Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus “José Santilli Sobrinho”

JOÃO PEDRO OLIVEIRA PRESTUPA

SOFTWARE PARA GESTÃO DE CLÍNICA VETERINÁRIA

Assis/SP
2024

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

JOÃO PEDRO OLIVEIRA PRESTUPA

SOFTWARE PARA GESTÃO DE CLÍNICA VETERINÁRIA

Projeto de pesquisa apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e à Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): João Pedro Oliveira Prestupa
Orientador(a): Me. Guilherme de Cleva Farto

Assis/SP
2024

Prestupa, João Pedro Oliveira

P942s Software para gestão de clínica veterinária / João Pedro Oliveira Prestupa. -- Assis, 2024.

47p.

Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) -- Fundação Educacional do Município de Assis (FEMA), Instituto Municipal de Ensino Superior de Assis (IMESA), 2024.

Orientador: Prof. Me. Guilherme de Cleva Farto

1. Sistemas de informações gerenciais. 2. Saúde animal. I Farto, Guilherme de Cleva. II Título.

CDD 003

Elaborada por Anna Carolina Antunes de Moraes – Bibliotecária – CRB-8/10982

SOFTWARE PARA GESTÃO DE CLÍNICA VETERINÁRIA

JOÃO PEDRO OLIVEIRA PRESTUPA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Prof. Me. Guilherme de Cleva Farto

Examinador: _____
Prof. Esp. Domingos de Carvalho Villela Júnior

AGRADECIMENTO

Agradeço a Deus primeiramente, por me dar saúde e forças para a conclusão deste capítulo tão importante na minha vida, pois sem ele isso não seria possível.

Aos meus pais que sempre me apoiaram e incentivaram nos estudos e em minhas escolhas, a eles dedico não só este trabalho mas esta conquista por um todo.

Aos meus amigos que muitos tenho uma amizade de vários anos, outros que conheci nessa jornada, mas que todos me agregaram e espero ter agregado muitos momentos únicos de diversão e principalmente aprendizado e motivação.

Também quero agradecer ao meu orientador Guilherme Cleva Farto que, sempre foi muito paciente e mostrava-se sempre interessado no projeto e em ajudar, um professor e orientador que sempre procura fazer mais do que o mínimo.

Por fim quero agradecer a todos os professores e alunos que estiveram comigo ao longo desta jornada, espero reencontrá-los no futuro.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

RESUMO

Um software contendo soluções para potencializar a agilidade no agendamento de consultas, controle de vacinação e estado do animal. Por meio de seu fácil e intuitivo uso, o cliente poderá marcar uma consulta online de acordo com a disponibilidade do veterinário. Assim evitando filas e agilizando o atendimento, além de acompanhar o histórico de seu animal por meio de uma plataforma online.

Palavras chave: animal, *web*, vacinação, agendamento;

ABSTRACT

A software containing solutions to enhance agility in scheduling appointments, vaccination control, and monitoring the animal's status. Through its easy and intuitive use, the customer will be able to schedule an appointment online according to the veterinarian's availability, thus avoiding queues and expediting the service. Additionally, they can track their animal's history through an online platform.

Key words: animal, *web*, vaccination, scheduling;

1. INTRODUÇÃO

Com o crescimento contínuo na área da saúde animal, tem-se observado um aumento na demanda por softwares que atendam eficazmente às necessidades de gerenciamento das clínicas veterinárias. De acordo com Jaffar (2021), o uso da tecnologia nesse setor tem experimentado um crescimento significativo nos últimos anos, em resposta a essas crescentes demandas. Este trabalho propôs o desenvolvimento de um sistema que proporcionasse um atendimento dinâmico aos clientes, visando evitar filas e reduzir as demoras no atendimento.

O software proposto por este trabalho surgiu quando foram notadas reclamações de clientes em clínicas veterinárias sobre a demora no atendimento, quando a agilidade falha e são marcados horários e os mesmos não são respeitados pelas clínicas, e a falta de um sistema que permita por meio de poucos cliques agendar uma consulta online.

Segundo Nascimento (2021) o mercado vem se tornando cada vez mais competitivo e, o melhor atendimento ao cliente é um diferencial enorme ainda mais quando se trata de uma área em expansão de acordo com Aguiar (2015).

Após uma pesquisa sobre os softwares já existentes, constatou-se a existência de deficiência na agilidade dos próprios. Tais falhas são devido a consultas marcadas em horários próximos causando a reclamação dos clientes, uma forma tradicional de agendamento por telefone ou presencial.

O objetivo deste trabalho está em reunir os pontos positivos encontrados nos sistemas pesquisados, bem como adicionar melhorias como, por exemplo, o agendamento dinâmico para o cliente.

O auto-crescimento na área de medicina animal fez com que a demanda sobre softwares nesse ramo aumentasse intensamente, segundo Aguiar (2015) a demanda de boas aplicações para gerência dessas clínicas é maior que as já existentes e, por isso, é uma área em expansão. Os já existentes estão em sua maioria ultrapassados e não suprem todas as necessidades. Uma solução encontrada foi por meio do desenvolvimento de uma aplicação Web com maior gerência e agendamento dinâmico para os pacientes.

De acordo com Nascimento (2021), o atendimento e a organização de qualidade se tornaram imprescindíveis no mundo corporativo. Dito isso, trouxe como solução nesta

aplicação o agendamento dinâmico, onde ao agendar uma consulta, o administrador do sistema vai validar aquele horário, fazendo com que o próximo cliente venha no tempo certo e não tenha que ficar esperando ser atendido.

Este trabalho tem como objetivo solucionar problemas de gestões e agilidade para uma clínica veterinária, trazendo dinâmica, produtividade e um melhor gerenciamento. Como objetivo específico a aplicação contará com: Agendamento de consultas online, controle do estado dos animais e vacinação e gerenciamento de horários.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

O presente trabalho tem por objetivo desenvolver um sistema para gerenciamento de uma clínica veterinária de médio e grande porte, com agendamentos dinâmicos entre veterinários e clientes, além de histórico de consultas, vacinações e a possibilidade do cliente monitorar seus animais e marcar consultas conforme a disponibilidade.

Com o objetivo de aumentar a produtividade do estabelecimento e levar mais agilidade e praticidade ao cliente final, a solução defendida por este trabalho deve auxiliar tanto para clínicas que cuidam de animais domésticos quanto animais de produção, levando o que há de mais novo em tecnologia e gestão de empresas ao ramo de cuidado de animais.

1.1.2 OBJETIVOS ESPECÍFICOS

Como objetivo específico, as seguintes funcionalidades responsáveis pela construção do aplicativo são:

- Agendamento dinâmico: o próprio cliente irá marcar sua consulta de acordo com a disponibilidade do veterinário;
- Gerenciamento de horário: o cliente e o administrador podem ver os horários livres de forma fácil;
- Históricos de consultas: o sistema vai disponibilizar dados sobre consultas anteriores do mesmo cliente;
- Relatórios para o cliente: o cliente poderá receber dados sobre seu animal;

- Interface: o sistema propõe uma interface limpa e intuitiva para ambos, tanto cliente quanto funcionários da clínica a fim de facilitar seu uso;

1.2 PÚBLICO ALVO

Este trabalho tem como público alvo gerentes de clínicas veterinárias de grande e médio porte que desejam aumentar sua produtividade, um maior controle sobre seu negócio, agilizar o agendamento de consultas e vacinas. Estes benefícios serão refletidos para o cliente final, pois, diminuirá significativamente o tempo de espera para vacinar ou consultar o animal.

Tendo em vista que o mercado está cada vez mais competitivo e em crescente estado segundo FREITAS (2016), portanto qualquer vantagem que se possa ter é válida e bem vinda. A solução proposta e desenvolvida neste trabalho fornece uma melhora na agilidade de qualquer clínica que queira ser destaque em sua região, levando a tecnologia até a medicina animal.

1.3 JUSTIFICATIVAS

Uma das principais dores que este trabalho propôs a atender é diminuir o tempo de espera no local. Como dito por Gomes (2023) e Rojas (2012) muitos destes estabelecimentos ainda adotam métodos ineficientes para agendamento, como ordem de chegada e gerência por meio de papel e caneta. Após diversas reclamações em diferentes segmentos, incluindo este, chegou-se à conclusão de que é necessário implementar uma solução para acabar com esses problemas. Além disso, o sistema conta com um histórico das consultas e vacinação do animal, como também uma avaliação de seu bem estar.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado nas seguintes partes:

- **Capítulo 1 - Introdução**
- **Capítulo 2 - Tecnologias e Ferramentas de Desenvolvimento**
- **Capítulo 3 - Análise e Especificação do Sistema**
- **Capítulo 4 - Desenvolvimento do Sistema**
- **Capítulo 5 - Conclusão**
- **Referências**

2. MÉTODO DE DESENVOLVIMENTO

A metodologia de pesquisa inclui a revisão bibliográfica e a coleta de relatos de profissionais na área para a realização de um sistema de gerência de clínica veterinária. Na revisão bibliográfica, são apresentados conceitos já existentes e novos para qualificar mais ainda a área abordada.

Já no desenvolvimento do software, foi utilizado as seguintes tecnologias: Linguagem de programação Java e Spring Framework para o back-end, no front-end as tecnologias implementadas foram React, Tailwind CSS que segundo Soares (2022) a combinação entre essas duas tecnologias é capaz de aumentar o nível de acessibilidade das páginas web, tendo em vista sua popularidade e praticidade e a biblioteca de componentes MUI Material.

O MUI Material segundo (mui, MUI Material) “funciona isoladamente. Ele é independente, e só injeta os estilos que ele precisa para exibir. Não conta com qualquer folha de estilo global como normalize.css”. A escolha desta tecnologia se deu por sua enorme variedade de componentes que, unidos ao React tornam o desenvolvimento do front-end muito mais rápido e prático.

A utilização do Spring vem da necessidade de abranger diferentes dispositivos que segundo Junior (2021) o desenvolvimento de uma *API REST* com o Spring Framework oferece desacoplamento e a vantagem de desenvolver apenas um back-end para vários front-end.

O conceito de *API* segundo Machado (2023) significa *Application Programming Interface*, e a implementação desse conceito nos permite executar funções dentro do sistema de forma padronizada, abstraindo toda a complexidade da tarefa, já o *REST* seria o modelo de desenvolvimento da própria.

2.1 QUANTUX-UX

Uma plataforma web destinada a criação de interfaces, seja web, desktop ou mobile, sua usabilidade é simples fazendo com que se torne uma ferramenta fácil e intuitiva de usar, perfeita para situações onde se precisa expor suas ideias de protótipos de sites antes de

desenvolvê-los de fato. Com uma ampla possibilidade de criação de interfaces, a escolha dessa tecnologia para este trabalho foi feita por sua fácil utilização e usabilidade.

2.2 JAVA

A linguagem de programação Java foi selecionada como base para o desenvolvimento da plataforma devido à sua robustez, escalabilidade e ampla aceitação no mercado. Amplamente utilizada em diferentes setores, como bancos, fintechs e e-commerce, a linguagem Java oferece capacidade para lidar com grandes volumes de dados e operações complexas, demonstrando sua confiabilidade e eficácia em ambientes exigentes.

Uma característica fundamental do Java é sua capacidade de ser compilado para bytecode, permitindo sua execução em qualquer plataforma que possua uma JVM (Java Virtual Machine), incluindo ambientes Windows, Linux, macOS e Android. Além disso, o Java oferece recursos de segurança robustos, protegendo a aplicação contra diversos tipos de ataques. Sua vasta biblioteca de recursos prontos para uso e o fato de ser uma linguagem de código aberto a tornam uma escolha ideal para o desenvolvimento de aplicações robustas, escaláveis e seguras.

2.3 SPRING FRAMEWORK

O Spring Boot, framework desenvolvido em Java, foi adotado neste trabalho devido à sua capacidade de agilizar o desenvolvimento web, tornando-o mais eficiente e versátil. Reconhecido por sua segurança e robustez, o Spring oferece uma ampla gama de recursos que podem ser utilizados, seja por meio das opções fornecidas pelo próprio Spring ou de ferramentas open-source compatíveis.

O suporte a longo prazo e as constantes atualizações do Spring tornam-no uma escolha ideal para o desenvolvimento de aplicações web, garantindo estabilidade e confiabilidade ao longo do tempo.

Uma das bibliotecas deste framework mais importantes que foram utilizadas foi o Spring Security, ela é usada para garantir a segurança na aplicação, permitindo a criação de

roles para cada usuário, controlar níveis de acesso, autenticar e cadastrar, filtros e respostas.

2.4 REACT

Uma biblioteca desenvolvida pelo Facebook em 2011, foi escolhida para o desenvolvimento do front-end da plataforma devido à sua abordagem inovadora de componentização. Essa abordagem permite a reutilização de funções ou componentes em diferentes partes do código, resultando em um código mais limpo, claro, produtivo e intuitivo de entender e desenvolver.

Com recursos como JSX (JavaScript XML), que combina HTML e JavaScript, o React facilita a escrita de interfaces de usuário, especialmente para desenvolvedores familiarizados com HTML. Além disso, bibliotecas como Next.js permitem a renderização do lado do servidor (server-side rendering - SSR), melhorando o SEO da aplicação ao facilitar a indexação do conteúdo inicial da página pelos mecanismos de busca.

2.5 TAILWIND

Tailwind CSS foi escolhido como framework de estilização devido à sua abordagem simplificada e eficiente para o desenvolvimento usando CSS. Ao transformar a forma tradicional de escrever CSS em simples tags HTML, o Tailwind CSS oferece uma maneira mais rápida e ágil de estilizar interfaces, tornando-se uma escolha ideal para este trabalho.

2.6 POSTGRESQL

Uma ferramenta poderosa e open-source, o PostgreSQL é um banco de dados relacional robusto que permite usar linguagem SQL para manipulação das tabelas e de fácil

conexão com qualquer *API*, a escolha dessa tecnologia se deu por sua alta confiabilidade e segurança.

2.7 CLEAN ARCHITECTURE

O conceito de arquitetura limpa foi desenvolvido pelo vangloriado Robert C. Martin ou mais conhecido como *Uncle* Bob, é uma estratégia de desenvolvimento de software separada por camadas, provendo uma estrutura organizada e de fácil manutenção.

Sua principal característica é sua separação e independência das camadas, como o desacoplamento da regra de negócio, da entidade, interface de usuário e frameworks, tendo como regra uma direção unidirecional em seu desenvolvimento, ou seja, somente as camadas externas acessando as internas e não o contrário como na Figura 6.

Quando a separação é feita de forma correta se tem diversos benefícios como a facilidade de escrita de testes automatizados, a possibilidade de troca de frameworks, manutenção mais fácil dentre outras, segundo o próprio autor do livro Clean Architecture Robert C. Martin “Uma boa arquitetura faz um sistema fácil de entender, fácil de desenvolver, fácil de manter e fácil de fazer o deploy. O principal objetivo é minimizar o custo de tempo e maximizar a produtividade do programador.” O funcionamento da arquitetura é demonstrado na Figura 1.

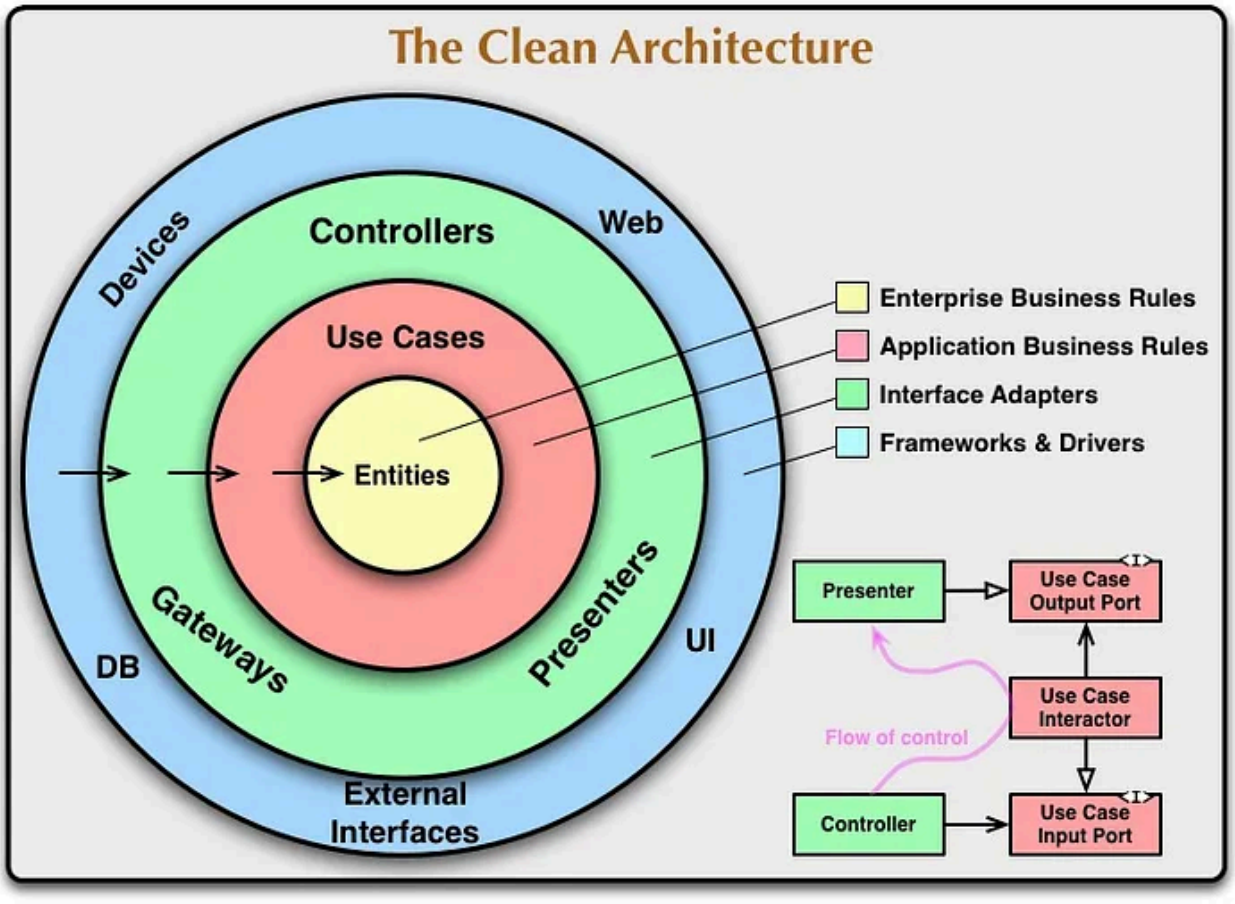


Figura 1 – Exemplo de arquitetura limpa

2.8 BCRYPT PASSWORDENCODER

Uma tecnologia presente também no ecossistema spring, é comumente usada para mascarar senhas e verificá-las em aplicações web. Foi criada para ser segura e adaptativa, feita por meio de algoritmos intensos, foi feita para conter tentativas de ataques hacker de força bruta.

2.9 JSON

É uma estrutura de dados em formato de texto, que serve para serem utilizados em diferentes tipos de sistemas. Por se tratar de um formato simples e leve ele é rápido e usado na maioria dos casos para transferência de dados, assim como nesta aplicação.

2.10 SWAGGER API

Uma biblioteca de código aberto que ajude a documentar o sistema, consumir API's, testar *endpoints,* e muito mais, permite o envio de json, arquivos, xml e está presente em quase todas as linguagens de programação back-end. Sua utilidade em teste e documentação foram cruciais para o desenvolvimento deste *software*.

3. ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

A análise feita para este trabalho baseia-se em relatos de profissionais da área e de um estudo dos sistemas já existentes, visando trazer novas funcionalidades.

3.1 LEVANTAMENTO DE REQUISITOS

1. Manter consultas;
2. Manter veterinários;
3. Manter clientes;
4. Manter histórico de consultas;
5. Manter vacinações;
6. Agendar consultas facilmente;
7. Solicitar agendamento de consulta online;
8. Avaliar bem-estar do animal;
9. Manter histórico de vacinações;

Para uma melhor visualização e entendimento das funcionalidades principais, foram desenvolvidos diagramas de caso de uso especificando tanto ações gerais, quanto específicas do sistema, feitas para fornecer um melhor entendimento do fluxo de funcionamento e dos processos que são feitos, assim como possíveis ações dos usuários. Portanto foram listados nas Figuras 1, 2 e 3 os casos de uso dos usuários cliente e administrador, que pode ser um veterinário ou um funcionário da clínica.

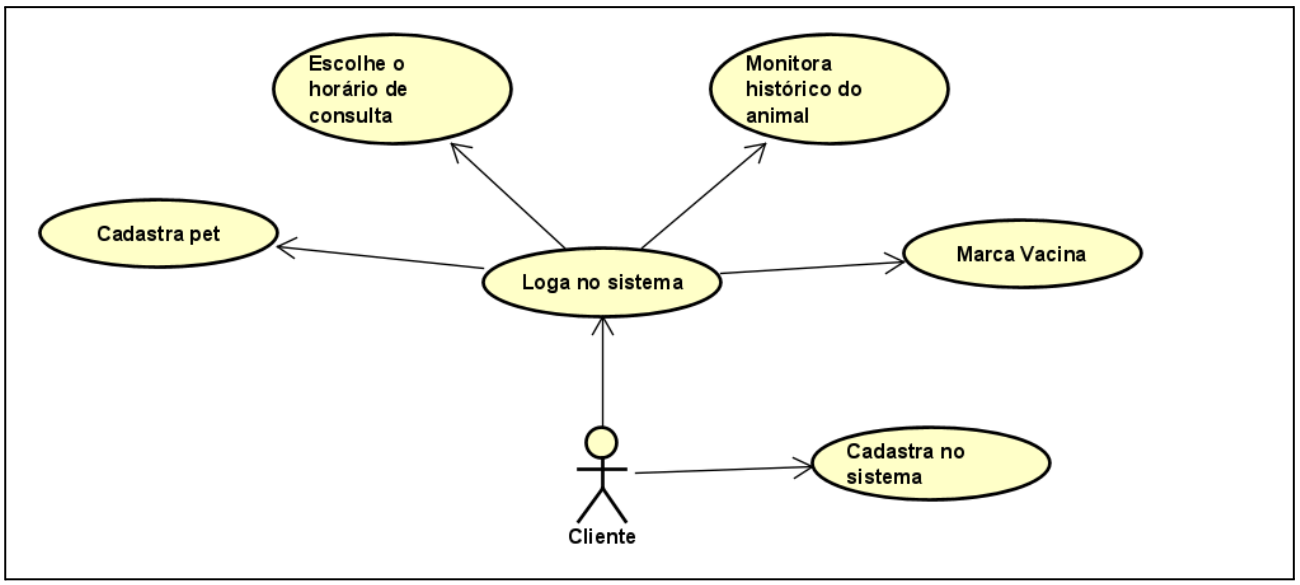


Figura 2 – Diagrama de Caso De Uso – Cliente

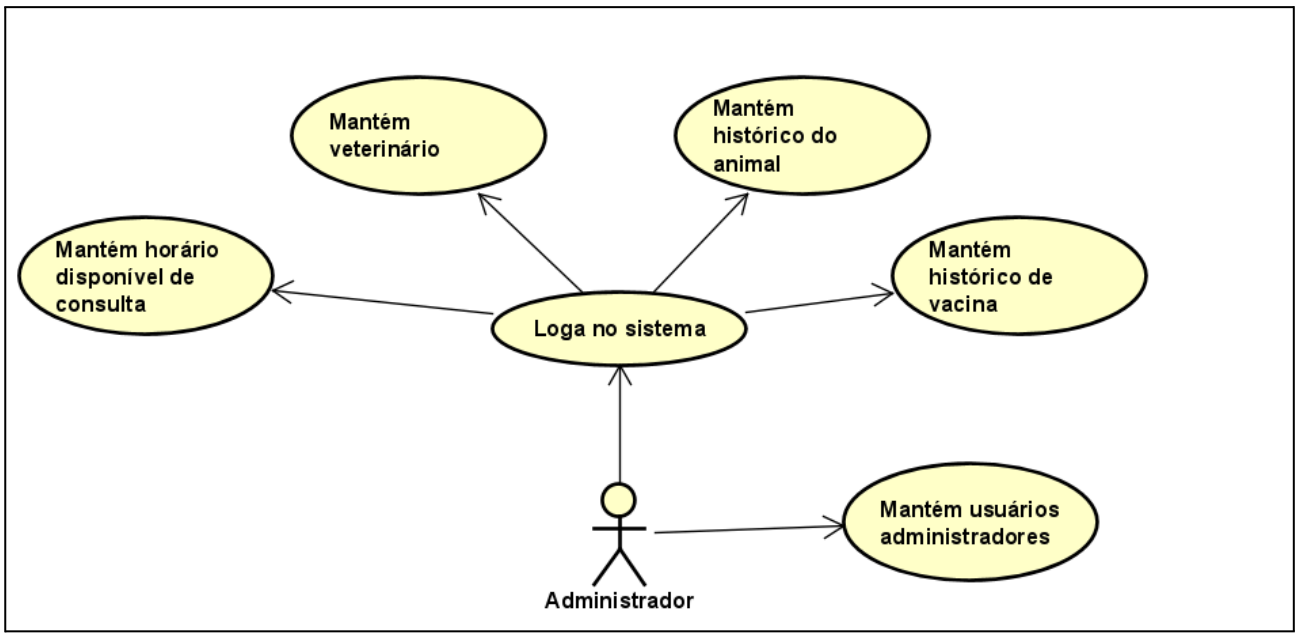


Figura 3 – Diagrama de Caso De Uso – Administrador

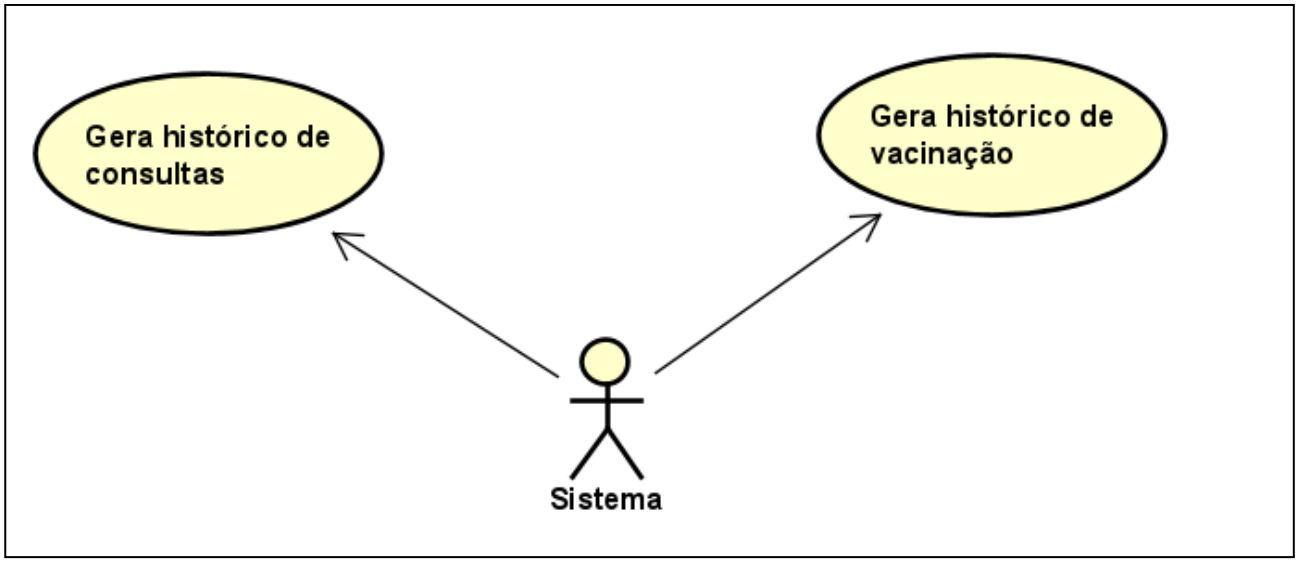


Figura 4 – Diagrama de Caso De Uso – Sistema

3.2 DIAGRAMA DE CASOS DE USO

Como mostrado na Figura 1 abaixo, o diagrama de caso de uso geral do sistema, provê um maior entendimento de suas funcionalidades

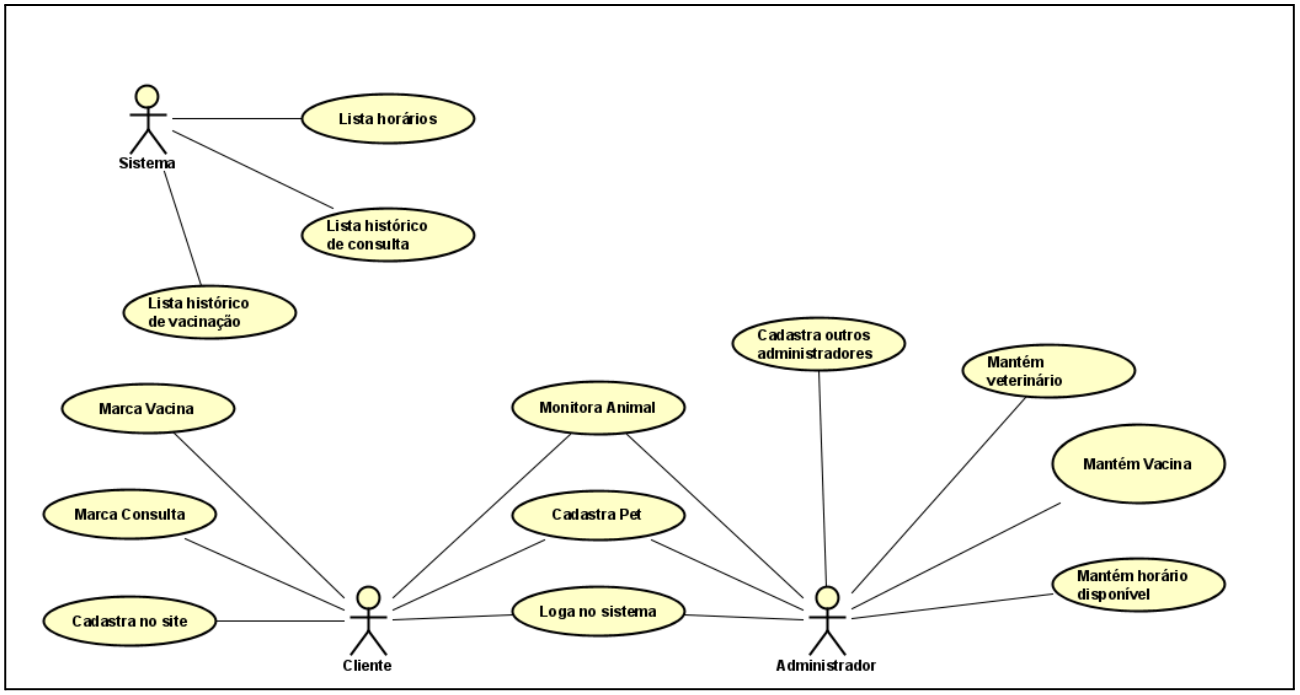
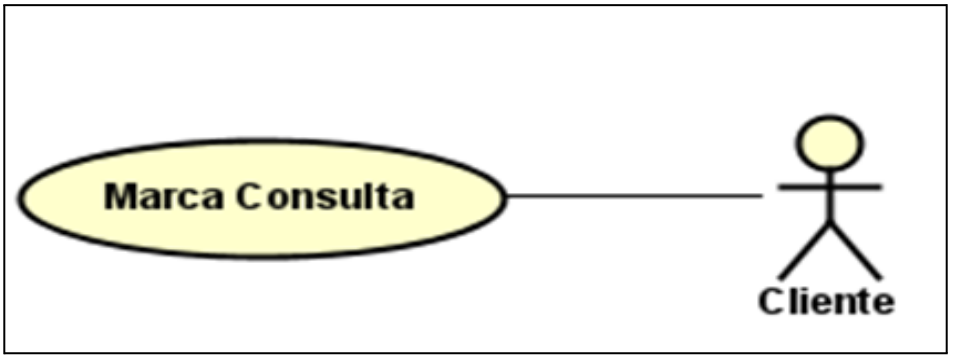


Figura 5 – Diagrama de Caso De Uso Geral

Abaixo nas Figuras 7.1.1, 7.1.2 e 7.1.3. Apresento de forma mais detalhada o fluxo e o funcionamento dos casos de uso principais, juntamente com o que é esperado do sistema.

3.2.1 UC 01 – MARCAR CONSULTA

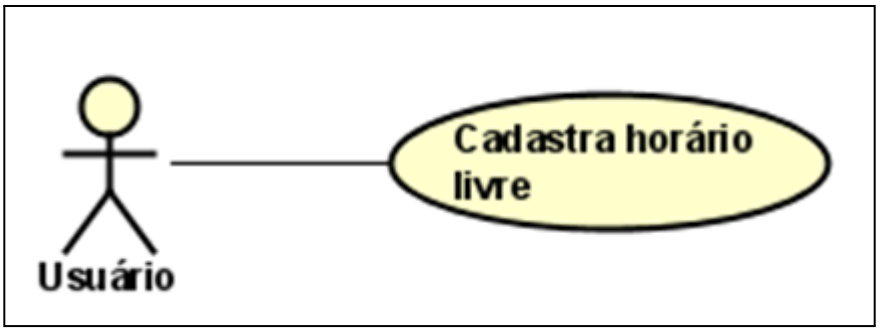


1 - Finalidade/Objetivo	Marcar consulta.
2 - Atores	Cliente.
3 - Pré-Condições	Estar logado.
4 - Evento inicial	O ator começa o caso de uso selecionando a opção de marcar consulta.
5 - Fluxo Principal	a) O sistema apresenta a interface com a lista de horários disponíveis; b) O sistema mostra as opções de marcar consulta; c) O ator seleciona o horário desejado. d) O ator preenche os dados necessários para cadastrar a consulta. e) O ator clica em cadastrar; f) O sistema exibe um pop-up de sucesso e volta para o fluxo “a”. g) Caso de uso encerrado.

6 - Fluxo Alternativo	**A1 – Não há horários disponíveis.** a) O ator clica em marcar consulta; b) O sistema exibe um aviso de “Sem horários disponíveis, entre em contato com a clínica”; c) Caso de uso encerrado.

Tabela 1: UC 01 – Marcar consulta

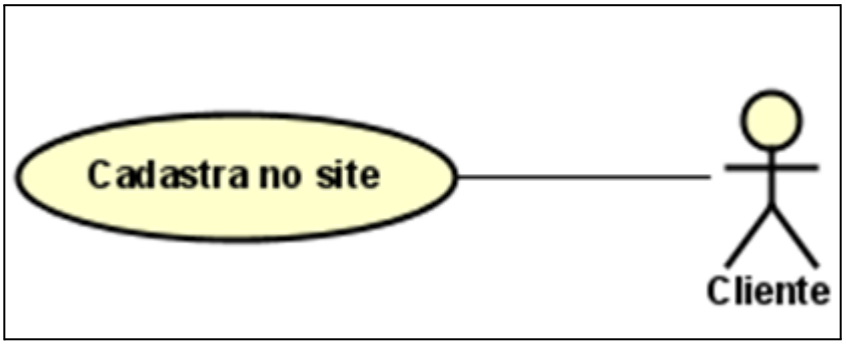
3.2.2 UC 02 – MARCAR HORÁRIO



1 - Finalidade/Objetivo	Cadastrar horário disponível
2 - Atores	Usuário.
3 - Pré-Condições	Estar logado como administrador.
4 - Evento inicial	O ator começa o caso de uso selecionando a opção de cadastrar horário.
5 - Fluxo Principal	a) O sistema apresenta a interface de cadastro de horários b) O sistema pede os dados de veterinário e horário. c) O ator clica em cadastrar e o horário é adicionado na lista de disponíveis. d) Caso de uso encerrado.

Tabela 2: UC 02 – Cadastrar Horário.

3.2.3 UC 03 – MANTER CLIENTE



1 - Finalidade/Objetivo	Marcar cliente.
2 - Atores	Cliente.
3 - Pré-Condições	Entrar no site.
4 - Evento inicial	O ator começa o caso de uso selecionando a opção de cadastrar.
5 - Fluxo Principal	a) O sistema apresenta a interface de cadastro; b) O ator preenche os formulários com os dados necessários; c) O ator clica em cadastrar. d) O sistema leva o cliente para a tela inicial. e) Caso de uso encerrado. f) O sistema exibe um pop-up de sucesso e volta para o fluxo “a”. g) Caso de uso encerrado.

Tabela 3: UC 03 – Manter cliente**.**

3.3 DIAGRAMA DE CLASSE

Os diagramas de classe servem para tornar mais fácil a visualização das funcionalidades dos métodos e dos atributos dos objetos, abaixo na Figura 1 será mostrado a estrutura das classes do projeto.

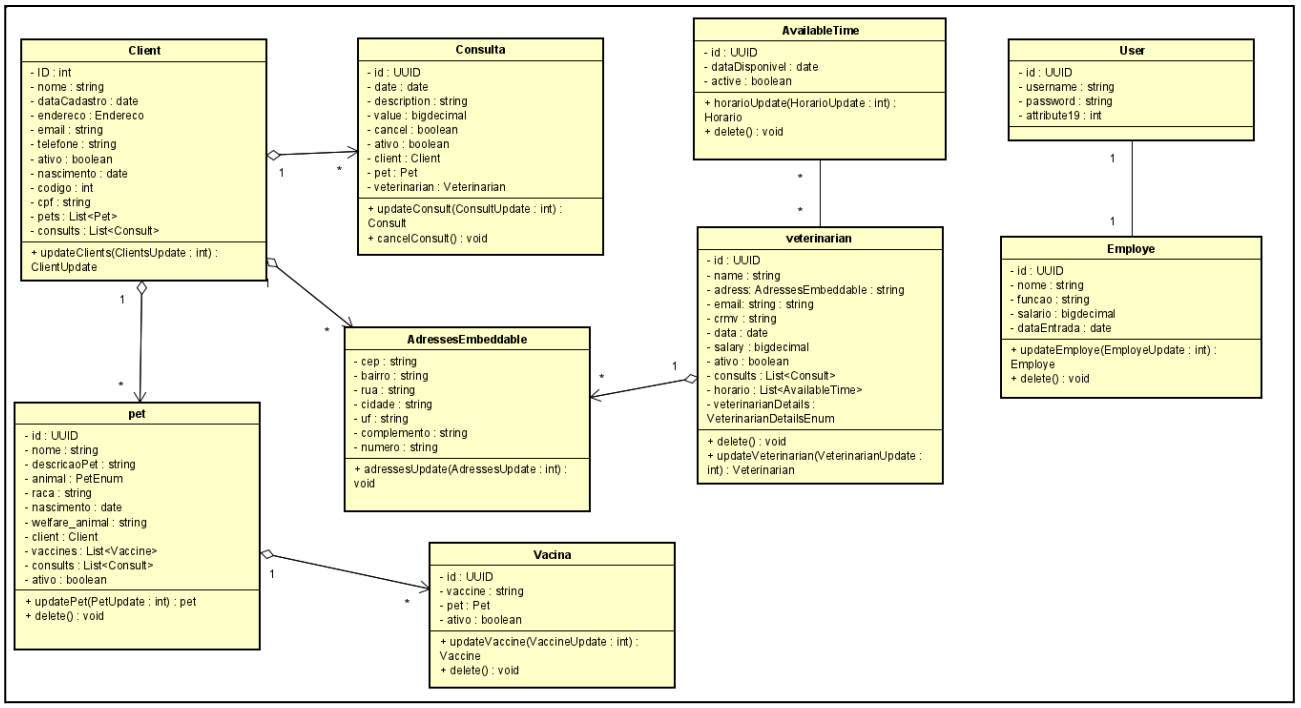


Figura 6 : Diagrama de Classe

3.4 DIAGRAMA DE ENTIDADE-RELACIONAMENTO

O diagrama de relacionamento de entidade é uma representação do banco de dados da aplicação, fator muito importante para a funcionalidade do sistema**,** como mostrado abaixo.

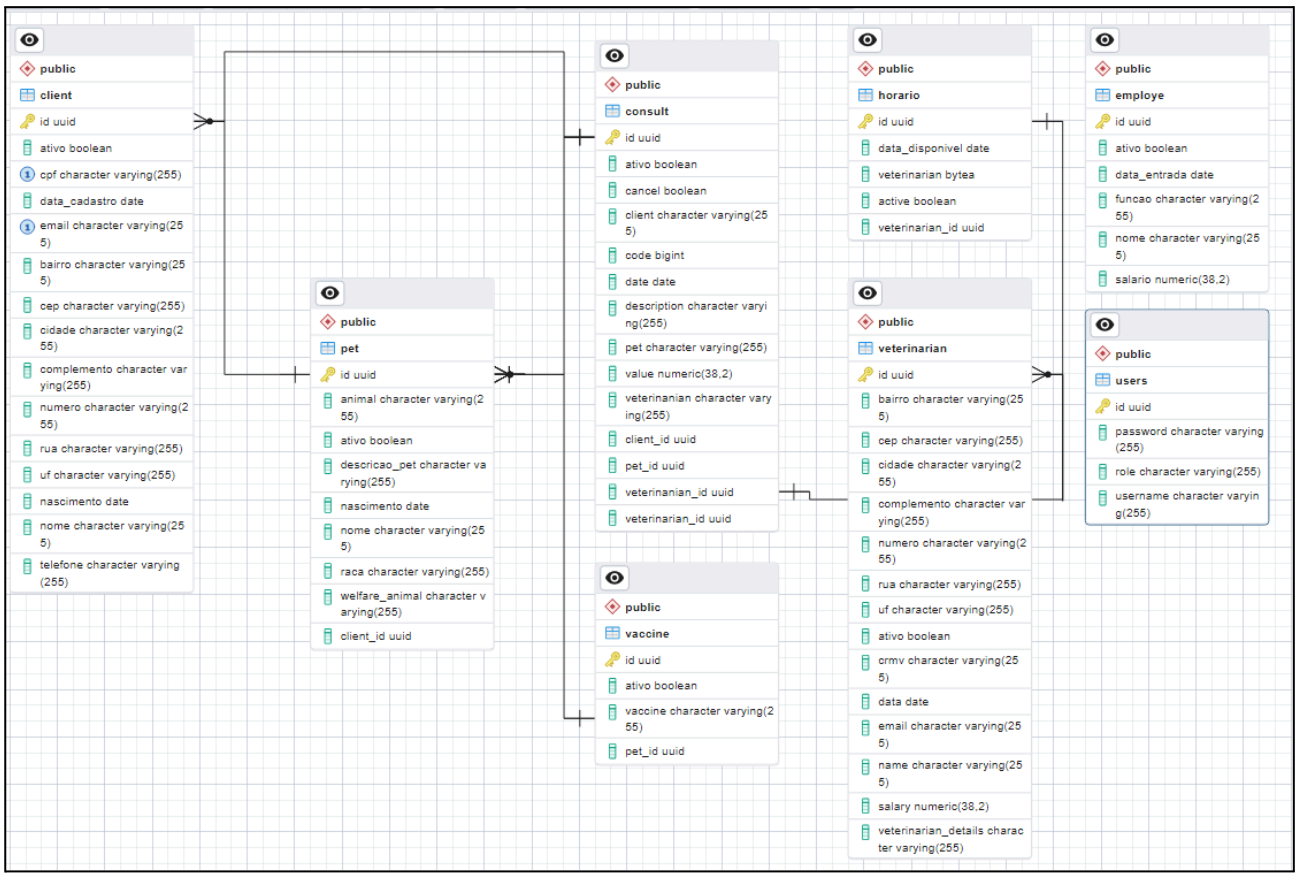


Figura 7 : Modelo de Entidade-Relacionamento

4. DESENVOLVIMENTO DO SISTEMA

4.1 FUNCIONAMENTO

A *API* deste trabalho foi desenvolvida utilizando-se como base a Clean Archicteture, não seguindo 100% a risca dessa arquitetura, limitando-se em utilizar-se dos conceitos de separar as camadas de acesso para web, conforme ilustrada na Figura 8, o sistema foi desenvolvido em camadas seguindo a regra de acesso somente pela parte externa e nunca pela interna. A ideia por trás dessa arquitetura é que cada camada tenha sua função e que acesse somente o que deve ou uma camada abaixo dela, com um acoplamento forte e um sistema de autenticação bem definido a aplicação se torna segura e robusta, preparada para ser escalável conforme sua necessidade.

O usuário final acessa a interface do front-end da aplicação e por meio dela faz as requisições que serão válidas pela *API*. O sistema conta com uma camada de segurança robusta feita por papéis, onde somente a rota de autenticação está liberada para acesso público, e o restante foi devidamente pensado para cada papel.

Os usuários do tipo "*User"* foram desenvolvidos para os funcionários da clínica, portanto seus acessos são mais liberados permitindo a visualização de todos os dados, a criação de outros usuários de seu mesmo tipo ou de cliente, edição e exclusão lógica de quase todos os dados.

O tipo "Client" foi desenvolvido pensando nos clientes, onde seus acessos foram limitados a visualização de somente os animais relacionados com seu usuário, visualizar dados de consultas anteriores e solicitar uma consulta no dia e horário desejado, cabendo ao funcionário por meio de uma tabela de consultas requisitadas aceitar ou modificar a consulta solicitada.

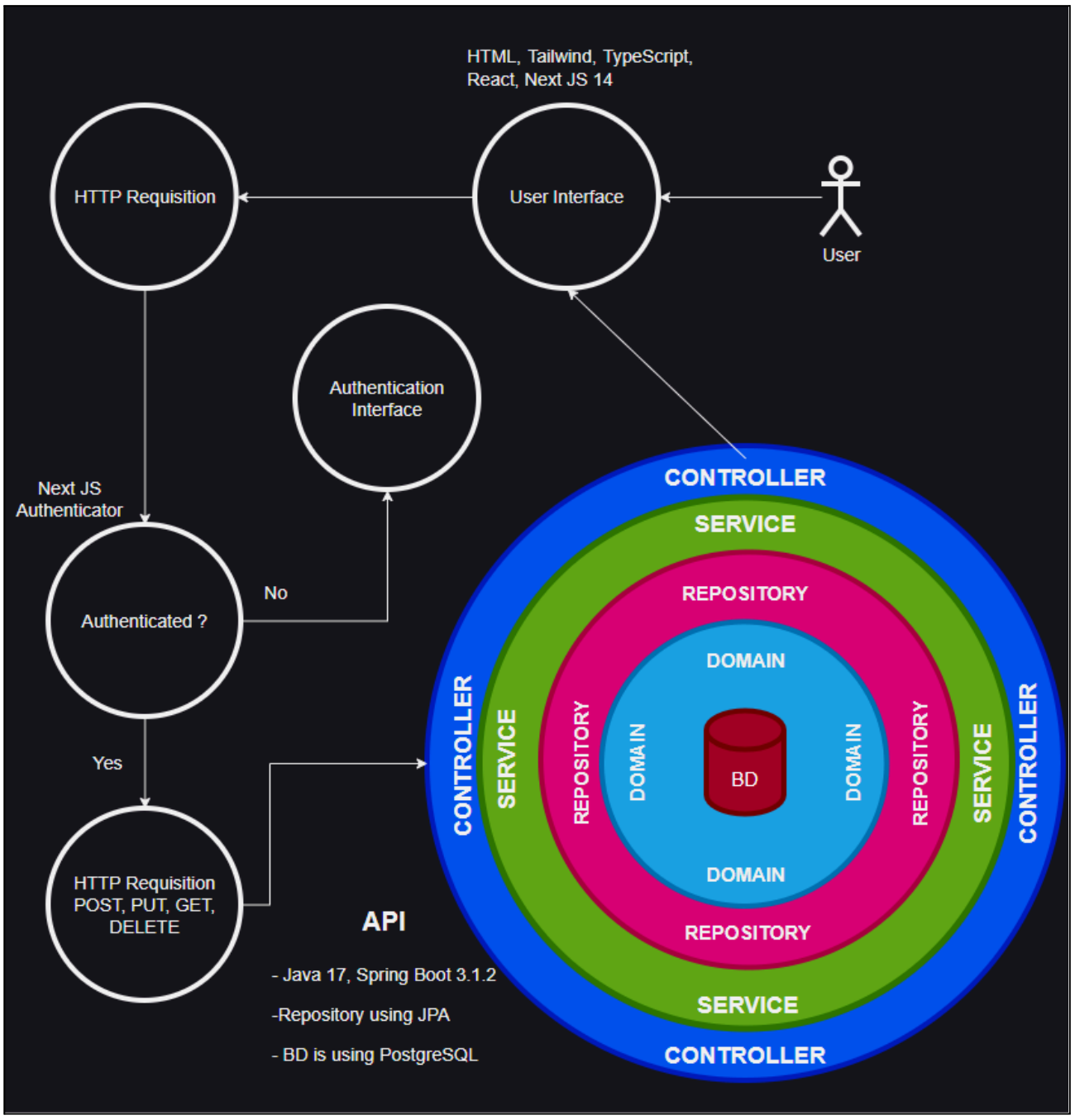


Figura 8: Diagrama Geral do Funcionamento do Sistema

4.2 AGENDAMENTO RÁPIDO DE CONSULTAS

Ao estar autenticado com as credenciais de administrador, o usuário poderá acessar a tela inicial do sistema, que contém breves porém valiosas informações rápidas como próximas consultas, pedidos de novas consultas dos clientes e vacinas, conforme na Figura 9.

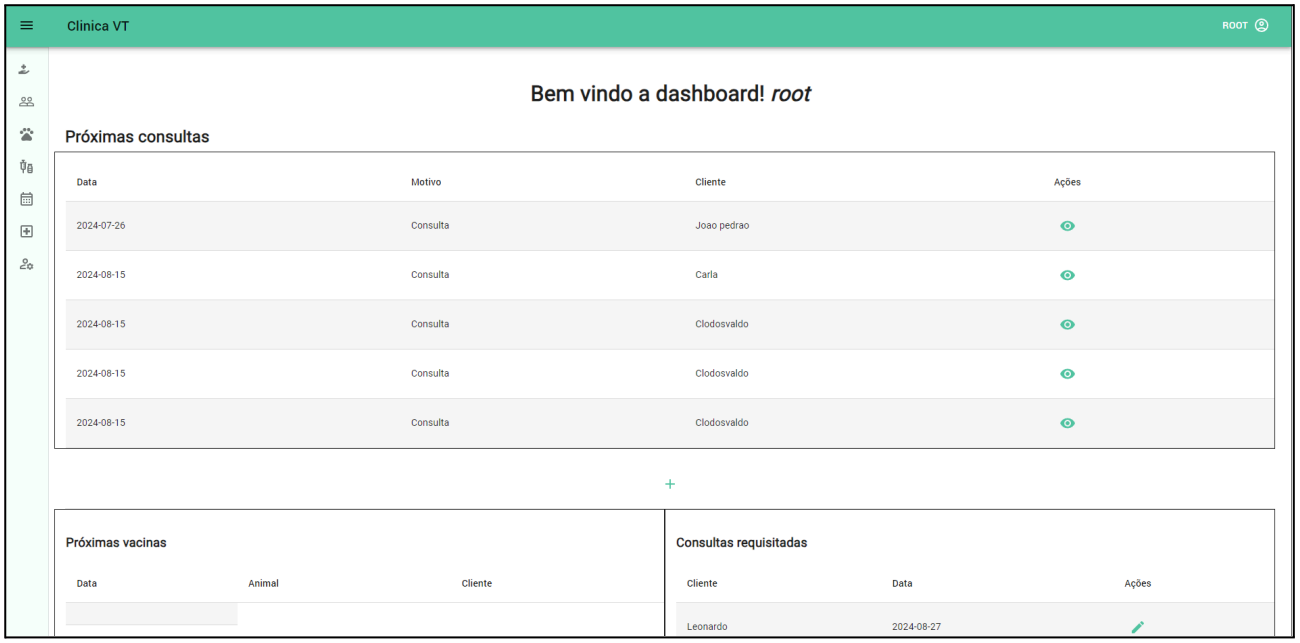


Figura 9: Tela inicial do sistema no modo admin - Onde contém breves informações.

4.3 LINHA DO TEMPO DOS ATENDIMENTOS

Esta funcionalidade traz consigo os históricos de atendimento dos veterinários, podendo ser por filtros ou não, obtendo informações valiosas a fim de produtividade dos veterinários como todas as consultas realizadas, horário, descrição da consulta, término e dia. Como mostrado na Figura 10.

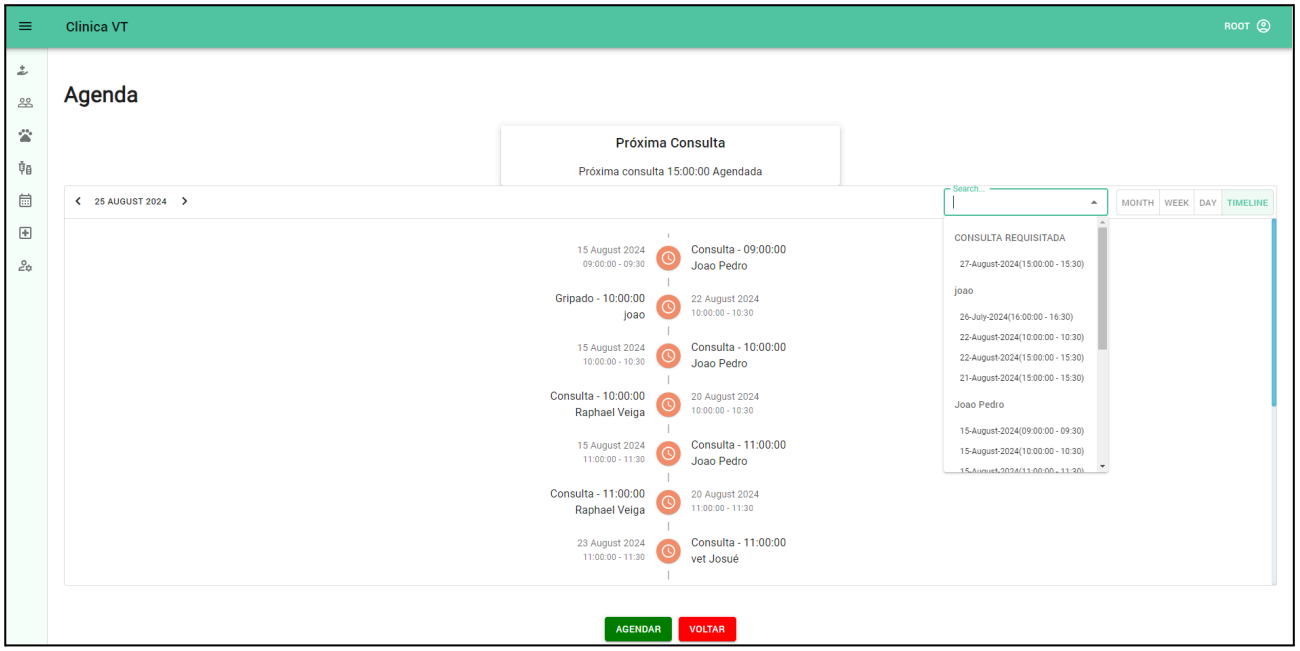


Figura 10: Tela de linha do tempo de atendimento - Pode-se encontrar o histórico de atendimento da clínica completo em forma de linha do tempo.

4.4 AGENDAMENTO RÁPIDO DE CONSULTAS

Com toda a agenda disponível com a funcionalidade de escolher os meses, visualizar todos os dias, consultas marcadas, horários e a descrição, o funcionário poderá encaixar uma nova consulta no melhor horário possível, assim evitando a espera no local, conforme mostrado na Figura 11.

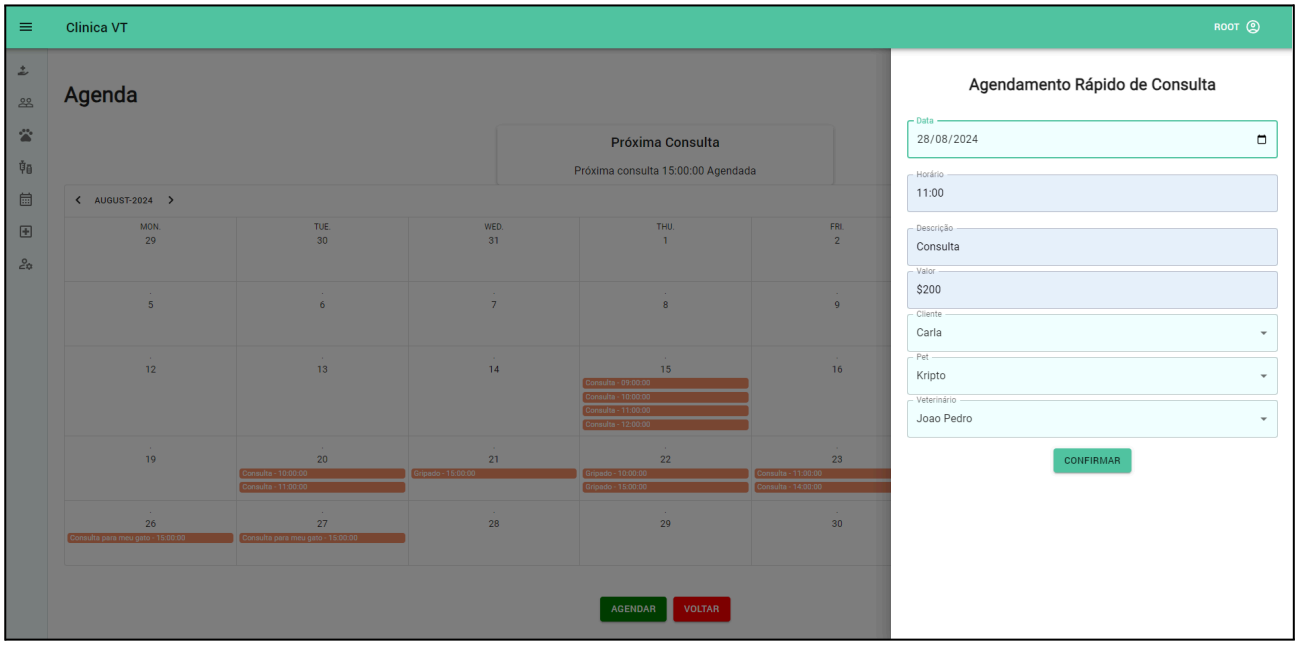


Figura 11: Agendamento Rápido de Consulta - Permite o cadastro rápido das consultas.

4.5 CADASTRO DE VETERINÁRIO

Para haver um cadastro de consulta é necessário antes estar cadastrado um veterinário, e, como o sistema também tem a possibilidade de gerir vários aspectos da clínica o veterinário é um deles, podendo ser adicionado com facilidade, sendo uma das informações mostradas para o cliente também, como mostrado na Figura 12.

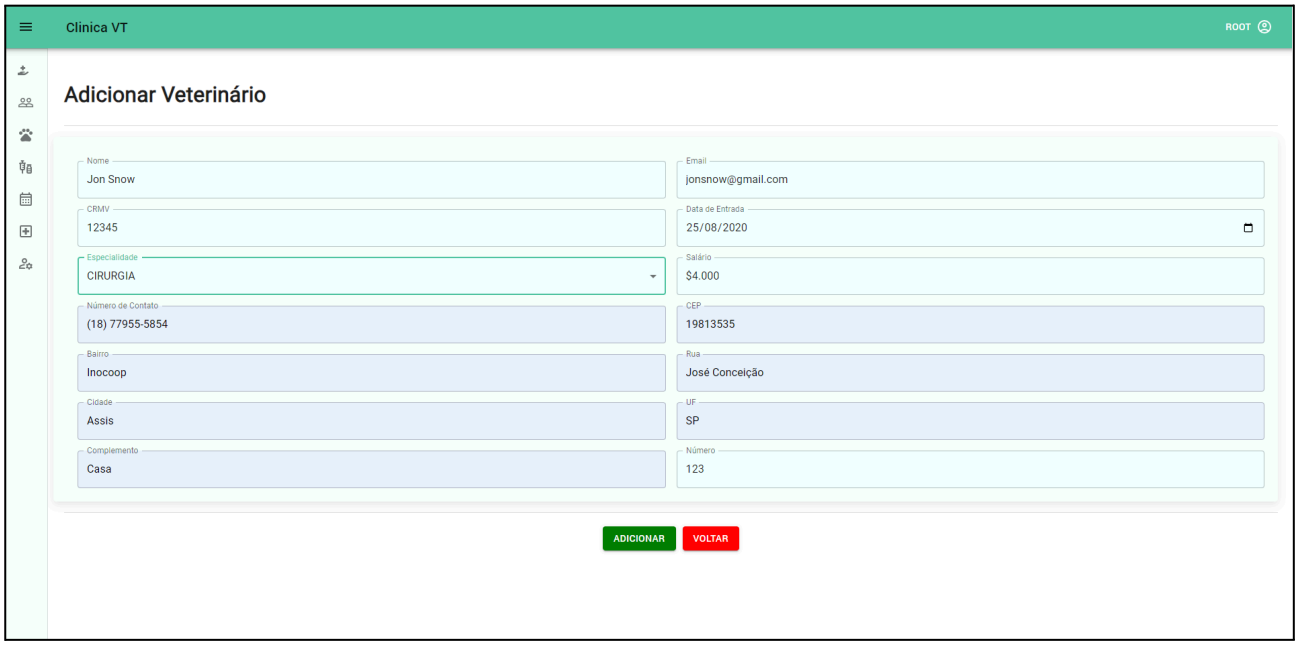


Figura 12: Cadastro de Veterinário - Permite que o administrador adicione veterinários no sistema.

4.6 ACOMPANHAMENTO DA FICHA DO ANIMAL

Por meio do botão de “Editar” representado por um ícone de lápis na tabela, o cliente pode acompanhar todo o histórico de seu animal de estimação obtendo informações relevantes como consultas realizadas, vacinas aplicadas, se está em tratamento entre outros, também permitindo que o mesmo atualize os dados de seu animal de estimação como ilustrado na FIgura 13..

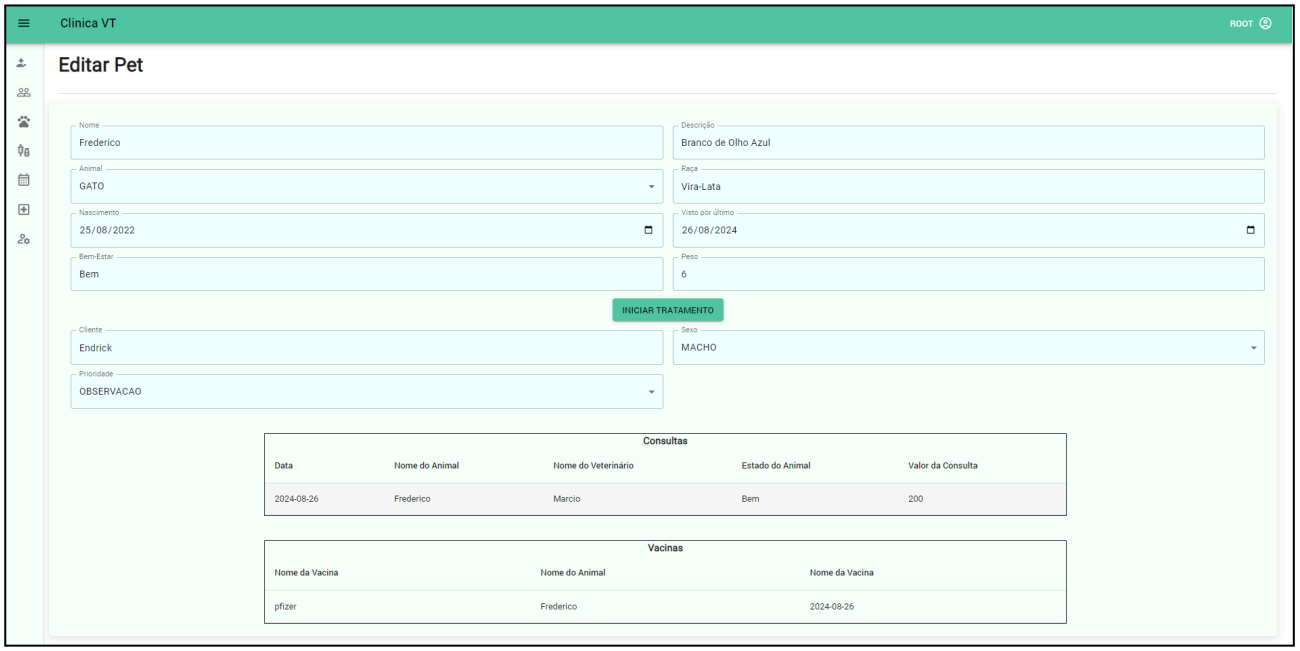


Figura 13: Visualização e edição de dados do animal - Permite que o cliente atualize e visualize os dados de seu animal de estimação.

4.7 REGISTROS DE USUÁRIOS

Como parte fundamental do sistema, cada pessoa que acessar precisa de um usuário seja ele administrador ou cliente, e para isso foi criado uma lógica com duas rotas diferentes para evitar conflitos e possíveis erros na criação dos usuários, sendo uma rota para administrador e outra para cliente, abaixo é mostrado um exemplo de rota para criar um acesso para funcionário da clínica**.**

O sistema de cadastro de funcionários foi implementado com bibliotecas do framework spring, como a spring security permitindo que fossem criados papéis para cada usuário e permissões exclusivos, assim segregando de forma correta o acesso de cada um.

Tendo em vista a segurança do sistema, todas as senhas foram guardadas em formato de hash pela biblioteca bcrypt passworderncoder, evitando assim qualquer problema de vazamento de dados ou outras formas de capturar o json.

```java
public void registerUser(UserDTO registerDto){
        if (this.repository.findByUsername(registerDto.username()) != null) throw new EntityExistsException("Usuário já existente.");
        String encryptedPassword = new BCryptPasswordEncoder().encode(registerDto.password());
        User newUser = new User(registerDto, encryptedPassword);

        if (jobRepository.findAll().stream().noneMatch(job -> job.getJob().equals(registerDto.job()))){
            Job job = jobRepository.save(new Job(registerDto.job()));
            newUser.getJob().add(job);
            this.repository.save(newUser);
        } else {
            Job job = jobRepository.findByJob(registerDto.job());
            newUser.getJob().add(job);
            this.repository.save(newUser);
        }

    }
```

Figura 14: Cadastro de usuários do tipo administrador - Permite que o funcionário da clínica se autentique com segurança.

4.8 AGENDAMENTO DE CONSULTA

Este método foi desenvolvido de forma com que evite ao máximo o engarrafamento de consultas, por meio de verificações certeiras, cada parte dessa função foi pensada de forma única.

As primeiras verificações são as de existência dos dados enviados, logo após uma lista de condicionais para verificar a validade do dia recebido, evitando assim que uma consulta seja marcada em horários que a clínica não funcione como em um domingo às três horas da tarde.

A próxima verificação é a mais importante e mais complexa, onde é feito a lógica de não haver duas consultas marcadas ao mesmo tempo com o mesmo veterinário, ou até mesmo muito próxima uma da outra, por meio dos parâmetros de data e hora o sistema bloqueia qualquer tentativa de cadastro em caso de estarem próximas, retornando uma exceção devidamente tratada.

Como trecho final do método vem os envios de e-mail, para notificar tanto o cliente quanto o veterinário, evitando o caso de um dos dois se esquecerem da consulta.

```java
@Transactional
    public Consult create(@NotNull FormConsultDTO consult) {

        Veterinarian veterinarian =
                veterinarianService.getById(UUID.fromString(consult.
getIdVeterinarian())).orElseThrow(() -> new RuntimeException(
"Veterinário não encontrado"));

        Client client =
                clientService.getById(UUID.fromString(consult.
getIdClient())).orElseThrow(() -> new RuntimeException(
"Cliente não encontrado"));

        Pet pet = petService.getById(UUID.fromString(consult.
getIdPet())).orElseThrow(() -> new RuntimeException(
"Pet não encontrado"));

        LocalDate date = consult.getDate();
        LocalTime hour = consult.getTime();

        DayOfWeek dayOfWeek = date.getDayOfWeek();

        if (dayOfWeek == DayOfWeek.SATURDAY || dayOfWeek ==
DayOfWeek.SUNDAY) {
            throw new RuntimeException(
"Consultas só podem ser agendadas em dias úteis");
        }

        if ((hour.isBefore(LocalTime.of(9, 0)) || hour.isAfter(
LocalTime.of(12, 0))) &&
                (hour.isBefore(LocalTime.of(14, 0)) || hour.isAfter(
LocalTime.of(19, 0)))) {
            throw new RuntimeException(
"Consultas só podem ser agendadas entre 9am-12pm e 14pm-19pm");
        }

```

Figura 15: Parte inicial do cadastro de consulta - Onde há a instanciação de entidades e verificação se existem ou não, também a verificação de dias úteis e horários de comércio.

```
Consult nearbyConsult = repository.findNearbyConsult(date, hour.
getHour());

        if (nearbyConsult != null) {
            LocalTime nextAvailableTime = repository.
findNextAvailableTime(date, hour);
            if (nextAvailableTime != null) {
                consult.setTime(nextAvailableTime);
            } else {
                LocalDate nextAvailableDay = repository.
findNextAvailableDay(date);
                if (nextAvailableDay != null) {
                    consult.setDate(nextAvailableDay);
                } else {
                    throw new RuntimeException(
"Não há datas disponíveis para agendamento.");
                }
            }
        }

        if (veterinarian.getEmail() != null) {
            String body = String.format(
"Olá %s, você tem uma consulta agendada para o dia %s às %s com o cl
iente %s e seu pet, %s. Bom trabalho! :)"
,
                    veterinarian.getName(), consult.getDate(),
consult.getTime(),
                    client.getName(), pet.getName());

            emailService.sendEMail(new EmailRecord(veterinarian.
getEmail(), body, "Consulta Agendada"));
        } else {
            throw new RuntimeException(
"O email do veterinário não foi informado.");
```

Figura 16: Segunda parte do cadastro de consulta - Aqui é feita a verificação se há alguma consulta marcada no horário próximo ao anterior, e também o início das notificações por e-mail.

```
}
        if (client.getEmail() != null) {
            String body = String.format(
"Olá %s, sua consulta foi agendada para o dia %s às %s com o veterinári
o %s. Nos vemos lá ! :)"
,
                    client.getName(), consult.getDate(), consult.
getTime(),
                    veterinarian.getName() == null ? "a ser definido" :
 veterinarian.getName());

            emailService.sendEMail(new EmailRecord(client.getEmail(),
body, "Consulta Agendada"));
        } else {
            throw new RuntimeException("O email não foi informado.");
        }

        pet.setLastSeenDate(consult.getDate());
        ConsultDto consultDto = new ConsultDto(consult, veterinarian,
pet, client);
        var newConsult = new Consult(consultDto);
        return repository.save(newConsult);
    }
```

Figura 17: Parte final do cadastro de consulta - A imagem mostra o final do método, onde é feito o envio do email e o tratamento de sua exceção.

4.9 APLICAÇÕES DE VACINA

Para um controle melhor das vacinas e também a possibilidade de fornecer um retorno ao cliente, esta funcionalidade traz consigo a data e a vacina aplicada no pet, visando manter a carteirinha de vacinação do animal sempre atualizada e em dia, com uma lógica simples . A lógica de vacinação foi feita por meio de listas na entidade de pet, capturando o id da vacina aplicada e adicionando em uma lista presente na classe. Já a data foi desenvolvida de forma similar, também sendo persistida em uma lista. Os dados são capturados de forma correta pois está interligado com a entidade, evitando assim cadastrar de forma errada uma data de vacinação em outro animal. O método é mostrado na Figura 18.

```
public void vaccineApply(UUID idPet, UUID idVaccine){  1 usage  JoaoPrestupa *
    if (petService.getById(idPet).isEmpty() || repository.findById(idVaccine).isEmpty()){
        throw new RuntimeException("Pet or Vaccine not found");
    }
    petService.getById(idPet).get().getVaccines().add(repository.getReferenceById(idVaccine));
    petService.getById(idPet).get().getVaccinesAppliedDate().add(LocalDate.now());
}
```

Figura 18: Lógica de vacinação - Foi criado duas listas, uma de vacinas e outra de data na entidade pet, para que os dados fossem enviados ao front-end com mais facilidade.

5. CONCLUSÕES

Com o crescimento na área da saúde animal juntamente com o mercado cada vez mais competitivo, é necessário modernizar e trazer o que há de melhor sempre que possível. Um estabelecimento seja ele qual for, se não tiver um bom controle organizacional, no contexto deste trabalho uma clínica veterinária, deve perder espaço e clientes, o que ocasiona em lucros menores.

Seja para manutenção e aumento de produtividade de clínicas para animais domésticos ou animais de produção, a solução proposta por este trabalho traz uma solução por meio de tecnologia para aumentar a produtividade no dia a dia e melhorar o atendimento ao cliente final, gerando uma maior conexão empresário-cliente. O resultado deve ser visto de forma gradual, após um bom tempo de uso com feedbacks dos clientes e dos funcionários que o utilizarem.

5.1 TRABALHOS FUTUROS

Apesar do progresso alcançado, há algumas funcionalidades que devem ser implementadas como gerenciamento das finanças da clínica, pensando em um maior controle, melhoria no sistema de cadastros de consultas no lado do cliente, permitindo com que o próprio selecione mais de um *pet* por vez caso tenha, e mais informações sobre o seu animal, adicionar a funcionalidade de manter medicamentos, uma sessão exclusiva para cirurgias e internações, além de melhorias de performance e de designer também são indispensáveis visando manter um padrão de qualidade para o cliente final sempre elevado.

REFERÊNCIAS

AGUIAR, Thiago Ferreira de. **Sistema de Gerenciamento de Hospitais Veterinários**. 2015. 83p. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná. Paraná. Cornélio Procópio, 2015.

ALI, Mudassir, **What is bcrypt password encoder**. 2024. Disponível em <https://medium.com/@mudassirali_79816/what-is-bcrypt-password-encoder-2dc7fe2131b2> Acesso em 26 ago, 2024.

BOSCO, João Pedro Cayres, **Docker Como Ferramenta Para Construção de Ambiente em Nuvem**. 2019. 41p. Trabalho de Conclusão de Curso. Faculdade de Tecnologia de Americana. São Paulo. Americana, 2019.

FREITAS, Frederico José Souto de. **Ensino de administração nos cursos de medicina veterinária e a visão dos profissionais sobre a gestão dos serviços veterinários para pequenos animais diante da expansão do mercado pet.** 2016. 123p. Dissertação de Mestrado. USP. São Paulo, 2016.

Gabriel, Fernandes Lemos. **Clean Architecture: Uma nova abordagem baseada em príncipios**. Disponível em <https://medium.com/@gabrielfernandeslemos/clean-architecture-uma-abordagem-baseada-em-princípios-bf9866da1f9c> Acesso em 26 ago, 2024.

Gomes, Adriano Gebert. **Sistema de agendamento de clínica veterinária: desenvolvimento de uma aplicação Web utilizando framework Yii 2.0**. 2023. 120p.Trabalho de conclusão de Curso. Facultad de Ingeniería, Ciencias Físicas y Matemática Carrera de Ingeniería Informática. Universidad Central del Ecuador. Quito, 2023.

JAFFAR, Nurnajlaa Bahirah. **The Development of Veterinary Clinic Management System Using Structured Approach.** 2021. 37pt. Applied Information Technology And Computer Science. Universiti Tun Hussein Onn. Malásia. Batu Pahat Johor, 2021.

JUNIOR, Edemilton Alcides Galindo. **Desenvolvimento de API REST com Spring Boot**. Disponível em

<https://www.publicacoes.unirios.edu.br/index.php/revistarios/article/view/102> Acesso em 24 nov, 2023.

MARTIN, Robert C. **Clean Architecture A Craftsman 's Guide to Software Structure and Design**. 1º. 12 de setembro de 2017.

MUI MATERIAL, **Getting Started Material UI. Dísponivel em**: https://v4.mui.com/pt/getting-started/usage/

Muniz, Antonio et al. **Jornada API na prática: Unindo conceitos e as experiências do Brasil para acelerar negócios com a tecnologia**. Brasport. 2023.

NASCIMENTO, Rosalina Moreira, Angélica Santos. **A qualidade do atendimento como /fator de crescimento empresarial**. 2021. 16p. Trabalho de Conclusão de Curso. Universidade Evangélica de Goiás. Goiás. Goiânia, 2021.

ORACLE, **O QUE É JSON**. ORACLE. Disponível em: <https://www.oracle.com/br/database/what-is-json/#:~:text=JavaScript%20Object%20Notation%20(JSON)%20é,vez%20que%20não%20requer%20desserialização>. Acesso em 27/08/2024.

ROJAS, Claudio Ivor Torres. **Sistema de Gestión para Clínica Veterinaria**. 2012. 141p. Trabalho de Conclusão de curso. Pontificia Universidad Católica de Valparaíso. Valparaíso, 2012.

SOARES, Italo Ribeiro. **Conformidade da biblioteca material UI para react com a web accessibility content guidelines**. 2022. 44p. Trabalho de Conclusão de Curso. Universidade Federal de Pernambuco. Pernambuco, 2022,

SWAGGER, **Streamline your workflow with unparalleled API specification support**. Disponível em <https://swagger.io>. Acesso em 28 nov, 2024