos arquivos.

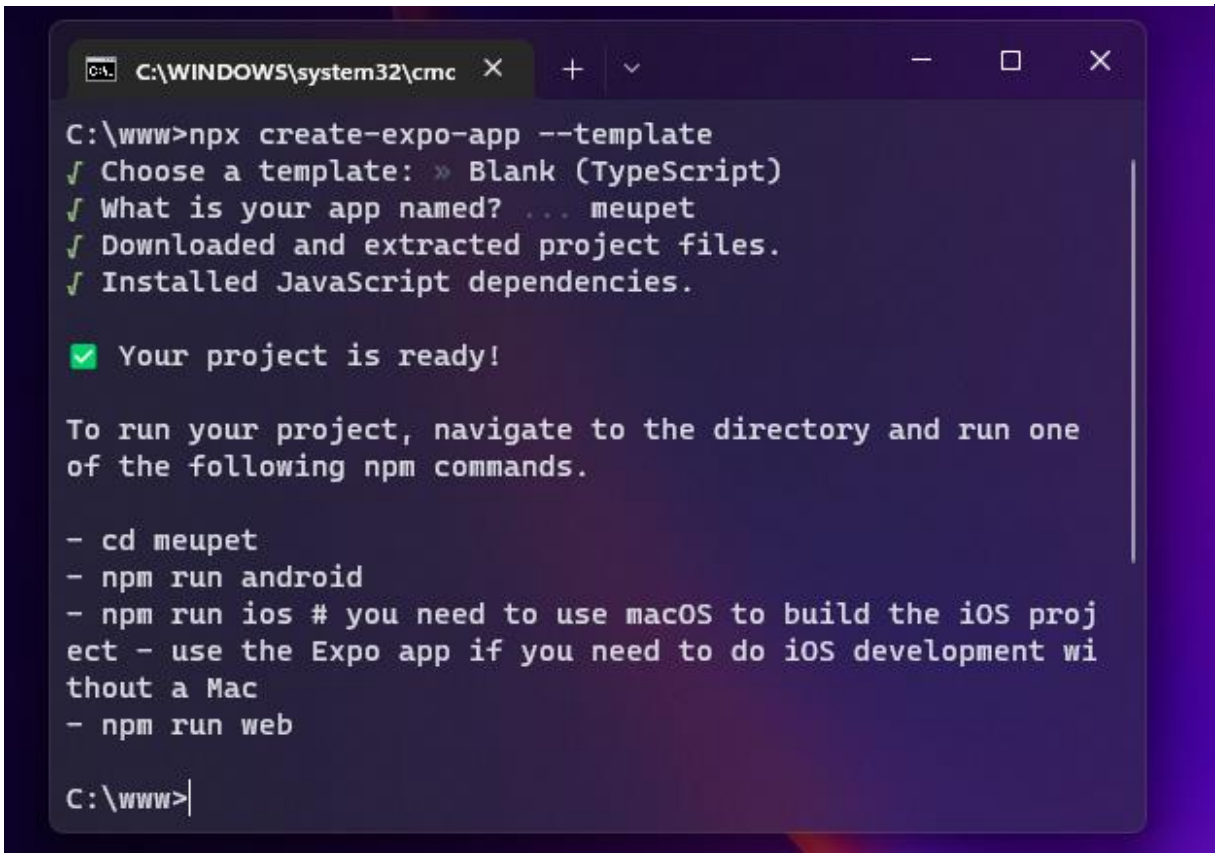
Figura 21:Instalação via npm



Fonte: próprio autor, 2022.

Aguardamos terminar o *Download* dos arquivos e quando terminar a tela ficará assim:

Figura 22: project ready



Fonte: próprio autor, 2022.

Pronto estamos com projeto feito, agora vamos abrir no *VSCode* o projeto logo abaixo mostraremos a estrutura do projeto já finalizado.

7. FUNCIONALIDADE DO APLICATIVO

Nesse tópico descrevemos as telas e funcionalidades do aplicativo.

Tela login.

Figura 23: Tela de login

Fonte: próprio autor, 2022.

Tela de início, logo abaixo temos o botão de login com *Google*, botão de autenticação.

Tela de Carregamento

Neste ponto após a confirmação da autenticação com conta *Google* entra na tela de carregamento aguardando o *call-back* de confirmação, após isso segue para tela de *Home*.

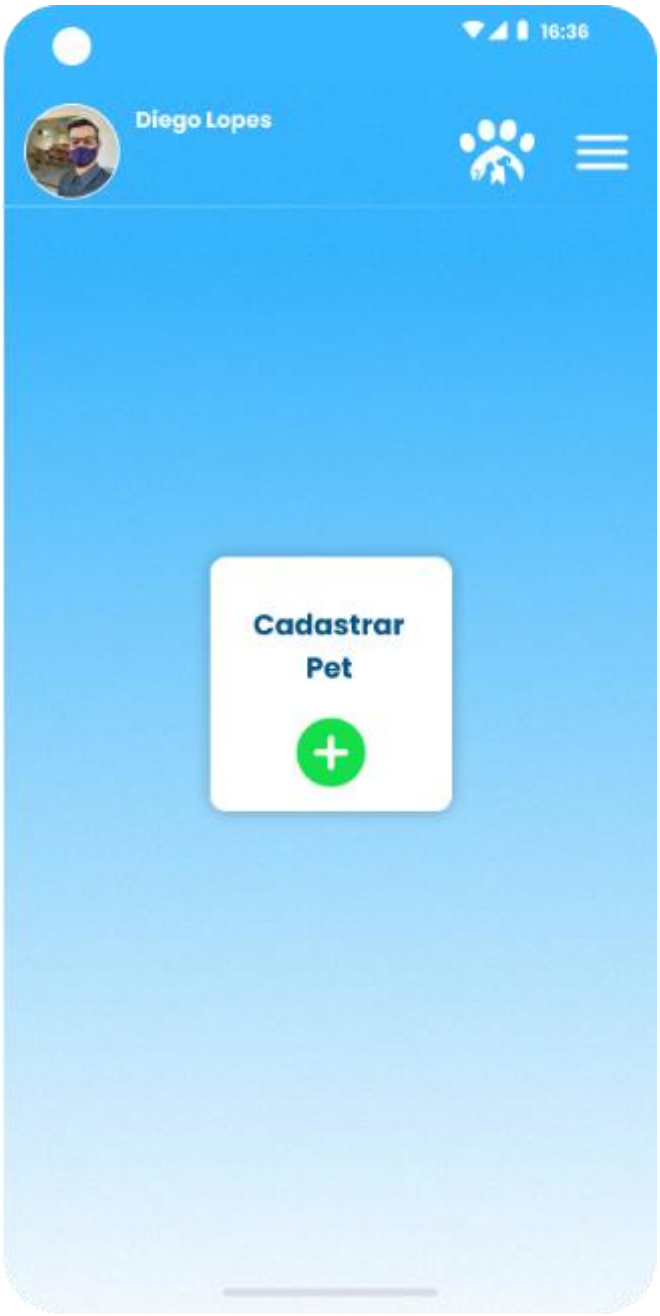
Figura 24: Tela de carregamento

Fonte: próprio autor, 2022.

Tela Home

Nesta parte já temos alguns dados fornecido pelo *Google* que é foto, imagem e endereço de *E-mail.*

Figura 25: Tela home/principal



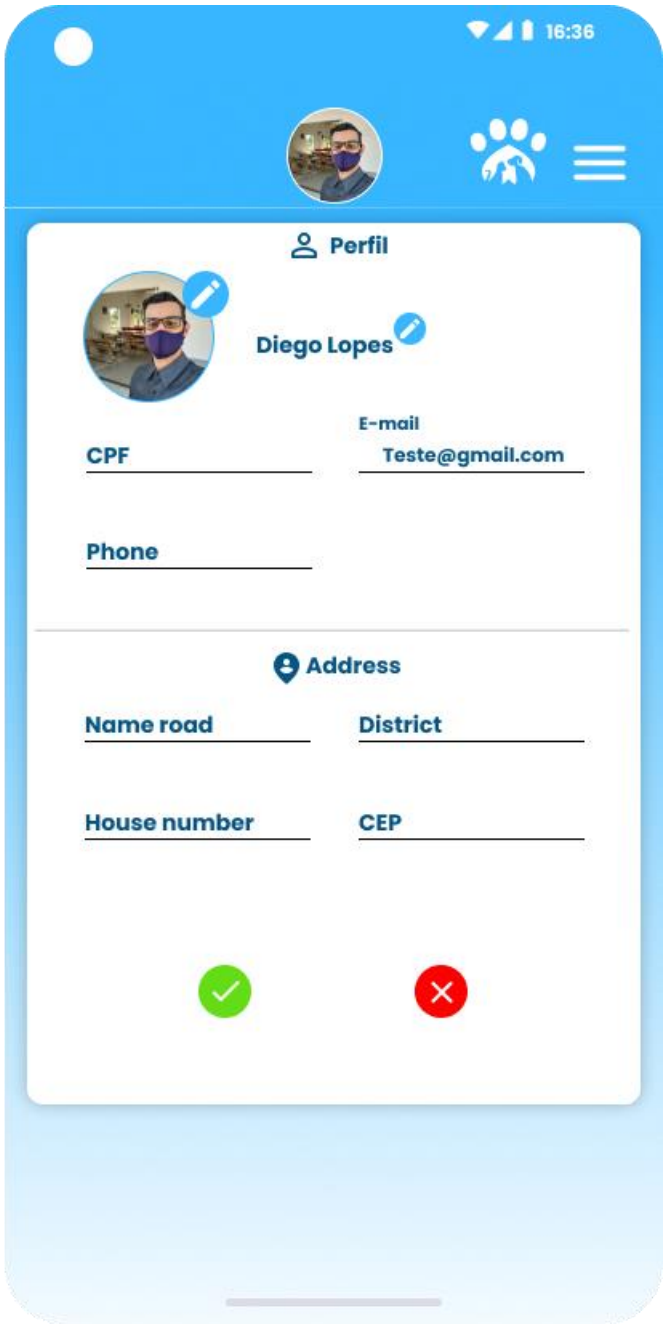
Fonte: próprio autor, 2022.

Na página *Home* temos algumas informações direta cadastro de um animal de estimação, e perfil do usuário.

Tela de perfil usuário

Clicando no ícone do usuário, temos algumas opções adicionais para serem preenchidos.

Figura 26: Tela de perfil usuário



Fonte: próprio autor, 2022.

Tela cadastrar animal de estimação

Neste ponto temos as telas de registro de animal de estimação, temos os campos para serem preenchidos com a informação dele, com possível adição de imagem do animal.

Figura 27: Tela de cadastrar animal

Fonte: próprio autor, 2022.

Tela com animal de estimação cadastrado

Logo após a criação do registro, a tela home muda de estado que se adapta de acordo com entrada de registro de animais, nestas imagens vemos dois casos quando temos apenas um animal registrado e quando temos vários.

O cartão de apresentação do registro apresenta uma imagem, nome, peso e idade. Pontos relevante de fácil acesso, caso tenha que responder ao veterinário ou outras pessoas.

Figura 28: Tela com animal estimação

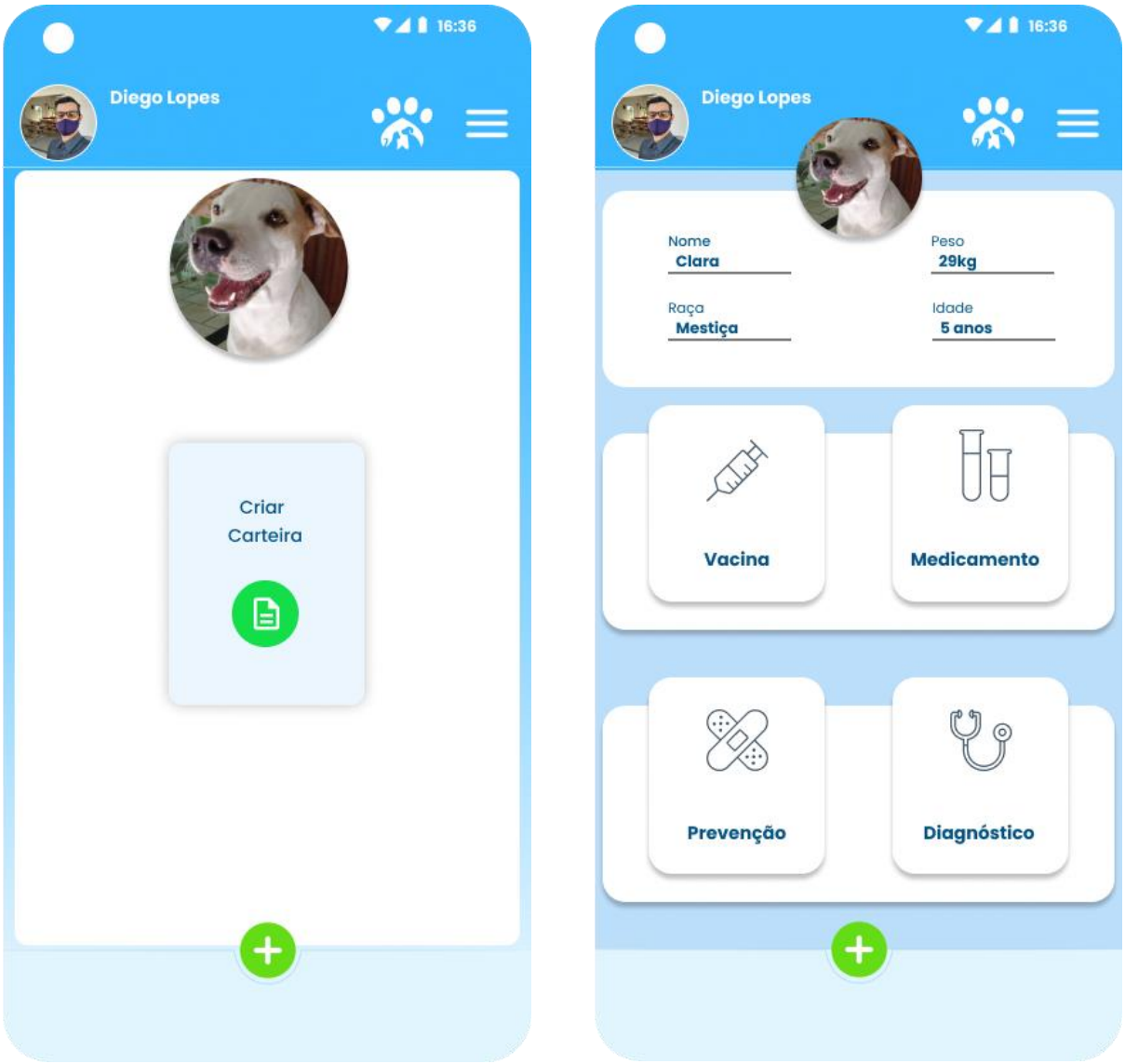
Fonte: próprio autor, 2022.

Tela da carteira de vacina

Ao tocar no cartão do animal de estimação abre mais uma opção que é a carteira de vacinação, este campo ao ser clicado gera a carteira, buscando os dados base do animal.

E abre mais opções para serem preenchida com passar do tempo, tais como: vacinação, medicamento, prevenção, diagnóstico.

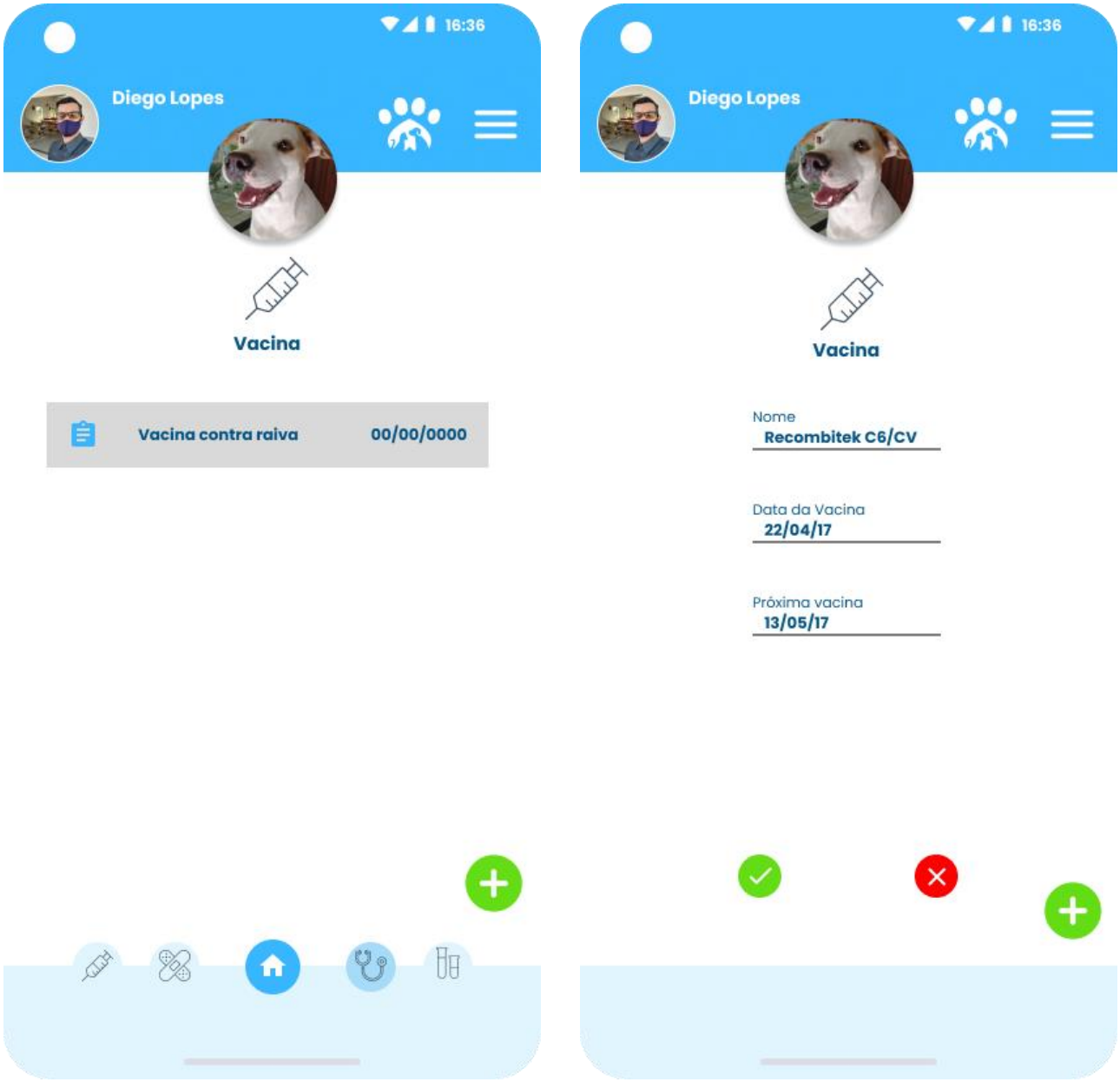
Figura 29: Carteira de vacina



Fonte: próprio autor, 2022.

Tela de cadastro de vacina e lista de vacina

Figura 30: Tela de cadastro de vacina e lista de vacina



Fonte: próprio autor, 2022.

8. CONSIDERAÇÕES FINAIS.

O propósito deste trabalho foi desenvolver um aplicativo mobile para gerenciar a carteira de vacinação, o usuário poderá navegar entre seus animais de estimação cadastrado no aplicativo e para cada um terá sua carteira com todos os seus dados do animal separado e organizado, a carteira terá categorias que facilita a identificação de certos remédios e prevenções aplicada no animal.

O conteúdo aqui apresentado como base teórica foi utilizado para modelagem do aplicativo por meio de diagrama UML, no qual descrevemos o caso de uso, especificamos os requisitos funcionais, criamos o diagrama de classe que foram utilizados para modelagem do projeto, utilizamos *Figma* para criação das interfaces para fácil programação.

Escolhemos o *React-Native* pois ele traz a possibilidade de mesmo código JavaScript pode ser compilado para outra plataforma mobile como o IOS. As modificações para outra plataforma é pouca isso possibilita uma escalabilidade alta. E reparo no código em um só ponto.

Há ideia dessa aplicação é ser gratuita para facilitar a propagação de usuários no aplicativo, e ser Open-Source para colaboração de desenvolvedores e ter um canal para receber doações para continuação do projeto. Também terá a parte de anúncio para o projeto autossustentar.

REFERÊNCIAS

Android. Android. 2022. Disponível em: https://www.android.com. Acesso em: 24 mai. 2022.

Arca Brasil. Vacinação nos cães. Arca Brasil. Disponível em: https://arcabrasil.org.br/vacinacao-caes/ Acesso em: 5 mai. 2022.

Android Developers. Disponível em: <https://developer.android.com/guide/platform?hl=pt-br>. Acesso em: 28 nov. 2022.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Elsevier, f. 185, 2006. 369 p.

CAETANO, ELAINE CRISTINA SALVARO. **AS CONTRIBUIÇÕES DA TAA**: TERAPIA ASSISTIDA POR ANIMAIS À PSICOLOGIA. criciúma, 2010, p. 14-29. Trabalho de Conclusão de Curso (Psicologia) - Universidade do Extremo Sul Catarinense, Criciúma, 2010.

COUTO, Lucas Silva; DOMINGUES, Marcos Aurélio. UM APLICATIVO MOBILE PARA EXPLORAÇÃO DE RELACIONAMENTOS DE ARTISTAS EM REDES DE MÚSICAS.

FERREIRA, Hélder. **Draw.IO: Desenhar diagramas nunca foi tão fácil**. pplware. Disponível em: https://pplware.sapo.pt/internet/draw-io-desenhar-diagramas-nunca-foi-tao-facil/. Acesso em: 25 mai. 2022.

FLANAGAN, D. **JavaScript:** O guia definitivo. Tradução de João Eduardo Nóbrega Tordello. 6. ed. Porto Alegre: Bookman, 2013. Tradução de: JavaScript: The Definitive Guide

FOWLER, Martin. **UML Essencial**: Um Breve Guia para Linguagem Padrão de modelagem de objetos. Tradução Addison Wesley Professional. 3. ed. Bookman, 2004. Tradução de: UML Distilled: A Brief Guide to the Standard Object Modeling Language.

GARGENTA, Marko; NAKAMURA, Masumi. **Learning Android**: Develop Mobile Apps Using Java and Eclipse. Sebastopol: O'Reilly Media, 2014.

Globo. **Brasileiros têm 52 milhões de cães e 22 milhões de gatos, aponta IBGE**. G1. São Paulo, 2015. Disponível em: http://g1.globo.com/natureza/noticia/2015/06/brasileiros-tem-52-milhoes-de-caes-e-22-milhoes-de-gatos-aponta-ibge.html. Acesso em: 26 mar. 2022.

GUEDES, G. T. **UML**: Uma abordagem prática. [S.l.]: Novatec Editora, 2008.

GUEDES, G. **UML 2** – Uma Abordagem Prática 2 ed. São Paulo: Novatec Editora Ltda. 2011. 45 pag.

JUSTINA, Della; ELENA, Eloiza. **Zoneamento geoambiental da reserva biológica do Jaru e zona de amortecimento - RO, como subsídio ao seu plano de manejo**. Rio Claro, 2009. Tese (Geociências e Educação Ambiental) - Universidade Estadual Paulista Júlio de Mesquita Filho, Rio Claro, 2009. Disponível em: https://repositorio.unesp.br/handle/11449/102891. Acesso em: 5 abr. 2022.

Visual Studio Code. **Getting Started**. Visual Studio Code. Disponível em: https://code.visualstudio.com/docs. Acesso em: 21 mai. 2022.

Mundo Animal. **Calendário de Vacinação em 2021**. Mundo animal. Disponível em: . Acesso em: 26 mai. 2021.

NICHELLE, ALINE MARIEL. **APLICATIVO MÓVEL PARA GERENCIAMENTO DE DADOS DE ANIMAIS DE ESTIMAÇÃO**. Pato Branco. 17 p. Trabalho de Conclusão de Curso (Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná, Pato Branco.

PINTO, Pedro**. Visual Studio Code: O melhor editor para programadores?**. pplware. 2017. Disponível em: https://pplware.sapo.pt/software/visual-studio-code-melhor-editor-programadores/. Acesso em: 12 mai. 2022.

PRESSMAN, Roger; MAXIN, Bruce. **Engenharia de Software**: Uma abordagem Profissional. Tradução João Eduardo Nóbrega Tortello. 8. ed. Porto Alegre: AMGH, 2016. Tradução de: Software Enginering: A Practitioner's Approach, 8th Edition.

REACTNATIVE, React Native.2022. Disponivel em: https://reactnative.dev/. Acesso em 20 nov. 2022.

SADALAGE, Pramod J.; FOWLER, Martin. **NoSQL Distilled**: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2012.

SAMAT, Sameer. **Android 12 Beta: Designed for you**. Google The Keyword. 2021. Disponível em: https://blog.google/products/android/android-12-beta. Acesso em: 25 mai. 2022.

SANTOS, Jocelaine. **Cinco benefícios que animais de estimação trazem à família**. Gazeta Do Povo. 2015. Disponível em: https://www.semprefamilia.com.br/pets/cinco-beneficios-que-animais-de-estimacao-trazem-a-familia/. Acesso em: 16 mai. 2022.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **FUNDAMENTOS DE SISTEMAS OPERACIONAIS**. Tradução Aldir José Coelho Correa da Silva. LTC, 2015. Tradução de: OPERATING SYSTEM CONCEPTS.

SILVA, R. P. E. **UML 2**: modelagem orientada a objetos. [S.l.]: Visual Books, 2007.

Sirius Interativa. **Figma: uma nova ferramenta para design de interface que está ganhando o mercado | Sirius Interativa**. Medium. 2019. Disponível em: https://medium.com/@Sirius_/figma-uma-nova-ferramenta-para-design-de-

interface-que-est%C3%A1-ganhando-o-mercado-sirius-interativa-2e78e0905b44. Acesso em: 5 jun. 2022.

STEELE, Colin. **mobile operating system**. TechTarget. 2020. Disponível em: https://searchmobilecomputing.techtarget.com/definition/mobile-operating-system. Acesso em: 28 mai. 2022.

TANENBAUM, Andrew S.; BOS, Hebert. **Sistemas operacionais modernos**. Tradução Daniel Vieira e Jorge Ritter. 4. ed. São Paulo: Person, 2016. Tradução de: Modern operating systems.

VALENTE, Jonas. **Brasil é 5° país em ranking de uso diário de celulares no mundo**. Agência Brasil. 2019. Disponível em: https://agenciabrasil.ebc.com.br/geral/noticia/2019-01/brasil-foi-5o-pais-em-ranking-de-uso-diario-de-celulares-no-mundo. Acesso em: 26 mar. 2022.

9. Apêndice

9.1. Código do Projeto

Logo abaixo colocaremos o código do projeto, ele será subdivido por cada tela, demostrando com é uma estrutura de um aplicativo feito com JavaScript. Logo abaixo estar os códigos fontes das telas propostas no projeto.

9.1.1. Tela Inicial

```
import React from "react";

import { LinearGradient } from "expo-linear-gradient";
import { FontAwesome5 } from '@expo/vector-icons'
import { useNavigation } from "@react-navigation/native";

import { styles } from "./styles";

import { ButtonFlutuante } from "../../components/buttonFlutuante";
import { Header } from "../../components/header";

export function Home() {
 const navigation = useNavigation();
 return (
  <LinearGradient colors={["#38B6FF", "#FFF"]} style={styles.container}>
   <Header />

   {/* Button add pet */}
   <ButtonFlutuante
    icon={<FontAwesome5 name="plus" size={24} color="white" />}
    onPress={() => navigation.navigate('Register')}
   />
  </LinearGradient>
```

```
  );
}
```

Fonte: próprio autor, 2022.

9.1.2. Tela de registro

Tela de registro é bem grande, pois ela contém a maior parte pois contém um formulário bem grande.

```
import React, { useState, useEffect } from "react";
import { ImageBackground, View, TouchableOpacity } from "react-native";

import { LinearGradient } from "expo-linear-gradient";

import { styles } from "./styles";
import { Header } from "../../components/header";
import { Input } from "../../components/input";
import { Button } from "../../components/button";

import { Feather, MaterialIcons } from "@expo/vector-icons";
import { useNavigation } from "@react-navigation/native";
import AsyncStorage from "@react-native-async-storage/async-storage";
import * as ImagePicker from 'expo-image-picker';

export function Register() {
  const [nome, setNome] = useState("");
  const [raca, setRaca] = useState("");
  const [genero, setGenero] = useState("");
  const [peso, setPeso] = useState("");
  const [apelido, setApelido] = useState("");
  const [dataNascimento, setDataNascimento] = useState("");
  const [dataAquisicao, setDataAquisicao] = useState("");
  const [corPredominante, setCorPredominante] = useState("");
  const [image, setImage] = useState(null);
  const [pets, setPets] = useState([]);
```

```
const navigation = useNavigation();

useEffect(() => {
  loadPetsRegiters();
}, []);

const loadPetsRegiters = async () => {
  const petsRegisters = await AsyncStorage.getItem("@pet");
  if (petsRegisters !== null) {
    setPets(JSON.parse(petsRegisters));
  }
};

const onHandleContinue = async () => {
  const data = {
    nome,
    raca,
    genero,
    peso,
    apelido,
    dataNascimento,
    dataAquisicao,
    corPredominante,
    image
  };

  const jsonValue = JSON.stringify([...pets, data]);
  await AsyncStorage.setItem("@pet", jsonValue);
  navigation.navigate("List");
};

const pickImage = async () => {

  let result = await ImagePicker.launchImageLibraryAsync({
```

```
    mediaTypes: ImagePicker.MediaTypeOptions.All,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
   });

   if (!result.cancelled) {
    setImage(result.uri);
   }
  };

  return (
   <LinearGradient colors={["#38B6FF", "#FFF"]} style={styles.container}>
    <Header />

    <View style={styles.content}>
     <View style={styles.modal}>
      {/* Image */}
      <ImageBackground source={{ uri: image }} style={styles.image}>
       <TouchableOpacity style={styles.editButton} onPress={pickImage}>
        <MaterialIcons name="edit" size={24} color="white" />
       </TouchableOpacity>
      </ImageBackground>
      <View style={styles.form}>
       <View style={styles.inputRow}>
        <Input placeholder="Nome" onChangeText={setNome} value={nome} />
        <Input placeholder="Raça" onChangeText={setRaca} value={raca} />
       </View>
       <View style={styles.inputRow}>
        <Input
         placeholder="Gênero"
         onChangeText={setGenero}
         value={genero}
        />
        <Input placeholder="Peso" onChangeText={setPeso} value={peso} />
```

```
</View>
<View style={styles.inputRow}>
 <Input
  placeholder="Apelido"
  onChangeText={setApelido}
  value={apelido}
 />
 <Input
  placeholder="Data de nascimento"
  onChangeText={setDataNascimento}
  value={dataNascimento}
 />
</View>
<View style={styles.inputRow}>
 <Input
  placeholder="Data da aquisição"
  onChangeText={setDataAquisicao}
  value={dataAquisicao}
 />
 <Input
  placeholder="Cor predominante"
  onChangeText={setCorPredominante}
  value={corPredominante}
 />
</View>

<View style={styles.buttonContainer}>
 <Button
  onPress={onHandleContinue}
  icon={<Feather name="check" size={24} color="white" />}
 />
 <Button
  color="error"
  icon={<MaterialIcons name="clear" size={24} color="white" />}
 />
```

```
        </View>
       </View>
      </View>
     </View>
   </LinearGradient>
  );
}
```

Fonte: próprio autor, 2022.

9.1.3. Tela de carteira criação de carteira de vacinação

```
import React from "react";
import { View, Image, TouchableOpacity, Text } from "react-native";

import { LinearGradient } from "expo-linear-gradient";
import { useNavigation, useRoute } from "@react-navigation/native";

import { Header } from "../../components/header";

import { styles } from "./styles";

import DocumentSvg from "../../../assets/icons/document.svg";

export const CreateWallet = () => {
  const route = useRoute();
  const navigation = useNavigation();
  const { pet } = route.params;

  return (
    <LinearGradient colors={["#38B6FF", "#FFF"]} style={styles.container}>
      <Header />

      <View style={styles.content}>
        <Image source={{ uri: pet?.image }} style={styles.image} />
```

```
      <TouchableOpacity
        style={styles.button}
        onPress={() => navigation.navigate("InfoWallet", { pet })}
      >
        <Text style={styles.textButton}>Criar{"\n"}Carteira</Text>
        <View style={styles.circle}>
          <DocumentSvg />
        </View>
      </TouchableOpacity>
    </View>
  </LinearGradient>
 );
};
```

Fonte: próprio autor, 2022.

9.1.4. Tela de criação de prevenção

```
import React from "react";
import { Image, View, Text } from "react-native";

import { LinearGradient } from "expo-linear-gradient";
import { Header } from "../../components/header";
import { styles } from "./styles";
import { useNavigation, useRoute } from "@react-navigation/native";

import PreventionSvg from "../../../assets/icons/prevencao.svg";
import { WalletItem } from "../../components/walletItem";
import { ButtonFlutuante } from "../../components/buttonFlutuante";
import { FontAwesome5 } from "@expo/vector-icons";


export const Prevention = () => {
 const route = useRoute();
```

```
 const navigation = useNavigation();
 const { pet } = route.params;
 return (
  <LinearGradient colors={["#38B6FF", "#FFF"]} style={styles.container}>
    <Header />

    <View style={styles.content}>
     <Image source={{ uri: pet?.image }} style={styles.image} />
     <PreventionSvg />
     <Text style={styles.title}>Prevenção</Text>
     <WalletItem
      text="vermicida"
      date="00/00/0000"
      onPress={() => navigation.navigate("PreventionInfo", { pet })}
     />
    </View>

    <ButtonFlutuante
     icon={<FontAwesome5 name="plus" size={24} color="white" />}
     onPress={() => navigation.navigate("PreventionInfo", { pet })}
    />
  </LinearGradient>
 );
};
```

Fonte: próprio autor, 2022.

9.1.5. Tela de vacinas

```
import React from "react";
import { Image, View, Text } from "react-native";

import { LinearGradient } from "expo-linear-gradient";
import { Header } from "../../components/header";
```

```
import { styles } from "./styles";
import { useNavigation, useRoute } from "@react-navigation/native";

import VaccineSvg from "../../../assets/icons/vacina.svg";
import { WalletItem } from "../../components/walletItem";
import { ButtonFlutuante } from "../../components/buttonFlutuante";
import { FontAwesome5 } from "@expo/vector-icons";

export const Vaccine = () => {
  const route = useRoute();
  const navigation = useNavigation();
  const { pet } = route.params;
  return (
    <LinearGradient colors={["#38B6FF", "#FFF"]} style={styles.container}>
      <Header />

      <View style={styles.content}>
        <Image source={{ uri: pet?.image }} style={styles.image} />
        <VaccineSvg />
        <Text style={styles.title}>Vacina</Text>
        <WalletItem
          text="Vacina contra raiva"
          date="00/00/0000"
          onPress={() => navigation.navigate("VaccineInfo", { pet })}
        />
      </View>

      <ButtonFlutuante
        icon={<FontAwesome5 name="plus" size={24} color="white" />}
        onPress={() => navigation.navigate("VaccineInfo", { pet })}
      />
    </LinearGradient>
  );
};
```

Fonte: próprio autor, 2022.

9.1.6. Tela de medicamento

```
import React from "react";
import { Image, View, Text } from "react-native";

import { LinearGradient } from "expo-linear-gradient";
import { Header } from "../../components/header";
import { styles } from "./styles";
import { useNavigation, useRoute } from "@react-navigation/native";

import MedicineSvg from "../../../assets/icons/medicamento.svg";
import { WalletItem } from "../../components/walletItem";
import { ButtonFlutuante } from "../../components/buttonFlutuante";
import { FontAwesome5 } from "@expo/vector-icons";


export const Medicine = () => {
 const route = useRoute();
 const navigation = useNavigation();
 const { pet } = route.params;
 return (
  <LinearGradient colors={["#38B6FF", "#FFF"]} 0style={styles.container}>
   <Header />

   <View style={styles.content}>
    <Image source={{ uri: pet?.image }} style={styles.image} />
    <MedicineSvg />
    <Text style={styles.title}>Medicação</Text>
    <WalletItem
     text="Antialérgico"
     date="00/00/0000"
     onPress={() => navigation.navigate("MedicineInfo", { pet })}
    />
   </View>
```

```
    <ButtonFlutuante
      icon={<FontAwesome5 name="plus" size={24} color="white" />}
      onPress={() => navigation.navigate("MedicineInfo", { pet })}
    />
  </LinearGradient>
 );
};
```

Fonte: próprio autor, 2022.

9.1.7. Tela de listar todos os animais de estimação

```
import React, { useEffect, useState } from "react";
import { FlatList } from "react-native";

import { LinearGradient } from "expo-linear-gradient";
import { useNavigation } from "@react-navigation/native";

import { Header } from "../../components/header";
import { styles } from "./styles";
import { PetCard } from "../../components/petCard";
import AsyncStorage from "@react-native-async-storage/async-storage";

export function ListAllPets() {
 const [pets, setPets] = useState([]);
 const navigation = useNavigation();

 async function loadAllPets() {
  const value = await AsyncStorage.getItem("@pet");
  if (value !== null) {
   setPets(JSON.parse(value));
  }
 }

 useEffect(() => {
```

```
  loadAllPets();
 }, []);

 return (
  <LinearGradient colors={["#38B6FF", "#FFF"]} style={styles.container}>
   <Header />

   <FlatList
    data={pets}
    keyExtractor={(_, key) => String(key)}
    renderItem={({ item }) => (
     <PetCard
      nome={item.nome}
      idade={item.dataNascimento}
      peso={item.peso}
      image={item.image}
      onPress={() => navigation.navigate('CreateWallet', { pet: item })}
     />
    )}
    style={{ paddingTop: 20 }}
    numColumns={2}
    contentContainerStyle={{
     flex: 1,
     justifyContent: "space-between",
   alignItems: 'center'
    }}
   />
  </LinearGradient>
 );
}
```

Fonte: próprio autor, 2022.