CENTRO UNIVERSITÁRIO CHRISTUS

SISTEMAS DE INFORMAÇÃO

FERNANDO DE LUCAS DA SILVA GOMES

INIMAL: UMA APLICAÇÃO NATIVA IOS PARA CONTROLE DE VACINAÇÃO DE PETS

FORTALEZA

2023

FERNANDO DE LUCAS DA SILVA GOMES

INIMAL: UMA APLICAÇÃO NATIVA IOS PARA CONTROLE DE VACINAÇÃO DE PETS

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Sistemas de Informação do Centro Universitário Christus, como requisito parcial para obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Gleidson Sobreira Leite

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Centro Universitário Christus - Unichristus
Gerada automaticamente pelo Sistema de Elaboração de Ficha Catalográfica do
Centro Universitário Christus - Unichristus, com dados fornecidos pelo(a) autor(a)

G633i Gomes, Fernando de Lucas da Silva.
INIMAL: uma aplicação nativa iOS para controle de vacinação de pets / Fernando de Lucas da Silva Gomes. - 2023.
36 f. : il. color.

Trabalho de Conclusão de Curso (Graduação) - Centro Universitário Christus - Unichristus, Curso de Sistemas de Informação, Fortaleza, 2023.
Orientação: Prof. Dr. Gleidson Sobreira Leite.

1. Desenvolvimento. 2. Aplicativos. 3. iOS. 4. Swift. 5. Desenvolvimento nativo.. I. Título.

CDD 004.07

FERNANDO DE LUCAS DA SILVA GOMES

INIMAL: UMA APLICAÇÃO NATIVA IOS PARA CONTROLE DE VACINAÇÃO DE PETS

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Sistemas de Informação do Centro Universitário Christus, como requisito parcial para obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em: 14 de Dezembro de 2023

BANCA EXAMINADORA

Prof. Dr. Gleidson Sobreira Leite (Orientador)
Centro Universitário Christus (Unichristus)

Prof. Ms. Francisco Edvan Chaves
Centro Universitário Christus (Unichristus)

Prof. Ms. Tiago Guimarães Sombra
Centro Universitário Christus (Unichristus)

AGRADECIMENTOS

Quero agradecer primeiramente ao jovem eu que sempre acreditou que seria possível estar aqui, ocupando uma cadeira da universidade e aos meus pais que sempre ressaltaram a importância dos estudos. Também agradeço a todos que sempre torceram por mim, das pessoas mais próximas aos que estão a centenas de quilômetros de distância. Quero expressar minha gratidão especial ao Sr. Gleidson, ao Sr. Edvan Chaves e ao Sr. Thiago Sombra. Além disso, gostaria de agradecer à instituição e aos professores que sempre me fizeram sentir no lugar certo.

Serei eternamente grato a todos!

"O maior desafio da tecnologia no século XXI não é o que podemos fazer. É o que devemos fazer"

(TIM COOK)

RESUMO

With the growing population of *pets* there are opportunities for various products and market opportunities that can help in the diverse areas in which pets or their owners need support. However, despite this growth, there is still a large percentage of animals that are not adequately vaccinated in order to prevent these animals from suffering from any disease. In this context, and seeking to contribute to the academic area and pet care, it is proposed to use information technology in the development of a *mobile* application that helps pet owners maintain control over vaccination of your animals. In conclusion, this work allowed us to present the concepts involved in the MASAM methodology and also put it into practice and create an application following the methodology, thus becoming a source of study and consultation for academics and professionals who wish to work in the development of applications for mobile devices.

Palavras-chave: Desenvolvimento. aplicativos. iOS. *swift*. Desenvolvimento nativo.

ABSTRACT

With the growing pet population comes opportunities for a variety of products and market opportunities that can assist in the many areas in which pets or their owners need support. However, despite this growth, there is still a large percentage of animals that are not vaccinated with the aim of preventing these animals from suffering from any disease. In this context, seeking to contribute to the academic area and pet care, it proposes the use of information technology in the development of a mobile application that helps pet owners maintain control over the vaccination of their animals. In conclusion, this work allowed us to present the concepts involved in the MASAM methodology and also put it into practice and create an application following the methodology, thus becoming a source of study and consultation for academics and professionals who wish to work in the development of applications for mobile devices

Keywords: dDevelopment. Applications. *iOS*. *Swift*. Native Development.

LISTA DE FIGURAS

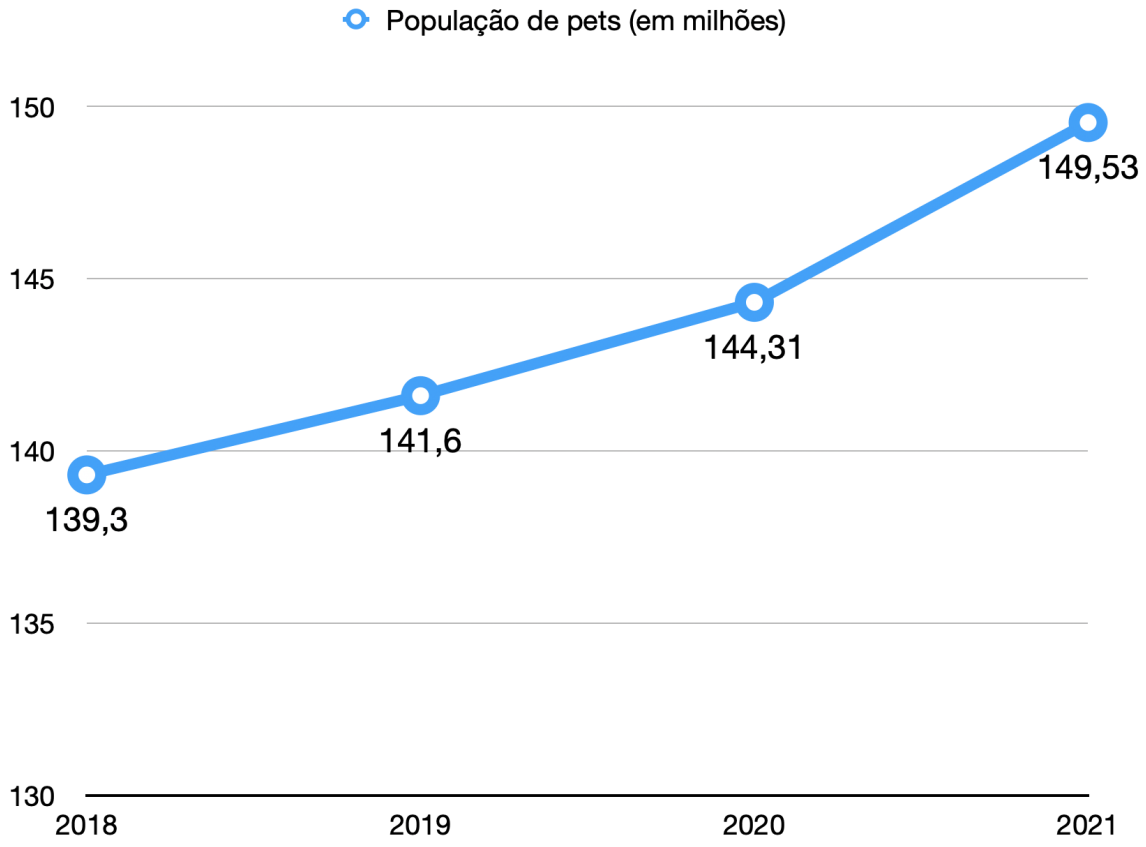
SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização e delimitação do tema

O número de *pets*, animais de estimação, no Brasil tem crescido com o passar dos anos. Segundo o IPB (IPB, 2022), a quantidade de animais de estimação até 2021 atingiu um quantitativo de cerca de 149,6 milhões, um aumento de 3,7% em relação ao ano anterior (Figura 1). Com isso, cerca de 70% da população brasileira possui um *pet* em casa ou conhece alguém que tenha, considerando os 215 milhões de brasileiros. Acompanhando esse crescimento, observa-se que devido isso houve crescimento voltado para esse mercado em 2022, que faturou 60,2 bilhões de reais. Esse valor, divulgado no balanço de 2022 pelo Instituto Pet Brasil, representa um aumento de 16.4% em relação ao ano anterior (LIMA, 2022).

Figura 1 – Crescimento da população de *pet* em milhões entre os anos de 2018 e 2021.



Fonte: IPB (2022).

Com o aumento populacional e crescimento de mercado, também acarretou o incremento da questão da assistência à saúde desses animais, cujos cuidados são de importância não

só para eles como também para os seres humanos que cuidam dos animais.

Uma etapa fundamental para garantir a saúde do *pet* é a vacinação. A função das vacinas é treinar o sistema para que, quando injetada em um corpo, produza anticorpos. Como um exemplo de atuação de vacinas, temos aquelas que liberam formas enfraquecidas ou mortas de vírus ou bactérias que, quando aplicada, permite que o sistema imunológico produza anticorpos para combater doenças infecciosas sem oferecer riscos à saúde (FERREIRA, 2022).

O desenvolvimento da vacinação e as novas tecnologias para a produção de vacinas permitiram o controle de diversas doenças em *pets*. Além de proteger os animais de estimação, o sucesso no controle da raiva em cães tornara esporádicos os casos em humanos. Apesar dos protocolos de vacinação terem variações, a maioria exige que além da aplicação inicial sejam aplicadas doses de reforço para que seja desenvolvida uma memória imunológica. Uma das ferramentas que se tornaram comum para garantir a vacinação dos *pets* é a carteira de vacinação, documento esse que normalmente é oferecido por profissionais da saúde após a vacinação do animal (AMARO *et al.*, 2016).

1.2 Problematização

Nesse contexto de crescente aumento de *pets* e a importância de sua vacinação, surge também uma questão preocupante, que seria a taxa efetiva de vacinação em animais de estimação. Em 2021, por exemplo, a cobertura de vacina contra a raiva em cães e gatos era de apenas 60,4% (BRASIL, 2022).

Diante desse cenário, surge a oportunidade de adoção de medidas para auxiliar os proprietários de *pets* na gestão ou controle de vacinas de forma a possibilitar, por exemplo, um acompanhamento de quais já tomaram e quando precisará ser feito um reforço. Uma solução para essa problemática seria a digitalização da carteira de vacinação. Com o avanço tecnológico, se tornou cada vez mais comum o uso de versões digitais de documentos como carteira de habilitação e título de eleitor. Segundo pesquisa do Comitê Gestor da *Internet* no Brasil, em 2019, contataram-se cerca de 134 milhões de usuários dela no Brasil, sendo que os brasileiros estão utilizando cada vez mais ela nos celulares do que em computadores. A pesquisa também informa que o número de acesso através de computadores tem caído ano após ano (BRASIL, 2020).

Diante desse cenário, este trabalho propõe a criação de um aplicativo com a finalidade de auxiliar os proprietários de *pets* na gestão e acompanhamento de vacinação, agindo como uma

carteira de vacinação pet digital deles. Com isso, as pessoas poderão ter registros das vacinações de seus *pets*, de forma a evitar esquecimentos e suas respectivas consequências. Além disso, esse trabalho também busca demonstrar as etapas de desenvolvimento de uma aplicação para dispositivos móveis de forma a inspirar novas soluções para o desenvolvimento de aplicações móveis para dispositivos *iOS*, contribuindo assim para o avanço deste mercado.

1.3 Objetivos

1.3.1 Objetivo geral

Diante do crescimento populacional de *pets* e a importância da vacinação dos mesmos, este trabalho tem como objetivo a construção de uma aplicação para registro do histórico de vacinação e lembretes de reforço. Tal aplicação servirá como uma carteira de vacinação digital e será construída para dispositivos móveis que contenham o sistema operacional *iOS*.

1.3.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Revisar a literatura a respeito do desenvolvimento mobile e os principais processos de desenvolvimento;
2. Selecionar as tecnologias adotadas para desenvolvimento da aplicação;
3. Idealizar a aplicação e suas funcionalidades para atingir o objetivo geral;
4. Desenvolver a aplicação e demonstrar as etapas deste desenvolvimento.

1.4 Justificativa

A escolha do desenvolvimento dessa aplicação foi motivada pela importância do processo de vacinação de *pets* e o crescimento populacional deles, buscando com que esse trabalho traga impactos acadêmicos e sociais.

1.5 Estrutura do trabalho

Este trabalho está estruturado da seguinte maneira: no capítulo 1, consta a introdução com contextualização, problematização, objetivos e justificativa, no capítulo 2, é apresentada a revisão da literatura das tecnologias selecionadas para o desenvolvimento do jogo, no capítulo 3,

é apresentada a metodologia; no capítulo 4, o aplicativo é descrito, e, por fim, no capítulo 5, a conclusão.

2 REVISÃO DA LITERATURA

Este capítulo apresenta a revisão de literatura com informações sobre o desenvolvimento de aplicações móveis, desenvolvimento *iOS*, etapas do desenvolvimento de aplicações móveis e as tecnologias utilizadas para o desenvolvimento do aplicativo proposto neste trabalho.

2.1 Desenvolvimento de aplicativos móveis

Segundo El-Kassas et al. (2015), o desenvolvimento de aplicativos móveis possui particularidades e restrições que o diferenciam do desenvolvimento de outros tipos de *software*. Uma das particularidades é que as aplicações são heterogêneas, pois os diferentes sistemas operacionais fazem com que aplicações rodem apenas para a plataforma que foi desenvolvida.

O desenvolvimento de aplicativos para dispositivos móveis possui duas abordagens principais, a nativa e a híbrida. Os aplicativos nativos são desenvolvidos utilizando as ferramentas, *Application Programming Interface* (API) e linguagens de programação específicas fornecidas pelas plataformas (EL-KASSAS *et al.*, 2015). As principais plataformas de aplicação móvel são *Android* e *iOS* (GOADRICH, 2011).

Segundo Reis (2019), as aplicações nativas são aquelas que são projetadas exclusivamente para uma plataforma específica, aproveitando todas as capacidades oferecidas por essa plataforma. Em outras palavras, aplicativos criados apenas para *iOS*, *Android*, *Windows Phone* e similares podem explorar plenamente os recursos do dispositivo, como a câmera, calendário, álbum de fotos, GPS e muito mais, graças à estrutura do sistema operacional em que operam. Esses aplicativos nativos são desenvolvidos sob medida para funcionar perfeitamente em dispositivos específicos, levando em consideração suas características únicas.

A criação de aplicativos móveis nativos ganhou destaque com a popularização dos *smartphones* na segunda metade da década anterior e continuou sendo uma tendência durante a década seguinte e até os dias atuais (REIS, 2019).

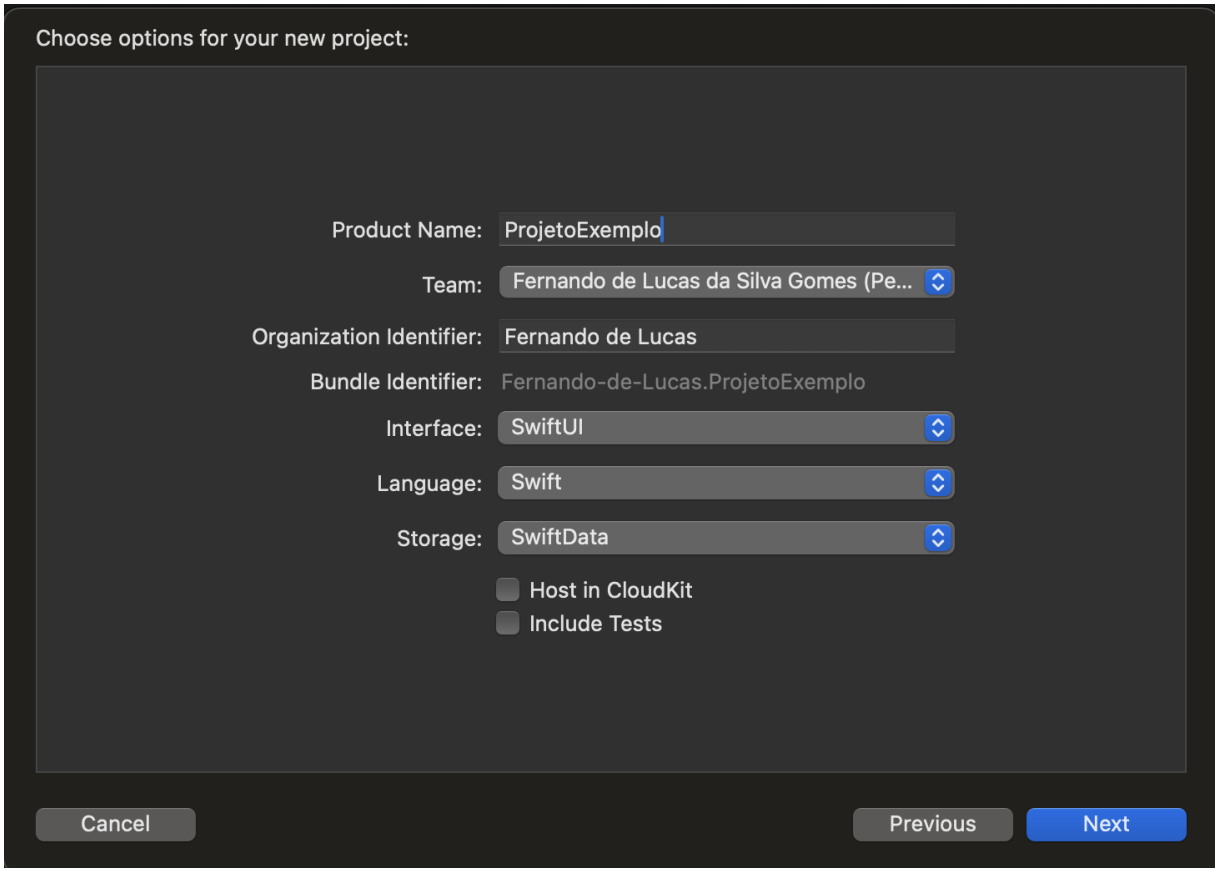
2.2 Desenvolvimento *iOS*

Desenvolvido pela *Apple*, o *iOS* foi feito para atender as necessidades de aparelhos móveis desenvolvidos pela empresa e foi lançado em Janeiro de 2007 juntamente com o primeiro *iPhone*. O desenvolvimento de aplicações *iOS* é vinculado à linguagem *Swift*, que é uma linguagem de *software* livre, e foi lançado pela *Apple* como um sucessor do Objective-C, a

linguagem anteriormente utilizada para o desenvolvimento de aplicações *iOS*. *Swift* por sua vez possui interoperabilidade com o Objective-C, permitindo que códigos escritos em Objective-C possam ser integrados à códigos *Swift* e vice-versa (*APPLE INC*., 2016).

Através do ambiente de desenvolvimento, o *Xcode*, é possível criar aplicações e testar em simuladores. Para desenvolver uma aplicação, o desenvolvedor cria um projeto no *Xcode* do tipo App, o tipo de projeto escolhido aqui, habilita *templates* para a configuração inicial do projeto, como por exemplo qual tipo de *framework* de interface e dados irá utilizar (Figura 2). Essa etapa não impede o desenvolvedor de utilizar um *framework* diferente, o *Xcode* utiliza essa informação apenas para gerar arquivos e trechos para facilitar a configuração inicial do projeto (*APPLE INC*., 2023?a).

Figura 2 – Tela de configuração de novo projeto no *Xcode*.



Fonte: Captura de tela do *software Xcode* feita pelo autor.

Um projeto no *Xcode* pode ser um app, um framework ou até mesmo um jogo. O desenvolvedor pode ainda criar um Workspace. *Workspace* é um contêiner para projetos relacionados, ele permite que se organize múltiplos projetos que tem relação explica ou implícita. Podemos por exemplo criar um workspace com uma aplicação e um ou mais outros projetos que

serão *frameworks* utilizados pela aplicação.

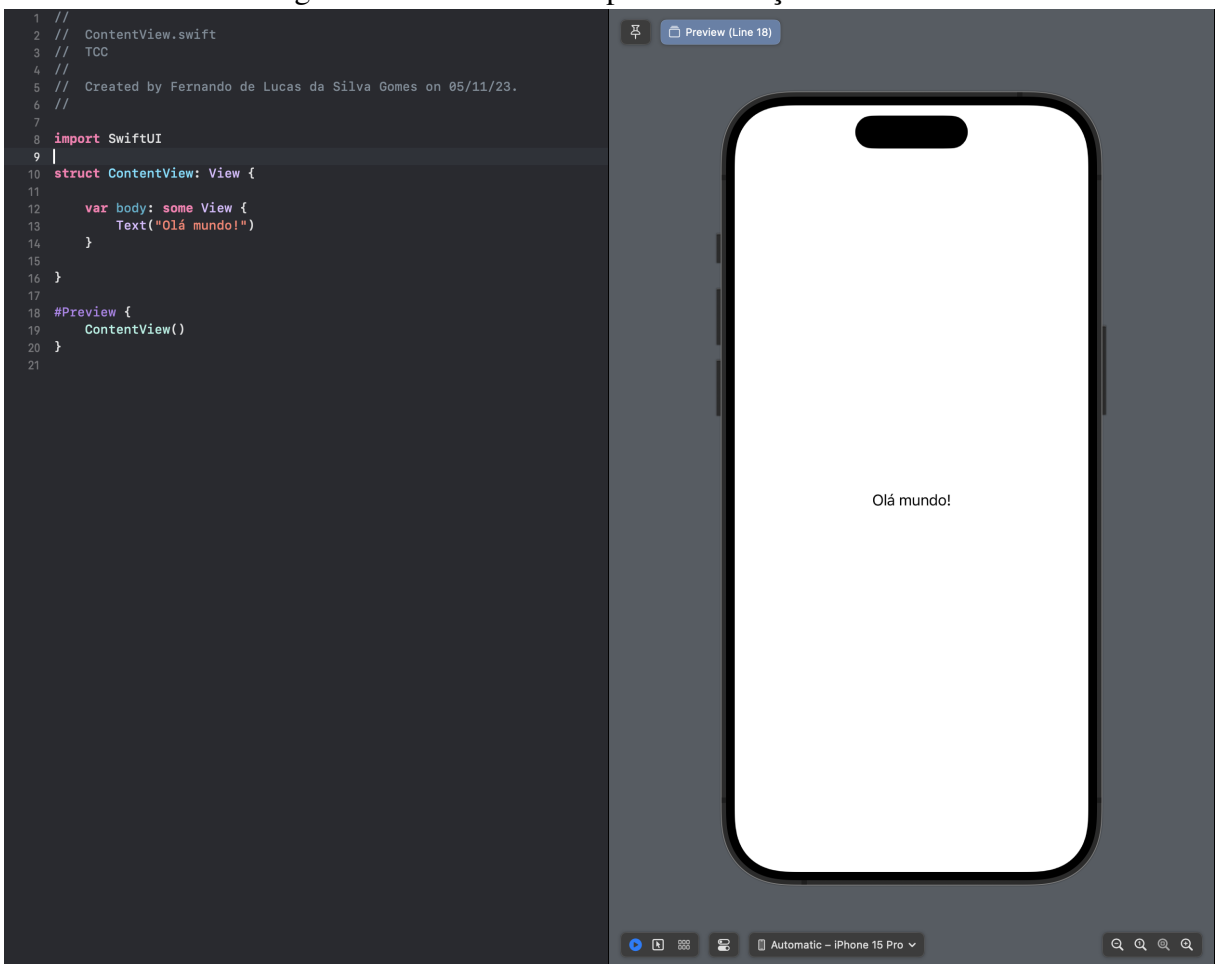
Frameworks são diretórios que oferecem interfaces para que o usuário possa realizar tarefas, como acessar funcionalidades do hardware, como câmera, acelerômetro e etc. Um framework pode possuir diversos recursos como código, arquivos de imagens, arquivos de textos ou arquivos de documentação em um único pacote . Segundo a *Apple Inc*., o sistema carrega os *frameworks* na memória conforme são necessários e compartilha uma cópia dos recursos quando possível (*APPLE INC*., 2013).

Existem *frameworks* de sistema, que são *frameworks* fornecidos pela própria *Apple* e *frameworks* externos, que podem bibliotecas que podem ser adicionadas aos projetos ou, conforme citado anteriormente, podem ser outros projetos em um mesmo workspace. Os principais *frameworks* fornecidos pela *Apple* utilizados no desenvolvimento da aplicação desenvolvida neste trabalho foram o *SwiftUI*, *EventUIKit*.

SwiftUI é um framework que permite a criação de interfaces para diferentes sistemas operacionais *Apple* utilizando a linguagem de programação *Swift*. O *Swift*UI permite com que atualizações nos dados atualizem automaticamente os componentes visuais que são afetados por essa mudança (*APPLE INC*., 2023?b). Anunciado em 2019, o *Swift*UI veio como uma nova forma de criar interface gráfica para aplicações *iOS*, antes disso, as plataformas *Apple* contavam com os *frameworks UIKit* e *AppKit* para realizar a mesma função. O *SwiftUI* veio como uma nova forma de realizar essa tarefa tornando a criação de componentes gráficos mais simples. Outra característica de importante destaque é que o *SwiftUI* funciona em todas as plataformas *Apple*, diferente de seu antecessor, o UIKit que atendia apenas dispositivos *iOS*, caso você desejasse criar aplicações para *MacOS* precisaria utilizar o framework *AppKit*. Outra vantagem em relação ao seu antecessor, é que o *SwiftUI* permite a visualização prévia da interface no *Xcode* (Figura 3), enquanto no *UIKit*, quaisquer alterações que fosse feita, exigiria que o desenvolvedor executasse a aplicação novamente em um simulador para visualizar a alteração, isso poderia tomar minutos do tempo de desenvolvimento, agravando ainda mais quando pensamos em aplicações grandes que contam com centenas de milhares de linhas de código.

O *SwiftUI* permite também que menos código seja necessário para o fluxo de atualizações de dados. Antes, nos *frameworks UIKit* e *AppKit*, quando uma interação de usuário ou mudança em dados precisava ser exibida na interface gráfica, o desenvolvedor precisava controlar essa atualização através de *delegates callbacks*, por exemplo. Já em *SwiftUI*, os componentes visuais podem ser vinculados à dados, quando os dados são atualizados, seja por eventos externos

Figura 3 – Ferramenta de pré-visualização do *Xcode*.



Fonte: Captura de tela do *software Xcode* feita pelo autor.

ou ações do usuário, o *SwiftUI* automaticamente atualiza as partes da interface que são afetadas por aquela mudança. Logo, o *SwiftUI* automaticamente executa parte do trabalho que antes precisaria ser implementado pelo desenvolvedor.

O *EventKitUI* permite que exibamos uma interface para que o usuário possa visualizar, selecionar e editar eventos ou lembretes no Calendário (*APPLE INC.*, 2023?c). Calendário é uma aplicação desenvolvida pela *Apple* e que já vem pré-instalada nos dispositivos, ela permite com que o usuário possa criar e editar eventos, compromissos e reuniões. Utilizando o *EventKitUI*, podemos no código, em um momento de nossa escolha, exibir uma tela para que o usuário cadastre um evento para que ele receba lembretes sobre o mesmo.

O *SwiftData* é um *framework* que permite salvar, buscar e filtrar dados localmente (*APPLE INC.*, 2023). Os dados ficam persistidos no dispositivo do usuário e só serão apagados através de código ou se o usuário desinstalar a aplicação. Utilizando o *Swift*Data, o desenvolvedor pode criar toda a camada de dados da aplicação através apenas do código, definindo propriedades, tipos e relacionamentos com outros objetos. Para definir os objetos que serão persistidos, o

desenvolvedor deve criar a classe no código e adicionar a anotação @Macro, a partir disso, o *SwiftUI* irá utilizar as propriedades da classe para definir o esquema que representará o objeto na hora de persisti-lo localmente. Para melhor entendimento usemos a classe exemplo Pet (Figura 4), que contém os valores *name* e *specie* em *Swift*, se pensarmos no banco de dados, o *SwiftData* persistirá uma tabela com nome *Pet*, contendo as colunas *name* e specie, em que ambas aceitam apenas valor *String*.

Figura 4 – Exemplo classe *Pet*.

```
public class Pet {

    public var name: String
    public var specie: String

    public init(
        name: String,
        specie: String
    ) {
        self.name = name
        self.specie = specie
    }

}
```

Fonte: Captura de tela do *software Xcode* feita pelo autor.

2.3 Etapas do desenvolvimento de aplicações

Em um dos estudos pioneiros em desenvolvimento ágil, Abrahamsson et al(2004) propõe uma abordagem de desenvolvimento chamada *Mobile D*, onde as fases seriam divididas em (1) Exploração, (2) Início, (3) Produção, (4) Estabilização, (5) Testes de sistema e correções.

Segundo El-Kassas et al. (2015), o ciclo de vida do desenvolvimento de aplicações móveis consiste em (1) Analisar a ideia da aplicação, (2) *Design* da interface de usuário, (3) Desenvolvimento da aplicação utilizando as ferramentas e linguagens de programação específicas da plataforma, (4) Teste da aplicação em diferentes dispositivos e (5) Publicação da aplicação na loja da plataforma alvo.

2.3.1 MASAM

Jeong et al(2008) propôs a Metodologia Ágil de *Software* de Aplicativos Móveis, chamada MASAM, que é uma metodologia recomendada para pequenas companhias que estão

focadas no desenvolvimento de aplicações móveis (FLORA; CHANDE, 2013).

O MASAM propõe o ciclo de desenvolvimento de aplicações móveis em quatro fases:

1. Preparação;
2. Personificação;
3. Desenvolvimento;
4. Comercialização.

A fase de preparação é indispensável para o sucesso do projeto. Nessa etapa é elaborado o conceito do produto através da análise do mesmo e do público alvo (FLORA; CHANDE, 2013).

Segundo Jeong et al (2008), na fase de personificação, a intenção de um cliente é representada de forma concreta. Nessa fase são construídos protótipos e verificado se os mesmos atendem às necessidades do usuário. Outras tarefas dessa etapa são desenvolvimento de *storyboards*, análise de requerimentos não funcionais, definição de arquitetura e gerenciamento de padrões.

Na fase de desenvolvimento, o produto é gradualmente construído e lançado para o cliente. Esta fase pode ser dividida em duas etapas principais: Preparação para Implementação e ciclo de Lançamento (FLORA; CHANDE, 2013).

Na etapa de Preparação para Implementação são realizadas as configurações do ambiente de desenvolvimento e definido o plano de desenvolvimento. Na etapa de Ciclo de Lançamento, é feito o plano de lançamento, ciclo de interação e lançamento (FLORA; CHANDE, 2013).

Na fase de Comercialização são realizados testes de aceitação com os usuários, testes de lançamento e por último, o lançamento do produto.

3 METODOLOGIA

Visando atingir o objetivo desse trabalho, que consiste em desenvolver uma aplicação *iOS* para controle de vacinas de *pet*, foi realizado uma revisão de literatura de trabalhos relacionados ao desenvolvimento de aplicações móveis, plataformas e novas tecnologias *Apple*.

Na revisão de literatura, foram consultados artigos científicos, publicações e sites oficiais a fim de coletar informações que levassem o correto processo de desenvolvimento da aplicação.

Após a revisão de literatura, pode-se então selecionar entre os ciclos de desenvolvimento propostos pelos diferentes autores, o que nos levaria a atingir o objetivo deste trabalho. Em seguida, foi possível elencar as atividades e dar início ao desenvolvimento da aplicação utilizando a metodologia MASAM. Contudo, a etapa de comercialização não se aplica ao escopo deste trabalho por envolver as etapas de lançamento do produto na loja digital da plataforma *Apple*, a *App Store*, assim como ser um trabalho acadêmico sem almejar lucros comerciais.

A seguir, serão listadas as fases realizadas seguindo a metodologia MASAM, adotada para o desenvolvimento do aplicativo mobile. A Metodologia foi selecionada em virtude da maior afinidade com o autor deste trabalho em virtude de já ter experiências anteriores com sua adoção.

3.1 Fase de Preparação

Nessa etapa, foram elencadas quais as funcionalidades do aplicativo, tecnologias para desenvolvimento *iOS* iriam ser utilizadas, público-alvo e prazos para desenvolvimento de funcionalidades. As funcionalidades foram elencadas através de casos de uso pensando na execução das etapas de cadastro de *pet*, vacina e visualização das informações. Essas funcionalidades foram pensadas de forma a serem a base para atingir o objetivo deste trabalho. As figuras 5, 6 e 7 ilustram os casos de uso elencados.

Figura 5 – Caso de uso de cadastro de *pet*.

Caso de Uso	Cadastro de Pet
Descrição	Permitir que o usuário possa realizar o cadastro com os dados de um pet
Pré-requisitos	Nenhum
Fluxo de eventos	(1) O usuário clica no botão de cadastrar; (2) É direcionado para a tela de cadastro; (3) Preenche os dados; (4) Clica em salvar.
Pós-condições	Os dados preenchidos são salvos no dispositivo do usuário

Fonte: Captura de tela do caso de uso feita pelo autor.

Figura 6 – Caso de uso de cadastro de vacina.

Caso de Uso	Cadastro de Vacina
Descrição	Permitir que o usuário possa realizar o cadastro com os dados de um aplicação de vacina
Pré-requisitos	Cadastro de Pet
Fluxo de eventos	(1) O usuário clica no botão de cadastrar; (2) É direcionado para a tela de cadastro de vacina; (3) Seleciona o pet que foi vacinado; (4) Preenche os dados da aplicação; (5) Informa se precisará ser feita uma nova dose; (6) Clica em salvar.
Pós-condições	Os dados preenchidos são salvos no dispositivo do usuário e caso o usuário informar que precisará ser feita uma nova dose, o lembrete da nova dose será adicionada ao calendário do usuário.

Fonte: Captura de tela do caso de uso feita pelo autor.

Figura 7 – Caso de uso de visualizar lembretes.

Caso de Uso	Visualizar lembretes
Descrição	Permitir que o usuário possa visualizar os lembretes de vacinação futuras.
Pré-requisitos	Cadastro de pet; Cadastro de vacinas.
Fluxo de eventos	(1) O usuário abre a aplicação (2) É exibida uma lista com os próximos lembretes de aplicação de vacina; (3) O usuário c
Pós-condições	

Fonte: Captura de tela do caso de uso feita pelo autor.

Nessa etapa também foi decidido que a aplicação seria construída com as tecnologias mais recentes da *Apple* para que este trabalho possa contribuir com a comunidade de desenvolvimento *iOS*, optamos por utilizar os *frameworks* mais recentes para desenvolvimento de interface e armazenamento de dados em *iOS*, o *SwiftUI* e o *CoreData*. Isso, porém, acaba por limitar a aplicação criada para dispositivos *iOS* 17+ pois apenas eles serão compatíveis.

3.2 Fase de Personificação

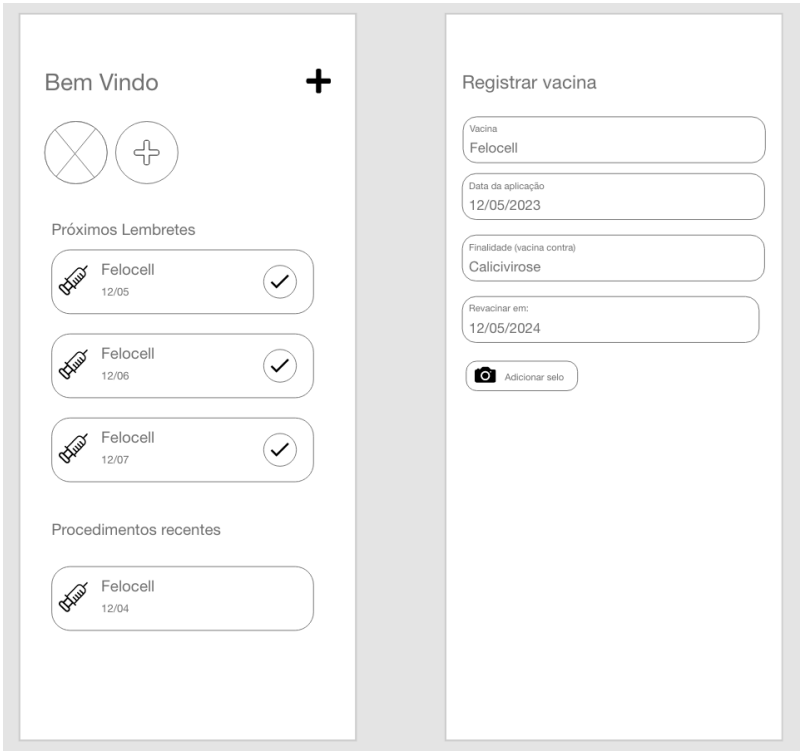
O MASAM estabelece duas atividades principais na fase de personificação (1) Entendimento das necessidades do usuário e (2) Arquitetura.

3.2.1 Entendimento das necessidades do usuário

Como parte fundamental dessa etapa, foram desenvolvidas as interfaces da aplicação. Inicialmente, foram desenvolvidos os primeiros rascunhos das interfaces utilizando a técnica *Crazy Eights*, uma técnica de *design* que consiste em esboçar até 8 ideais em 8 minutos (*GOOGLE INC.*, 2023?).

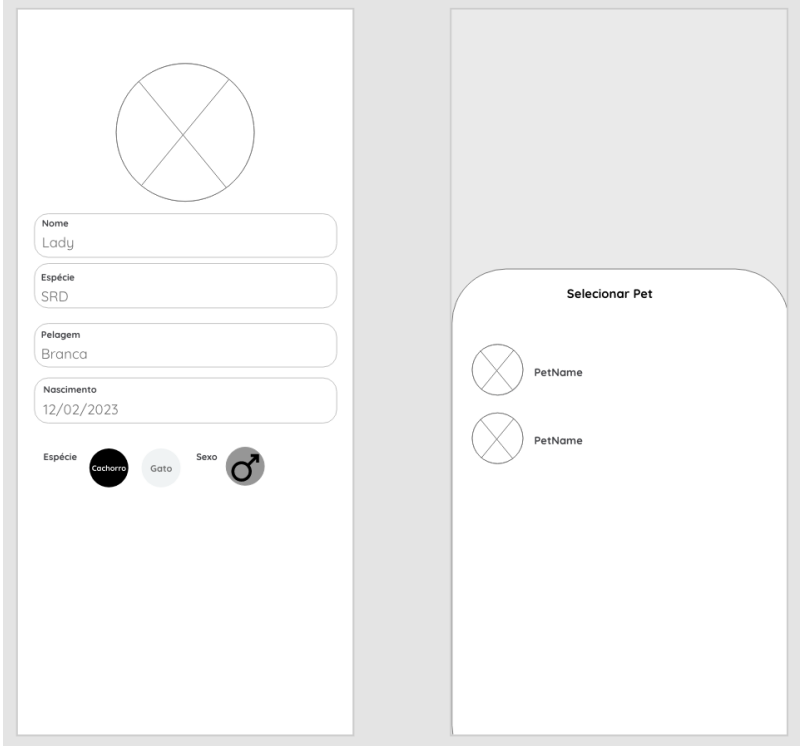
Posteriormente, esses rascunhos foram transformados em protótipos de baixa fidelidade (ilustrados nas Figuras 8 e 9) utilizando a ferramenta *Adobe XD*.

Figura 8 – Protótipos de baixa fidelidade das telas de cadastro de *pet* e selecionar *pet*.



Fonte: Captura de tela do *software* XD feita pelo autor.

Figura 9 – Protótipos de baixa fidelidade das telas de cadastro de pet e selecionar pet.

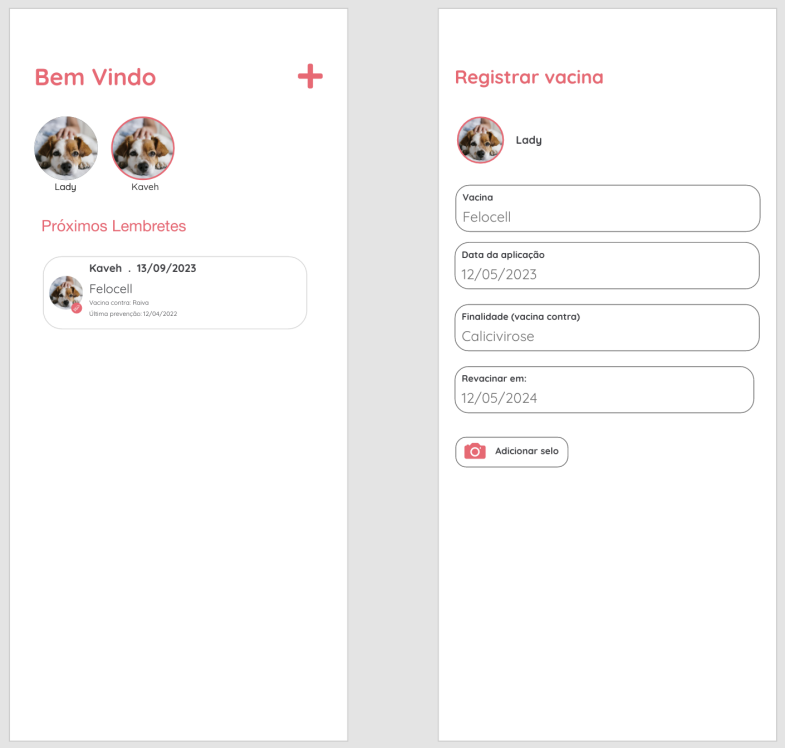


Fonte: Captura de tela do *software* XD feita pelo autor.

Os protótipos de baixa fidelidade representam componentes visuais sem detalhes, cores ou formas definitivas, isso permite com que sejam desenvolvidos e editados mais rapidamente com o objetivo de validar a ideia. A validação foi feita verificando se o protótipo executava todas as funcionalidades elencadas na fase de preparação.

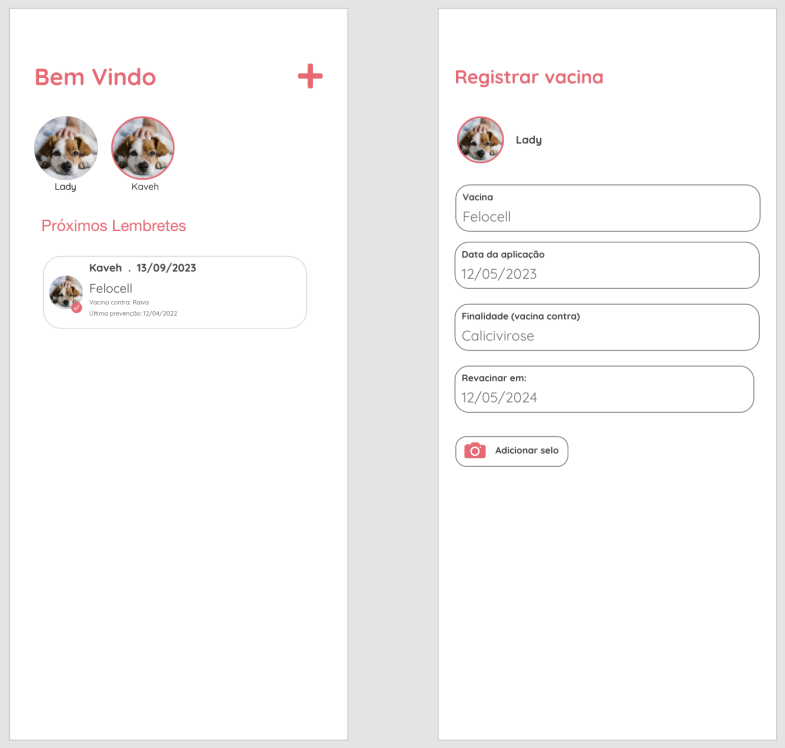
Uma vez que os protótipos de baixa fidelidade estavam validados, eles foram adaptados para protótipos de alta fidelidade (ilustrados nas Figuras 10 e 11). Nessa etapa, foram escolhidas as cores, fontes e formas finais e tamanhos padrões dos componentes visuais.

Figura 10 – Protótipos de alta fidelidade das telas de cadastro de *pet* e selecionar *pet*.



Fonte: Captura de tela do *software* XD feita pelo autor.

Figura 11 – Protótipos de alta fidelidade das telas de cadastro de *pet* e selecionar *pet*.



Fonte: Captura de tela do *software* XD feita pelo autor.

3.2.2 Arquitetura

Durante esta fase foram definidas a arquitetura de design de interfaces bem, além de alguns padrões para o código. A arquitetura de design escolhida foi o *Model-View-ViewModel* por ser uma arquitetura utilizada em projetos *Apple* utilizando *Swift*UI (*APPLE INC.*, 2022).

O MVVM separa as regras de negócio da aplicação da interface. Isso é possível graças a distribuição de responsabilidades entre três objetos: (1) A *View* responsável por renderizar a interface do usuário, (2) o *ViewModel* responsável pelas interações de usuário e a estado da View, e (3) o *Model* que lida com os dados (SYROMIATNIKOV; WEYNS, 2014).

Além disso, foi estabelecido como padrão que todos os arquivos de interface teriam testes unitár*iOS*. Isso foi possível utilizando a arquitetura MVVM criando protocolos para os objetos utilizados pelas *Views* e *ViewModels*.

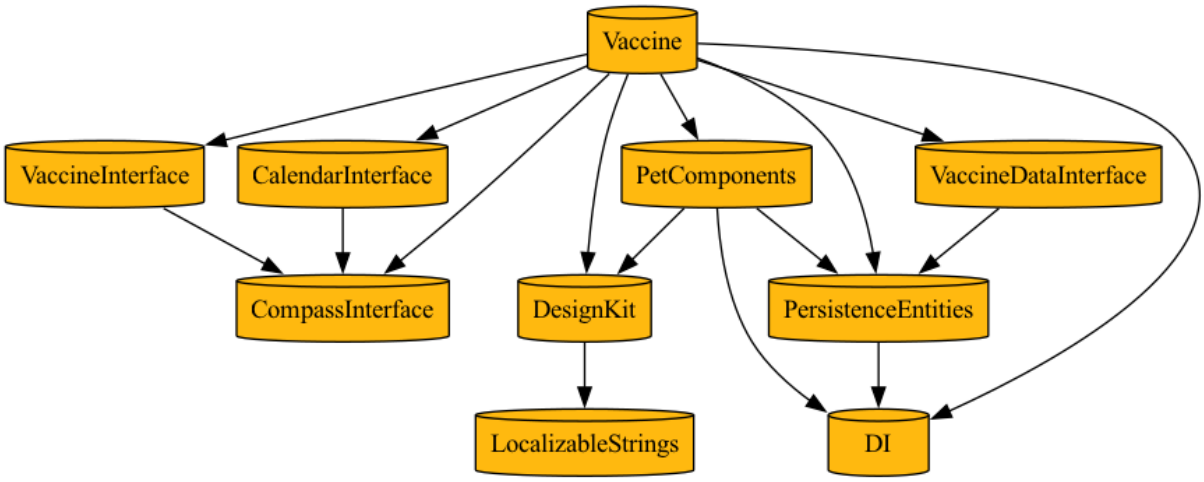
Nos testes, foram validados se todos os métodos presentes na *ViewModel* estavam sendo executados como esperados e se todas as inovações de métodos e dados fornecidos pela *ViewModel* estavam sendo executados corretamente nas *Views*.

3.3 Fase de Desenvolvimento

Durante esta fase foi feita a configuração do ambiente de desenvolvimento. Para isso, foi utilizado o *Tuist* e *Xcode* para a geração do projeto do ambiente. Como citado anteriormente, *Xcode* é o ambiente de desenvolvimento fornecido pela *Apple* para a criação e gerenciamento de projetos *iOS*, *WatchOS* ou *MacOS*. Já o *Tuist*, é uma ferramenta de comando de linha que facilita a criação, manutenção e interação de projetos *Xcode* (Tuist Inc., 2023).

Nessa fase, o uso do *Tuist* foi fundamental para que fosse possível a modularização da aplicação em diferentes projetos, cada um desses projetos representa um módulo da aplicação, que podem ser utilizados pela aplicação principal ou por outros módulos. A figura 12, ilustra um dos módulos criados para uso desta aplicação e seu gráfico de dependências.

Figura 12 – Gráfico de dependências do módulo *Vacine*.



Fonte: Gráfico de depedências gerado pela ferramenta *Tuist*.

Graças a modularização, foi possível a criação de projetos separados que se integravam e, para cada funcionalidade da aplicação, foi possível separar em módulos de renderização e navegação de telas, e módulos de dados. Isso foi fundamental para o desenvolvimento de um código escalável, pois os fluxos de telas de uma funcionalidade puderam ser controlados pelo módulo e serem acessíveis apenas a partir de protocolos, ou seja, a implementação da interface fica encapsulada e inacessível por outros módulos.

A Figura 13 ilustra um trecho de código presente no módulo de Vaccine, que invoca a interface de calendário da aplicação utilizando apenas o protocolo.

Essa mesma estrutura se aplica aos módulos de dados, ou seja, caso um módulo precise utilizar uma funcionalidade relacionada a camada de dados como por exemplo, salvar os

Figura 13 – Trecho de código do módulo Vaccine.

```
func navigateToCalendar(eventModel: CalendarEventModel) {
    guard let compass else {
        return
    }
    let coordinator: CalendarCoordinatorProtocol = DIContainer.appContainer.get()

    DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
        coordinator.addNewEvent(compass: compass, eventModel: eventModel)
    }
}
```

Fonte: Captura de tela do *software Xcode* feita pelo autor.

dados de um cadastro, é necessário utilizar o protocolo adequado. A Figura 14 ilustra um dos protocolos da camada de dados, o *VaccineRepositoryProtocol*, caso seja necessário inserir ou deletar o cadastro de uma vacinação, basta invocar um dos métodos do protocolo e apenas essas funcionalidades podem ser executadas, qualquer outra ação referente aos dados de vacinação não serão possíveis, evitando assim comportamentos indesejados no código.

Figura 14 – Protocolo de acesso aos dados de vacinas.

```
11  public protocol VaccineRepositoryProtocol {
12      func insert(_ vaccine: VaccineApplication)
13      func delete(_ vaccine: VaccineApplication)
14  }
```

Fonte: Captura de tela do *software Xcode* feita pelo autor.

4 O APLICATIVO INIMAL

4.1 Informações gerais

Através das etapas relatadas na metodologia deste trabalho, foi possível desenvolver a aplicação *iNimal*. O *iNimal* é um aplicativo que busca auxiliar donos de *pet* a manterem os dados da vacinação desses animais em um meio digital. Através dele, o usuário da aplicação pode cadastrar e gerenciar dados dos *pets* que possui, cadastrar aplicações de vacina que ocorreram e adicionar lembretes ao calendário com a data das reaplicações da vacina, se necessário.

4.2 Funcionalidades da aplicação

Nessa seção discutiremos sobre as funcionalidades e como elas podem ser executadas na aplicação.

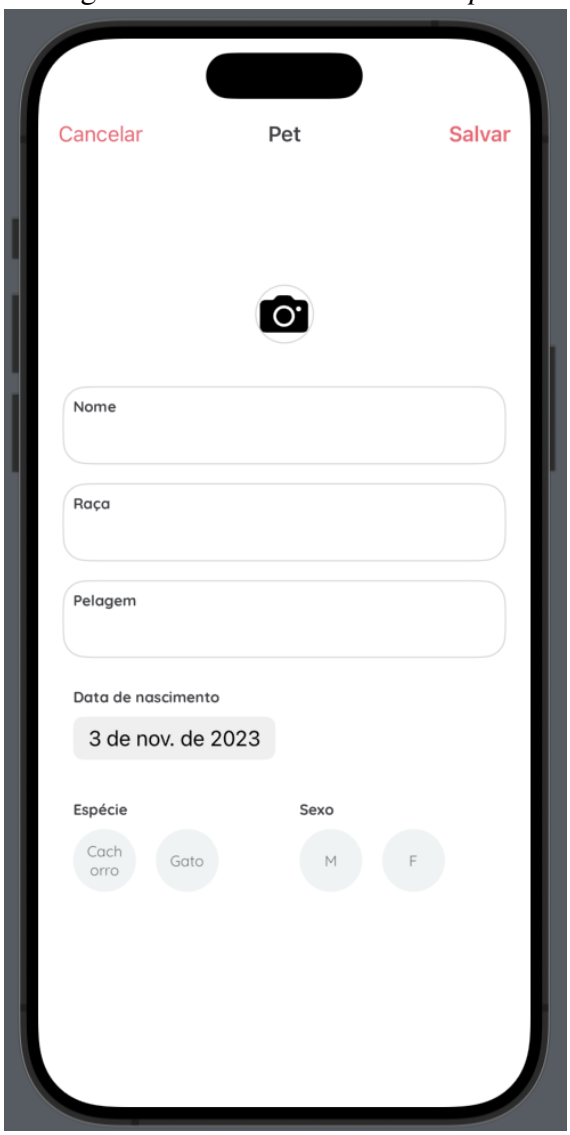
4.2.1 Cadastro de Pet

O cadastro de *pet* é o primeiro registro que deve ser feito pelo usuário. Nele, o usuário deve preencher os dados abaixo referente ao seu *pet*:

1. Nome;
2. Raça;
3. Pelagem;
4. Data de nascimento;
5. Espécie;
6. Sexo.

Uma vez preenchidos esses dados, a opção de salvar é habilitada, feito isso, os dados do *pet* são salvos no dispositivo. Essa etapa é fundamental para permitir que ao cadastrar uma vacina, seja possível vincular qual foi o animal vacinado. Dessa forma o usuário pode visualizar quais vacinas foram feitas em um *pet* e quais estão agendadas para o futuro. A Figura 15 ilustra a tela de cadastro.

Figura 15 – Tela de cadastro de *pet*.



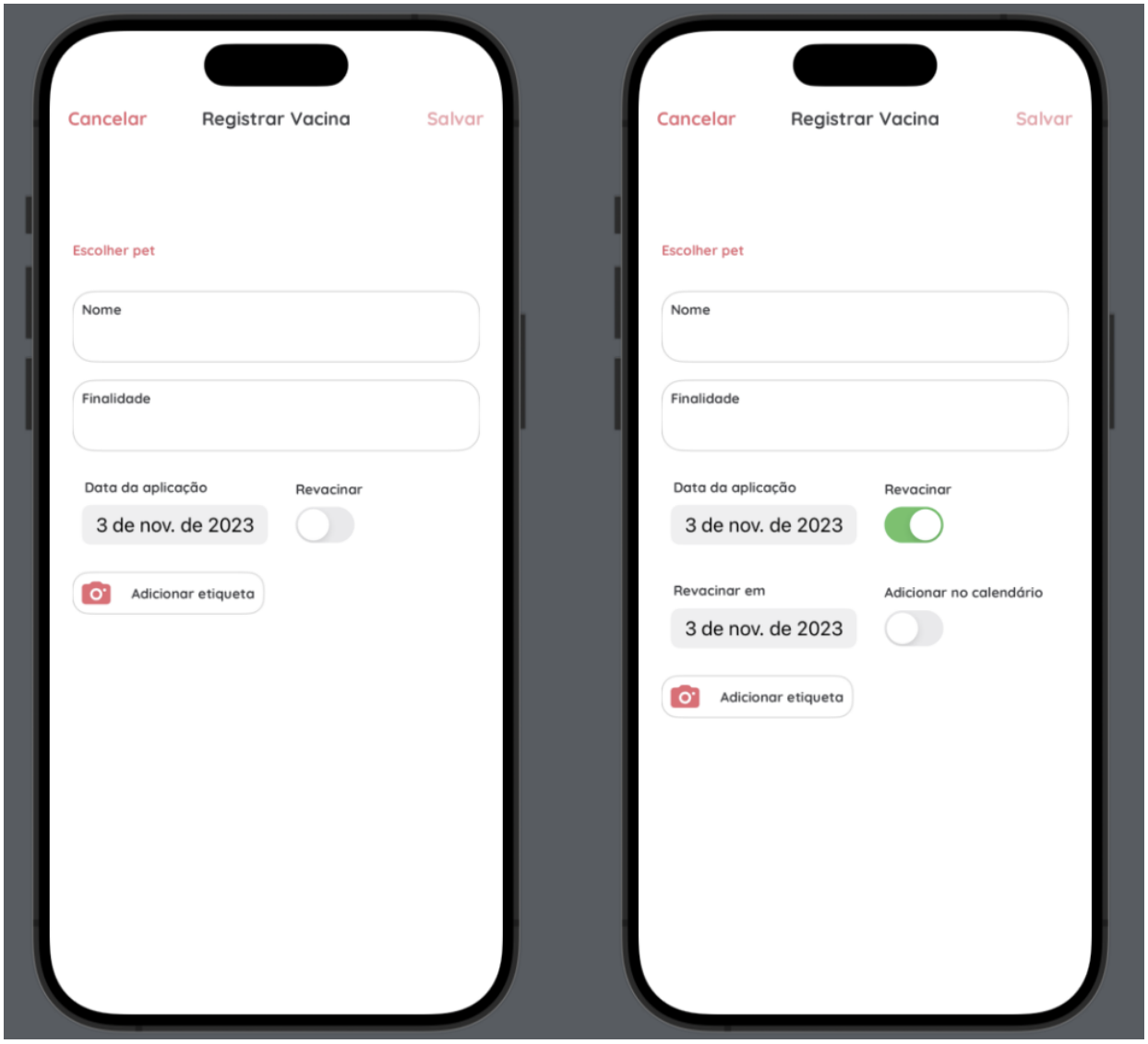
Fonte: Captura de tela do simulador presente no *software Xcode* feita pelo autor.

4.2.2 Cadastro de Vacina

No cadastro de vacina o usuário pode realizar o cadastro de uma aplicação de vacina. Nessa etapa, o usuário pode preencher o nome da vacina, data da aplicação, qual a finalidade da vacina e em qual *pet* foi feita a aplicação da vacina. Apenas esses dados são obrigatórios para o cadastro da vacina e uma vez preenchidos o botão de salvar é habilitado, porém, o usuário pode adicionar o selo, uma imagem da embalagem da vacina para manter no registro caso seja necessário futuramente. Além disso, o usuário pode selecionar a opção de revacinar, ao clicar nessa opção é habilitado o campo de data da reaplicação. Esse dado é importante para casos em que o *pet* precisa tomar mais de uma dose da vacina, geralmente doses de reforço. Na figura

16, é possível visualizar a tela de cadastro de vacina com a opção de revacinar desabilitada e habilitada.

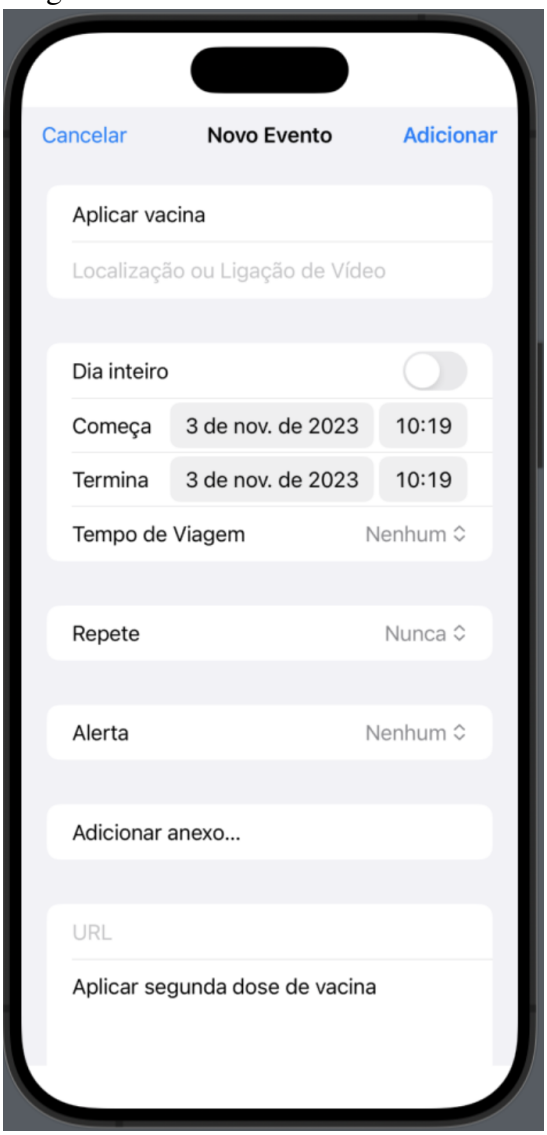
Figura 16 – Tela de cadastro de vacina com a opção de revacinar desabilitada à esquerda e habilitada à direita.



Fonte: Captura de tela do simulador presente no *software Xcode* feita pelo autor.

Caso o usuário tenha preenchido a data de reaplicação, após clicar em salvar, é exibida uma tela de cadastro de evento. A Figura 17 demonstra a tela de cadastro de evento. Nessa etapa, é possível adicionar um evento de lembrete ao calendário do usuário que servirá de lembrete para a reaplicação da vacina. Alguns dados dessa etapa como nome do evento e a data são preenchidos previamente com base nos dados da vacina, porém antes de salvar o usuário pode modificar esses dados e adicionar lembretes ou adicionar esse dado a um calendário específico.

Figura 17 – Tela de cadastro de evento.

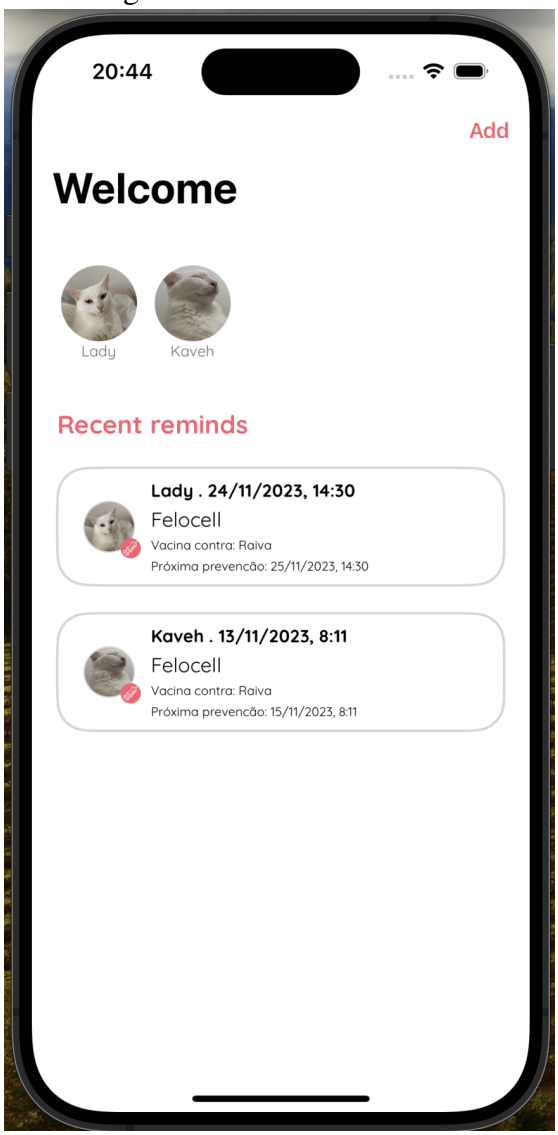


Fonte: Captura de tela do simulador presente no *software Xcode* feita pelo autor.

4.2.3 Tela de entrada

A figura 18 ilustra a tela de entrada, essa tela é exibida assim que a aplicação é aberta. Nela são exibidos ícones com imagens dos *pets* e uma lista de lembretes próximos e lembretes recentes. Os lembretes próximos é uma lista de futuras reaplicações de vacina, enquanto lembretes recentes mostram a lista de aplicações de vacina que foram feitas anteriormente.

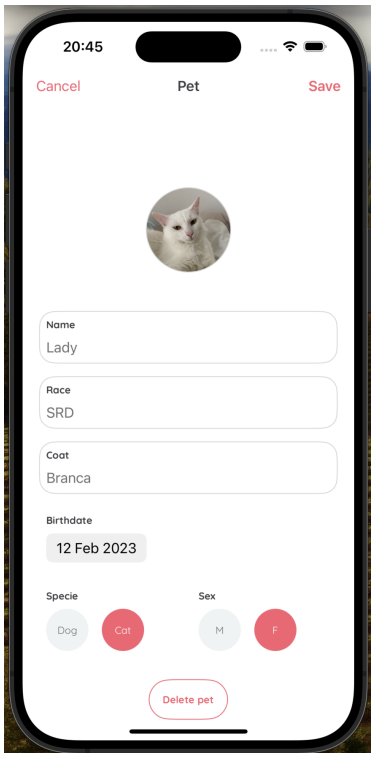
Figura 18 – Tela de entrada.



Fonte: Captura de tela do simulador presente no *software Xcode* feita pelo autor.

Ao clicar no ícone de um *pet*, as listas são filtradas para exibir apenas os lembretes do *pet* selecionado. O usuário pode clicar novamente no ícone para desabilitar o filtro. Caso o usuário clique duas vezes no ícone, é exibida a tela de edição do *pet*, onde ele pode atualizar ou deletar o cadastro do *pet*, caso haja vacinas cadastradas para o *pet* deletado, os registros de vacinas também são deletados. Clicar em uma das vacinações listadas irá abrir a tela de edição da vacina, similar à tela de edição de *pet*, o usuário pode editar os dados e/ou deletar o cadastro. A Figura 19 demonstra a tela de edição do cadastro de *pet*.

Figura 19 – Tela de editar cadastro de *pet*.



Fonte: Captura de tela do simulador presente no *software Xcode* feita pelo autor.

Na tela de entrada, se o usuário clicar e segurar por 4 segundos é habilitada a deleção múltipla, onde o usuário pode deletar diversos cadastros (*pets* ou vacinas). Quando a deleção múltipla é ativada, um ícone para deletar é exibido em cada um dos componentes visuais dos itens cadastrados, ao clicar neles o usuário pode deletar o item em questão. A Figura 20 demonstra a tela de entrada com a deleção múltipla ativada.

Figura 20 – Lista de vacinas com a opção de deleção múltipla ativa.

Fonte: Captura de tela do simulador presente no *software Xcode* feita pelo autor.

5 CONCLUSÃO

Visando contribuir com a área acadêmica através da revisão de literatura, este trabalho permitiu a construção de uma aplicação *mobile* para registro de histórico de vacinação. Nesse contexto foram apresentados os conceitos de desenvolvimento *mobile* e processos de desenvolvimento, além de permitir que fosse elencada a melhor metodologia de desenvolvimento para atingir este objetivo e com isso demonstrar o processo de desenvolvimento utilizando esta metodologia.

Como contribuição, o *iNimal*, uma vez disponibilizado para *download*, também pode facilitar o processo de gerenciamento da vacinação de *pets*, trazendo junto a melhoria na qualidade de vida de vários pets. Uma das maiores limitações foi o processo de lançamento na loja digital de produtos *Apple*, a *App Store*, pois ela exige uma conta de desenvolvedor que possui um plano anual com custo de 99 (noventa e nove) dólares, abaram por impossibilitar o lançamento até a publicação deste trabalho. Porém o aplicativo poderá ser disponibilizado ao repositório acadêmico para *download* pelos interessados. Outra limitação para o desenvolvimento da aplicação foi que ela foi desenvolvida apenas pelo autor.

A experiência e aprendizado permitiram não só a elaboração deste trabalho, mas também o desenvolvimento de uma aplicação escalável e que utiliza das mais recentes tecnologias de desenvolvimento *iOS*, como os *frameworks* citados.

Como trabalhos futuros, recomenda-se o uso das metodologias citadas no desenvolvimento de aplicações nativas *Android* ou híbridas e expansão das funcionalidades do aplicativo. Também se destaca a importância de seguir a etapa de comercialização do produto desenvolvido neste trabalho, uma vez que não foi possível realizar essa atividade.

Conforme a liberação para usuários reais seja feita, poderá ser feita a expansão das funcionalidades da aplicação, além da coleta de *feedbacks* de tutores de animais e profissionais da saúde.

Como etapas futuras, a aplicação continuará sendo expandida para que os donos de pets tenham todo o controle de vacinação, consultas e outros dados médicos dos *pets* disponíveis em seu *smartphone*. Além disso, como próximos passos para melhoria da experiência da aplicação como uma carteira de vacinação digital, a aplicação irá:

- Exibir uma lista de sugestões de nomes de vacinas utilizando como referência as vacinas mais comuns e cadastros anteriores para que o usuário não precise preencher o nome novamente sempre que precisar cadastrar uma nova;

- Persistir em nuvem os dados dos *pets*. Dessa forma, o usuário poderá ter a segurança de que não perderá os dados dos pets se precisar trocar de celular ou caso apague o aplicativo;
- Permitir ao usuário exportar um arquivo PDF com os dados do *pet* e o histórico de vacinação;
- Enviar notificações de etapas de vacinação pública que podem ser de interesse do usuário.

REFERÊNCIAS

AMARO, F. do P. A.; MACZUGA, J. M.; CARO, L. F. A vacinologia em cães e gatos. **Archives of Veterinary Science ISSN 1517-784X**, v. 21, n. 1, p. 01–10, 2016.

APPLE INC. ***What are frameworks?*** 2013. Disponível em: <https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html>. Acesso em: 13 de Outubro de 2023.

APPLE INC. ***Swift - Apple*** **(BR**. 2016. Disponível em: <http://www.apple.com/br/swift/>.> Acesso em: 27 de Agosto de 2023.

APPLE INC. ***Implementing an Inline PhotoPicker***. 2022. Disponível em: <https://developer.apple.com/documentation/photokit/implementing_an_inline_photos_picker>. Acesso em: 12 de Setembro de 2023.

APPLE INC. ***Create an Xcode project for an app***. 2023? Disponível em: <https://developer.apple.com/documentation/xcode/creating-an-xcode-project-for-an-app>. Acesso em: 12 de Setembro de 2023.

APPLE INC. ***Driving changes in your UI with state and bindings***. 2023? Disponível em: <https://developer.apple.com/tutorials/swiftui-concepts/driving-changes-in-your-ui-with-state-and-bindings>. Acesso em: 12 de Setembro de 2023.

APPLE INC. ***EventKit UI***. 2023? Disponível em: <https://developer.apple.com/documentation/eventkitui>. Acesso em: 28 de Julho de 2023.

APPLE INC. ***SwiftData***. 2023. Disponível em: <https://developer.apple.com/xcode/swiftdata>. Acesso em: 13 de Outubro de 2023.

BRASIL, C. G. da Internet no. **TIC DOMICÍLIOS 2019**. 2020. Disponível em: <https://cetic.br/media/analises/tic_domicilios_2019_coletiva_imprensa.pdf>. Acesso em: 15 de Agosto de 2023.

BRASIL, M. da S. **COBERTURA VACINAL de CÃES E GATOS**. 2022. Disponível em: <https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/r/raiva/cobertura-vacinal-de-caes-e-gatos>. Acesso em: 15 de Agosto de 2023.

EL-KASSAS, W. S.; ABDULLAH, B. A.; YOUSEF, A. H.; WAHBA, A. M. *Taxonomy of Cross-Platform Mobile Applications Development Approaches*. **Ain Shams Engineering Journal. ISSN 2090-4479**, 2015.

FERREIRA, A. C. B. H. **Qual a importância das vacinas?** 2022. Disponível em: <https://unilavras.edu.br/2021/05/14/importancia-das-vacinas-vacinacao>. Acesso em: 20 de Agosto de 2023.

FLORA, H.; CHANDE, S. V. *A REVIEW AND ANAYSIS ON MOBILE APPLICATION DEVELOPMENT PROCESSES USING AGILE METHODLOGIES*. **International Journal of Research in Computer Scienc**, 2013.

GOADRICH, M. P. R. M. H. *Smart smartphone development: ios versus android*. **SIGCSE '11**, p. 607–612, 2011.

GOOGLE INC. ***Crazy 8s - Design Sprint Kit***. 2023? Disponível em: <https://designsprintkit.withgoogle.com/methodology/phase3-sketch/crazy-8s>. Acesso em: 11 de Julho de 2023.

IPB. **Censo *Pet* IPB**. 2022. Disponível em: <https://institutopetbrasil.com/fique-por-dentro/amor-pelos-animais-impulsiona-os-negocios-2-2>. Acesso em: 11 de Agosto de 2023.

LIMA, M. **Brasil é o terceiro país com mais *pets***. 2022. Disponível em: <https://forbes.com.br/forbes-money/2022/10/brasil-e-o-terceiro-pais-com-mais-pets-setor-fatura-r-52-bilhoes/>. Acesso em: 11 de Agosto de 2023.

REIS, A. C. S. dos. Um estudo comparativo entre modelos de desenvolvimento de aplicações móveis. An optional note. 2019.

SYROMIATNIKOV, A.; WEYNS, D. *A Journey through the Land of Model-View-Design Patterns*. **IEEE/IFIP Conference on Software Architecture**, p. 21–30, 2014.

Tuist Inc. ***Get Started***. 2023. Disponível em: <https://docs.tuist.io/tutorial/get-started>. Acesso em: 20 de Outubro de 2023.