PROTÓTIPO DE APLICATIVO PARA AUXÍLIO AO ATENDIMENTO DOMICILIAR POR MÉDICOS VETERINÁRIOS

Vitor Silva Mello Martins, Geraldo Henrique Neto

Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

vitor.martins4@fatec.sp.gov.br,
geraldo.henrique01@fatec.sp.gov.br

Resumo. *Este artigo descreve um protótipo de aplicação para auxiliar o atendimento domiciliar por médicos veterinários, através de um prontuário veterinário. Para a realização do protótipo foram detalhadas a engenharia de requisitos através dos diagramas de classe, de caso de uso e do modelo entidade-relacionamento que foram feitas nos softwares Astah UML e BrModelo. As telas do protótipo foram feitas em aplicação Just in Mind. A aplicação começou a ser desenvolvida com o framework Flutter, com arquitetura MVC (Model-View-Controller) e seguirá com a implementação da estrutura de comunicação entre a aplicação e o banco de dados, além de melhorias relativas às regras de negócios e aparência da aplicação.*

Abstract*. This paper describes a prototype for an application tool that helps veterinarians as a veterinary record. For the prototype, the requirements engineering was stablished through the use case diagram, the class diagram and the entity relationship diagram were sketched using Astah UML software and the software BrModelo. All prototype screens were developed using the Just in Mind application. The application is being developed with Flutter framework, with MVC architecture (Model-View-Controller) and it will continue with the implementation of the communication structure between the app and the database (backend), as well as improvements in the business model and the app's front end.*

1. Introdução

A revolução dos smartphones, iniciada na década de 2000 com os celulares *palmtops* e depois com os *smartphones*, trouxe ao mundo soluções de *software* que antes só eram possíveis através dos computadores de mesa. Com o aprimoramento dos aparelhos celulares e das tecnologias de desenvolvimento, é possível desenvolver soluções customizadas de alta performance que atendem diferentes mercados e resolvem necessidades que eram limitadas pela tecnologia da época. Um desses mercados é o de médicos veterinários que atendem somente à domicílio e necessitam de um prontuário veterinário que seja de fácil acesso e contenha todas as informações necessárias para um atendimento eficaz e eficiente. Com esse problema em questão, este trabalho pretende sanar essa necessidade através do desenvolvimento de um protótipo de aplicativo móvel de prontuário veterinário.

2. Referencial Teórico

Nesta seção é apresentado o referencial teórico utilizado neste projeto, com definições dos termos chave e explicações sobre as ferramentas utilizadas.

- **Prontuário Veterinário**

A definição de prontuário de acordo com (SANTERAMO, TREMORI, SIQUEIRA, 2021) é:

> O prontuário é um documento que compila toda a história clínica de um paciente. Essa história clínica consiste na estruturação organizada dos procedimentos médicos realizados incluindo todos os documentos pertinentes ao atendimento do animal e apresentando todas as fases da evolução do atendimento clínico, desde as primeiras manifestações clínicas observadas, os exames solicitados e procedimento cirúrgico, quando executado, dentre outros.

O prontuário na medicina veterinária é definido como a documentação produzida na prática da clínica veterinária, que serve como fonte de informação sobre o paciente (HAMMERSCHMIDT, 2017). Entretanto, quando o prontuário digital é utilizado, o software disponível deverá permitir a elaboração de cópias de segurança e possuir acesso restrito e protegido por senhas (SANTERAMO, TREMORI, SIQUEIRA, 2021). É de suma importância, portanto, que a aplicação deverá respeitar e cumprir a elaboração de cópias quanto restringir acesso através do uso de senhas.

- **Framework Flutter**

Em relação ao *framework* Flutter para desenvolvimento *mobile*, este foi escolhido pois o Flutter é o *kit* de ferramentas de interface do Google para desenvolvimento de aplicações bonitas e nativamente compiladas em dispositivos móveis, na web, *desktop* e dispositivos embarcados utilizando um único código-fonte (FLUTTER, 2021).

Uma vez que a plataforma Flutter aborda o desenvolvimento de aplicações de forma híbrida, isto é, permitir criar aplicações móveis, *desktop* e web utilizando apenas um único código fonte, ela se torna uma ferramenta muito poderosa pois alcança grande escalabilidade com pouco esforço. O desenvolvimento de aplicações híbridas pode ter três abordagens (TAQTILE, 2016): encapsular um sistema web em containers nativos e apresentá-los como aplicativos (são chamados de progressive web apps ou simplesmente PWA); escrever a aplicação em determinada linguagem e então este ser traduzido para a linguagem nativa da plataforma; e por fim um misto entre a segunda opção e código nativo da plataforma.

- **Arquitetura Flutter**

De acordo com a documentação oficial, as aplicações desenvolvidas em Flutter, durante a fase de desenvolvimento, são executadas em uma Virtual Machine (VM) que possibilita uma recompilação rápida apenas dos componentes que sofreram mudanças, não necessitando reconstruir toda a aplicação. Entretanto, na etapa de *releasing* (lançamento da aplicação) todo o código é compilado diretamente para

código de máquina (intel x64, ARM ou JavaScript). O Flutter é construído com C, C++, Dart e Skia (um motor gráfico de renderização 2D) e, como mostrado na Figura 1, possui uma arquitetura de camadas.

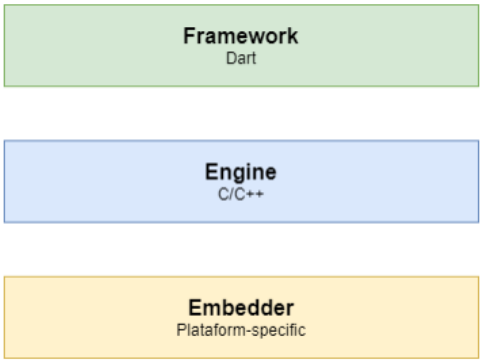

Figura 1. Arquitetura do Flutter

Fonte: Adaptado da Documentação Oficial

Na camada mais baixa do diagrama está o *Embedder*. Cada plataforma possui um *Embedde*r específico que pode ser escrito em Java e C++ para Android, Objective C/Objective-C++ para iOS e macOS, e C++ para Windows e Linux. O *Embedder* provê um ponto de entrada para a aplicação e é responsável por coordenar acesso a serviços do sistema operacional, como: renderização, acessibilidade, entrada de dados e gerencia o laço de eventos (Event Loop).

Ainda tomando como referência a Figura 1, tem-se a *Engine* ou também chamada de *Flutter Engine*. Este componente, em sua maior parte, é escrito em C++ e tem as primitivas necessárias para dar suporte a todas aplicações Flutter. Este mecanismo é responsável por rasterizar os componentes da tela sempre que for preciso atualizar a interface. Nesta etapa está incluído a implementação de baixo nível da API do Flutter, como gráficos através do Skia, *layout* de textos, entrada e saída de arquivos, dados de rede etc. No topo do diagrama está o *Framework* Flutter. Este é o componente mais utilizado pelos desenvolvedores. A Figura 2 detalha os principais componentes da arquitetura Flutter.

Detalhando a parte de Framework de baixo para cima, temos:

- ***Foundation*:** provê abstrações para serem utilizadas por serviços como *Animation* (animação), *Painting* (pintura) e *Gestures* (gestos)

- ***Rendering*:** provê abstração para trabalhar com *layouts*. Nesta camada é possível criar uma árvore de componentes renderizados.

- ***Widgets*:** Nesta camada possibilita criar abstração de composição. Basicamente as classes utilizadas nesta camada serão tratadas como *render objects* na camada de renderização.

- **Material e Cupertino:** Esta camada provê um conjunto de *widgets* pré-definidos de acordo com Material e iOS design e quem podem ser utilizadas na criação de interfaces.

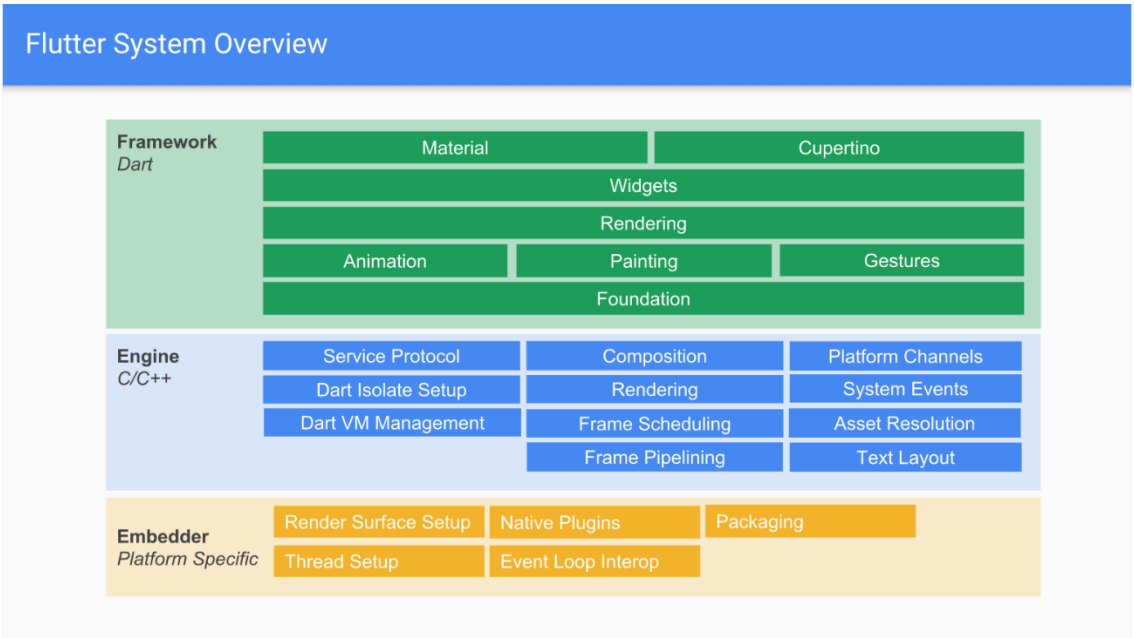

Figura 2. Componentes do Flutter

Fonte: Blog Zup, 2021

2.1 Engenharia de Software

Engenharia de software é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado. É, portanto, uma abordagem sistemática para a produção de software; ela analisa questões práticas de custo, prazo e confiança, assim como as necessidades dos clientes e produtores do software. De qualquer maneira, o software deve prover a funcionalidade e o desempenho requerido pelo usuário; além disso, deve ser confiável e fácil de manter (Sommerville, 2011)

Para este projeto, a aplicação desenvolvida será uma aplicação *stand-alone*. Essas são as aplicações executadas em um computador local, como um PC ou um *smartphone*. Elas contêm toda a funcionalidade necessária e não precisam estar conectadas a uma rede.

2.2 Sistemas de Informação

Os Sistemas de Informação – SI podem ser definidos como um conjunto estruturado de sistemas que estão interrelacionados e possuem algumas funções essenciais como, coletar, realizar a manipulação, armazenar de forma segura e disseminar os dados e informações. Todas as atividades, sejam elas realizadas na vida pessoal ou profissional, envolvem os SI, desde o manuseio de um e-mail, de sistemas bancários até a realização um pedido de compra para uma empresa.

Abaixo, um fluxograma do processo de funcionamento de um SI.

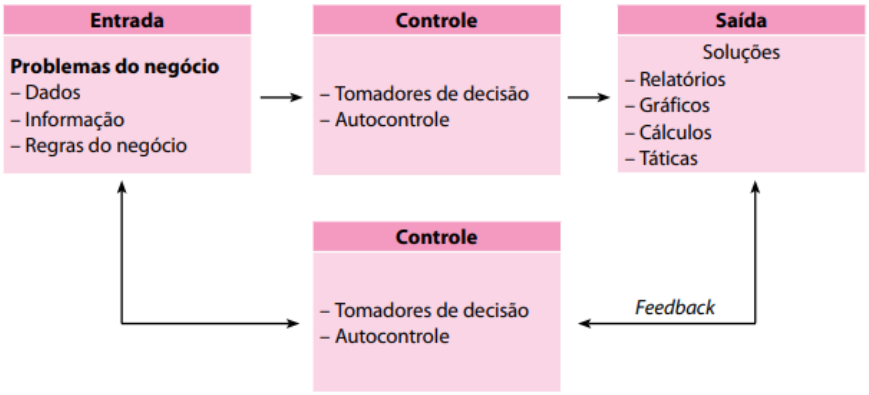

Figura 3. Fluxograma geral de um Sistema de Informação

Fonte: Gonçalves, 2015

A entrada de dados é considerada o início dos eventos que são capturados para executar os processos e as atividades. Por exemplo, para registrar os dados cadastrais de um cliente, é necessário que os dados sejam inseridos no sistema e assim armazenados no banco de dados, onde serão formados dados concretos sobre o perfil do cliente.

O processamento dos dados permitirá que dados registrados e armazenados no banco de dados sejam transformados em informação significativa, entendível para o usuário ou para o cliente. Como exemplo de processamento de dados podemos citar uma solicitação ou execução do sistema para gerar relatórios financeiros, cálculo de custos e despesas, relatórios de índices, classificação e conclusão de atividades desenvolvidas na organização. Todos os dados processados pelo sistema resultarão em uma resposta, como também serão transformados, convertidos em informações organizadas e entendíveis para o usuário da máquina.

O controle está envolvido com a entrada e processamento dos dados, possui como responsabilidade verificar se todos os procedimentos estão sendo seguidos e se o sistema está realizando seu objetivo principal. O controle nada mais é do que monitorar, analisar e identificar alguma anomalia ou algum problema com os dados de entrada e de processamento. Observando que o controle é um processo de responsabilidade do sistema, este possui componentes e configurações para apresentar como está o desempenho do sistema, em relação à entrada e processamento de dados.

A saída é um processo muito importante para os usuários e clientes dos sistemas, pois é através da saída de dados que podem ser analisados os relatórios, analisar a saúde da empresa e realizar novas estratégias de negócio. Esta atividade é uma sequência dos outros processos, pois primeiramente devem ser introduzidos, armazenados, processados e controlados os dados de sistemas, para que a informação chegue para o usuário. A informação de saída possui vários objetivos, como podem ser transmitidos através de várias formas, como por exemplo relatórios gerenciais, cálculos sobre a empresa, gráfico dos processos, atividades e produção, são alguns dos objetivos da saída de dados.

2.3 Banco de Dados

O sistema de gerenciamento de banco de dados (SGBD) é geralmente um conjunto de bibliotecas, aplicativos e utilitários, extremamente importante para o armazenamento, gerenciamento e manipulação dos dados. Ele também fornece facilidades para pesquisar e atualizar registros. (MATTHEW, STONES, 2005).

3. Resultados e Discussão

Nesta seção são detalhados o planejamento e escopo do projeto. Assim como os resultados dos principais requisitos para o sistema, e os diferentes diagramas que auxiliaram na implementação do prontuário.

3.1 Arquitetura

Para a arquitetura do sistema será utilizado o padrão MVC (Model – View – Controller) que separa o projeto do software em três camadas independentes: o modelo (manipulação da lógica de dados), a visão (a interface do usuário) e o controlador (fluxo de aplicação). Esta separação facilita a manutenção do código, que pode ser reutilizado em outros projetos (Figura 4).

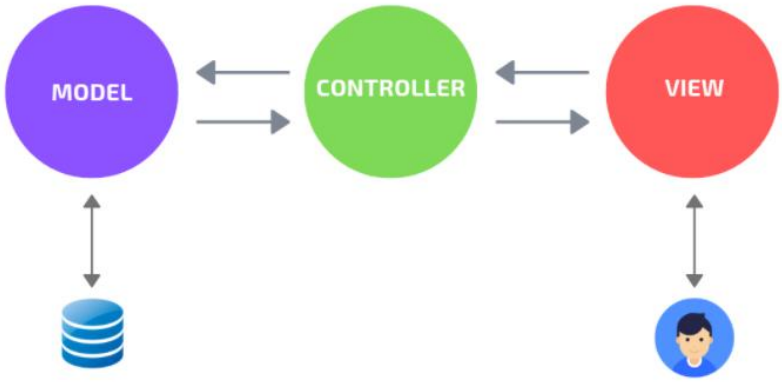

Figura 4. Fluxograma geral do padrão MVC

Fonte: Blog Treinaweb, 2020

O MVC é utilizado em muitos projetos devido que sua arquitetura possibilita a divisão do projeto em camadas muito bem definidas. A utilização do padrão MVC traz como benefício o isolamento das regras de negócios da lógica de apresentação, que é a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem a necessidade de alterar as regras de negócios, proporcionando muito mais flexibilidade e oportunidades de reuso das classes.

3.2 Engenharia de Requisitos

O sistema consiste no gerenciamento de um prontuário veterinário eletrônico e no gerenciamento dos dados cadastrais de seus respectivos proprietários. O sistema deve emitir

diversos tipos de relatórios e consultas, possibilitando um melhor gerenciamento dessas informações. Os principais requisitos estão listados na seção 3.2.1 – Documento de Requisitos

3.2.1 Documento de Requisitos

A seguir estão listados os requisitos funcionais e não funcionais do Prontuário Veterinário.

3.2.1.1 Requisitos Funcionais

1. O sistema deve permitir a inclusão, alteração e remoção de pacientes, com os seguintes atributos: nome do tutor, nome do paciente, raça, data de nascimento, vacinas, sexo, pelagem, histórico de atendimento e identificação.

2. O sistema deve permitir a inclusão, alteração e remoção dos tutores com os seguintes atributos: nome, endereço, telefone, data de nascimento, e-mail, animais.

3. O sistema deve permitir a inclusão, alteração e remoção de fotos e documentos como receitas médicas, imagens de anamnese, vídeos e/ou quaisquer arquivos que complementem o prontuário.

4. O sistema deve permitir a inclusão, alteração e remoção de medicamentos e vacinas de seu banco de dados.

5. O sistema deve permitir a consulta dos dados cadastrais de pacientes e tutores, junto do histórico de atendimento.

6. O sistema deve permitir a impressão de um relatório contendo os pacientes e as vacinas em atraso no período (por exemplo, semanal, quinzenal ou mensal), contendo, para cada dia do período, o nome do paciente e do respectivo tutor, o telefone, o e-mail, o tipo da vacina, a data da última aplicação e a data prevista para nova vacinação.

7. O sistema deve permitir a consulta de medicamentos e vacinas, contendo nome, nome comercial, princípio ativo, dosagem entre outros.

8. O sistema deverá possibilitar o agendamento de vacinas, avisando com antecedência a data de vacinação, além de também avisar em caso de vacinação atrasada.

9. O sistema deve fornecer facilidades para a realização de backups dos arquivos do sistema.

10.O sistema deve possuir senhas de acesso e identificação para dois tipos de usuários: administrador do sistema e veterinário

3.2.1.2 Requisitos Não Funcionais

C1. Confiabilidade

11. O sistema deve ter capacidade para recuperar os dados perdidos da última operação que realizou em caso de falha.

C2. Eficiência

12. O sistema deve responder a consultas em menos de 5 segundos.

C3. Portabilidade
13. O sistema deve ser executado em smartphones Android, no mínimo versão 6. O sistema deve ser capaz de armazenar os dados em base de dados sequenciais

3.2.2 Diagramas

A continuação segue uma descrição detalhada dos diferentes diagramas necessários para projetar o Prontuário Veterinário. Todos os diagramas foram desenvolvidos através do *software* Astah UML.

3.2.2.1 Diagrama de Classes

A figura 5 abaixo detalha as relações entre diferentes classes, objetos e seus atributos. No caso do prontuário as principais classes serão o tutor, animal, veterinário, consulta e medicamentos.

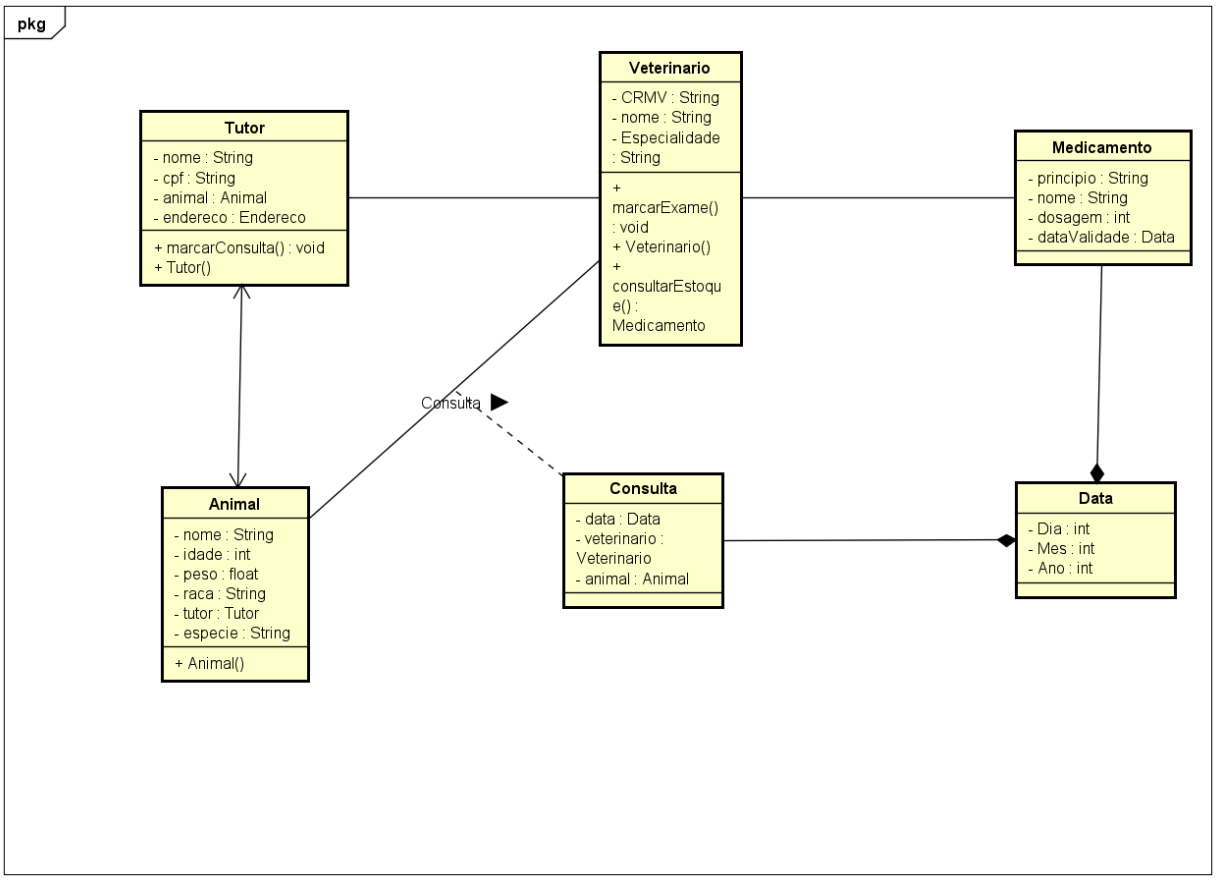

Figura 5. Diagrama de Classes
Fonte: Desenvolvido pelo próprio Autor

3.2.2.2 Diagrama de Casos de Uso

Nele, temos os atores Veterinário, Cliente (tutor), e o Sistema, representados pelos bonecos-palito. Foram detalhados os casos de uso entre estes atores através das elipses amarelas. No diagrama de casos de uso temos que alguns casos possuem relação entre si, por exemplo, o caso Cadastrar Cliente obrigatoriamente irá fazer uma chamada para o caso de uso Cadastrar Animal, pois não há como ter um cadastro de cliente sem animal. O caso Prescrever Exame possui muitas semelhanças com o caso Prescrever Receita, por isso ele possui a relação *extend* (Figura 6).

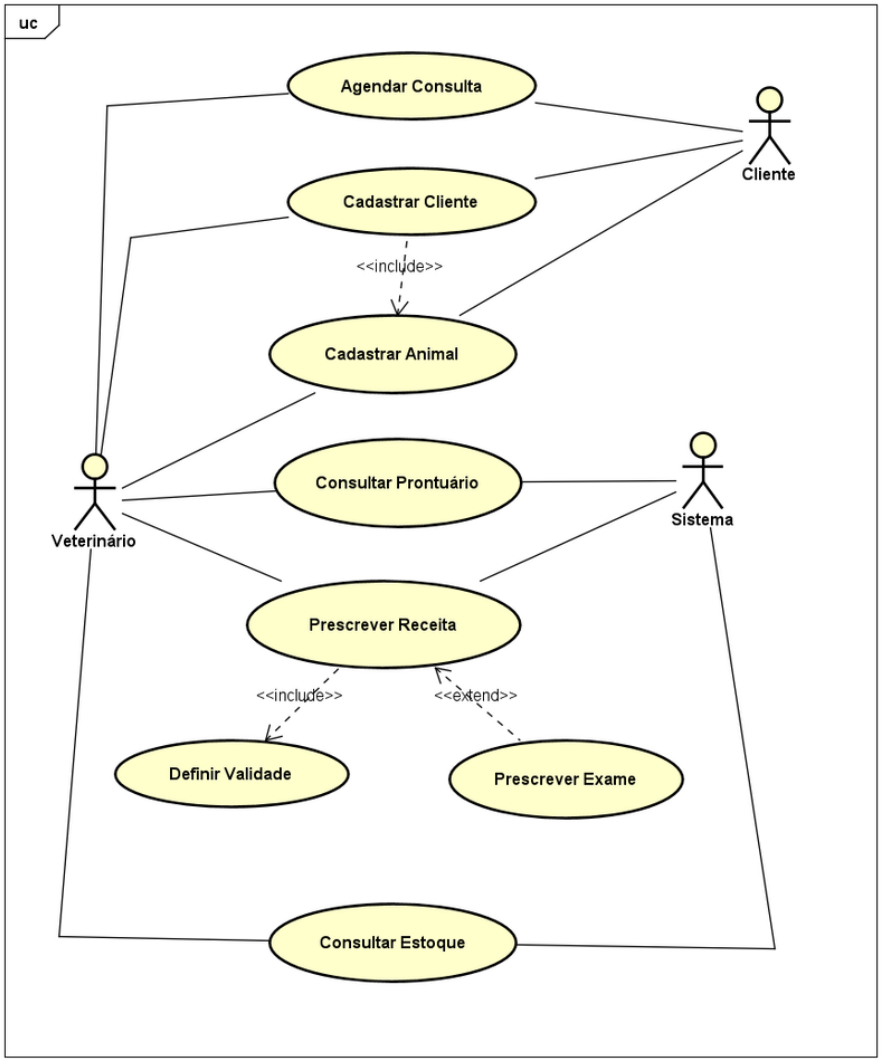

Figura 6. Diagrama de Caso de Uso
Fonte: Desenvolvido pelo próprio Autor

3.2.2.3 Diagrama de Entidade Relacionamento

A continuação segue o diagrama de Entidade Relacionamento do Prontuário Veterinário

(Figura 7) desenvolvido através do software BrModelo. As principais entidades são Tutor, Animal, Veterinário e Medicamentos, representadas pelos retângulos; já as relações são representadas através dos losangos. Cada círculo representa uma característica/atributo da entidade, sendo que os círculos em negrito são os atributos identificadores.

As relações sempre variam entre um valor mínimo e um valor máximo, representados pelos números a esquerda e a direita, dentro dos parênteses. Por exemplo, a relação Tutora entre o Tutor e o seu Animal, pode ser lida como "Um tutor tutora no mínimo um, no máximo N animais", enquanto a relação reversa pode ser lida como "Um animal é tutorado por exatamente um tutor".

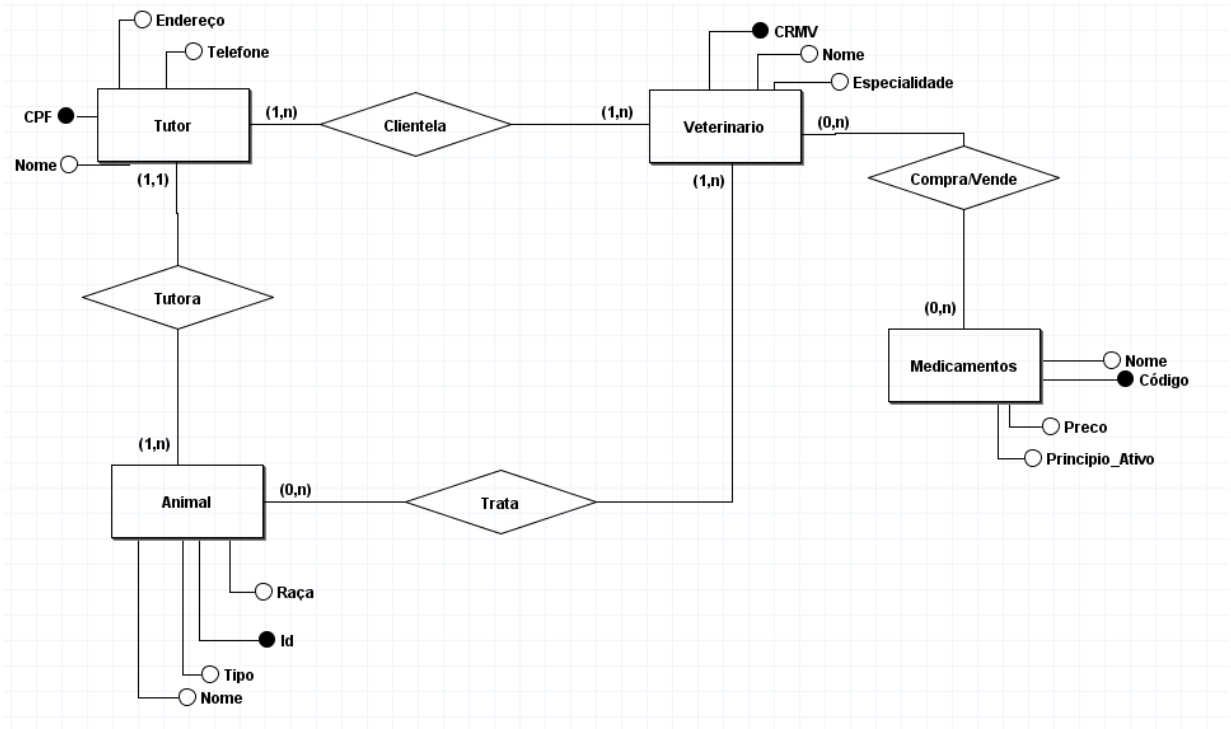

Figura 7. Diagrama de Entidade Relacionamento
Fonte: Desenvolvido pelo próprio Autor

3.3 Desenvolvimento

Inicialmente foi desenvolvido o protótipo de diferentes interfaces no *software* de prototipagem Just In Mind. Na figura 8 estão representadas a tela de login e do menu principal. Na tela de login será necessário que o usuário cadastre um e-mail e uma senha com no mínimo 8 dígitos para acesso ao sistema, conforme requisito de segurança. Na tela do menu principal, serão apresentadas informações gerenciais de consultas, estoque e solicitações de novos agendamentos.

Figura 8. Tela de Login e Tela do Menu Principal
Fonte: Desenvolvido pelo próprio Autor

Em seguida, na figura 9, temos as telas de busca de clientes e a tela de detalhes. Na tela de busca, é possível encontrar o cadastro dos clientes (tutores), onde é apresentado o nome do tutor e de seu(s) animal(is). Também, é possível adicionar um cadastro de cliente através do botão flutuante. Para consultar as informações cadastrais, detalhes, anamneses, dentre outras informações, é necessário clicar em um registro de cliente. Ao fazer isso, a tela de detalhes é apresentada.

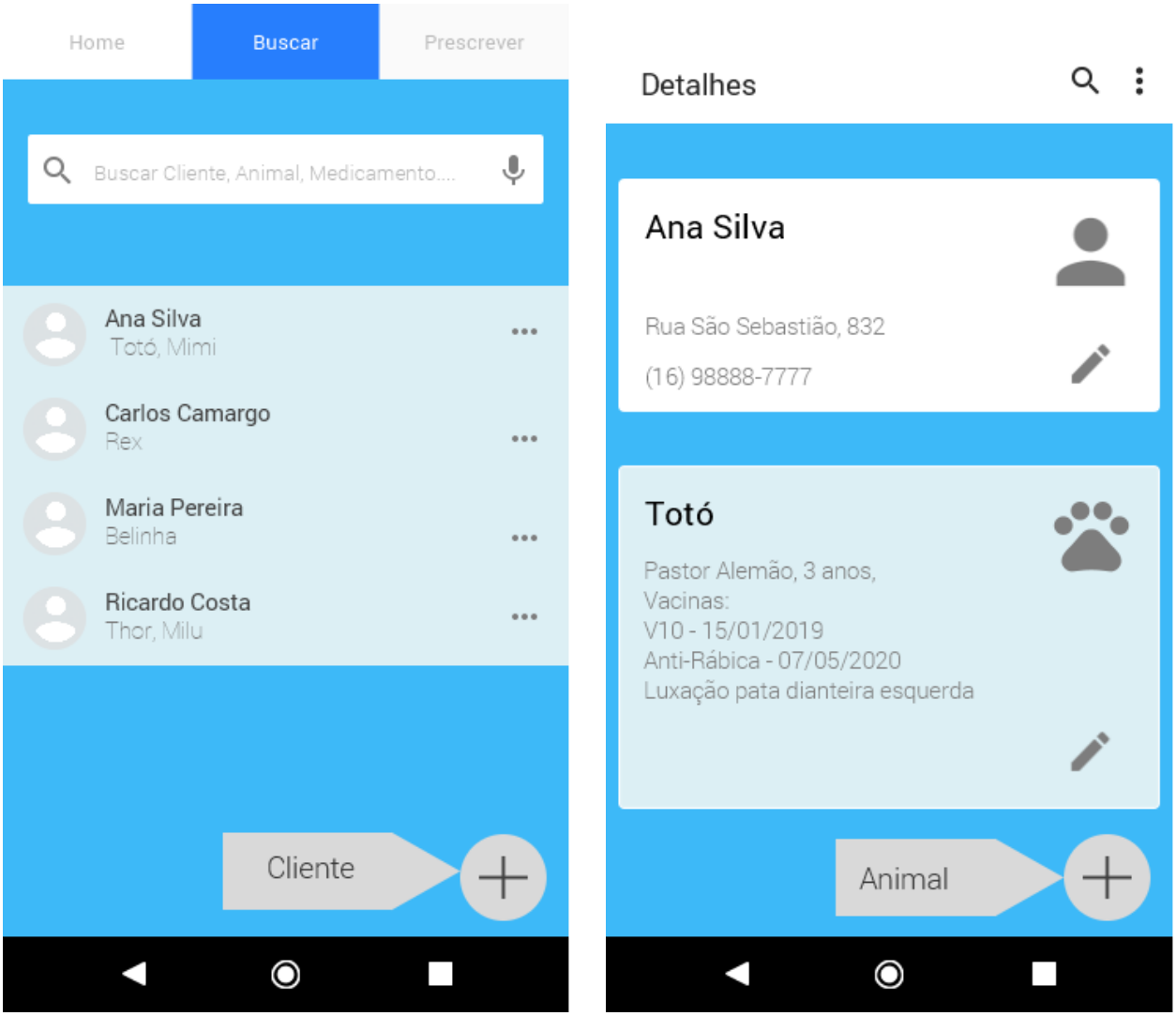

Figura 9. Tela de Busca de Clientes e Tela de Detalhes
Fonte: Desenvolvido pelo próprio Autor

A figura 10 abaixo detalha a tela de estoque dos medicamentos, consultada através da tela home. A tela de estoque mostra os medicamentos estocados, suas quantidades e data de vencimento. Também, é possível cadastrar um medicamento ao estoque. Já na tela de prescrição, é possível prescrever uma receita ou um exame e enviar para algum cliente cadastrado no banco de dados.

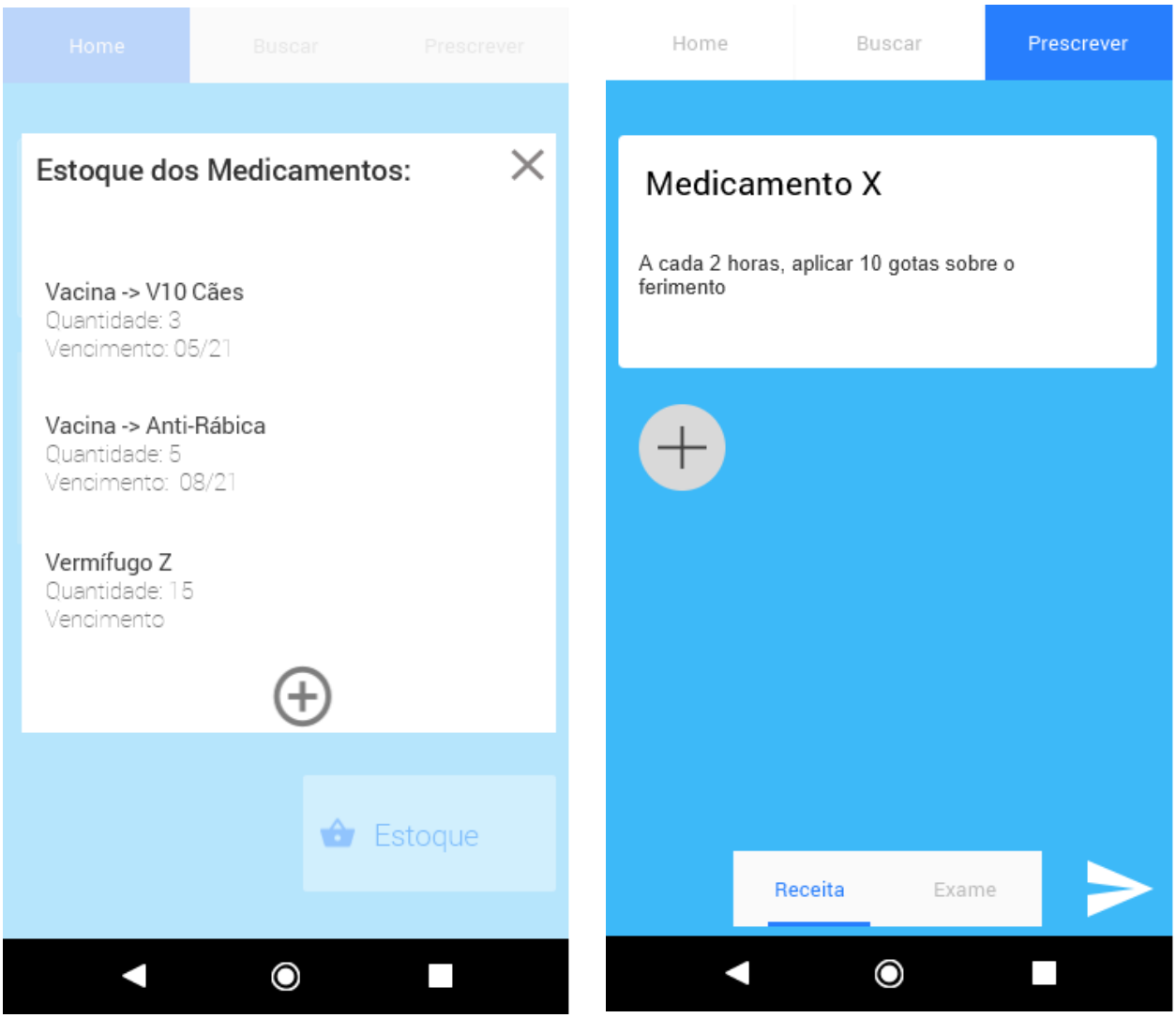

Figura 10. Tela de Estoque e Tela de Prescrição
Fonte: Desenvolvido pelo próprio Autor

Após a implementação do protótipo, foi iniciado o desenvolvimento do aplicativo através do framework Flutter (figura 11) e da interface de desenvolvimento Virtual Studio Code. Observa-se que um sistema de auxílio ao médico veterinário é viável de ser desenvolvido quando respeitado os requisitos estabelecidos e respeitando as boas práticas de programação.

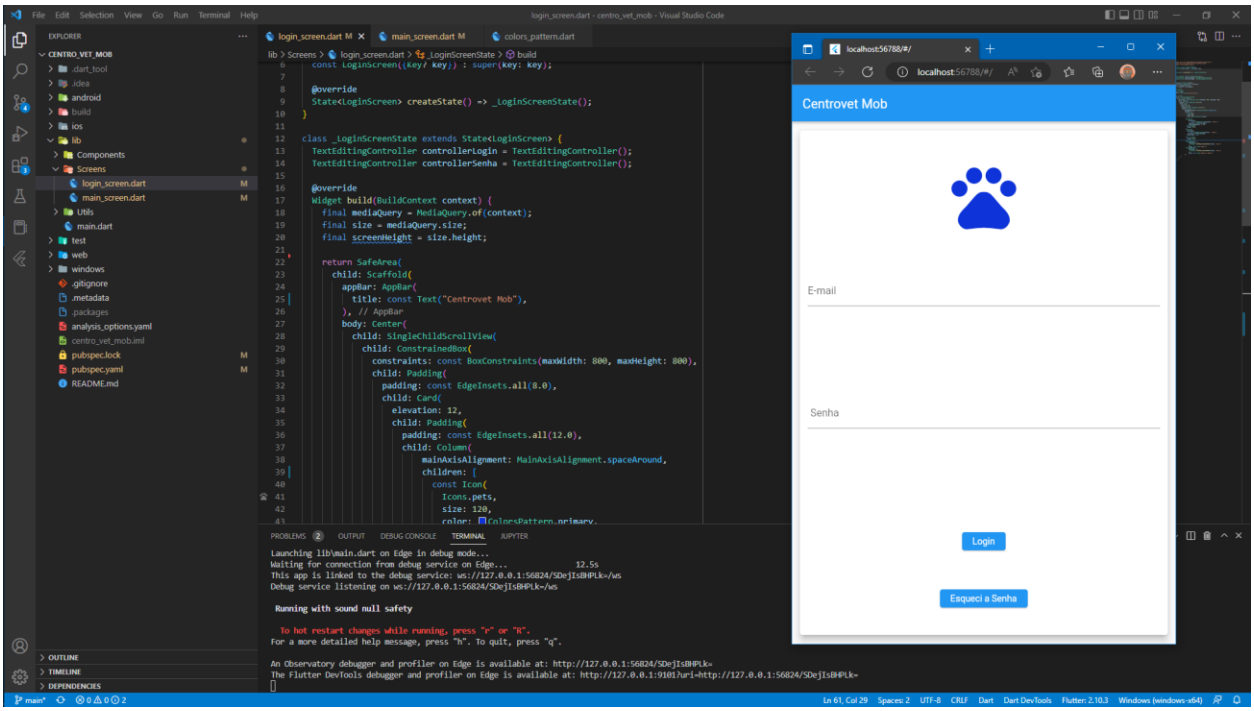

Figura 11. Tela de Login em Flutter
Fonte: Desenvolvido pelo próprio Autor

4. Conclusão

Neste trabalho foi possível fazer uma revisão ampla da parte teórica dos requisitos funcionais e não funcionais do prontuário veterinário. Além disso foi possível prototipar parte das telas do aplicativo. Para os próximos passos do desenvolvimento do aplicativo, serão realizadas as outras telas do protótipo, junto com a estrutura de comunicação entre a aplicação e o banco de dados, além de melhorias relativas às regras de negócios e aparência da aplicação.

5. Referências Bibliográficas

ASEF, Mazen. **Simples, mas complexo! O que é e como funciona o Node.js. TecMundo, 2019.** Disponível em: <https://www.tecmundo.com.br/mercado/137805-o-node-js.htm>. Acesso em: 19 ago. 2022.

FLUTTER, 2021. **Flutter home page.** Disponível em: https://flutter.dev, <Acesso em 23 de maio de 2022>

HAMMERSCHMIDT, J. O prontuário médico-veterinário: requisitos e importância. In: TOSTES, R. A; REIS, S. T; CASTILHO, V. V. **Tratado de Medicina Veterinária Legal.** 1. ed. Curitiba: MedVep, 2017. p. 120-126.

MATTHEW, Neil; STONES, Richard. **Beginning Databases with PostgreSQL From Novice to Professional**. 2. ed. Apress, 2005.

NODEBR. **O QUE É NODEJS?.** 2016. Disponível em: <http://nodebr.com/o-que-e-node-js/>. Acesso em 19 ago. 2022.

RED HAT. **O que é uma API? .** [S.D]. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Acesso em 22 maio 2022.

SANTERAMO J.; ROCHA T. M. T.; SIQUEIRA A. de. Aspectos técnicos, éticos e legais na elaboração do prontuário médico-veterinário**. Revista de Educação Continuada em Medicina Veterinária e Zootecnia do CRMV-SP**, v. 19, n. 1, 10 maio 2022.

TAQTILE. **Híbrido vs Nativo**. [S. L.]: Medium, 2016. Disponível em: <https://medium.com/taqtilebr/h%C3%ADbrido-vs-nativo-c8591df0dce6>. Acesso em: 19 ago. 2022