

NIVERSIDADE FEDERAL FLUMINENSE

KALEBE SILVA GOUVEA

**VETVAX MANAGER: DESENVOLVIMENTO DE UM SISTEMA DE
GERENCIAMENTO DE VACINAÇÃO ANIMAL**

NITERÓI

2024

KALEBE SILVA GOUVEA

**VETVAX MANAGER: DESENVOLVIMENTO DE UM SISTEMA DE
GERENCIAMENTO DE VACINAÇÃO ANIMAL**

Trabalho de Conclusão de Curso
submetido ao Curso de Tecnologia em
Sistemas de Computação da
Universidade Federal Fluminense como
requisito parcial para obtenção do título
de Tecnólogo em Sistemas de
Computação.

Orientador: Prof. Esp. Wellington de Souza Silva

NITERÓI

2024

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

G719v Gouvea, Kalebe Silva
VETVAX MANAGER: DESENVOLVIMENTO DE UM SISTEMA DE
GERENCIAMENTO DE VACINAÇÃO ANIMAL / Kalebe Silva Gouvea. -
2024.
74 f.: il.

Orientador: Wellington Silva.
Trabalho de Conclusão de Curso (graduação)-Universidade
Federal Fluminense, Instituto de Computação, Niterói, 2024.

1. Desenvolvimento de software. 2. Vacinação de animal. 3.
Sistema de banco de dados. 4. Projeto de base de dados. 5.
Produção intelectual. I. Silva, Wellington, orientador. II.
Universidade Federal Fluminense. Instituto de Computação.
III. Título.

CDD - XXX

KALEBE SILVA GOUVEA

**VETVAX MANAGER: DESENVOLVIMENTO DE UM SISTEMA DE
GERENCIAMENTO DE VACINAÇÃO ANIMAL**

Trabalho de Conclusão de Curso
submetido ao Curso de Tecnologia em
Sistemas de Computação da
Universidade Federal Fluminense como
requisito parcial para obtenção do título
de Tecnólogo em Sistemas de
Computação.

Este trabalho foi defendido e aprovado pela banca em 28/06/2024.

BANCA EXAMINADORA

Prof. Esp. Wellington de Souza Silva - Orientador
UFF - Universidade Federal Fluminense

Prof. Dr. Gustavo Silva Semaan - Avaliador
UFF - Universidade Federal Fluminense

Dedico este trabalho especialmente a
Deus.

AGRADECIMENTOS

Agradeço especialmente a Deus, pelo imenso cuidado, amor e suporte que me concedeu durante toda minha trajetória.

À minha esposa, Natália dos Santos Magalhães Gouvea, por todo apoio, cuidado e ajuda que me proporcionou ao decorrer da minha formação acadêmica.

Aos meus pais, Leandro da Cruz Gouvea e Antonia Marta Silva Gouvea e aos meus irmãos Ana Letícia Silva Gouvea e Nicolas Silva Gouvea por terem me acompanhado com muito amor durante toda minha trajetória de vida até aqui.

Ao meu orientador, Wellington Silva, pelo tempo disposto e direcionamento dados a mim durante esse trabalho.

“Não tentes ser bem-sucedido, tenta antes ser um homem de valor.”

Albert Einsten

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de gerenciamento de vacinação, semelhante ao *Conecte SUS* para animais domésticos e rural. A estruturação da base de dados relacional necessária para seu funcionamento até a escrita do código-fonte que implementa as interações do sistema. O objetivo inicial era desenvolver um sistema de gerenciamento de clínicas veterinárias com funcionalidades abrangendo todos os processos comuns em uma clínica. No entanto, devido às limitações de tempo, o escopo foi reduzido para focar exclusivamente no gerenciamento da vacinação, atendendo especificamente aos proprietários dos animais. Na primeira etapa, são apresentados exemplos de sistemas semelhantes de gerenciamento de clínicas médicas, utilizados como base para a elaboração do sistema veterinário. Nessa etapa, também são discutidos o padrão de projeto MVC, que foi adotado para o desenvolvimento. Além disso, os casos de uso elaborados e os requisitos funcionais e não funcionais derivados desses casos de uso. Na segunda etapa, são detalhados os elementos responsáveis pela construção do sistema, incluindo o diagrama de entidade-relacionamento, o mapeamento do modelo relacional derivado desse diagrama, a definição das tabelas do banco de dados, os componentes do código-fonte do sistema e as telas finais após a implementação.

Palavras-chave: Sistema de vacinação, projeto de banco de dados, MVC.

ABSTRACT

This work presents the development of a vaccination management system, similar to *Conecte SUS*, but for domestic and rural animals. The structuring of the necessary relational database up to writing the source code that implements the system interactions. The initial objective was to develop a veterinary clinic management system with functionalities covering all common processes in a clinic. However, due to time limitations, the scope was reduced to focus exclusively on vaccination management, specifically catering to animal owners. In the first stage, examples of similar medical clinic management systems, used as a basis for the veterinary system, are presented. This stage also discusses the MVC design pattern adopted for development, the use cases created, and the functional and non-functional requirements derived from these use cases. In the second stage, the elements responsible for building the system are detailed, including the entity-relationship diagram, the mapping of the relational model derived from this diagram, the definition of database tables, the system's source code components, and the final screens after implementation.

Keywords: Vaccination System, Database design, MVC.

LISTA DE FIGURAS

Figura 1 - Tela de Dashboard - iClinic.....	18
Figura 2 - Tela de agendamento de consultas - iClinic.....	20
Figura 3 - Tela de prontuário eletrônico - iClinic	21
Figura 4 - Tela de teleatendimento - iClinic.....	22
Figura 5 - Tela de gestão de clínica - iClinic	23
Figura 6 - Tela de agendamento de consultas - Apolo	25
Figura 7 - Tela de prontuário eletrônico - Apolo.....	25
Figura 8 - Tela de gerenciamento financeiro - Apolo.....	26
Figura 9 - Tela de envio de e-mails personalizados - Apolo.....	26
Figura 10 - Diagrama de casos de uso completo.....	34
Figura 11 - Diagrama de casos de uso restringido	35
Figura 12 - Diagrama Entidade Relacionamento	40
Figura 13 – Diagrama ER gerado pelo DBeaver.....	48
Figura 14 - Consulta de todos os animais pertencentes a um proprietário.....	51
Figura 15 - Consulta de vacinas disponíveis por espécie.....	51
Figura 16 - Consulta de cartão de vacina por animal	52
Figura 17 - Vacinação agendada para um animal	52
Figura 18 - Lembrete recebido por e-mail após execução do script	54
Figura 19 - Tela de login do usuário	57
Figura 20 - Tela de cadastro do usuário.....	57
Figura 21 - Tela inicial exibindo os animais do proprietário.....	57
Figura 22 - Tela de cadastro de animal	58
Figura 23 - Tela de detalhes do animal	58
Figura 24 - Tela de alteração de dados do animal.....	59
Figura 25 - Tela de vacinas agendadas e aplicadas do animal.....	60
Figura 26 - Tela de detalhes de vacinação agendada	60
Figura 27 - Tela de agendamento de vacinação.....	61
Figura 28 - Tela de detalhes da vacina aplicada.....	61
Figura 29 - Tela de registro de vacina no cartão de vacinação do animal.....	62
Figura 30 - Tela de alteração de dados de vacina agendada para o animal	62
Figura 31 - Tela de alteração de dados de vacina aplicada no animal	63
Figura 32 - Tela de cartilha de vacinação disponível por espécie.....	63

Figura 33 - Tela de alteração de dados do proprietário	63
--	----

LISTA DE TABELAS

Tabela 1 - Algoritmo de base de dados.....	37
--	----

LISTA DE ABREVIATURAS E SIGLAS

DER	Diagrama Entidade Relacionamento
MER	Modelo Entidade Relacionamento
MVC	<i>Model-View-Controller</i>
MVP	<i>Model-View-Presenter</i>
MVVM	<i>Model-View-ViewModel</i>
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	<i>Structured Query Language</i>
TI	Tecnologia da Informação
UI	Interface de Usuário
XAML	<i>Extensible Application Markup Language</i>

SUMÁRIO

1	INTRODUÇÃO	15
2	CONCEITOS E REVISÃO DE LITERATURA.....	17
2.1	O SISTEMA ICLINIC	18
2.2	O SISTEMA NINSAÚDE APOLO.....	23
2.3	COMPARAÇÃO ENTRE SISTEMAS AVALIADOS.....	27
2.4	DIFERENCIAIS PROPOSTOS PARA UM NOVO SOFTWARE	29
2.5	LIMITAÇÕES DO SISTEMA PROPOSTO	30
2.6	ARQUITETURA DE SOFTWARE	31
2.7	CASOS DE USO	33
2.8	REQUISITOS PARA UM SISTEMA DE GERENCIAMENTO DE VACINAÇÃO ANIMAL	35
2.8.1	Requisitos funcionais	36
2.8.2	Requisitos não funcionais.....	36
2.9	PROJETO DE BASE DE DADOS.....	37
3	SISTEMA PARA GERENCIAMENTO DE VACINAÇÃO ANIMAL.....	39
3.1	DIAGRAMA ENTIDADE-RELACIONAMENTO	39
3.1.1	Entidades	40
3.1.1.1	<i>Usuário.....</i>	40
3.1.1.2	<i>Proprietário</i>	41
3.1.1.3	<i>Animal</i>	41
3.1.1.4	<i>Espécie</i>	41
3.1.1.5	<i>Cartilha.....</i>	42
3.1.1.6	<i>Vacina</i>	42
3.1.1.7	<i>Agenda.....</i>	42
3.1.2	Relacionamentos	43
3.2	MAPEAMENTO DE MODELO RELACIONAL.....	44
3.3	DEFINIÇÃO DE TABELAS NO BANCO DE DADOS.....	44
3.4	INSERÇÃO DE DADOS E CONSULTAS.....	48
3.4.1	Inserções	48
3.4.2	Consultas.....	51
3.5	COMPONENTES DO SISTEMA.....	52
3.5.1	Models.....	53

3.5.2	Controllers.....	53
3.5.3	Views	53
3.5.4	Script de automatização de lembretes de vacinação	54
3.6	APRESENTAÇÃO DE TELAS DO SISTEMA	56
4	CONSIDERAÇÕES FINAIS.....	65
4.1	TRABALHOS FUTUROS	66
	REFERÊNCIAS	67
	APÊNDICE A - VaccineController	68
	APÊNDICE B - VaccineDetailsView	72

1 INTRODUÇÃO

A vacinação animal é fundamental para a saúde pública e a economia agropecuária, prevenindo doenças que podem afetar tanto os animais quanto os humanos. Além disso, a vacinação mantém os animais domésticos saudáveis, protegendo-os contra diversas doenças infecciosas que podem comprometer sua qualidade de vida e do coletivo [1]. A gestão eficaz da vacinação é crucial para garantir que os animais recebam as vacinas necessárias no tempo apropriado, minimizando o risco de surtos de doenças. No entanto, muitas clínicas veterinárias ainda dependem de métodos manuais ou sistemas inadequados para gerenciar esse processo, resultando em erros, perdas de informações e dificuldades no acompanhamento do histórico de vacinação dos animais.

Diante desse cenário, este trabalho propõe o desenvolvimento de um sistema de gerenciamento de vacinação animal, com foco específico de auxiliar as clínicas veterinárias e os proprietários de animais. O objetivo é criar uma ferramenta que simplifique o processo de gerenciamento de vacinas, aumente a precisão e a eficiência das operações e ofereça uma interface intuitiva para os usuários.

Inicialmente, o projeto visava a criação de um sistema abrangente de gerenciamento de clínicas veterinárias, capaz de suportar todos os processos comuns em uma clínica. No entanto, devido às limitações de tempo e recursos, o escopo foi ajustado para concentrar-se exclusivamente no gerenciamento de vacinação. Mesmo com essa mudança, o sistema proposto mantém funcionalidades críticas que atendem às principais necessidades dos proprietários de animais.

O sistema desenvolvido permite que os proprietários realizem o cadastro completo de seus animais, listem todos os animais sob sua responsabilidade e acessem as fichas cadastrais detalhadas. Além disso, os usuários podem visualizar uma cartilha de vacinas disponíveis, agendar eventos futuros de vacinação e visualizar os próximos eventos agendados. A carteira de vacinação digital oferece acesso fácil aos detalhes das vacinas aplicadas, e o sistema permite o registro de novas vacinas na carteira de vacinação do animal. Para garantir que os proprietários não percam datas importantes, o sistema envia lembretes de aplicação de vacinação.

A metodologia adotada para o desenvolvimento do sistema incluiu a análise de sistemas semelhantes de gerenciamento em clínicas médicas, utilizados como base para a elaboração do sistema veterinário. O padrão de projeto MVC (Model-View-Controller) foi escolhido para organizar a estrutura do sistema, garantindo uma separação nítida das responsabilidades por camada de desenvolvimento e facilitando a manutenção e a escalabilidade do sistema.

Na primeira etapa do projeto, foram identificados e definidos os requisitos funcionais e não funcionais do sistema, com base em casos de uso detalhados. Esses requisitos orientaram a criação do design do sistema, incluindo diagramas de entidade-relacionamento e o mapeamento do modelo relacional. Na segunda etapa, foi realizada a implementação do sistema, desde a codificação dos componentes até a definição das tabelas do banco de dados e a criação das interfaces de usuário.

Este documento é organizado em 4 capítulos. Além desse, os demais incluem os seguintes conteúdos:

- No capítulo 2, são apresentados os conceitos e a revisão de literatura, incluindo a análise de sistemas semelhantes. Também são abordados a arquitetura de software, os casos de uso, os requisitos funcionais e não funcionais para um sistema de gerenciamento de vacinação animal e o projeto de base de dados proposto para o sistema.
- O capítulo 3 detalha o desenvolvimento do sistema, começando com o diagrama entidade-relacionamento, seguido pelo mapeamento do modelo relacional, definição das tabelas no banco de dados, componentes do sistema e um algoritmo de automatização do envio de lembretes de vacinação, finalizando com a apresentação das telas do sistema.
- No capítulo 4, são apresentadas as considerações finais, que discutem os resultados obtidos, a eficácia do sistema e sugestões para trabalhos futuros.

2 CONCEITOS E REVISÃO DE LITERATURA

Este capítulo tem como objetivo examinar de forma crítica sistemas de gerenciamento de clínicas e consultórios similares existentes, buscando entender as suas abordagens, funcionalidades, avanços e limitações com o intuito de derivar suas funcionalidades para o contexto do gerenciamento de saúde animal. Um sistema de gerenciamento de clínicas e consultórios é uma ferramenta essencial projetada para auxiliar na administração, digitalização e organização eficaz das operações em ambientes de saúde, facilitando tarefas como agendamento de consultas, gestão de prontuários médicos, controle de recursos e informações financeiras, e otimização das operações clínicas. Por meio da análise comparativa, objetivamos oferecer de forma aprofundada sobre as inovações que o sistema proposto apresenta para o cenário das consultas veterinárias, além de identificar oportunidades de aprimoramento com base em lições extraídas de tais sistemas.

Iniciaremos esta etapa explorando o panorama dos sistemas de consultas médicas que compartilham semelhanças temáticas e funcionais com o projeto em foco. Visto que o mercado não possui muitas opções de sistemas voltados a clínicas veterinárias e as opções disponíveis exigem assinatura para acessar os recursos ofertados. Finalmente, por meio da revisão abrangente, apresentaremos as funcionalidades e particularidades dos sistemas escolhidos.

Prosseguindo, abordaremos as inovações e diferenciais específicos do sistema desenvolvido no contexto deste estudo. Detalharemos as características distintas que definem o sistema e como essas inovações respondem as especificações no âmbito dos cuidados da saúde animal, buscando mostrar também como ele pode contribuir para o cenário veterinário de modo relevante.

No entanto, é crucial reconhecer que sistemas inovadores não estão isentos de desafios e limitações. Portanto, exploraremos as limitações inerentes ao sistema em análise. Com base nessa avaliação crítica, buscaremos identificar as oportunidades para o aprimoramento contínuo do sistema, aproveitando os ensinamentos obtidos por meio das experiências dos sistemas relacionados.

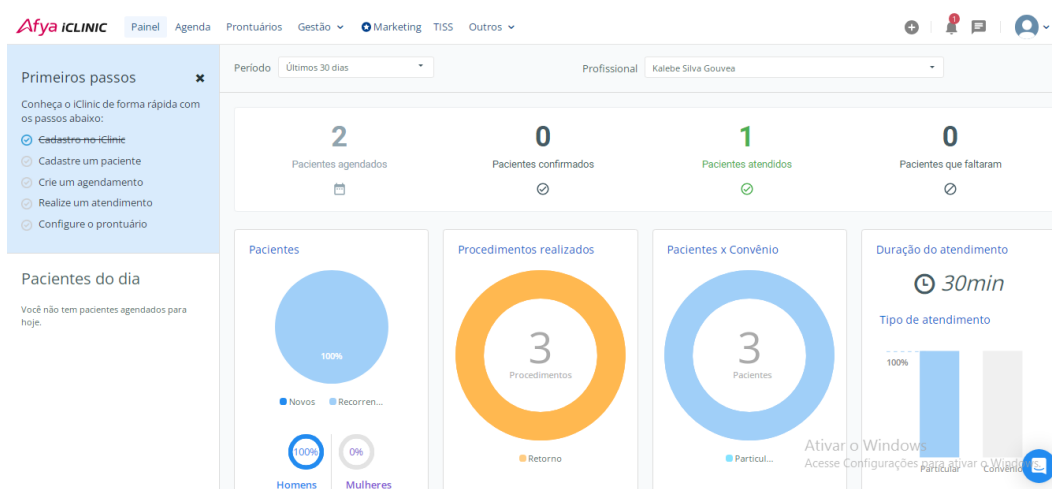
Por meio da análise detalhada dos sistemas relacionados, este capítulo visa enriquecer a compreensão não apenas sobre o sistema de gerenciamento de saúde animal em si, mas também sobre seu posicionamento no cenário da tecnologia de saúde. Ao situar o sistema em uma perspectiva mais ampla, estaremos preparados para avaliar suas contribuições e potencial de aprimoramento, estabelecendo um sólido alicerce para a conclusão deste trabalho.

2.1 O SISTEMA ICLINIC

O iClinic é o sistema de gestão médica que se apresenta como solução abrangente e moderna para clínicas e consultórios médicos [2]. Tendo variedade de recursos projetados para otimizar os processos, melhorar a qualidade do atendimento ao paciente e simplificar as tarefas administrativas. Portanto, ele é uma ferramenta relevante para profissionais de saúde.

A tela inicial exibe um *dashboard*, que é um painel visual que apresenta informações, métricas e indicadores importantes da clínica, contendo algumas informações relevantes, como um resumo dos pacientes que têm consultas agendadas para o dia, com nome, hora e status da consulta, alguns indicadores de desempenho que exibem, por exemplo, quantidades de pacientes atendidos, média de tempo dos atendimentos, gráficos indicando a quantidade de atendimentos realizados em um período, de distribuição etária dos pacientes etc.

Figura 1 - Tela de Dashboard - iClinic



Fonte: Reprodução própria. Disponível em: <https://iclinic.com.br>. Acesso em: 18/09/2023

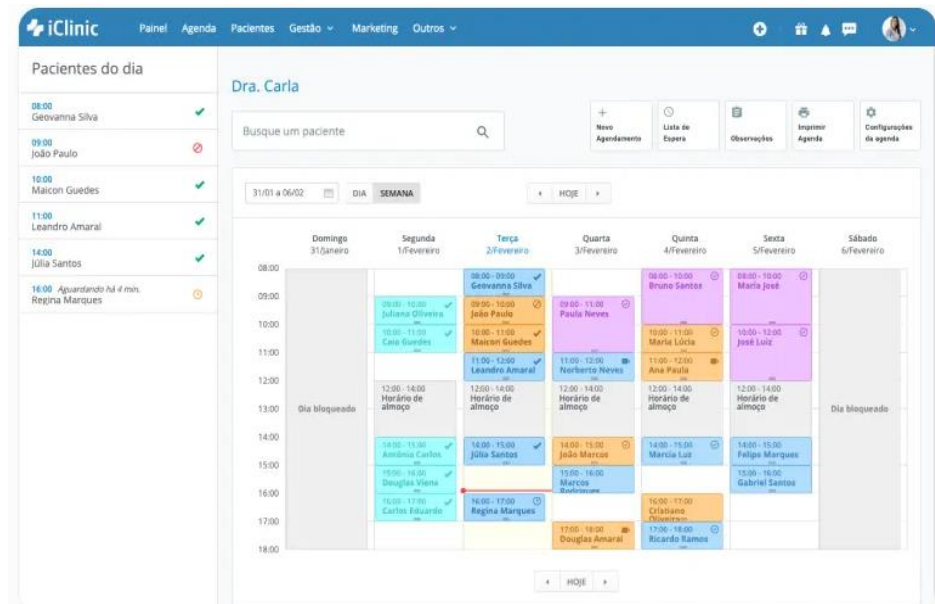
Uma das principais funcionalidades do iClinic é a capacidade de agendamento de consultas pela plataforma. Os médicos e suas equipes podem facilmente visualizar a agenda, marcar novas consultas e organizar compromissos. Isso ajuda, por exemplo, a evitar conflitos e atrasos, garantindo um fluxo de pacientes mais controlável, centralizar e organizar informações de agendamento em um único local otimizando a gestão clínica. Como parte dessa funcionalidade, o iClinic permite a sincronização da agenda com calendários externos, como o Google Calendar e o Calendário da Apple.

Ela oferece a capacidade de visualização de todos os pacientes agendados para um determinado dia, dispostos por horário, com a opção de personalização para aprimorar a visualização.

No âmbito do controle, os profissionais de saúde podem gerenciar a duração das consultas e os valores de cada uma diretamente do painel. Da mesma forma, acessam aos status dos agendamentos, possibilitando a identificação rápida dos pacientes confirmados, aguardando atendimento na sala de espera e os que faltaram às consultas.

Ademais, os profissionais têm a capacidade de bloquear horários que não estarão disponíveis para agendamento, além de enviar lembretes e confirmações de consultas por meio de e-mail, SMS ou WhatsApp.

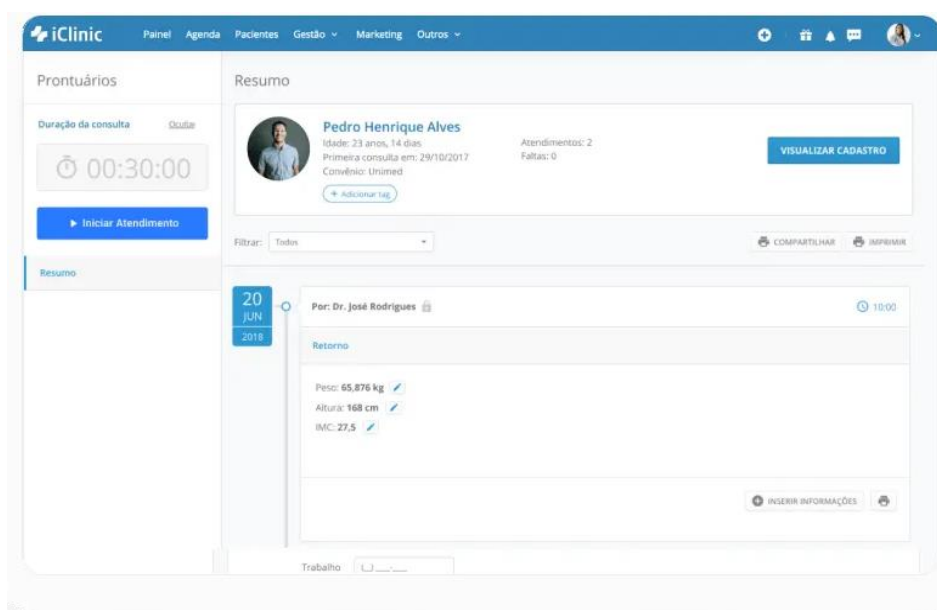
Figura 2 - Tela de agendamento de consultas - iClinic



Fonte: iClinic. Disponível em: <https://iclinic.com.br>. Acesso em: 18/09/2023

Além disso, o iClinic também oferece como funcionalidade o sistema de prontuário eletrônico integrado, permitindo que os profissionais de saúde armazenem e acessem os registros médicos dos pacientes de forma organizada. Isso elimina a necessidade do médico utilizar arquivos físicos, contribuindo com o meio ambiente e facilita o acompanhamento do histórico médico, diagnósticos anteriores, prescrições e resultados de exames.

Figura 3 - Tela de prontuário eletrônico - iClinic



Fonte: iClinic. Disponível em: <https://iclinic.com.br>. Acesso em: 18/09/2023

Tal funcionalidade também permite a visualização abrangente do histórico do paciente por meio de gráficos e tabelas que incluem informações como peso, medidas, registro de vacinas e outros dados originados de exames e de outras fontes. A capacidade de adicionar lembretes através de tags nos prontuários assegura que informações cruciais relacionadas aos pacientes não sejam esquecidas. Além disso, há a capacidade de anexar documentos essenciais diretamente ao sistema e criar orçamentos distintos para cada paciente. Ela também oferece diversas calculadoras, incluindo a de Índice de Massa Corporal (IMC), Curva de Crescimento, Idade Gestacional e Data Provável de Parto. Além disso, possibilita a consulta de qualquer código da Classificação Internacional de Doenças (CID), contribuindo para atendimentos mais ágeis e precisos.

Com o intuito de proporcionar maior flexibilidade nas consultas e mitigar ausências de pacientes, o iClinic também disponibiliza uma funcionalidade para o agendamento de teleatendimentos. Adicionalmente, permite a gravação das sessões de consulta, reforçando a segurança inerente a esse modelo de atendimento.

Figura 4 - Tela de teleatendimento - iClinic

Agendar consulta

Escolha o tipo de consulta que deseja agendar:

Clínica iClinic (Avenida Presidente Vargas, 1265 - Sala 401)

Sáb 20/02/21	Dom 21/02/21	Seg 22/02/21	Ter 23/02/21
08:00	08:00	08:00	08:00
08:20	08:20	08:20	08:20
08:40	08:40	08:40	08:40
09:00	09:00	09:00	09:00
Mais..	Mais..	Mais..	Mais..

Fonte: iClinic. Disponível em: <https://iclinic.com.br>. Acesso em: 18/09/2023

Para melhorar as taxas de comparecimento às consultas remotas, o sistema também permite o envio de lembretes de teleatendimentos diretamente para os números de WhatsApp dos pacientes. Essa prática auxilia na melhora das taxas de comparecimento e pontualidade das consultas, beneficiando tanto os pacientes quanto os profissionais de saúde.

O iClinic também possui um recurso para a gestão da clínica. Isso engloba um gerenciamento de estoque (como um almoxarifado) dos materiais que é realizado com eficácia, permitindo o controle de entradas e saídas por meio de registros históricos. Além disso, o sistema emite alertas quando os níveis de estoque estão baixos ou quando os itens se aproximam da data de vencimento.

Além das funcionalidades descritas anteriormente, o iClinic também possui outras que não serão descritas com detalhe nesse trabalho, mas que valem a pena serem citadas. A plataforma oferece recursos voltados ao marketing do iClinic, proporcionando aos profissionais de saúde uma ferramenta eficaz para construir e manter relacionamentos com pacientes, ao mesmo tempo em que oferece recursos automatizados e personalizados para atrair novos pacientes. Também apresenta módulo de gestão de finanças, que proporciona o controle financeiro eficaz e simplificado, permitindo que os profissionais de saúde gerenciem suas finanças com

precisão, mantenham um registro detalhado das transações e otimizem o relacionamento com convênios médicos e pacientes.

Figura 5 - Tela de gestão de clínica - iClinic

Meu Estoque

ADICIONAR PRODUTO ENTRADA SAÍDA

TODOS ESTOQUE BAIXO VENCE EM 30 DIAS PRODUTOS VENCIDOS

🔍 Pesquise por nome ou código do produto

Exibindo: 10 produtos

PRODUTO	CÓD.	QUANT.	VENCE EM 30 DIAS	VENCIDO	ESTOQUE MÍNIMO
<input type="checkbox"/> ALGODÃO 500G ROLO	00059	19	0	0	15
<input type="checkbox"/> PAPEL LENÇOL ROLO	00126	13	0	0	20
<input type="checkbox"/> AGULHA DESCARTÁVEL CX 100 UN...	00049	9	0	0	3
<input type="checkbox"/> ÁGUA OXIGENADA 10 VOLUMES - ...	00047	22	0	0	10
<input type="checkbox"/> MONONILON 5.0 CX C/ 24 UNID. ...	00122	10	0	0	10
<input type="checkbox"/> MONONILON 4.0 CX C/ 24 UNID. ...	00121	20	0	0	10
<input type="checkbox"/> ALCOL 70° - 1LT	00046	15	0	0	5
<input type="checkbox"/> MONONILON 3.0 CX C/ 24 UNID. ...	00120	10	0	0	10
<input type="checkbox"/> MONONILON 2.0 CX C/ 24 UNID. ...	00119	10	0	0	10
<input type="checkbox"/> MONONILON 6.0 CX C/ 24 UNID. ...	00118	10	0	0	10

Central de suporte

Fonte: iClinic. Disponível em: <https://iclinic.com.br>. Acesso em: 18/09/2023

Em resumo, o iClinic é uma solução comercial, abrangente e eficiente que atende às necessidades complexas de clínicas e consultórios médicos. Com suas funcionalidades de agendamento, prontuário eletrônico, teleatendimento, gestão clínica, marketing e gestão financeira, o sistema contribui para uma experiência mais eficiente para médicos, funcionários e pacientes, ao mesmo tempo em que promove uma melhor organização das informações médicas.

2.2 O SISTEMA NINSAÚDE APOLO

O sistema Apolo foi desenvolvido pela Ninsaúde Software, uma startup especializada em tecnologia da informação com uma trajetória no mercado de saúde desde 2013 [3]. A empresa se apresenta como uma startup compromissada com a qualidade e segurança em suas soluções, com sua atuação concentrada no mercado brasileiro privado de saúde.

No ano de 2018, a Ninsaúde lançou o Ninsaúde Apolo, o sistema abrangente e contemporâneo projetado para simplificar a gestão de clínicas para diversas especialidades médicas. Esse software oferece uma variedade de funcionalidades, incluindo agendamento de consultas, prontuário eletrônico, faturamento de convênios médicos e ferramentas de engajamento e retenção de pacientes. Entre essas ferramentas destacam-se o marketing por email, check-in do paciente, pesquisas de satisfação e outras funcionalidades destinadas a melhorar a experiência do paciente e fortalecer o relacionamento clínica-paciente.

A plataforma oferece diversas funcionalidades que auxiliam no trabalho do cotidiano do profissional com o objetivo de tornar mais rápido e objetivos os processos em um consultório ou clínica.

Uma das funcionalidades presentes é a de agendamento de consultas, tanto pelo paciente quanto pela clínica. O sistema fornece acesso ao calendário de agendamento, onde a equipe pode visualizar a disponibilidade de horários dos médicos e escolher espaços apropriados para as consultas. Informações sobre o agendamento, incluindo o nome do paciente, motivo da consulta, profissional para o atendimento, informações de convênio estão presentes durante o preenchimento.

Ainda como parte dessa funcionalidade, o Apolo oferece confirmação de agendamentos via SMS ou WhatsApp, agendamento online da consulta por parte do paciente e também agendamento recorrente, onde é possível cadastrar agendamentos que se repetem diária, semanal, mensal ou até mesmo anualmente.

Figura 6 - Tela de agendamento de consultas - Apolo

Fonte: Ninsaúde. Disponível em: <https://www.apolo.app/>. Acesso em: 18/09/2023

Outro recurso presente no sistema é o prontuário eletrônico. Ela permite que os médicos registrem informações sobre os pacientes, incluindo histórico médico, alergias, medicamentos em uso, doenças preexistentes e histórico familiar. Assim, fornece um panorama da saúde do paciente no sistema. Além disso, permite anotações durante as consultas, registrando sintomas, diagnósticos, prescrições eletrônicas e planos de acompanhamento. Os resultados de exames médicos, como análises de sangue, ressonâncias magnéticas e radiografias, podem ser anexados ao prontuário eletrônico. A funcionalidade também permite armazenar um histórico completo de todas as consultas anteriores do paciente, facilitando a revisão de informações passadas e o acompanhamento de progressos ao longo do tempo.

Figura 7 - Tela de prontuário eletrônico - Apolo

Fonte: Ninsaúde. Disponível em: <https://www.apolo.app/>. Acesso em: 18/09/2023

O sistema também apresenta uma solução de gerenciamento financeiro que desempenha um papel fundamental na gestão eficaz das finanças de clínicas médicas e consultórios. Essa ferramenta oferece uma série de recursos que auxiliam na organização, controle e otimização das operações financeiras. Dentre esses recursos, cabe destacar a opção de cadastrar títulos de recebimento ou pagamento com periodicidade semanal, mensal ou anual, com a capacidade de parcelar títulos facilmente; a possibilidade de pagamento de comissões devidas aos profissionais, como o rendimento e quantidade de pacientes atendidos na clínica; a opção de gerar relatórios em planilha eletrônica, oferecendo flexibilidade na análise de dados e na criação de gráficos personalizados, além de permitir o arquivamento de informações importantes.

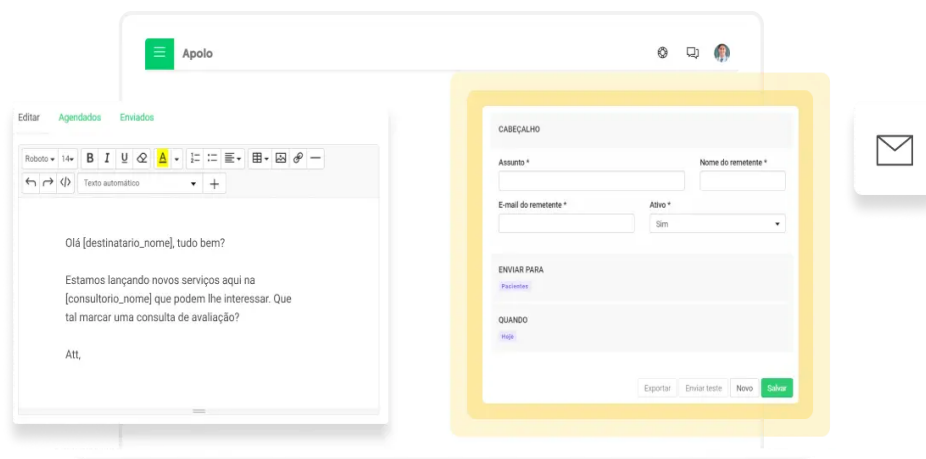
Figura 8 - Tela de gerenciamento financeiro - Apolo

A captura de tela mostra a interface do sistema Apolo para o lançamento de recebimentos. No topo, há uma barra de navegação com o nome 'Apolo' e ícones de configurações, mensagens e perfil. O menu lateral à esquerda contém ícones para home, calendário, pacientes, documentos, relatórios e configurações. O formulário principal está dividido em seções: 'Lançar recebimento' com abas para 'Título único', 'Título parcelado' e 'Título recorrente' (destacado); 'INFORMAÇÕES DO TÍTULO' com campos para Descrição, Tipo de pagamento (Indiferente), Conta bancária (Indiferente), Receber de (Paciente), Paciente, Valor (0,00), Tipo de recibo (Usar bloco de recibos), Categoria (Indiferente), Subcategoria, Começar em (dd/mm/aaaa), Recorrente (Mensal), Terminar em (dd/mm/aaaa), Terminar após (Ocorrências ou Sem término com 'Repetir sempre' selecionado) e uma área de Observação. No rodapé, há uma barra de status com o URL 'https://apolo.ninsaude.com/view/fluxo_geral#di/lançarreceitaRecorrente' e botões para 'Ativar o Windows', 'Acesse Configurações para', 'Limpar' e 'Salvar'.

Fonte: Ninsaúde. Disponível em: <https://www.apolo.app/>. Acesso em: 18/09/2023

O profissional de saúde ou administrador do sistema também pode contar com uma funcionalidade para retenção de clientes, ou marketing, possibilitando, por exemplo, a criação de e-mails de prospecção para seus pacientes; de envio de mensagens automática de felicitação de aniversariantes ou de outras datas comemorativas; de solicitar a confirmação de consultas e procedimentos, podendo ser efetuadas por meio de métodos como chamadas telefônicas automatizadas, mensagens via WhatsApp, SMS e notificações.

Figura 9 - Tela de envio de e-mails personalizados - Apolo



Fonte: Ninsaúde. Disponível em: <https://www.apolo.app/>. Acesso em: 18/09/2023

2.3 COMPARAÇÃO ENTRE SISTEMAS AVALIADOS

Notável a convergência de funcionalidades essenciais entre os softwares iClinic e Ninsaúde Apollo, ambos desempenhando um papel crucial na otimização da gestão e operação de clínicas e consultórios médicos. Tais funcionalidades compartilhadas desempenham um papel fundamental na melhoria dos serviços de saúde e na satisfação tanto de profissionais da saúde quanto de pacientes.

Primeiramente, ambos os sistemas oferecem robustas soluções para o agendamento de consultas, simplificando o processo de marcação de compromissos médicos. A funcionalidade oferece a capacidade tanto para médicos quanto para secretários marcarem compromissos clínicos. Isso proporciona flexibilidade na gestão de agendas, permitindo que o próprio médico acesse o sistema para agendar suas consultas de acordo com sua disponibilidade. Esse recurso não apenas beneficia a organização das agendas dos profissionais de saúde, mas também promove um atendimento mais ágil e conveniente para os pacientes. Os pacientes podem, muitas vezes, agendar consultas diretamente online, o que aumenta a conveniência e a acessibilidade aos serviços médicos.

Também é possível encontrar em ambos os sistemas funcionalidades relacionadas à prescrição eletrônica de medicamentos (receitas médicas) e à solicitação e gerenciamento de exames. Essas funcionalidades permitem uma maior

organização e dinamismo nas clínicas, possibilitando que os documentos sejam padronizados, digitalizados e reunidos em uma área de fácil acesso para os médicos.

Outra característica em comum é a implementação do prontuário eletrônico, permitindo o armazenamento seguro e acessível de registros médicos dos pacientes. Essa funcionalidade agiliza a recuperação de informações clínicas cruciais, contribuindo para diagnósticos mais precisos e um acompanhamento mais eficiente dos tratamentos.

Ambos os sistemas também reconhecem a importância crescente do teleatendimento na prestação de serviços de saúde. Eles oferecem soluções para consultas remotas, facilitando o acesso dos pacientes à assistência médica, independentemente da localização geográfica. Essa funcionalidade ganhou destaque particularmente durante a pandemia de COVID-19, quando as consultas virtuais se tornaram uma necessidade. Os sistemas costumam incluir recursos como suporte a teleconsulta e compartilhamento de tela para tornar as consultas online eficazes e seguras.

A respeito à gestão financeira, tanto o iClinic quanto o Ninsaúde Apolo apresentam recursos para automatizar e simplificar o controle das finanças das clínicas. Isso inclui o registro de pagamentos de pacientes, o cálculo automático de repasses a parceiros e médicos, além de possibilitar a geração de guias para convênios médicos, agilizando o processo de faturamento. Os relatórios financeiros e gráficos fornecidos por esses sistemas auxiliam na análise das finanças da clínica, tornando mais fácil tomar decisões informadas.

Por fim, ambos os sistemas reconhecem a importância da comunicação no atendimento na área da saúde. Eles oferecem ferramentas para engajamento de pacientes, como pesquisas de satisfação, e-mail marketing e outras estratégias que visam melhorar a relação entre clínicas e pacientes. A capacidade de personalizar campanhas de marketing e segmentar pacientes com base em suas necessidades e preferências é fundamental para a eficácia dessas estratégias.

Essas funcionalidades em comum entre o iClinic e o Ninsaúde Apolo ilustram como esses softwares se alinham com as necessidades em constante evolução do setor de saúde e como essas necessidades podem se repetir no contexto veterinário.

Mesmo que os sistemas abordados não tenham os animais como foco principal, eles podem facilmente ser adaptados para resolver problemas semelhantes nesse nicho. Permitir que uma clínica veterinária ofereça soluções de agendamento de consultas, acompanhamento de prontuário eletrônico, gerenciamento de receitas e exames, além de gerenciamento financeiro e de marketing, facilita a gestão eficaz e aprimora a qualidade dos serviços veterinários oferecidos. O foco na conveniência, segurança, eficiência financeira e engajamento do paciente torna esses sistemas valiosos aliados para profissionais de veterinária e administradores de clínicas.

2.4 DIFERENCIAIS PROPOSTOS PARA UM NOVO SOFTWARE

Embora os sistemas iClinic e Ninsaúde Apolo ofereçam uma ampla gama de funcionalidades para a gestão eficiente de clínicas e consultórios médicos, uma lacuna inexplorada reside na falta de um módulo dedicado ao gerenciamento de vacinação. Essa ausência é notável, especialmente considerando a importância crucial da vacinação na saúde preventiva, tanto humana quanto veterinária. A inclusão de tal funcionalidade não só complementaria os recursos existentes, mas também abriria novas oportunidades de aplicação na gestão veterinária.

Um aspecto distintivo e altamente benéfico da proposta de gerenciamento de vacinação é a capacidade de permitir que os proprietários tenham controle direto sobre as informações vacinais de seus animais. Essa autonomia é valiosa, pois independe da necessidade de autorização prévia do veterinário no sistema. Destacam-se diversas vantagens dessa abordagem: os proprietários podem inserir dados de vacinação de seus animais independentemente do profissional veterinário que os atenda, o acesso digitalizado ao histórico vacinal do animal é facilitado e a gestão de lembretes de vacinação auxilia os proprietários a manterem o calendário vacinal sempre atualizado. Essas funcionalidades não apenas promovem uma maior conscientização sobre a importância da vacinação, mas também contribuem para a saúde e o bem-estar dos animais de estimação.

Portanto, a incorporação de um módulo de gerenciamento de vacinação em sistemas de gestão de saúde animal não apenas preenche uma lacuna significativa, mas também representa um avanço crucial na promoção da saúde preventiva, tanto

para humanos quanto para animais. Ao fornecer aos proprietários a autonomia para controlar diretamente as informações vacinais de seus animais de estimação, essa funcionalidade não apenas simplifica o processo, mas também fortalece a conscientização e a responsabilidade dos proprietários na manutenção da saúde de seus companheiros animais. Assim, a inclusão desse diferencial não só enriquece a oferta dos sistemas existentes, mas também reflete um compromisso com o bem-estar abrangente dos proprietários e seus animais.

2.5 LIMITAÇÕES DO SISTEMA PROPOSTO

Embora o novo sistema proposto ofereça uma abordagem inovadora para o gerenciamento de vacinação de animais, é importante reconhecer algumas limitações que acompanham essa fase inicial de desenvolvimento. Essas limitações são abordadas de forma estratégica para garantir a entrega eficiente de uma funcionalidade essencial ao sistema.

O desenvolvimento de um sistema abrangente de gestão de saúde animal é uma tarefa complexa e demorada. Por isso, o escopo do sistema será inicialmente limitado ao gerenciamento de vacinação, assegurando que essa funcionalidade diferencial seja implementada de maneira eficaz. Embora outras funcionalidades comuns, como agendamento de consultas e prescrição eletrônica de receitas, não estejam presentes na versão inicial, essa abordagem permite uma entrega mais rápida e focada. Futuras atualizações podem expandir o sistema para incluir essas funcionalidades adicionais, tornando-o ainda mais abrangente e útil para os usuários.

O sistema foi projetado com o proprietário do animal como o principal usuário, o que capacita os proprietários na gestão das informações vacinais de seus animais. Embora veterinários e administradores não sejam os principais atores na fase inicial, essa decisão simplifica o uso e facilita a adoção do sistema pelos proprietários. Essa abordagem centrada no usuário promove a conscientização e responsabilidade dos proprietários na manutenção da saúde de seus animais. No futuro, funcionalidades adicionais podem ser desenvolvidas para incluir a participação de veterinários e administradores, permitindo uma integração mais ampla e colaborativa no gerenciamento de saúde animal.

Essas limitações foram cuidadosamente consideradas para garantir uma implementação bem-sucedida e eficiente das funcionalidades de gerenciamento de vacinação. Ao focar inicialmente em uma área específica e crucial, o sistema pode oferecer uma solução eficaz e imediata para uma necessidade identificada, com a flexibilidade de expandir suas capacidades em iterações futuras.

2.6 ARQUITETURA DE SOFTWARE

A arquitetura de software desempenha um papel fundamental no desenvolvimento de sistemas web modernos, fornecendo uma estrutura organizacional que separa as preocupações relacionadas à interface do usuário, à lógica de negócios e ao acesso a dados. Neste tópico, serão explorados três padrões arquitetônicos: o Model-View-Controller (MVC), o Model-View-Presenter (MVP) e o Model-View-ViewModel (MVVM). Os dois últimos são evoluções do tradicional padrão MVC e foram projetados para melhorar a separação de preocupações, modularidade e testabilidade.

O Modelo-Visão-Controlador ou *Model-View-Controller* é uma abordagem que desassocia a interface do usuário (View) da lógica de negócios e do acesso aos dados (Model), facilitando a manutenção, a escalabilidade e a reutilização de código em sistemas web [4, p.349]. A estrutura MVC é composta por três componentes principais:

- A Model, também conhecido como "objeto-modelo", abrange toda a lógica de negócios e os dados específicos da aplicação. Ela gerencia a manipulação e a estruturação dos dados, incluindo o acesso a fontes externas, como bancos de dados e serviços web.
- Já, a View é responsável pela apresentação da interface do usuário. Ela utiliza os dados fornecidos pelo model para renderizar conteúdo dinâmicos e interativos na interface web. A View é desacoplada da lógica de negócios, permitindo diferentes representações da mesma informação.
- Por fim, o Controller atua como intermediário entre o Model e a View. Ele recebe as interações dos usuários através da interface e decide como responder, manipulando os dados e atualizando o model conforme necessário. Também

atualiza a View com os dados do model, garantindo uma separação de responsabilidades.

No contexto de uma aplicação web baseada em MVC, o fluxo de interação segue um padrão bem definido: o usuário interage com a interface (View), gerando eventos que são capturados pela Model. O Controller processa esses eventos, atualiza a Model conforme necessário e atualiza a View com os novos dados, proporcionando uma experiência dinâmica e responsiva ao usuário.

O MVP, introduzido na década de 1990, divide o aplicativo em Model, View e Presenter [5]. A Model gerencia os dados e a lógica de negócios sem se comunicar diretamente com a view ou o presenter. A View é responsável por exibir os dados e depende do Presenter para atualizações e tratamento de entradas do usuário. O presenter serve como ponte entre a Model e a View, garantindo a apresentação correta dos dados e lidando com a entrada do usuário. O MVP melhora a separação de preocupações e facilita a testabilidade e modularidade, mas pode ser mais complexo e exigir mais código padrão.

O MVVM, desenvolvido pela Microsoft, também separa o aplicativo em três componentes principais: Model, View e ViewModel [5]. A Model lida com dados e lógica de negócios, enquanto a View, geralmente projetada com *Extensible Application Markup Language* (XAML), exibem os dados ao usuário. O ViewModel conecta a Model e a View, usando observáveis e comandos para facilitar a comunicação. No MVVM, o ViewModel não se comunica diretamente com a view, o que melhora a testabilidade e a separação de preocupações. O MVVM é adequado para a Interfaces de Usuários (UI) complexas e se destaca em projetos que utilizam frameworks como WPF, UWP e Angular. No entanto, pode ser mais complexo de implementar devido à necessidade de compreender ligação de dados e comandos.

A escolha do padrão Model-View-Controller (MVC) para o desenvolvimento do sistema de gerenciamento de vacinação animal foi baseada na sua simplicidade e ampla adoção, que facilitam a separação de preocupações e a manutenção do código. Embora MVP e MVVM ofereçam melhorias na testabilidade e modularidade, especialmente em UIs complexas, o MVC é suficiente para as necessidades do projeto.

2.7 CASOS DE USO

Os casos de uso são uma técnica fundamental na engenharia de requisitos [9, p.74]. Eles são uma forma de capturar e especificar os requisitos de um sistema de software do ponto de vista dos usuários. Um caso de uso descreve uma interação entre um ator (usuário ou sistema externo) e o sistema em questão para atingir um objetivo específico.

Eles são úteis porque fornecem uma visão clara e concisa dos requisitos funcionais do sistema, focando nas interações entre os atores e o sistema [9, p.74]. Eles também ajudam a identificar e compreender os requisitos do sistema de uma perspectiva orientada ao usuário, permitindo aos desenvolvedores entender melhor o comportamento necessário do sistema e validar os requisitos com os *stakeholders*.

Um sistema abrangente de gestão de saúde animal deve incorporar diversas funcionalidades essenciais, como agendamento de consultas e vacinações, administração de medicamentos, acompanhamento de carteira de vacinação, de receitas veterinárias, de exames, entre outras. Os requisitos para um sistema desse porte são identificados no diagrama de caso de uso apresentado da figura 10.

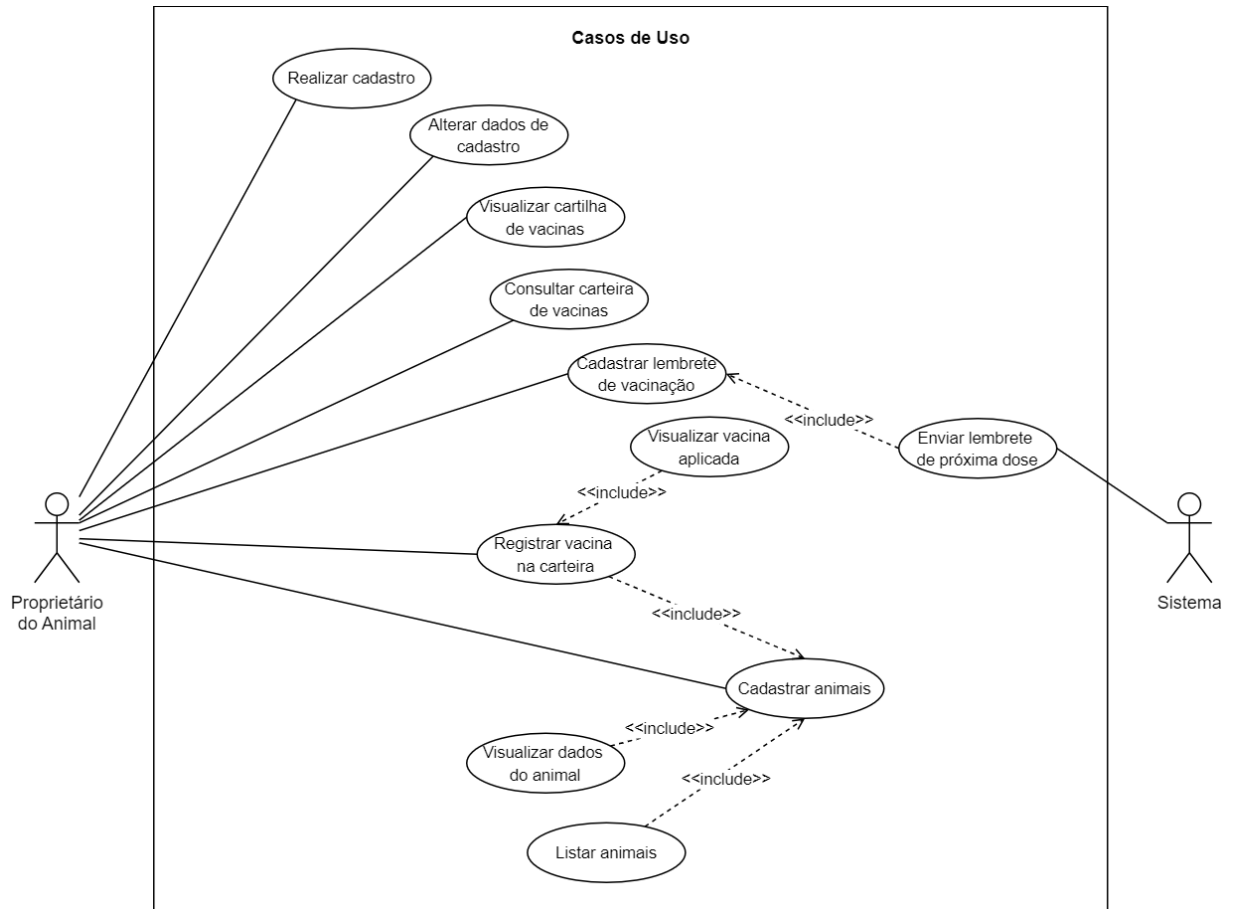
Figura 10 - Diagrama de casos de uso completo



Fonte: Reprodução própria

Dado que a implementação completa desse sistema se mostra desafiadora dentro dos limites de tempo deste trabalho, optou-se por uma abordagem mais focada, concentrando-se principalmente no gerenciamento da vacinação animal com foco voltado para os proprietários dos animais, como citado anteriormente. O diagrama de caso de uso abaixo ilustra o escopo reduzido do sistema proposto, delineando os objetivos que se pretende alcançar com ele, vide a Figura 11.

Figura 11 - Diagrama de casos de uso restringido



Fonte: Reprodução própria

2.8 REQUISITOS PARA UM SISTEMA DE GERENCIAMENTO DE VACINAÇÃO ANIMAL

Ao examinar o diagrama de caso de uso citado anteriormente, buscou-se identificar os elementos fundamentais que podem ser utilizados em um sistema voltado ao gerenciamento de vacinação animal através de requisitos de software, frequentemente classificados como requisitos funcionais e não funcionais, contribuindo para a sua eficácia e abrangência. Cabe destacar que outros requisitos poderiam ser citados para o sistema proposto, mas o escopo desse trabalho se limitará apenas a alguns desses:

2.8.1 Requisitos funcionais

Os requisitos funcionais são declarações de serviços fornecidos por um sistema, de como ele deve se comportar em determinadas situações, descrevendo o que um sistema deve ou não fazer [7]. Abaixo, encontram-se os requisitos funcionais definidos para o sistema abordado neste trabalho.

- 1 - O sistema permite que o proprietário realize o seu cadastro.
- 2 - O sistema permite que o proprietário realize o cadastro completo do seu animal.
- 3 - O sistema permite listar todos os animais que o proprietário possui.
- 4 - O sistema permite que o proprietário tenha acesso a uma ficha cadastral com os dados completos do seu animal.
- 5 - O sistema permite que o proprietário visualize a cartilha de vacinas disponíveis para os animais.
- 6 - O sistema permite que o proprietário agende eventos futuros de vacinação para um animal em sua agenda.
- 7 - O sistema permite que o proprietário visualize os próximos eventos de vacinação agendados para o seu animal.
- 8 - O sistema permite que o proprietário visualize a carteira de vacinação de seu animal.
- 9 - O sistema permite que o proprietário tenha acesso a detalhes de vacinas aplicadas contidas na carteira de vacinação.
- 10 - O sistema permite que o proprietário registre vacinas na carteira de vacinação do animal.
- 11 - O sistema permite o envio de lembretes de vacinação futura ao proprietário.

2.8.2 Requisitos não funcionais

- 1 - O sistema permite que o proprietário receba notificações por e-mail de vacinas quando estiver próximo do vencimento.
- 2 - O sistema será escalável para suportar o crescimento das informações e a adição de novas funcionalidades sem degradação significativa do desempenho.

3 - O sistema será projetado para ser modular, o que significa que suas partes individuais podem ser modificadas ou substituídas conforme necessário.

2.9 PROJETO DE BASE DE DADOS

A necessidade de armazenamento e manutenção de dados em um sistema é vital para assegurar a integridade e eficiência das informações. O armazenamento adequado permite que dados sejam organizados de forma estruturada, facilitando o acesso e análise. No entanto, a mera acumulação de dados não é suficiente; a manutenção contínua é essencial para garantir a consistência ao longo do tempo. Nesse contexto, um Sistema Gerenciador de Bancos de Dados (SGBD) desempenha um papel indispensável. Ele oferece uma estrutura robusta para gerenciar dados, implementando medidas de segurança, controle de acesso e backup. O SGBD não apenas organiza eficientemente as informações, mas também contribui significativamente para a confiabilidade do sistema, garantindo a disponibilidade dos dados quando necessário e a preservação da integridade ao longo do tempo [8, p.14].

Visto isso, é importante que o sistema proposto possua um projeto de base de dados. Ele define a estrutura, relacionamentos e lógica de armazenamento dos dados, garantindo a integridade e a recuperação eficaz da informação. Neste trabalho, para atingir esse objetivo, será seguida uma sequência de passos definidos no algoritmo a seguir, tendo como entrada os requisitos do sistema descritos anteriormente e ao fim, uma base de dados adequada para o sistema como saída.

Tabela 1 - Algoritmo de base de dados

Entrada: Requisitos de sistema

Saída: Base de dados de acordo com o modelo relacional

INÍCIO:

1. Elaborar o diagrama ER
 - 1.1. Validar o diagrama elaborado
2. Mapear o modelo relacional através do diagrama ER
 - 2.1. Normalizar o modelo de acordo com as ordens 1FN, 2FN e 3FN
3. Transformar o modelo relacional em tabelas

3.1. Instanciar tabelas utilizando SQL

3.2. Povoar e testar tabelas utilizando SQL

FIM

Na etapa 1, será elaborado o diagrama de entidade relacionamento. O Diagrama de Entidade e Relacionamento (DER) é uma representação gráfica do Modelo de Entidade e Relacionamento (MER) que por sua vez é um modelo de dados conceitual de alto nível, popularmente utilizado para projeto conceitual de aplicações de banco de dados [8].

Na etapa 2, será realizada a normalização do modelo relacional a partir do modelo de entidade e relacionamento. A normalização é o processo de organização de dados de uma tabela de modo a reduzir a redundância e as anomalias de dados [8]. Ela ocorre quando o mesmo dado é armazenado em mais de uma tabela, o que pode levar a inconsistências e erros. As anomalias de dados são problemas que podem ocorrer quando os dados são inseridos, atualizados ou excluídos.

A normalização é um processo importante para garantir a integridade e a confiabilidade dos dados. Ela também pode ajudar a melhorar o desempenho das consultas e a reduzir o tamanho do banco de dados. A normalização nesse trabalho será apresentada através das 3 formas normais: 1FN (os dados são atômicos, ou seja, não podem ser divididos em partes menores), 2FN (todos os atributos não chave são dependentes da chave primária) e 3FN (todos os atributos não chave são dependentes apenas da chave primária e não de outros atributos não chave).

Por fim, na etapa 3, será realizada a transformação do diagrama relacional em tabelas. Esse processo é importante para a implementação de um banco de dados, pois o modelo relacional fornece uma representação visual dos dados que serão armazenados no banco de dados. A instanciação é realizada utilizando o comando CREATE TABLE do SQL. As tabelas instanciadas devem ser povoadas com dados, utilizando o comando INSERT INTO. Após a população das tabelas, é importante realizar testes para verificar se os dados estão sendo armazenados corretamente. Os testes podem ser realizados utilizando o comando SELECT do SQL.

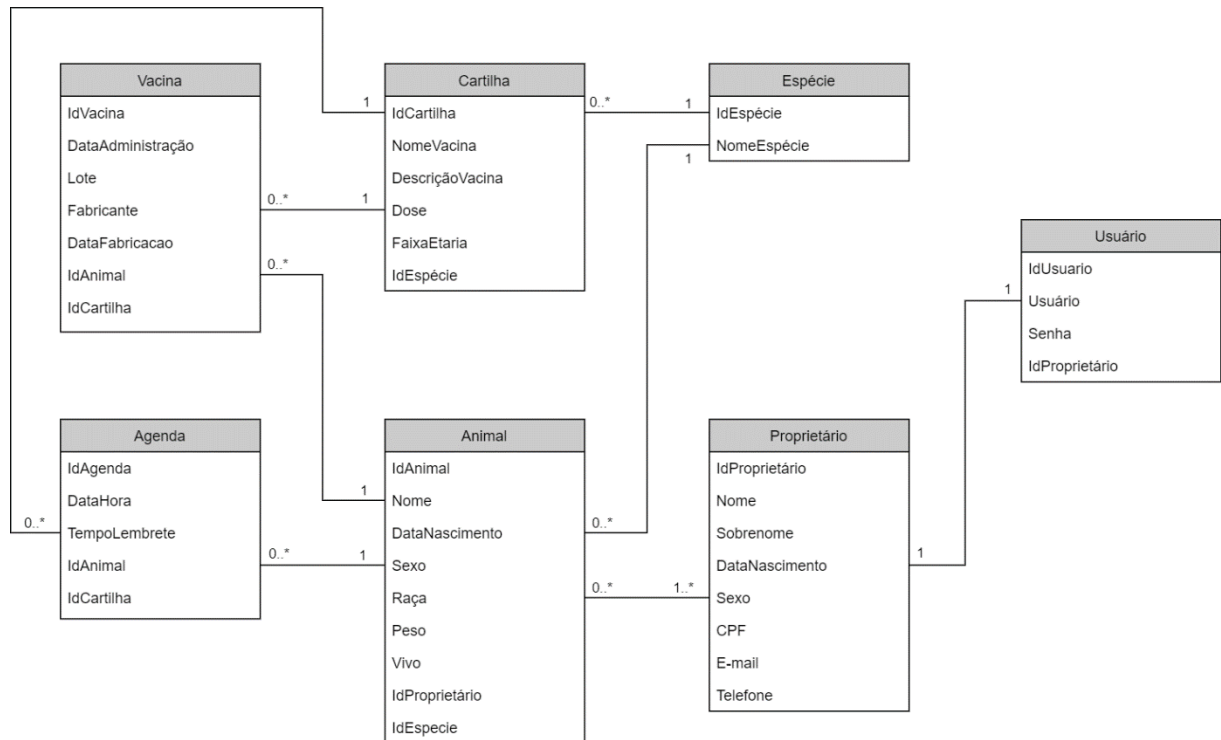
3 SISTEMA PARA GERENCIAMENTO DE VACINAÇÃO ANIMAL

Nesta seção apresentamos o desenvolvimento do sistema para gerenciamento de vacinação animal, detalhando desde a modelagem de dados até a implementação e apresentação das telas do sistema. Inicia-se com a construção do diagrama entidade-relacionamento (DER), que define as principais entidades e seus relacionamentos. Em seguida, são descritos o mapeamento do modelo relacional, a definição das tabelas no banco de dados e a inserção de dados, que serão testados através das consultas. O capítulo também explora a estrutura do sistema, incluindo seus componentes principais: models, controllers e views, além do script para automatização de lembretes de vacinação. Finalmente, são apresentadas as interfaces do sistema, demonstrando suas funcionalidades principais.

3.1 DIAGRAMA ENTIDADE-RELACIONAMENTO

A figura abaixo mostra o modelo entidade relacionamento elaborado para o sistema de gerenciamento de vacinação animal, considerando os requisitos elaborados na seção 1.9 deste trabalho. O DER será analisado e discutido, com o objetivo de entender sua estrutura e funcionalidade.

Figura 12 - Diagrama Entidade Relacionamento



Fonte: Reprodução própria

3.1.1 Entidades

O diagrama elaborado apresenta as entidades para Usuário, Proprietário, Animal, Espécie, Cartilha, Vacina e Agenda. Seus atributos são descritos conforme serão exibidos no diagrama ER.

3.1.1.1 Usuário

A entidade Usuário representa o conjunto de credenciais que serão utilizadas pelo proprietário no sistema. Ela possui os atributos:

- IdUsuario: identificador de usuário do proprietário no sistema
- Usuário: nome de usuário escolhido para login
- Senha: senha escolhida para login
- IdProprietario: identificador do proprietário no sistema

3.1.1.2 Proprietário

A entidade Proprietário representa os dados pessoais de uma pessoa física proprietária de um animal. Ela possui os atributos:

- IdProprietario: identificador do proprietário no sistema
- Nome: primeiro nome do proprietário
- Sobrenome: sobrenome do proprietário
- DataNascimento: data de nascimento do proprietário
- Sexo: gênero do proprietário
- CPF: número de identificação do proprietário
- Email: endereço de e-mail do proprietário
- Telefone: número de telefone do proprietário

3.1.1.3 Animal

A entidade Animal representa um animal que pertence a um proprietário no sistema, possuindo os atributos:

- IdAnimal: identificador do animal no sistema
- Nome: nome do animal no sistema
- DataNascimento: data de nascimento do animal
- Sexo: gênero do animal
- Raça: descreve a raça do animal (ex: labrador, siamês, mangalarga)
- Peso: último peso do animal
- IdProprietario: identificador do proprietário do animal
- IdEspécie: identificador da espécie na qual o animal pertence

3.1.1.4 Espécie

A entidade Espécie representa as categorias de espécie cadastradas no sistema e possui os atributos:

- IdEspécie: identificador da espécie na qual o animal pertence

- NomeEspécie: nome da espécie do animal (ex: cachorro, gato, cavalo)

3.1.1.5 Cartilha

A entidade Cartilha representa a cartilha de vacinação com as vacinas existentes por espécie de animal. Ela possui os atributos:

- IdCartilha: identificador da vacina contida na cartilha de vacinação
- NomeVacina: nome da vacina
- DescriçãoVacina: descrição dada a vacina, como as doenças combatidas por ela
- Dose: identificação da dose (ex: 1ª Dose, 2ª Dose, Reforço)
- FaixaEtaria: período determinado para vacinação do animal (A partir de 6 meses de idade, anualmente)
- IdEspécie: identificador da espécie na qual o animal pertence

3.1.1.6 Vacina

A entidade Vacina representa um registro de informações sobre a vacina aplicada no animal. Ela possui os atributos:

- IdVacina: identificador da vacina aplicada no sistema
- DataAdministração: data em que foi aplicada a vacina no animal
- Lote: registro do lote da vacina aplicada
- Fabricante: nome do fabricante da vacina
- DataAdministracao: data agendada de administração do medicamento
- IdAnimal: identificador do animal em que foi aplicada a vacina
- IdCartilha: identificador da vacina contida na cartilha de vacinação

3.1.1.7 Agenda

A entidade Agenda representa o conjunto de eventos agendados para vacinação do animal. Essa entidade possui os atributos:

- IdAgenda: identificador do evento registrado na agenda
- DataHora: data e hora em que deverá ser aplicada a vacina
- TempoLembrete: quantidades de dias antecedentes a vacinação na qual o lembrete deve ser enviado
- IdAnimal: identificador do animal em foi agendada a vacinação
- IdCartilha: identificador da vacina a ser aplicada contida na cartilha de vacinação

3.1.2 Relacionamentos

O diagrama elaborado também apresenta relacionamentos entre as entidades descritas no item 3.1.1, como pode-se ver abaixo:

- Um Usuário pertence a apenas um proprietário;
- Um Proprietário possui apenas uma credencial de Usuário;
- Um Proprietário pode possuir nenhum ou muitos Animais;
- Um Animal pode pertencer a um ou muitos Proprietários;
- Um Animal pertence a uma única Espécie;
- Um Animal pode ter sido vacinado com nenhuma ou muitas Vacinas;
- Um Animal pode possuir nenhuma ou muitas vacinas agendadas em sua Agenda;
- Uma Espécie pode conter nenhum ou muitos Animais;
- Uma Espécie pode possuir nenhuma ou muitas vacinas registradas na Cartilha;
- Uma vacina na Cartilha pertence a uma única Espécie;
- Uma vacina da Cartilha pode estar contida em nenhuma ou muitas Vacinas aplicadas;
- Uma vacina da Cartilha pode ser encontrada em nenhum ou muitos eventos da Agenda;
- Uma Vacina contém informação de uma única vacina da Cartilha;
- Uma Vacina pode ser aplicada em um único Animal;
- Um evento na Agenda contém uma única vacina da Cartilha;

3.2 MAPEAMENTO DE MODELO RELACIONAL

Nesta seção, será realizado o mapeamento do diagrama de entidade relacionamento apresentado, utilizando um conjunto de regras conhecido como normalização de banco de dados. Para atingir esse objetivo, serão seguidas as etapas de normalização conforme descritas abaixo:

- Primeira Forma Normal (1FN): Remover repetições de grupos de colunas, garantindo que cada coluna contenha apenas valores atômicos.
- Segunda Forma Normal (2FN): Remover dependências parciais, garantindo que cada coluna em uma tabela dependa totalmente da chave primária.
- Terceira Forma Normal (3FN): Remover dependências transitivas, garantindo que não haja colunas que dependam de outras colunas que não sejam a chave primária.

Para a Primeira Forma Normal, foi validado que todas as colunas das tabelas Usuário, Proprietário, Animal, Espécie, Cartilha, Vacina e Agenda contém apenas um valor atômico, não sendo necessário nenhuma modificação. Logo, todas as tabelas encontram-se na 1FN.

Para a Segunda Forma Normal, foi validado que todos os atributos das tabelas Usuário, Proprietário, Animal, Espécie, Cartilha, Vacina e Agenda dependem totalmente de suas respectivas chaves primárias. Portanto, todas as tabelas encontram-se na 2FN.

Para a Terceira Forma Normal, foi validado que não houve dependências transitivas nas tabelas Usuário, Proprietário, Animal, Espécie, Cartilha, Vacina e Agenda, pois todas as colunas dependem diretamente da chave primária de suas respectivas tabelas.

Portanto, nenhuma das tabelas abordadas precisou ser modificada, pois todas já estão na Terceira Forma Normal (3FN) e atendem aos requisitos de normalização.

3.3 DEFINIÇÃO DE TABELAS NO BANCO DE DADOS

Nesta seção, serão apresentadas as instruções SQL responsáveis por criar as tabelas que compõem o banco de dados do sistema de gerenciamento de vacinação

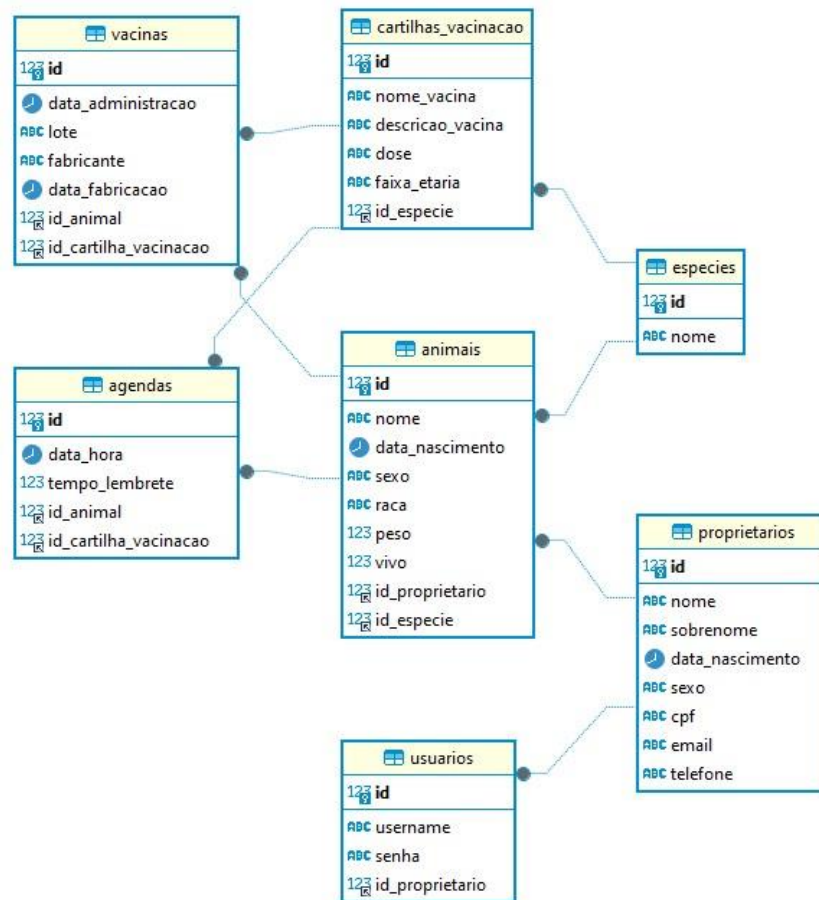
animal, que utiliza o MySQL, e o diagrama ER gerado pelo cliente DBeaver, após a execução dessas instruções. As tabelas foram projetadas para armazenar informações relevantes descritas nas etapas anteriores de modelagem do banco de dados:

```
CREATE TABLE usuarios (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    username VARCHAR(100) NOT NULL,  
    senha VARCHAR(100) NOT NULL,  
    id_proprietario INT NOT NULL,  
    FOREIGN KEY (id_proprietario) REFERENCES proprietarios(id)  
);  
  
CREATE TABLE proprietarios (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    nome VARCHAR(50) NOT NULL,  
    sobrenome VARCHAR(50) NOT NULL,  
    data_nascimento DATE NOT NULL,  
    sexo CHAR(1) NOT NULL,  
    cpf VARCHAR(11),  
    email VARCHAR(100) NOT NULL,  
    telefone VARCHAR(20) NOT NULL  
);  
  
CREATE TABLE especies (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    nome VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE animais (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    nome VARCHAR(50) NOT NULL,  
    data_nascimento DATE NOT NULL,  
    sexo CHAR(1) NOT NULL,  
    raca VARCHAR(50) NOT NULL,  
    peso DECIMAL(5,2) NOT NULL,  
    vivo BOOLEAN NOT NULL,  
    id_proprietario INT NOT NULL,  
    id_especie INT NOT NULL,  
    FOREIGN KEY (id_proprietario) REFERENCES proprietarios(id),  
    FOREIGN KEY (id_especie) REFERENCES especies(id)  
);  
  
CREATE TABLE cartilhas_vacinacao (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    nome_vacina VARCHAR(100) NOT NULL,  
    descricao_vacina VARCHAR(1000) NOT NULL,  
    dose VARCHAR(50) NOT NULL,  
    faixa_etaria VARCHAR(50) NOT NULL,  
    id_especie INT NOT NULL,  
    FOREIGN KEY (id_especie) REFERENCES especies(id)  
);  
  
CREATE TABLE vacinas (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    data_administracao DATE NOT NULL,  
    lote VARCHAR(50) NOT NULL,  
    fabricante VARCHAR(100) NOT NULL,  
    data_fabricacao DATE NOT NULL,  
    id_animal INT NOT NULL,  
    id_cartilha_vacinacao INT NOT NULL,  
    FOREIGN KEY (id_animal) REFERENCES animais(id),  
    FOREIGN KEY (id_cartilha_vacinacao) REFERENCES cartilhas_vacinacao(id)  
);
```

```
CREATE TABLE agendas (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    data_hora DATETIME NOT NULL,  
    tempo_lembrete TINYINT NOT NULL,  
    id_animal INT NOT NULL,  
    id_cartilha_vacinacao INT NOT NULL,  
    FOREIGN KEY (id_animal) REFERENCES animais(id),  
    FOREIGN KEY (id_cartilha_vacinacao) REFERENCES cartilhas_vacinacao(id)  
);  
  
CREATE TABLE usuarios (  
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
    email VARCHAR(100) NOT NULL,  
    senha VARCHAR(100) NOT NULL,  
    id_proprietario INT NOT NULL,  
    FOREIGN KEY (id_proprietario) REFERENCES proprietarios(id)  
);
```


Figura 13 – Diagrama ER gerado pelo DBeaver



Fonte: Reprodução própria

3.4 INSERÇÃO DE DADOS E CONSULTAS

Nesta seção, serão apresentadas algumas das instruções SQL responsáveis por popular as tabelas criadas para o banco de dados do sistema. Algumas tabelas necessitam de muitos dados serem inseridos e, pela grande quantidade de código, não serão tratadas aqui. Após as inserções, também serão executadas algumas consultas para exibir os registros realizados nas tabelas e os relacionamentos possíveis entre elas.

3.4.1 Inserções

```
-- PROPRIETARIOS
```

```
INSERT INTO proprietarios
```

```
(nome, sobrenome, data_nascimento, sexo, cpf, email, telefone)
```

```
VALUES
```

```
('João', 'Silva', '1990-03-15', 'M', '12345678901', 'joao.silva@email.com', '11987654321'),
('Maria', 'Souza', '1985-08-20', 'F', '98765432109', 'maria.souza@email.com', '21123456789'),
('Pedro', 'Santos', '1978-05-10', 'M', '45678901234', 'pedro.santos@email.com', '(31555555555));
```

```
-- USUARIOS
```

```
INSERT INTO usuarios
```

```
(username, senha, id_proprietario)
```

```
VALUES
```

```
('joao.silva', '$2a$11$e//iGfhuE14AtSm.yaUyeSco4YPCvvkvmrmlmp3ksk/qXVR.OOve', 1), --
SENHA: 123456
('maria.souza', '$2a$11$e//iGfhuE14AtSm.yaUyeSco4YPCvvkvmrmlmp3ksk/qXVR.OOve', 2), --
SENHA: 123456
('pedro.santos', '$2a$11$e//iGfhuE14AtSm.yaUyeSco4YPCvvkvmrmlmp3ksk/qXVR.OOve', 3); --
SENHA: 123456
```

```
-- ESPECIES
```

```
INSERT INTO especies
```

```
(nome)
```

```
VALUES
```

```
('Cachorro'),
('Gato'),
('Cavalo');
```

```
-- ANIMAIS
```

```
INSERT INTO animais
```

```
(nome, data_nascimento, sexo, raca, peso, vivo, id_proprietario, id_especie)
```

```
VALUES
```

```
('Totó', '2019-04-10', 'M', 'SRD', 8.50, 1, 1, 1),
('Frajola', '2020-01-15', 'M', 'Siames', 5.20, 1, 2, 2),
('Pé de pano', '2018-08-05', 'M', 'Puro-sangue inglês', 450.00, 1, 3, 3),
('Amora', '2017-12-20', 'F', 'Persa', 4.80, 1, 1, 2),
('Pandora', '2019-06-25', 'F', 'Golden Retriever', 15.70, 1, 2, 1);
```

```
-- EXEMPLOS DE VACINAS DISPONIVEIS NA CARTILHA DE VACINACAO

INSERT INTO cartilhas_vacinacao (nome_vacina, descricao_vacina, dose, faixa_etaria,
id_especie)
VALUES
-- CÃES
('V8', 'Previne contra Adenovirose 2, Cinomose (CDV), Parvovirose, Coronavirose, Parainfluenza
canina, Hepatite canina e Leptospirose canina (dois tipos). A V10 é uma alternativa a V8.', '1ª
Dose', '6 a 8 semanas', 1),
('Anti-rábica', 'Previne contra raiva.', '1ª Dose', '16 semanas', 1),
-- GATOS
('Quádrupla Felina', 'Previne contra Renotraqueíte, Calicivirose, Panleucopenia felina e
Chlamydia psittaci', '1ª Dose', '6 a 8 semanas', 2),
('Anti-rábica', 'Previne contra raiva', 'Única', 'A partir de 1 ano, anualmente', 2),
-- CAVALOS
('Tétano', 'Previne contra o tétano', '1ª Dose', 'Início de desmama', 3),
('Influenza', 'Previne contra a Influenza', '2ª Dose', '30 dias após a desmama', 3);
-- VACINAS APLICADAS
INSERT INTO vacinas (data_administracao, lote, fabricante, data_fabricacao, id_animal,
id_cartilha_vacinacao)
VALUES
-- CACHORRO VACINADO
('2019-08-17', '2F3021A', 'FabricanteA', '2019-08-01', 1, 1),
-- GATO VACINADO
('2019-04-17', '4F3021B', 'FabricanteC', '2019-04-01', 2, 3),
-- CAVALO VACINADO
('2019-03-17', '5L3021C', 'FabricanteA', '2019-03-01', 3, 5);
-- AGENDAMENTOS DE VACINAS FUTURAS
INSERT INTO agendas
(data_hora, tempo_lembrete, id_animal, id_cartilha_vacinacao)
VALUES
-- PARA O CACHORRO DE ID 1
('2024-07-02 00:00:00', 3, 1, 2),
-- PARA O GATO DE ID 2
('2024-08-09 00:00:00', 7, 2, 4),
-- PARA O CAVALO DE ID 3
('2024-09-06 00:00:00', 3, 3, 6);
```

3.4.2 Consultas

Figura 14 - Consulta de todos os animais pertencentes a um proprietário

```
-- SELECIONA TODOS OS ANIMAIS POR PROPRIETARIO
select a.id, a.nome, a.data_nascimento, a.sexo, a.raca, e.nome especie, a.peso, a.vivo
from animais a
join proprietarios p on p.id = a.id_proprietario
join especies e on e.id = a.id_especie
where p.id = 1;
```

animais(+) 1 X

select a.id, a.nome, a.data_nascimento, a.sexo, a.raca, e.nome especie, a.peso, a.vivo

	123 id	ABC nome	data_nascimento	ABC sexo	ABC raca	ABC especie	123 peso	123 vivo
1	1	Totó	2019-04-10	M	SRD	Cachorro	8,5	1
2	4	Amora	2017-12-20	F	Persa	Gato	4,8	1

Fonte: Reprodução própria

Figura 15 - Consulta de vacinas disponíveis por espécie

```
-- SELECIONA TODAS AS VACINAS DISPONIVEIS POR ESPECIE
select cv.id, cv.nome_vacina, cv.descricao_vacina, cv.dose, cv.faixa_etaria, e.nome as especie
from cartilhas_vacinacao cv
join especies e on e.id = cv.id_especie
where cv.id_especie = 3;
```

cartilhas_vacinacao(+) 1 X

select cv.id, cv.nome_vacina, cv.descricao_vacina, cv.dose, cv.faixa_etaria, e.nome as especie

	123 id	ABC nome_vacina	ABC descricao_vacina	ABC dose	ABC faixa_etaria	ABC especie
1	23	Tétano	Previne contra o tétano	1ª Dose	Início de desmama	Cavalo
2	24	Tétano	Previne contra o tétano	2ª Dose	30 dias após a desmama	Cavalo
3	25	Tétano	Previne contra o tétano	Única	A partir de 4 anos, anualmente	Cavalo
4	26	Tétano	Previne contra o tétano	Única	Durante gestação, anualmente	Cavalo
5	27	Influenza	Previne contra a Influenza	1ª Dose	Início de desmama	Cavalo
6	28	Influenza	Previne contra a Influenza	2ª Dose	30 dias após a desmama	Cavalo
7	29	Influenza	Previne contra a Influenza	Única	A partir de 4 anos, anualmente	Cavalo
8	30	Influenza	Previne contra a Influenza	Única	Durante gestação, anualmente	Cavalo
9	31	Encefalomielite (Leste e Oeste)	Previne contra a Encefalomielite	1ª Dose	Início de desmama	Cavalo
10	32	Encefalomielite (Leste e Oeste)	Previne contra a Encefalomielite	2ª Dose	30 dias após a desmama	Cavalo

Fonte: Reprodução própria

Figura 16 - Consulta de cartão de vacina por animal

```
-- CARTEIRA DE VACINA DO ANIMAL
select v.id, cv.nome_vacina, v.data_administracao, v.lote, v.fabricante, v.data_fabricacao, cv.dose
from vacinas v
join cartilhas_vacinacao cv on cv.id = v.id_cartilha_vacinacao
where
v.id_animal = 3;
```

vacinas(+) 1 X

select v.id, cv.nome_vacina, v.data_administracao | Insira uma expressão SQL para filtrar os resultados (use Ctrl+Espaço)

	123 id	ABC nome_vacina	data_administracao	ABC lote	ABC fabricante	data_fabricacao	ABC dose
1	13	Tétano	2019-03-17	5L3021C	FabricanteA	2019-03-01	1ª Dose
2	14	Tétano	2018-04-20	6P9152F	FabricanteA	2018-04-15	2ª Dose
3	15	Influenza	2020-08-03	7C6289K	FabricanteB	2020-08-30	1ª Dose
4	16	Influenza	2020-07-15	2F4791L	FabricanteC	2020-07-03	2ª Dose
5	17	Anti-rábica	2021-01-15	2C9685C	FabricanteB	2021-01-01	1ª Dose
6	18	Anti-rábica	2021-02-02	9R3424L	FabricanteA	2021-01-30	2ª Dose

Fonte: Reprodução própria

Figura 17 - Vacinação agendada para um animal

```
-- VACINAS AGENDADAS PARA O ANIMAL
select a.id, a.data_hora, a2.nome, cv.nome_vacina, cv.dose, a.tempo_lembrete
from agendas a
join animais a2 on a2.id = a.id_animal
join cartilhas_vacinacao cv on cv.id = a.id_cartilha_vacinacao
where a2.id = 3;
```

agendas(+) 1 X

select a.id, a.data_hora, a2.nome, cv.nome_vacina | Insira uma expressão SQL para filtrar os resultados (use Ctrl+Espaço)

	123 id	data_hora	ABC nome	ABC nome_vacina	ABC dose	123 tempo_lembrete
1	24	2024-08-03 00:00:00	Pé de pano	Tétano	Única	1
2	25	2024-09-06 00:00:00	Pé de pano	Influenza	Única	3
3	26	2024-09-15 00:00:00	Pé de pano	Encefalomielite (Leste e Oeste)	Única	7

Fonte: Reprodução própria

3.5 COMPONENTES DO SISTEMA

Nesta seção, detalharemos alguns dos componentes do sistema desenvolvido, divididos em Models, Controllers, Views e o script de automatização de lembretes de vacinação. Outros componentes foram desenvolvidos e não serão apresentados nesse trabalho, devido a extensão do código escrito, porém podem ser conferidos em seu repositório no Github [10]. Em cada subtópico mostraremos uma parte do código implementado, começando pelos models, que representam os dados e regras de negócio, passando pelos controllers, responsáveis pela lógica de aplicação, e finalizando com as views, que definem a interface com o usuário. Para o desenvolvimento desse sistema, foram utilizados a linguagem de programação C# e

o framework .NET 7. Por fim, será apresentado o script que automatiza o envio de lembretes de vacinação, destacando seu papel crucial na funcionalidade do sistema. Para o script, foi utilizado a linguagem de programação Python.

3.5.1 Models

A model Vaccine descrita abaixo é responsável por representar os dados relacionados a vacinas aplicadas ao animal.

```
namespace VetVaxManager.Models
{
    public class Vaccine
    {
        public int VaccinId { get; set; }
        public DateTime DateOfAdministration { get; set; }
        public string Lot { get; set; }
        public string Manufacturer { get; set; }
        public DateTime DateOfManufacture { get; set; }
        public Animal Animal { get; set; }
        public VaccinationSchedule VaccinationSchedule { get; set; }
    }
}
```

3.5.2 Controllers

A controller VaccineController é responsável por orquestrar toda a lógica relacionada a vacinas no sistema e sua implementação pode ser conferida no Apêndice A desse trabalho. No código, são utilizados métodos para realizar criação, criação, deleção e atualização de dados relacionados.

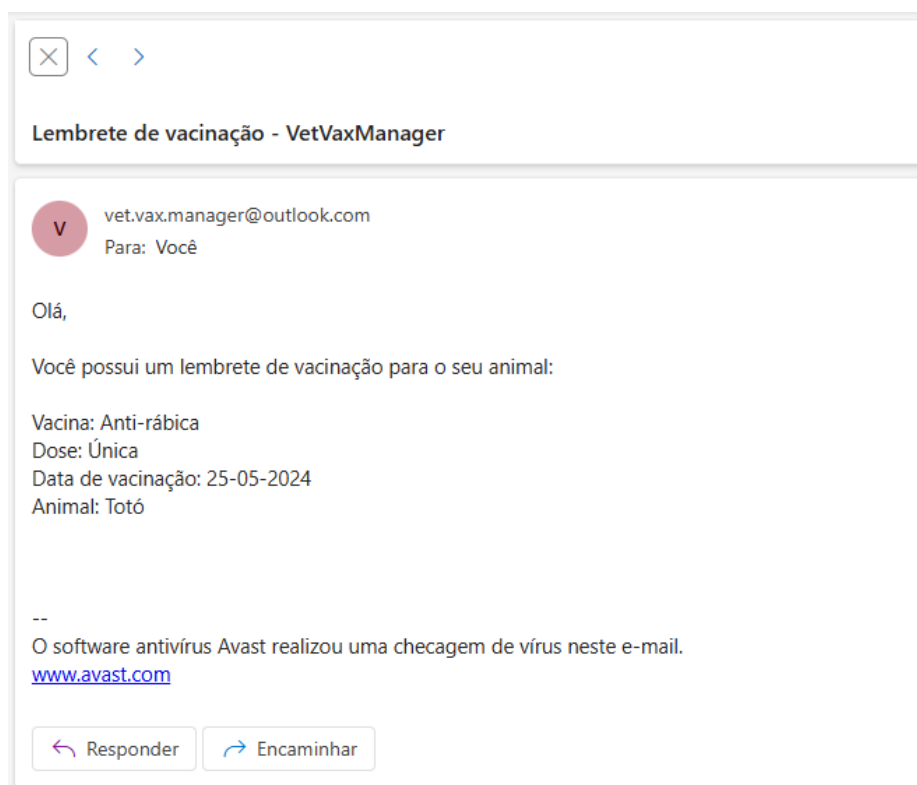
3.5.3 Views

A View VaccineDetailsView é responsável por exibir ao usuário todas as informações referentes a Vacina aplicada em seu animal. Por meio dela, outras views podem ser acessadas, como a edição e a deleção da vacina. A implementação do código pode ser vista no Apêndice B desse trabalho.

3.5.4 Script de automatização de lembretes de vacinação

O código desenvolvido para o envio de lembretes de vacinação tem um funcionamento simples: é executada uma consulta SQL para obter os registros de vacinação que precisam de lembretes, comparando a data atual com a data do evento ajustada pelo tempo de lembrete. Para cada registro recuperado, o sistema envia um e-mail ao proprietário do animal lembrando-o da vacinação. Para que a funcionalidade de envio de lembretes por e-mail funcione adequadamente, é necessário que a execução do script seja agendada como uma rotina diária, por exemplo, através do Agendador de Tarefas do sistema operacional Windows.

Figura 18 - Lembrete recebido por e-mail após execução do script



Fonte: Reprodução própria

O código pode ser conferido abaixo:

```
import mysql.connector
from datetime import datetime
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

def get_today_records():
    # Conexão com o banco de dados
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="root",
        database="tccdb"
    )

    cursor = conn.cursor(dictionary=True)

    today = datetime.today().strftime('%Y-%m-%d')

    query = """
    SELECT
    ag.data_hora AS EventDateTime,
    ag.tempo_lembrete AS ReminderDays,
    an.nome AS Animal,
    c.nome_vacina AS Vaccine,
    c.dose AS Dose,
    p.email AS Email
    FROM agendas ag
    INNER JOIN animais an ON an.id = ag.id_animal
    INNER JOIN cartilhas_vacinacao c ON c.id = ag.id_cartilha_vacinacao
    INNER JOIN proprietarios p ON p.id = an.id_proprietario
    WHERE DATE(ag.data_hora) = DATE_ADD(%s, INTERVAL ag.tempo_lembrete DAY)
    """

    cursor.execute(query, (today,))

    records = cursor.fetchall()

    cursor.close()
    conn.close()

    return records
```



```

def send_email(record):
    sender_email = "email_address"
    receiver_email = record['Email']
    password = "email_password"

    message = MIMEMultipart()
    message["From"] = sender_email
    message["To"] = receiver_email
    message["Subject"] = "Lembrete de vacinação - VetVaxManager"

    body = "Olá,\n\nVocê possui um lembrete de vacinação para o seu animal:\n\n"
    body += f"Vacina: {record['Vaccine']}\nDose: {record['Dose']}\nData de vacinação: {record['EventDateTime'].strftime('%d-%m-%Y')}\nAnimal: {record['Animal']}\n\n"

    message.attach(MIMEText(body, "plain"))

    with smtplib.SMTP("smtp-mail.outlook.com", 587) as server:
        server.starttls()
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, message.as_string())

def main():
    records = get_today_records()
    if records:
        for record in records:
            send_email(record)

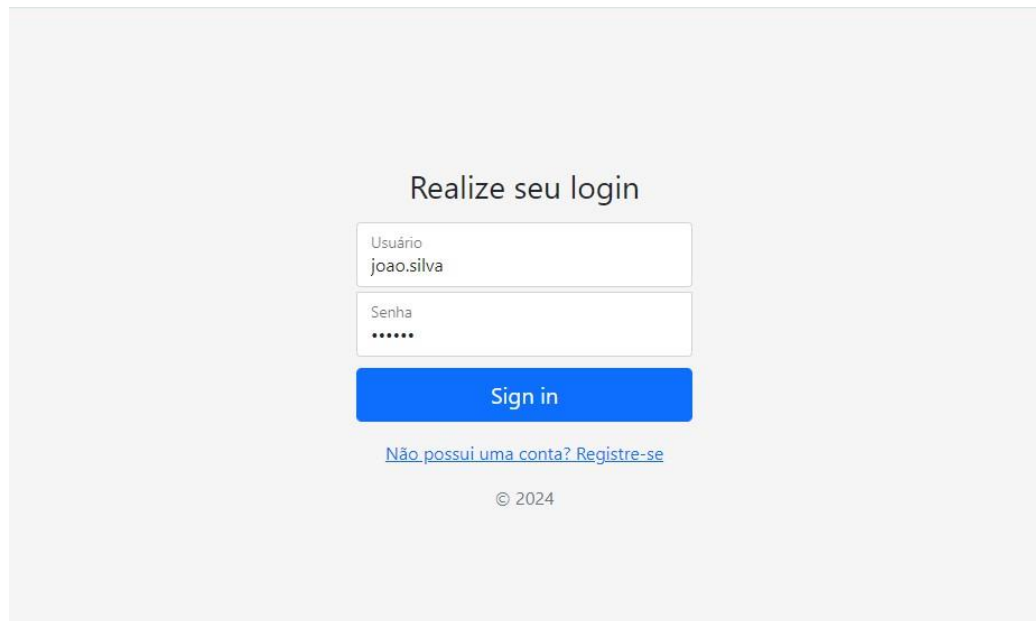
if __name__ == '__main__':
    main()

```

3.6 APRESENTAÇÃO DE TELAS DO SISTEMA

Nesta seção, serão apresentadas as telas capturadas do sistema desenvolvido em funcionamento. As capturas de tela ilustram a interface do usuário e demonstram as principais funcionalidades do sistema, incluindo o cadastro do usuário, o cadastro de animais, a visualização de informações detalhadas, a gestão de vacinas e o agendamento de eventos. Esta apresentação visual facilita a compreensão do funcionamento do sistema e evidencia a eficiência e a usabilidade da solução desenvolvida.

Figura 19 - Tela de login do usuário



Realize seu login

Usuário
joao.silva

Senha

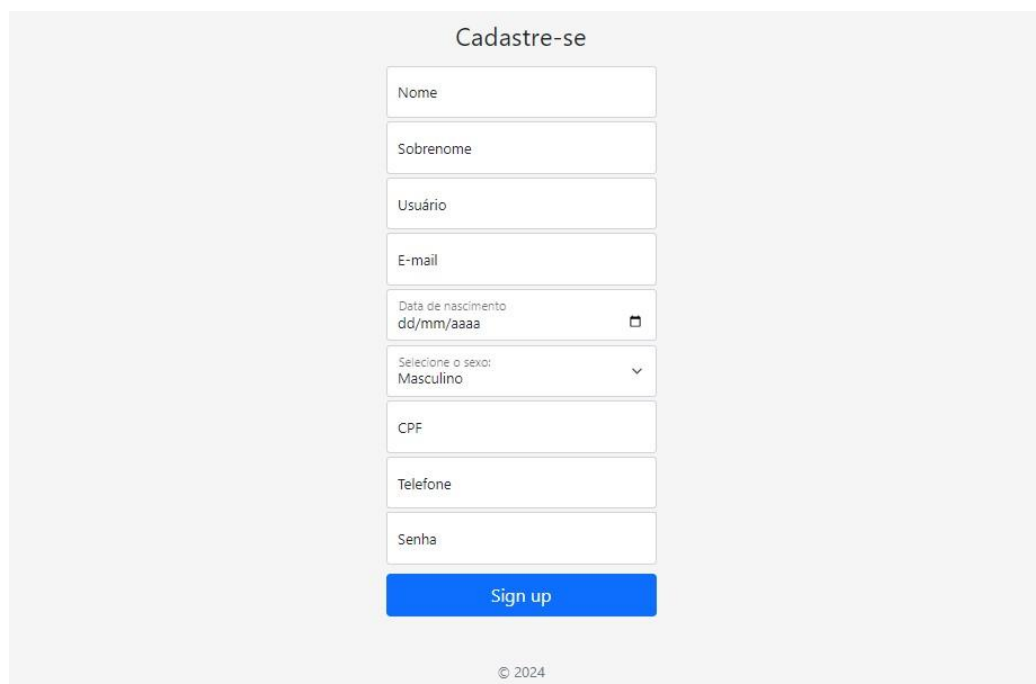
Sign in

[Não possui uma conta? Registre-se](#)

© 2024

Fonte: Reprodução própria

Figura 20 - Tela de cadastro do usuário



Cadastre-se

Nome

Sobrenome

Usuário

E-mail

Data de nascimento
dd/mm/aaaa

Selecione o sexo:
Masculino

CPF

Telefone

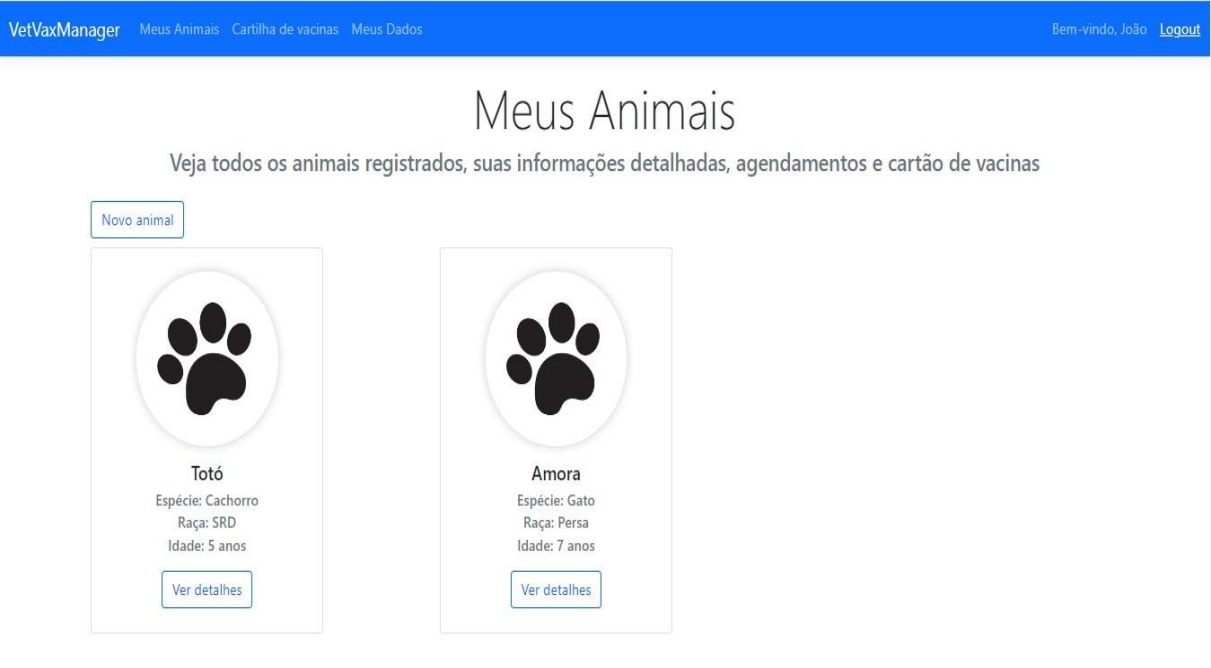
Senha

Sign up

© 2024

Fonte: Reprodução própria

Figura 21 - Tela inicial exibindo os animais do proprietário



Fonte: Reprodução própria

Figura 22 - Tela de cadastro de animal

VetVaxManager Meus Animais Cartilha de vacinas Meus Dados

Registrar animal:

Preencha os campos abaixo para registrar um novo animal:

Nome:

Data de nascimento:

Selecione o sexo:

Raça:

Peso:

Selecione a espécie:

[Registrar animal](#)


© 2024 - VetVaxManager - Licença

Fonte: Reprodução própria

Figura 23 - Tela de detalhes do animal

VetVaxManager Meus Animais Cartilha de vacinas Meus Dados Bem-vindo, João [Logout](#)

Dados do animal:



Totó
Sexo: M | Espécie: Cachorro | Raça: SRD
Idade: 5 anos | Peso: 8,50 kg | Estado: Vivo
[Editar](#) [Deletar](#)

Agenda:

[Agendar vacina](#)

V8
...

Gripe Canina
...

Anti-rábica
...

Fonte: Reprodução própria

Figura 24 - Tela de alteração de dados do animal

Editar animal:
Preencha os campos abaixo para editar informações do animal:

Nome:

Data de nascimento:

Selecione o sexo:

Raça:

Peso:

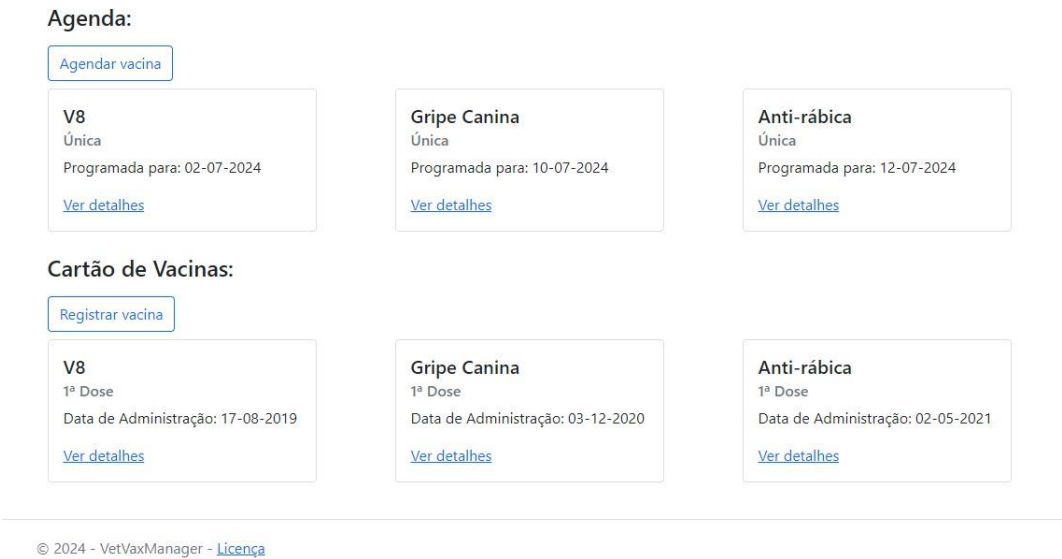
Selecione a espécie:

Selecione o estado:

[Editar animal](#) [Voltar](#)

Fonte: Reprodução própria

Figura 25 - Tela de vacinas agendadas e aplicadas do animal



Fonte: Reprodução própria

Figura 26 - Tela de detalhes de vacinação agendada



Fonte: Reprodução própria

Figura 27 - Tela de agendamento de vacinação

VetVaxManagerMeus AnimaisCartilha de vacinasMeus Dados

Agendar vacina:

Preencha os campos abaixo para criar um lembrete de vacinação:

Data da vacinação:

dd/mm/aaaa

Lembrar antes (dias):

1

Selecione a vacina:

V8 - 1ª Dose

Criar lembrete

Voltar

Fonte: Reprodução própria

Figura 28 - Tela de detalhes da vacina aplicada

VetVaxManagerMeus AnimaisCartilha de vacinasMeus Dados

Detalhes da vacina:

V8

1º Reforço

Data de administração: 20-09-2018

Data de fabricação: 15-09-2018

Fabricante: FabricanteB

Lote: 3P9152D

Voltar

Editar

Deletar

Fonte: Reprodução própria

Figura 29 - Tela de registro de vacina no cartão de vacinação do animal

VetVaxManager Meus Animais Cartilha de vacinas Meus Dados

Registrar vacina:
Preencha os campos abaixo para registrar uma vacina no cartão de vacinação:

Selecione a vacina:

V8 - 1ª Dose

Data de administração:

dd/mm/aaaa

Lote:

Fabricante:

Data de fabricação:

dd/mm/aaaa

Registrar vacina

Voltar

Fonte: Reprodução própria

Figura 30 - Tela de alteração de dados de vacina agendada para o animal

Meus Animais Cartilha de vacinas Meus Dados

Editar evento:
Preencha os campos abaixo para editar um lembrete de vacinação:

Data da vacinação:

02/07/2024

Lembrar antes (dias):

3

Selecione a vacina:

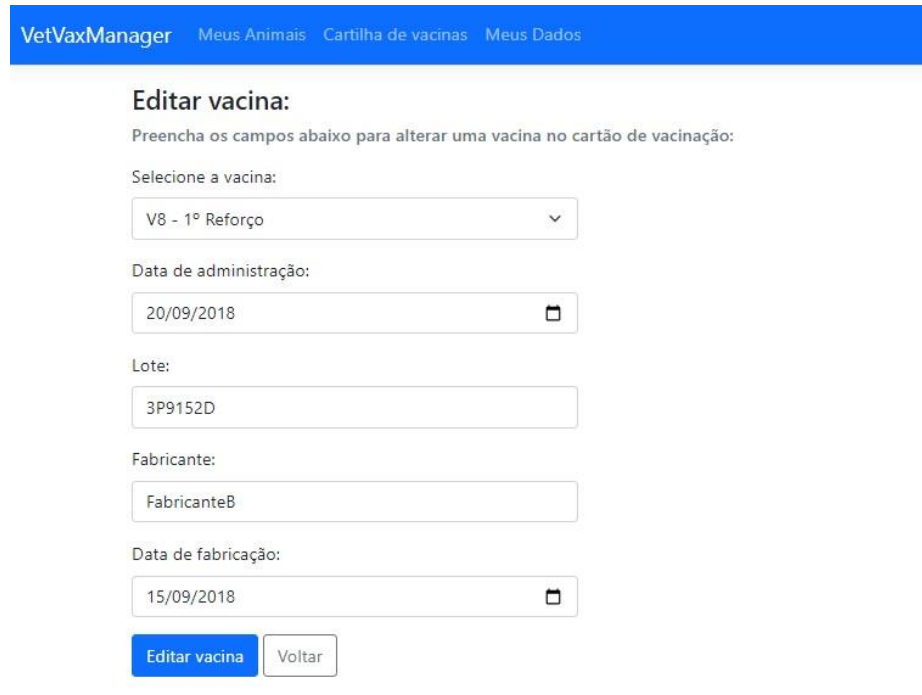
V8 - Única

Editar lembrete

Voltar

Fonte: Reprodução própria

Figura 31 - Tela de alteração de dados de vacina aplicada no animal



VetVaxManager Meus Animais Cartilha de vacinas Meus Dados

Editar vacina:

Preencha os campos abaixo para alterar uma vacina no cartão de vacinação:

Selecione a vacina:

V8 - 1º Reforço

Data de administração:

20/09/2018

Lote:

3P9152D

Fabricante:

FabricanteB

Data de fabricação:

15/09/2018

Editar vacina Voltar

Fonte: Reprodução própria

Figura 32 - Tela de cartilha de vacinação disponível por espécie



Cartilha de Vacinação

Informações detalhadas sobre todas as vacinas disponíveis, categorizadas por espécies animais

Cachorro **Gato** **Cavalo**

V8 - 1ª Dose
6 a 8 semanas

Descrição: Previne contra Adenovirose 2, Cinomose (CDV), Parvovirose, Coronavirose, Parainfluenza canina, Hepatite canina e Leptospirose canina (dois tipos). A V10 é uma alternativa a V8.

Dose: 1ª Dose

Faixa Etária: 6 a 8 semanas

Espécie: Cachorro

V10 - 1ª Dose
6 a 8 semanas

V8 - 1º Reforço

Fonte: Reprodução própria

Figura 33 - Tela de alteração de dados do proprietário

Alterar dados:

Preencha os campos abaixo para editar suas informações:

Nome:

Sobrenome:

Data de nascimento:

Selecione o sexo:

CPF:

E-mail:

Telefone:

Fonte: Reprodução própria

4 CONSIDERAÇÕES FINAIS

Nesta seção, apresentamos as conclusões derivadas do desenvolvimento e implementação do sistema de gerenciamento de vacinação animal, avaliamos a eficácia das soluções propostas e discutimos as contribuições e limitações do trabalho, além de sugestões para trabalhos futuros.

O desenvolvimento deste sistema teve como principal objetivo fornecer uma ferramenta robusta e eficiente para proprietários de animais, permitindo o cadastro completo como proprietário, o cadastro de seus animais, a visualização e gestão das vacinas, e o agendamento de eventos de vacinação. A adoção do padrão arquitetônico MVC demonstrou-se adequada, proporcionando uma clara separação de responsabilidades entre os componentes do sistema e facilitando a manutenção e evolução do código.

A implementação das funcionalidades propostas, como o cadastro do proprietário, o cadastro de animais, a visualização de fichas cadastrais, a gestão de vacinas e o envio de lembretes de vacinação, foi bem-sucedida. O sistema mostrou-se intuitivo e funcional, atendendo às expectativas e necessidades dos usuários finais. As capturas de tela apresentadas evidenciam a clareza e a usabilidade da interface desenvolvida.

Apesar dos resultados positivos, algumas limitações foram identificadas. A redução do escopo inicial para focar exclusivamente no gerenciamento de vacinação, devido a restrições de tempo, implicou na exclusão de outras funcionalidades que poderiam enriquecer o sistema, como a gestão de consultas, tratamentos veterinários, gestão de exames e receitas veterinárias e também na exclusão de atores como veterinários e administradores do sistema. Além disso, a complexidade técnica e os desafios associados à integração com outras plataformas e serviços externos representam áreas que necessitam de atenção futura.

4.1 TRABALHOS FUTUROS

Para trabalhos futuros, sugerimos a expansão do escopo para incluir a gestão completa de clínicas veterinárias, contemplando funcionalidades adicionais como a gestão de consultas, tratamentos, gestão dos exames e receitas veterinárias, prontuário eletrônico e relatórios financeiros. A adição de mais atores como veterinários e administradores que interagissem com as funcionalidades também agregaria maior valor ao sistema. A integração com sistemas externos, como a API de autenticação de usuários utilizando a conta do Google e API de integração com o Google Agenda, também pode proporcionar maior valor aos usuários. A adoção de padrões arquitetônicos mais avançados, como MVP ou MVVM, pode ser considerada para melhorar a testabilidade e modularidade em sistemas mais complexos. Logo, o sistema é uma iniciativa em criar um escopo de funcionalidades hoje existentes no Conecte SUS para o cenário de animais domésticos e rural, no qual este trabalho é um módulo de vacinação do sistema VETVAX, que ao decorrer da evolução do sistema apresentará demais módulos, como consultar, exames, medicamentos, rede de saúde, agendamentos, atendimentos, diário de saúde, ou seja, tais módulos serão desenvolvidos de forma independentes que serão integrados em um sistema maior que denominado de Conecta Pet.

Em resumo, o sistema de gerenciamento de vacinação animal desenvolvido neste trabalho cumpriu seu objetivo de proporcionar uma ferramenta eficiente e acessível para os proprietários de animais. As contribuições deste TCC são significativas, oferecendo uma base sólida para futuras melhorias e expansões, com potencial para impactar positivamente a gestão de saúde animal.

REFERÊNCIAS

- [1]. ABINPET. **Abinpet esclarece a importância de vacinar o pet**. 2016. Disponível em: <https://abinpet.org.br/2016/06/abinpet-esclarece-a-importancia-de-vacinar-o-pet/>. Acesso em: 22 jun. 2024
- [2]. AFYA, IClinic. **iClinic: Software médico para clínicas e consultórios**, 2023. Página inicial. Disponível em: <https://lps.iclinic.com.br>. Acesso em: 25 set. 2023.
- [3]. NINSAÚDE. **Solução completa na gestão de consultórios e clínicas - Ninsaúde Apolo**, 2023. Página inicial. Disponível em: <https://www.apolo.app>. Acesso em: 25 set 2023.
- [4]. PRESSMAN, Roger. **Engenharia de Software: uma abordagem profissional**. 7. ed. Porto Alegre: AMGH, 2011.
- [5]. APPMASTER. Padrões de arquitetura: MVC, MVP e MVVM explicados. 2023. Disponível em: <https://appmaster.io/pt/blog/padrees-arquiteticos-mvc-mvp-e-mvvm>. Acesso em: 27 mai. 2024.
- [6]. BONDI, André. **Characteristics of scalability and their impact on performance**. 2000. Disponível em: <https://dl.acm.org/doi/epdf/10.1145/350391.350432>. Acesso em: 25 set. 2023
- [7]. Coronel, Morris, Rob. **Database System: Design, Implementation and Management**, 9 ed. Boston: Cengage Learning, 2011.
- [8]. Elmasri, Ramez. **Sistemas de bancos de dados**. 6. Ed. São Paulo: Pearson Addison Wesley, 2011.
- [9]. SOMMERVILLE, Ian. **Engenharia de Software**. 9. Ed. São Paulo: Pearson Prentice Hall, 2011.
- [10]. GOUVEA, Kalebe. **VetVaxManager**. 2024. Disponível em: <https://github.com/KalebeGouvea/vet-vax-manager>. Acesso em 05 mai. 2024

APÊNDICE A - VaccineController

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Globalization;
using System.Reflection.PortableExecutable;
using VetVaxManager.Models;
using VetVaxManager.Repository;

namespace VetVaxManager.Controllers
{
    [Authorize]
    public class VaccineController : Controller
    {
        IAnimalRepository _animalRepository;
        IVaccineRepository _vaccineRepository;
        ICalendarRepository _calendarRepository;
        public VaccineController(IAnimalRepository animalRepository,
            IVaccineRepository vaccineRepository, ICalendarRepository calendarRepository)
        {
            _animalRepository = animalRepository;
            _vaccineRepository = vaccineRepository;
            _calendarRepository = calendarRepository;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult VaccinationSchedule()
        {
            var vaccinationSchedule =
                _vaccineRepository.GetVaccinationSchedules();
            if (vaccinationSchedule == null)
            {
                return NotFound();
            }
            return View(vaccinationSchedule);
        }

        public IActionResult VaccineDetails(int id)
        {
            if (id <= 0)
            {
                return BadRequest("Invalid Vaccine ID.");
            }

            var vaccine = _vaccineRepository.GetVaccineById(id);
            if (vaccine == null)
            {
                return NotFound();
            }
            return View(vaccine);
        }
    }
}

```

```

public IActionResult DeleteVaccine(int id, int animalId)
{
    if (id <= 0 || animalId <= 0)
    {
        return BadRequest("Invalid Vaccine and/or Animal ID.");
    }

    var vaccine = _vaccineRepository.DeleteVaccineById(id);

    var redirectId = animalId;
    return RedirectToAction("Details", "Animal", new {id = redirectId});
}

public ActionResult NewVaccine(int id)
{
    if (id <= 0)
    {
        return BadRequest("Invalid Animal ID.");
    }

    var animal = _animalRepository.GetAnimalById(id);
    if (animal == null)
    {
        return NotFound();
    }

    ViewBag.AnimalId = id;
    ViewBag.VaccinationSchedules = _vaccineRepository.GetVaccinationSchedules()
        .Where(v => v.Specie.SpecieId == animal.Specie.SpecieId)
        .ToList();

    return View();
}

[HttpPost]
public ActionResult NewVaccine(Vaccine vaccine)
{
    if (vaccine == null)
    {
        return BadRequest("Vaccine data is required.");
    }

    var errors = ModelState.Values.SelectMany(v => v.Errors);
    if (ModelState.IsValid)
    {
        var animal = _animalRepository.GetAnimalById(vaccine.Animal.AnimalId);
        if (animal == null)
        {
            return NotFound("Animal not found.");
        }
        vaccine.Animal = animal;
    }
}

```

```

        var vaccinationSchedule = vaccine.VaccinationSchedule =
            _vaccineRepository.GetVaccinationSchedules()
                .FirstOrDefault(v =>
v.VaccinationScheduleId == vaccine.VaccinationSchedule.VaccinationScheduleId);
        if (vaccinationSchedule == null)
        {
            ModelState.AddModelError("VaccinationSchedule", "Invalid
vaccination schedule.");
        }

        var vaccineId = _vaccineRepository.NewVaccine(vaccine);
        return RedirectToAction("Details", "Animal", new { id =
vaccine.Animal.AnimalId });
    }

    var animalRollback =
        _animalRepository.GetAnimalById(vaccine.Animal.AnimalId);
    if (animalRollback == null)
    {
        return NotFound("Animal not found.");
    }
    vaccine.Animal = animalRollback;
    return View(vaccine);
}

public IActionResult EditVaccine(int id)
{
    if (id <= 0)
    {
        return BadRequest("Invalid Vaccine ID.");
    }

    var vaccine = _vaccineRepository.GetVaccineById(id);
    if (vaccine == null)
    {
        return NotFound();
    }

    var vaccinationSchedule = _vaccineRepository.GetVaccinationSchedules()
        .Where(v => v.Specie.SpecieId ==
vaccine.Animal.Specie.SpecieId)
        .ToList();
    if (vaccinationSchedule == null)
    {
        return NotFound();
    }

    ViewBag.VaccinationSchedules = vaccinationSchedule;
    return View(vaccine);
}

```

```

[HttpPost]
public ActionResult EditVaccine(Vaccine vaccine)
{
    var animal =
        _animalRepository.GetAnimalById(vaccine.Animal.AnimalId);
    if (animal == null)
    {
        return NotFound();
    }
    vaccine.Animal = animal;

    var vaccinationSchedule =
        _vaccineRepository.GetVaccinationSchedules()
                           .FirstOrDefault(v =>
v.VaccinationScheduleId == vaccine.VaccinationSchedule.VaccinationScheduleId);
    if (vaccinationSchedule == null)
    {
        return NotFound();
    }
    vaccine.VaccinationSchedule = vaccinationSchedule;

    var errors = ModelState.Values.SelectMany(v => v.Errors);
    if (ModelState.IsValid)
    {
        var updatedVaccine =
            _vaccineRepository.UpdateVaccine(vaccine);
        return RedirectToAction("Details", "Animal", new { id =
vaccine.Animal.AnimalId });
    }

    vaccine.Animal = animal;

    return View(vaccine);
}
}
}

```


APÊNDICE B - VaccineDetailsView

```
@{
    ViewData["Title"] = "Detalhes da vacina";
    @model VetVaxManager.Models.Vaccine;
}

<div class="container my-2">
    <h4 class="mb-3">Detalhes da vacina:</h4>
    <div class="row row-cols-1 row-cols-md-3 g-4">
        <div class="col">
            <div class="card" style="width: 50rem;">
                <div class="card-body">
                    <h5 class="card-
title">@Model.VaccinationSchedule.Name</h5>
                    <h6 class="card-subtitle mb-2 text-
muted">@Model.VaccinationSchedule.Dose</h6>
                    <p class="card-text"><b>Data de administração:</b>
@Model.DateOfAdministration.ToString("dd-MM-yyyy")</p>
                    <p class="card-text"><b>Data de fabricação:</b>
@Model.DateOfManufacture.ToString("dd-MM-yyyy")</p>
                    <p class="card-text"><b>Fabricante:</b>
@Model.Manufacturer</p>
                    <p class="card-text"><b>Lote:</b> @Model.Lot</p>
                </div>
            </div>
        </div>
    </div>
</div>
<div class="container mt-2">
    <a class="btn btn-outline-secondary" asp-area="" asp-controller="Animal"
asp-action="Details" asp-route-id="@Model.Animal.AnimalId"
role="button">Voltar</a>
    <a class="btn btn-secondary" asp-area="" asp-controller="Vaccine" asp-
action="EditVaccine" asp-route-id="@Model.VaccineId" role="button">Editar</a>
    <button type="button" class="btn btn-danger" data-bs-toggle="modal" data-
bs-target="#deleteVaccineModal">Deletar</button>
</div>

<!-- Modal -->
<div class="modal fade" id="deleteVaccineModal" tabindex="-1" aria-
labelledby="deleteVaccineModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title"
id="deleteVaccineModalLabel">Deletar</h5>
                <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                Tem certeza que deseja deletar a vacina do cartão?
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Fechar</button>
                <a class="btn btn-danger" asp-area="" asp-controller="Vaccine"
asp-action="DeleteVaccine" asp-route-id="@Model.VaccineId" asp-route-
animalId="@Model.Animal.AnimalId" role="button">Confirmar</a>
            </div>
        </div>
    </div>
</div>
</div>
```