

Ração Simplex

Automação para formulação de dietas para cães e gatos

Chrystian Portella da Silva¹, Ana Paula Canal¹

¹Sistemas de Informação – Universidade Franciscana (UFN)
R. dos Andradas, 1614 – Centro, Santa Maria – RS, 97010-030

{c.portella, apc}@ufn.edu.br

Abstract. *This paper aims to present the development of a web software that helps in dog or cat food planning following the formulations proposed by the National Research Council. Its coding is done using .NET and Blazor technologies, the PostgreSQL database for data warehousing and best practices from the agile project management framework Scrum with collaboration of DevOps development culture. As for the results the automation of the process was achieved, which resulted in a gain of agility in the construct of diets for dogs and cats.*

Resumo. *Este trabalho tem como objetivo apresentar o desenvolvimento de um software web que auxilie no planejamento alimentar de cães ou gatos seguindo as formulações propostas pelo National Research Council. Sua codificação é realizada por meio das tecnologias .NET e Blazor, o banco de dados PostgreSQL para armazenamento de dados e como metodologia de desenvolvimento são utilizadas boas práticas do framework ágil Scrum, com colaboração da cultura de desenvolvimento DevOps. Como resultados atingiu-se a automação do processo o que mostrou-se um ganho de agilidade na elaboração de dietas de cães e gatos.*

1. Introdução

A indústria de animais de estimação está em grande expansão [Salzman 2000] um dos segmentos que representa grande faturamento é a linha de alimentação (*pet food*) e em decorrência de tutores de *pets* cada vez mais estarem investindo no bem-estar do animal, sua saúde e longevidade a procura por dietas que promovam a longevidade dos animais tem se tornado cada vez mais frequente.

Percebendo-se a importância da formulação de uma dieta adequada para a compra da ração apropriada para animais de estimação por parte dos tutores e a necessidade de utilização de diferentes equações matemáticas, além de variados padrões de referências que entidades tais como a *National Research Council* (NRC) publicam, foi realizado o desenvolvimento de um software web para automação deste processo para que Zootecnistas façam a elaboração de dietas para cães ou gatos com maior agilidade.

O desenvolvimento do projeto é realizado utilizando-se de tecnologias de relevante suporte de comunidades de código aberto - .NET, Blazor e PostgreSQL - e uma metodologia de desenvolvimento ágil que contempla um ambiente de constante mudança, o Scrum, em conjunto do DevOps, uma cultura de entrega contínua para cultivar o compartilhamento de ideias entre os *stakeholders*¹ do projeto aprimorando seu desenvolvimento.

¹partes interessadas no projeto

1.1. Justificativa

Entidades têm avançando nos últimos anos, com pesquisas que promovem maior entendimento sobre as necessidades nutricionais dos animais, e por efeito, uma grande evolução na alimentação ocorreu. Sabe-se hoje que os alimentos tem por objetivo não apenas nutrir como também promover saúde, bem estar e longevidade do animal de estimação [Ogoshi et al. 2015].

Pelo empenho e melhora ou manutenção da boa saúde dos animais de estimação, o aumento pela procura de dietas com rações adequadas por parte dos tutores tem sido cada vez mais notável.

Com base na observação do processo de criação de uma dieta nutricional realizado por uma Zootecnista constatou-se a perda de produtividade em decorrência da quantidade de parâmetros, equações e tabelas necessários para a realização do trabalho, surgiu assim a ideia de automatizar essa tarefa utilizando tecnologias que possam facilitar e agilizar esse processo. Essa automação tem como resultado o aumento de produtividade de zootecnistas qualificados que almejam criar dietas nutricionais para os tutores de cães ou gatos que buscam pela melhora e/ou manutenção da boa saúde em seus animais de estimação.

1.2. Objetivo geral

Desenvolver um software web que automatize o planejamento de dietas para cães ou gatos utilizando as tecnologias de codificação de sistemas .NET e Blazor, com o Sistema de Gerenciamento de Banco de Dados PostgreSQL.

1.3. Objetivos específicos

- Realizar estudos acerca de formulação de dietas de cães e gatos;
- Elaborar escopo do projeto utilizando boas práticas do *framework* ágil Scrum para gerenciamento do projeto em conjunto com a cultura de desenvolvimento DevOps;
- Fazer uso do *framework* Blazor em conjunto do .NET para codificação da aplicação;
- Manipular e armazenar os dados produzidos pela aplicação utilizando o sistema de gerenciamento de banco de dados PostgreSQL.

2. Referencial teórico

Nesta seção serão abordados os tópicos de pesquisa e as tecnologias que se fazem necessárias para o desenvolvimento deste projeto.

2.1. Nutrição Animal

É adequado definir a nutrição animal como a ciência que estuda e integra o conjunto de processo que se realizam nos alimentos, manuseando-os para todas as suas funções fisiológicas vitais tendo como principal objetivo suprir todos os nutrientes, que são elementais em todas as fases de vida animal: crescimento, manutenção, atividade física, gestação, lactação, trabalho e outros [Couto and Real 2019].

Tais nutrientes tem como classificação e agrupamento de acordo com acordo com sua natureza química e biológica (água, carboidratos, proteínas, lipídios, minerais e vitaminas).

Um animal bem nutrido depende principalmente de sua alimentação regular e equilibrada. O fornecimento de nutrientes essenciais para pleno funcionamento das células e tecidos do corpo deve ter uma correta quantidade e boa variabilidade [Couto and Real 2019].

Em seu trabalho [Couto and Real 2019, p.136] definem “exigências nutricionais como: quantidade de energia e nutrientes que o animal deve ingerir para satisfazer suas necessidades diárias”. Percebe-se que a cada dia as exigências nutricionais estão mais bem determinadas graças aos estudos como o do NRC.

É importante também salientar que existem fatores que afetam exigências nutricionais que são anatomia e fisiologia do trato digestório, genética, características físicas e organolépticas, qualidade das matérias-primas, nível energético, manejo da alimentação, higiene e profilaxias, sanidade, ambiente, aspecto ecológico [Couto and Real 2019].

Para realizar as formulações de dietas o primeiro passo é a coleta de informações básicas do *pet* tais como nome, idade, sexo, peso, porte, informações de comorbidades e se o animal possui preferências de paladar, tais como, por exemplo, tamanho dos grãos.

Após essa coleta de parâmetros pode-se dividir o processo manual em duas grandes etapas: a primeira é o cálculo da exigência nutricional energética e a segunda é a definição das exigências de nutriente essenciais.

Para o cálculo da exigência nutricional energética, define-se que taxa metabólica é a quantidade de energia gasta por um animal em um determinado período de tempo. É importante para entender o conceito de metabolismo basal (MB), pois este é o conjunto de processos onde a energia transforma-se em calor e é liberada do animal em jejum, repouso e perante a ambientes de termo-neutros. Compreendendo-se tem-se que [Couto and Real 2019]:

$$MB_{kcal/dia} = 70_{kcal} * PM$$
$$PM_{kg} = PC_{kg}^P$$

Em que, têm-se:

- MB como metabolismo basal sendo definido como conjunto de processos metabólicos onde a energia é transformada em calor e liberada do animal em estado de jejum, repouso e perante neutralidade térmica;
- PM como peso metabólico;
- PC como o peso corporal;
- E a potência P é uma constante de referência, para cães é utilizado o valor 0,75 podendo ter variações conforme proporção entre a superfície do corpo, taxa metabólica e a massa corporal se alteram no decorrer de seu crescimento ou em diferentes condições fisiológicas ou corporais.

As exigências nutricionais para cães e gatos são apresentadas pela NRC e para elaboração de sua dieta um passo importante é definir a demanda energética diária do animal que depende de sua fase de crescimento ou adulto, estado fisiológico, grau de atividade física, ambiente e condição corporal. Para referência energéticas para cães tem-se a Tabela 1 [Couto and Real 2019]:

Tabela 1. Comparação da energia de manutenção para portes de cães [Couto and Real 2019, p.140]

	Pequeno	Médio	Grande	Gigante
PV (KG)	8	40	65	100
PV (KG) = PV ^{0,75}	5	16	24	32
EM Manutenção	700	2240	3360	4480
Kcal EM/PV	88	56	52	45

Cães em crescimento e adulto tem diferentes exigências diárias de energia metabolizável (E.M.), os *pets* em crescimento apresentam maior exigência nos primeiros meses de vida e vão sendo reduzidos conforme o crescimento até a fase adulta. Uma importante equação é a predição de energia metabolizável para cães desmamados [Couto and Real 2019].

$$E.M. = (130 * PVA^{0,75}) * 3,2 * [e^{(-0,87 * F)}] - 0,1$$

Esta equação exige o peso vivo atual (PVA), obtido pelas pesagens semanais ou mensais e o peso esperado para maturidade (PVE) além do fator F que é a relação entre PVA e PVE (PVA/PVE).

Para cães adultos é exigido apenas o peso vivo atual para determinação do peso metabólico (PM) que multiplicado por um fator F pré-estabelecido com base na atividade física do *pet* e pode ser guiado pelas condições [Couto and Real 2019]:

- Cães com pouco estímulo e exercícios físicos: 95kcal/PM;
- Cães com maior atividade e criados em extenso espaço físico: 130-140 kcal/PM;
- Cães com médio e grande porte: 180-200 kcal/PM.

Têm-se então a equação:

$$E.M.kcal/dia = F * PVA^{0,75}$$

Na segunda etapa do processo o profissional pode seguir as tabelas recomendadas pela NRC considerando Unidade/Dia para 1000 Kcal conforme fase de vida. Estas recomendações são apresentadas nas Tabelas 1, 2, 3, 4 e 5 que estão no Anexo A.

A partir dos resultados das equações e recomendações das tabelas o profissional formulador da dieta pode recomendar uma ração que atenda as exigências nutricionais da dieta.

No que concerne aos gatos há variações nutricionais, porém o processo é essencialmente o mesmo: coleta de parâmetros, cálculos das energias e consultas nas recomendações considerando Unidade/Dia para 1000 Kcal.

Tem-se a equação de energia metabolizável após o desmame [Couto and Real 2019]:

$$E.M. = (100 * PVA^{0,67}) * 6,732 * [e^{(-0,189 * p)}] - 0,66$$

Também existe a equação de energia metabolizável para manutenção, porém com uma ressalva, os expoentes propostos para determinação do peso metabólico seguem um padrão para os expoentes do peso vivo atual conforme o escore corporal proposto por Laflamme (1997) e ilustrado na Tabela 14 do Anexo D. As equações são:

- Gatos adultos magros: $E.M.kcal/dia = 100_{kcal} * PVA^{0,67}$
- Gatos adultos obesos: $E.M.kcal/dia = 130_{kcal} * PVA^{0,40}$
- Gatos adultos exóticos²: $E.M.kcal/dia = 55 \text{ a } 260 \text{ (kcal)} * PVA^{0,75}$

As recomendações dos nutrientes para gatos podem ser visualizadas nas Tabelas 6, 7, 8, 9 e 10 que estão no Anexo A.

Por fim é importante considerar que não apenas as concentrações de aminoácidos devem ser satisfeitas diariamente, mas também o nutricionista deve considerar as proporções adequadas entre os aminoácidos, que podem ser estabelecidas pela teoria da proteína real conforme ilustra o gráfico da Tabela 11 [Couto and Real 2019] no Anexo A.

2.2. Tecnologias

Esta subseção irá fazer o referencial teórico das tecnologias a serem usadas para o desenvolvimento do trabalho .NET em conjunto da arquitetura MVC³, Blazor e o SGBD PostgreSQL.

2.2.1. .NET Blazor

O .NET é uma popular plataforma de desenvolvimento de software e contém diversas características tais como estrutura multiplataforma onde pode desenvolver e executar web aplicativos no Windows, Linux e macOS; código aberto com envolvimento da comunidade; Estrutura integrada de injeção de dependência (DI) e também servidor web integrado chamado Kestrel que pode-se integrar com servidores web populares tais como Nginx ou Apache.

Blazor é uma estrutura para a construção de interfaces de usuário da web interativas que operam no navegador do cliente e uma ferramenta integrante do .NET. Com ele é possível criar componentes de interface de usuário usando C#, HTML e CSS [Joshi 2019]. Ele se divide em duas partes o Blazor *Server Side* (Blazor Server) e o Blazor *Client Side* (Blazor WebAssembly).

O *Server Side* como o nome sugere é tudo o que é executado pelo servidor da aplicação .NET aqui está inserido o padrão .NET MVC com os controladores e modelos. Já o *Client Side* é tudo aquilo que é executado e mostrado (visualizações) no navegador do usuário. O *Client Side* manipula o DOM (*Document Object Model* - Modelo de Objeto de Documentos), ou seja, as interações que o usuário faz com o navegador. A comunicação se dá por intermédio da tecnologia SignalR que realiza uma comunicação bidirecional seguindo os padrões do protocolo HTTP exibindo em tempo de execução as alterações solicitadas ao servidor pelo cliente [Joshi 2019].

A Figura 1 ilustra como o Blazor realiza a comunicação *Server Side* com o *Client Side* de maneira bidirecional.

²Persa, Siames, Snowshoe

³Model-View-Controller

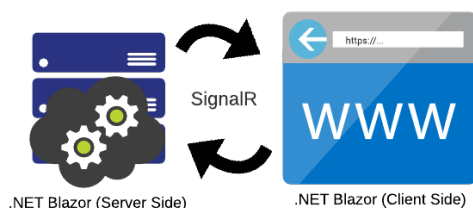


Figura 1. Comunicação Blazor

2.2.2. PostgreSQL

O PostgreSQL é descrito como um sistema de gerenciamento de banco de dados objeto-relacional (SSGBDOR)⁴ construído usando como base POSTGRES Versão 4.2 pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley.

Inclui grande parte dos tipos de dados do ISO SQL:1999, como INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP e suporta o armazenamento de objetos binários, dentre figuras, sons ou vídeos e também compatível com ACID⁵ [Politowski and Maran 2014]. Além de todos estes recursos o PostgreSQL também é um SGBD *open-source*, isto é, livre para uso e este foi o motivo decisivo para uso neste trabalho.

2.3. Scrum

O Scrum pode ser descrito como um *framework* simples para gerenciar projetos complexos. Seus maiores benefícios de acordo [Sabbagh 2014, p.4] são as entregas frequentes de retorno ao investimento dos clientes, redução dos riscos do projeto, maior qualidade no produto gerado, visibilidade do progresso do projeto, redução do desperdício, aumento de produtividade.

O Scrum é o *framework* adequado quando os usuários não sabem exatamente o que desejam até ter um protótipo de uso para o projeto e quando os requisitos não são completamente entendidos e/ou mudam com muita frequência [Sabbagh 2014].

Em vista de utilizar boas práticas do Scrum é importante destacar que o Scrum tem três papéis, seis eventos e três artefatos. Neste trabalho o papel do *Product Owner* é essencial para o desenvolvimento do mesmo, já o papel do Time de Desenvolvimento e do *Scrum Master* é acumulado por uma única pessoa. Em relação aos seus eventos apenas a *Sprint*, a *Sprint Planning*, a *Sprint Review* e a *Sprint Retrospective*, em princípio, serão necessários para desenvolvimento do trabalho. E referente aos artefatos todos os três (*Product Backlog*, *Sprint Backlog* e o Incremento do Produto) poderão ser utilizados.

Estabelecendo-se que uma Visão de Produto é um objetivo a ser alcançado temos que o *Product Owner* é aquele que define, comunica e mantém a Visão do Produto o

⁴Um banco de dados objeto-relacional (ORD), ou sistema de gerenciamento de banco de dados objeto-relacional (ORDBMS ou SGBDOR), e um sistema de gerenciamento de banco de dados relacional que permite aos desenvolvedores integrar ao banco de dados seus próprios tipos de dado e métodos

⁵Atomicidade, Consistência, Isolamento e Durabilidade

mais constante possível no decorrer do projeto. Sendo único no time, trabalha com os clientes do projeto e com quaisquer outras partes interessadas que possam contribuir para o entendimento e definição da Visão do Produto [Sabbagh 2014].

O Time de Desenvolvimento é um time multidisciplinar que realiza o trabalho de desenvolvimento do produto além de fazer a própria gestão e gerência da progressão das metas negociadas com o *Product Owner*. O *Scrum Master* por sua vez é responsável por garantir que os impedimentos que o Time de Desenvolvimento encontre durante seu trabalho sejam resolvidos e também ser um facilitador nas reuniões Scrum e iterações entre o *Product Owner* e o Time de Desenvolvimento.

Nos artefatos têm-se que o primeiro que deve ser decidido é o *Product Backlog*. Uma lista ordenada, incompleta e dinâmica de itens que representam o que é produzido ao longo do projeto. Os itens da lista são organizados por ordem de importância sendo o mais alto, o mais importante e por essa razão o mais detalhado até o menos importante e mais abstrato [Sabbagh 2014]. É comum o *Product Backlog* ser organizado em um quadro em que há pelo menos três divisões (A fazer, Fazendo, Feito), para gestão.

Outro artefato de grande importância para o desenvolvimento deste trabalho, porém não é parte integrante do *framework* Scrum é a *User Story*. Muito utilizada em times ágeis para representar os itens do *Product Backlog*, ela é a descrição sob o ponto de vista dos usuários do produto que tratam de necessidades ou objetivos de negócios de uma forma simples e leve [Sabbagh 2014].

Sobre os eventos existem as *Sprints*, elas são como um ciclo onde ocorre no início da *Sprint* o planejamento (*Sprint Planning*), depois o desenvolvimento e no final da *Sprint* a validação e adaptação do que foi desenvolvido (*Sprint Review* e *Sprint Retrospective*). Todas as *Sprints* e seus eventos devem ter uma duração máxima ou fixa definida chamada de *timebox*. Essas *timeboxes* limitam o tempo para que um objetivo seja alcançado criando um ritmo e regularidade de entregas.

A *Sprint Planning* é o planejamento do ciclo (*Sprint*) durante a *planning* demonstra-se os itens do topo do *Product Backlog*, alguns desses itens são selecionados agrega-se um plano de ação para desenvolvê-los. Ao término da *Sprint Planning* é esperado o *Sprint Backlog* que é o conjunto de itens selecionados e seu respectivo plano de ação.

A *Sprint Review* ocorre ao término da *Sprint*, nesta etapa o *Product Owner* se reúne com os *stakeholders* para avaliar *feedbacks* do incremento entregue do produto. Nessa etapa geralmente várias sugestões e alterações surgem dando a oportunidade para planejar os próximos passos.

Por fim, referente aos eventos, existe a *Sprint Retrospective* onde se faz uma inspeção do time e dos processos neste momento ocorre uma auto-avaliação do time onde são destacados pontos a melhorar, planos de ação e aspectos a serem mantidos.

O ideal é que os *Sprints* tenham uma duração fixa e todas as *Sprints* tenham a mesma duração, normalmente elas tem duração de 2 até 4 semanas, porém variam conforme a necessidade do projeto.

A dinâmica da *Sprint* inicia-se a partir do *Product Backlog* priorizado onde serão selecionadas as funcionalidades que serão desenvolvidas na *Sprint*. Após o término da

Sprint é esperado que o incremento do produto seja entregue, no caso de um sistema, como este trabalho, é uma parte funcional do sistema. Os demais itens a serem desenvolvidos seguem a ordem de importância definida pelo *Product Owner*. Durante as entregas o *Product Owner* pode verificar mudanças e essas mudanças serem inseridas no *Product Backlog* e também classificadas com sua devida prioridade. O processo é repetido até que o produto final esteja pronto contemplando todas as mudanças solicitadas [Sabbagh 2014]. A Figura 2 ilustra o funcionamento de um ciclo completo do Scrum.

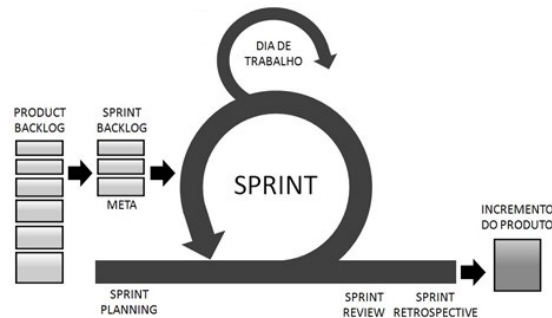


Figura 2. O ciclo do Scrum. Adaptado de [Sabbagh 2014, p.42]

2.4. DevOps

Seguindo a gestão de desenvolvimento ágil em que há iterações curtas e entregas frequentes outras preocupações surgem no processo: suporte, monitoramento, segurança, disponibilidade, desempenho, usabilidade, dentre outras. Com isso o time precisa estar preparado para reagir proativa e rapidamente, por isso algumas empresas possuem divisões de responsabilidades para o time de desenvolvimento e o time de operações. O time de desenvolvimento introduz as mudanças e o time de operações a estabilidade. Porém mudanças tendem a trazer instabilidade criando assim um conflito de interesses aos times e para mitigar este problema criam-se processos que definem o modo de trabalho e a responsabilidade de cada equipe [Sato 2014].

Em um ciclo normal [Sato 2014] elucida que a equipe de desenvolvimento faz um pacote do software que precisa ir para produção, escreve sua documentação para configurar o sistema e/ou instalar o sistema em um ambiente de produção e repassa a responsabilidade para o time de operações. Essa passagem de bastão – também conhecida em inglês pelo termo *hand off* – cria um gargalo no processo de levar código do desenvolvimento e teste para produção. Também é comum chamar esse processo de *deploy* em ambiente de produção. À medida que com o tempo passa o processo fica mais burocrático devido o aumento da complexidade do sistema, tendo por consequência a redução na frequência de *deploys* acumulando-os e aumentando os riscos do projeto como um todo, criando um ciclo vicioso diminuindo a capacidade de respostas à rápidas mudanças no produto o que impacta nas etapas anteriores do processo de desenvolvimento. Contempla-se a necessidade da separação entre os times de desenvolvimento e de operações, o *hand-off* de código existente entre eles e a cerimônia envolvida no processo de *deploy* acabam criando o problema conhecido como Última Milha [Duncan 2009].

Essa Última Milha é a fase final do processo de desenvolvimento quando o mesmo atende todos os requisitos do ciclo e está pronto para ser implantado (realizar o *deploy*)

em produção. Esta fase envolve atividades de verificação de validação e estabilidade, dentre elas pode-se citar testes de unidade, integração, sistema, desempenho, segurança, usabilidade, homologação com usuários (também conhecido como *User Acceptance Tests* ou *UAT*), ensaios de deploy, migração de dados e outros. [Sato 2014].

Com o objetivo de atenuar o ciclo vicioso e manter um nível de qualidade ao processo de Última Milha introduz-se o conceito de "Entrega Contínua"[Humble and Farley 2014] ou também *continuous delivery (CD)* que transforma o processo de *deploy* em algo regular.

Assim, inspirando-se em métodos ágeis, um novo movimento surge para trazer para as equipes de operações e desenvolvimento a mesma linha de raciocínio o movimento DevOps, permite aumentar o fluxo de trabalho completado – maior periodicidade de *deploys* – e manter a estabilidade e robustez do ambiente de produção [Sato 2014].

Como resultado as práticas de DevOps ajudam a quebrar o ciclo vicioso com a automação de processos como ilustrado na Figura 3.

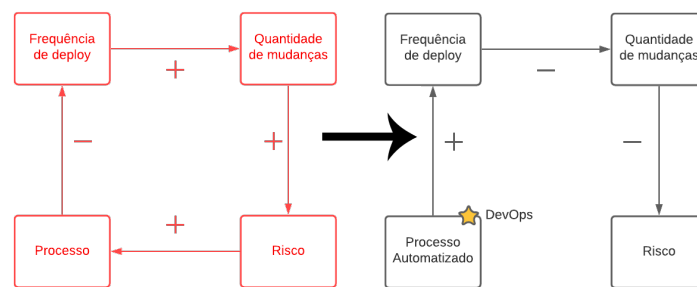


Figura 3. Ciclo vicioso e novo ciclo com DevOps. Adaptado de [Sato 2014]

Têm-se então que o movimento DevOps é a união de pessoas, processos e tecnologias para entregar continuamente valor ao cliente em decorrência da cadeia de ferramentas (*toolchain*) formada por planejamento (*plan*), compilação (*create*), testes (*verify*), empacotamento (*package*), implementação (*deploy*), configuração (*configure*) e monitoramento (*monitor*). A Figura 4 ilustra o conceito de desenvolvimento e operações juntos na cadeia de ferramentas.

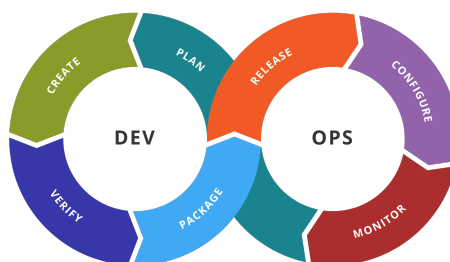


Figura 4. Ilustração exibindo etapas em uma cadeia de ferramentas DevOps [Wikimedia 2016]

3. Trabalhos Relacionados

Em Modelo computacional para formulação de rações de mínimo custo para pequenos ruminantes utilizando programação linear a proposta de [Santos et al. 2006, p.1-2] “tem por finalidade apresentar a elaboração de uma ferramenta a ser utilizada na área de nutrição de ruminantes, para otimizar o uso de alimentos procurando obter melhor eficiência animal e formular rações de mínimo custo para pequenos ruminantes”. Em sua modelagem computacional utilizando-se da programação linear e o método Simplex aplicam uma abordagem diferente de seus correlatos empregando uma limitação de fibra ao modelo para realização dos cálculos de rações, para [Santos et al. 2006, p.2-3] “esta pode ser considerada uma carência que deve ser atendida em curto prazo, o que veio a ser uma grande motivação para a realização deste trabalho”. [Santos et al. 2006] em seu trabalho representou exigências nutricionais dos animais, uma vez que a ração é elaborada pra suprir tais exigências. A tabela 12 de restrições de [Santos et al. 2006] pode ser visualizada no Anexo B.

Já *Optimization for Animal Diet Formulation: Programming Technique* [Saxena et al. 2016] revela que durante várias décadas diferentes formas de programação matemática foram realizadas para resolver o propósito de combinação dos nutrientes e para preencher os nutrientes dos animais da melhor maneira. O trabalho reporta dois objetivos ração animal de menor custo e vida útil máxima da ração animal. Para atingir seus objetivos utilizou do método simplex para investigar, analisar e determinar a forma mais eficiente de encontrar seus resultados esperados e como tecnologia a linguagem de programação Java. Por fim os resultados são calculados pela técnica de programação proposta para 8 modelos. Quatro modelos de minimização de custos são resolvidos para gado leiteiro de diferentes classes de peso e quatro modelos foram resolvidos para minimizar o conteúdo de água da mistura de ração para maximizar a vida útil para diferentes classes de peso [Saxena et al. 2016]. Tais resultados para minimização de custos e minimização do conteúdo de água são apresentados na tabela 13 no Anexo B.

Como considerações sobre os trabalhos relacionados temos que o projeto SANTOS et al. (2006) juntamente com o de Saxena et al. (2016) se correlacionam com este projeto ao serem importantes fontes de conhecimento sobre a aplicação da nutrição animal juntamente com sua importância econômica, além da validação de ganho de produtividade através da construção de software para profissionais que necessitam de automação em seus processos. O diferencial do presente trabalho se mostra na validação do processo para animais monogástricos⁶, especificamente para cães e gatos transcendendo ruminantes.

4. Escopo

Em conjunto da metodologia de gestão de projetos Scrum para geração do *Product Backlog* que foi feita junto do Product Owner, uma Zootecnista especialista em nutrição animal, foram utilizadas as *User Stories*. As *User Stories* podem ter a seguinte estrutura para facilitar o seu entendimento. Segundo sugerido por Cohn (2004)

“Eu como um(a) (papel) quero (função) para que (agregar valor ao negócio)” [Cohn 2004, p.81, tradução nossa]

⁶animais não ruminantes que apresentam um estômago simples [Nielsen 2002]

Este formato é aderido e após uma reunião com a *Product Owner* foi gerado o *Product Backlog* com as *User Stories* a seguir já ordenadas por prioridade sendo a *User Story* mais alta a mais prioritária e a última menos prioritária.

1. Eu como uma zootecnista quero poder digitar os parâmetros do animal (nome, idade, sexo, peso, porte, informações de comorbidades e se o animal possui exigências -preferências de paladar-, tais como, por exemplo, tamanho dos grãos) para que eu consiga realizar os cálculos energéticos.
2. Eu como uma zootecnista quero poder clicar em um botão que realize os cálculos das necessidades energéticas automaticamente para que eu consiga visualizar os resultados.
3. Eu como uma zootecnista quero poder imprimir os resultados para que eu consiga entregar e orientar o tutor quais rações ele poderá comprar para seu animal de estimação.

Com estas três *User Stories* é acordado que cada *Sprint* deverá ter duas semanas. Ainda conforme as orientações do *framework* Scrum as *Sprints* de duas semanas é recomendado uma *timebox* de aproximadamente quatro horas para a *Sprint Planning*, duas horas para as *Sprint Review* e uma hora e meia para a *Sprint Retrospective*.

4.1. Diagrama de classes

O diagrama de classes não é sugerido pelo Scrum ou pelo DevOps, porém em vista de ter uma visão ampla do negócio foi feita uma representação visual utilizando-o para guiar o desenvolvimento do software.

Ele é um artefato da UML (*Unified Modeling Language*) que representa a estrutura e relações das classes que servem de modelo para os objetos que serão utilizados no sistema [Booch et al. 2006]. O diagrama de classe também ajuda a prever e modelar possíveis tarefas que surgem ao longo do processo de desenvolvimento. A Figura 5 ilustra o diagrama de classes do projeto apresentando as entidades e parte do escopo do projeto.

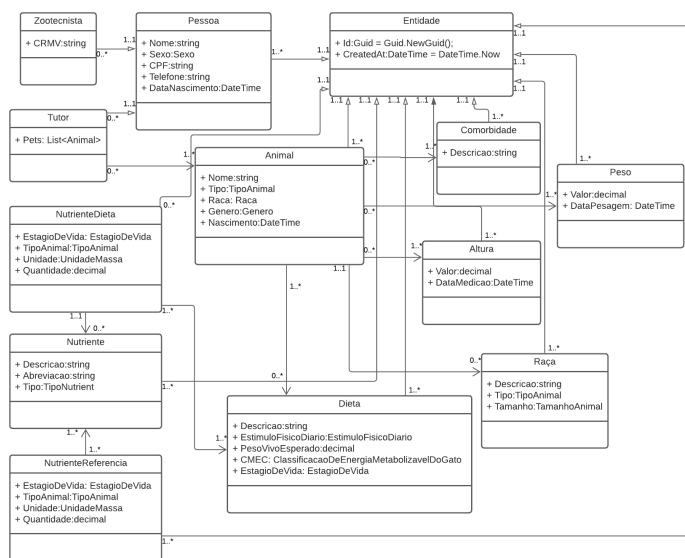


Figura 5. Diagrama de classes para ilustração do domínio do projeto

5. Resultados

Para organização do projeto e aplicação da metodologia de desenvolvimento Scrum em conjunto da cultura DevOps, a ferramenta Azure DevOps foi utilizada. Mais informações sobre a ferramenta podem ser obtidas no Anexo C.

A Azure DevOps permitiu a organização das histórias de usuário (*User Stories*) em itens do *backlog* de produto (*Product Backlog Item*). Cada história de usuário foi dividida em *Product Backlog Item* tendo *tasks* para a solução desses *Product Backlog Item*. O projeto foi dividido em 8 *Sprints*, cada uma com um entregável de valor para o cliente com planejamento de ao menos um *Product Backlog Item* de valor para o *Product Owner* por *Sprint*.

Como o Scrum propõem reuniões ocorreram durante o desenvolvimento do projeto infelizmente devido a diversas indisponibilidades da Zootecnista *Product Owner* elas não puderam ser cerimônias com as *timeboxes* propostas em todas as *Sprints*, ainda assim refinamentos do *Product Backlog* ocorreram durante o processo de desenvolvimento do produto. A Figura 6 expõe o quadro de entregáveis em uma das fases finais de desenvolvimento.

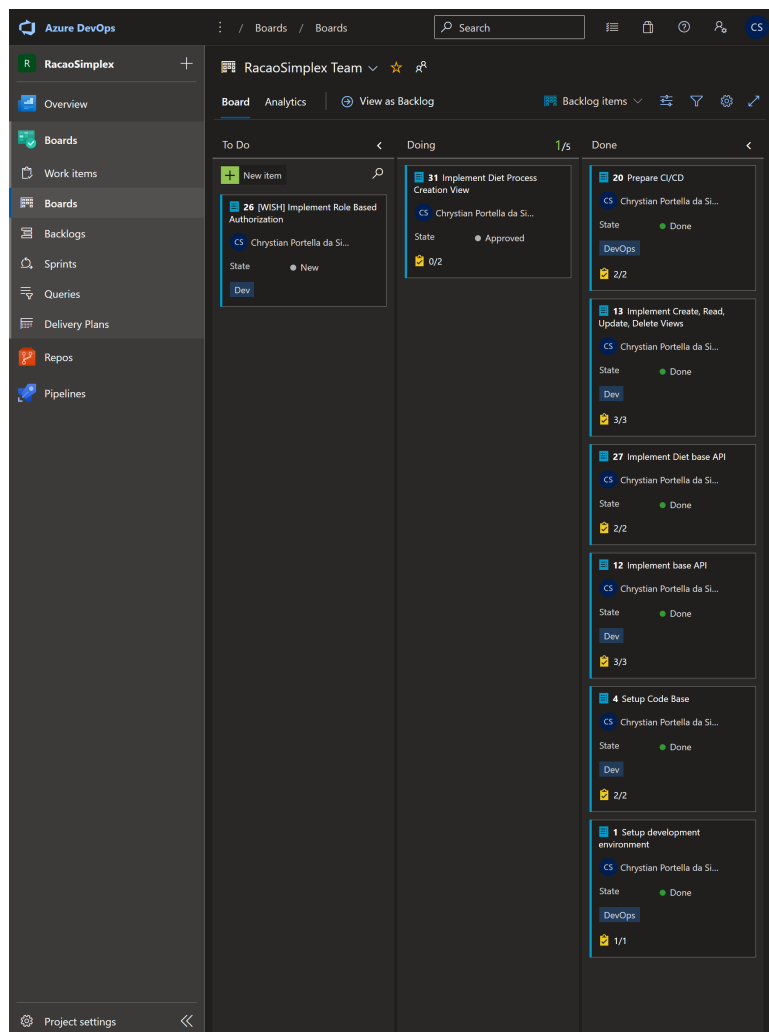


Figura 6. Captura de tela do quadro com os entregáveis

Para codificação foi montado a arquitetura base para as operações das entidades, em conjunto do *MVC* também foi adotado o padrão *MVVM*⁷. Mais informações sobre esse padrão de projeto podem ser obtidas no Anexo E. No escopo do *MVC* as *controllers* que agem como um intermediário entre a camada de visualização do Blazor e de infraestrutura de *back-end*. A comunicação entre as camadas de *front-end*, a interface gráfica de interação do usuário e *back-end*, a estrutura de apoio às ações do usuário processados na máquina, ocorre através do protocolo REST⁸ e *payloads* que são os dados que realmente interessam sem metadados, em formato do tipo JSON⁹, onde o *MVVM* foi essencial para a tradução do formato da *model* que representa a entidade para a *view* que exhibe as informações ao usuário.

Para apoio no desenvolvimento das telas de interação de usuário, além do Blazor foi usada a biblioteca MudBlazor que é um conjunto de componentes baseada no conjunto de *designs* da Google, o Material Design. O MudBlazor possui diversos componentes e *templates* para um rápida e responsiva produção de interfaces.

O produto final deu valor ao Zootecnista com o controle sobre os dados de animais, tutores, raças, nutrientes de dieta e suas referências. Operações de criação e leitura foram feitas em todas as entidades. Captura de telas de cada uma das principais entidades podem ser vistas no Apêndice A.

Ainda sobre o *back-end* no que concerne ao DevOps são adotados recomendações diversas de automações uma destas recomendações são os testes unitários ilustrado na Figura 7.

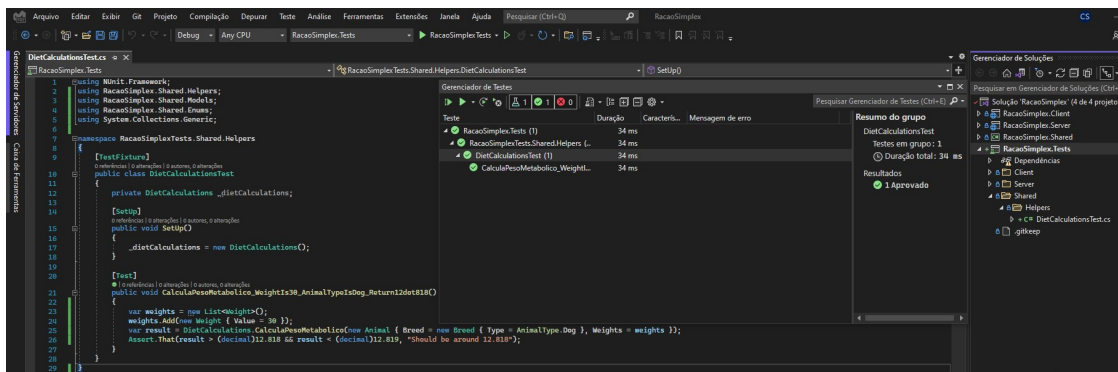


Figura 7. Captura de tela de teste unitário executado com sucesso

O Azure Devops também automatiza os processo de Integração Contínua (*CI*)/Entrega contínua (*CD*) através do Azure DevOps Pipelines, a parte de Ops da *toolbox*. Porém esse é um serviço pago e durante a execução deste projeto não foi possível configurar um teste gratuito no serviço. Foi então escolhido o Github Actions como alternativa para automação de testes e a parte de *CI*. Todo o código-fonte foi transferido para um repositório no Github onde lá o processo de automação de testes ocorreu com sucesso, conforme ilustra a Figura 8.

⁷Model-view-viewmodel

⁸Representational State Transfer

⁹JavaScript Object Notation

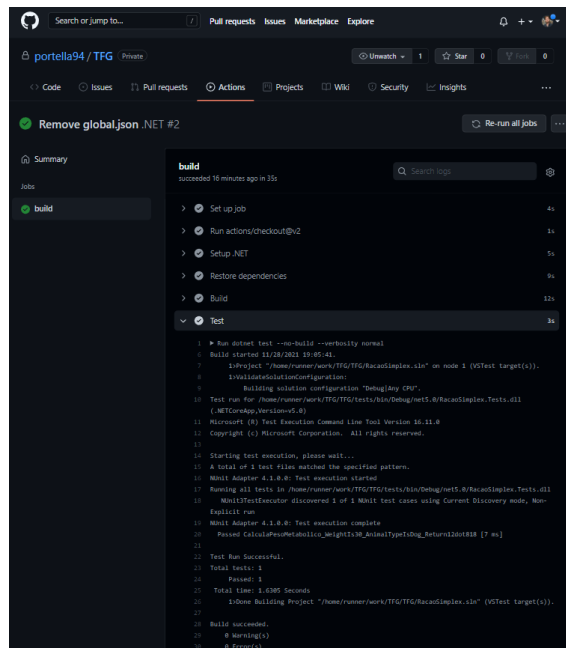


Figura 8. Captura de tela de teste automatizado após *commit* no Github

Para criação da dieta, o Zootecnista deve cadastrar seus dados, depois cadastrar o tutor do animal e o Animal em que a dieta irá ser feita. A produção da Dieta é toda automatizada os requerimentos para a elaboração é a pesagem do animal atualizada além de uma descrição, sua fase de vida, uma estimativa de estímulo físico diário. Para animais em fase de crescimento também é necessário definir um Peso Vivo Esperado e em caso de gatos a classificação de energia metabolizável. Um trecho de código das operações referente à produção da dieta pode ser visualizada na Figura 9.

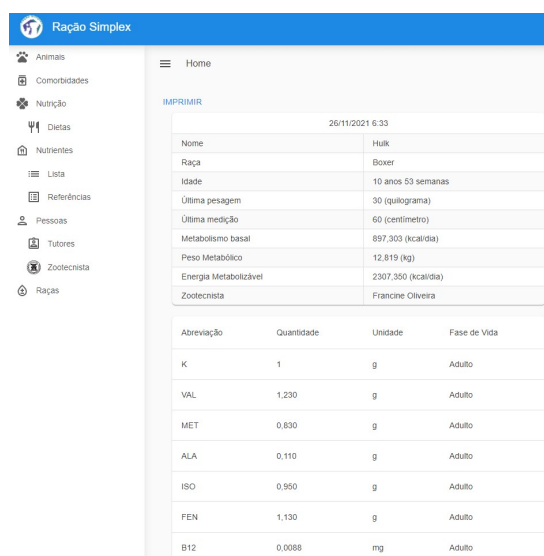
```
using RacaoSimplex.Shared.Enums;
using RacaoSimplex.Shared.Models;
using System;
using System.Linq;

[You, 7 hours ago | 1 author (You)]
namespace RacaoSimplex.Shared.Helpers
{
    [3 references | You, 7 hours ago | 1 author (You)]
    public class DietCalculations
    {
        [2 references]
        public static decimal CalculaPesoMetabolico(Animal animal)
        {
            var peso = Convert.ToDouble(animal.Weights.LastOrDefault().Value);
            double potencia = 0;
            switch (animal.Breed.Type)
            {
                case AnimalType.Dog:
                    potencia = 0.75;
                    break;
                case AnimalType.Cat:
                    potencia = 0.67;
                    break;
            }
            return Convert.ToDecimal(Math.Pow(peso, potencia));
        }

        [1 reference]
        public static decimal CalculaMetabolismoBasal(Animal animal)
        {
            return 70 * CalculaPesoMetabolico(animal);
        }
    }
}
```

Figura 9. Trecho de código de cálculos para a produção da Dieta

O resultado final da criação da dieta é ilustrado na Figura 10.



Nome	Hulk
Raça	Boxer
Idade	10 anos 53 semanas
Última pesagem	30 (quilograma)
Última medição	60 (centímetro)
Metabolismo basal	897.303 (kcal/dia)
Peso Metabólico	12.819 (kg)
Energia Metabolizável	2307.350 (kcal/dia)
Zootecnista	Francine Oliveira

Abreviação	Quantidade	Unidade	Fase de Vida
K	1	g	Adulto
VAL	1.230	g	Adulto
MET	0.830	g	Adulto
ALA	0.110	g	Adulto
ISO	0.950	g	Adulto
FEN	1.130	g	Adulto
B12	0.0088	mg	Adulto

Figura 10. Captura de tela do software com dieta realizada pela automação

6. Conclusões

A conclusão do desenvolvimento do produto ocorreu conforme a aprovação da Zootecnista *Product Owner* do projeto. Foi comprovada a melhora no processo de elaboração de dietas para cães e gatos bem como também um aumento significativo na agilidade comparado ao processo manual.

O *framework* de desenvolvimento ágil se mostra essencial para adaptabilidade e entrega contínua do projeto, ainda assim por questões de tempo limite de projeto do time de desenvolvimento, a funcionalidade de controle de usuário solicitado pela *Product Owner* durante os refinamentos de *backlog*, não foi possível ser entregue principalmente em virtude da falta de disponibilidade da Zootecnista *Product Owner*, acometendo em atrasos nas *Sprints* resultando na quantidade de entregáveis.

O DevOps influencia na velocidade dos entregáveis permitindo um entregável com poucos ou praticamente sem impacto no software em produção através das automações de *deploy* e testes automatizados. Por fim as tecnologias de produção de software .NET e Blazor, fomentam uma produção de alto rendimento com reaproveitamento de código como destaque.

Como projetos futuros, fica o desejo de implementação para um controle de usuário para que os tutores possam acessar o histórico de dietas de seu animal, a busca de composição da ração, conforme as características do animal e também ampliar para animais ruminantes com intuito pela busca da ração de custo mínimo.

Referências

- Booch, G., Rumbaugh, J., and Jacobson, I. (2006). UML: guia do usuário. Elsevier Brasil.
- Cohn, M. (2004). User stories applied: For agile software development. Addison-Wesley Professional.
- Couto, H. P. and Real, G. C. (2019). Nutrição e Alimentação de Cães e Gatos. Aprenda Fácil Editora, Viçosa - MG.
- Duncan, S. (2009). The thoughtworks® anthology, essays on software technology and innovation. *Software Quality Professional*, 11(2):53.
- Humble, J. and Farley, D. (2014). Entrega contínua: como entregar software de forma rápida e confiável.
- Joshi, B. (2019). Beginning Database Programming Using ASP. NET Core 3. Springer.
- Laflamme, D. (1997). Development and validation of a body condition score system for dogs. *Canine Practice (Santa Barbara, Calif.: 1990)(USA)*.
- Nielsen, K. S. (2002). Fisiologia animal: adaptação e meio ambiente. *Com. Imp. Ltda*, page 611.
- Ogoshi, R. C. S., Reis, J. S. d., Zangeronimo, M. G., and Saad, F. (2015). Conceitos básicos sobre nutrição e alimentação de cães e gatos. *Ciência Animal*, 25(1):64–75.
- Politowski, C. and Maran, V. (2014). Comparação de Performance entre PostgreSQL e MongoDB. *X Escola Regional de Banco de Dados. SBC*, pages 1–10.
- Rossberg, J. (2019). An overview of azure devops. *Agile Project Management with Azure DevOps*, pages 37–66.
- Sabbagh, R. (2014). Scrum: Gestão ágil para projetos de sucesso. Editora Casa do Código.
- Salzman, M. (2000). Pet trends: The state of the pet industry. *Vital speeches of the day*, 67(5):147.
- Santos, F. A., Rodrigues, M., and Lisboa Filho, J. (2006). Modelo computacional para formulação de rações de mínimo custo para pequenos ruminantes utilizando programação linear. In *SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO*, volume 13.
- Sato, D. (2014). DevOps na prática: entrega de software confiável e automatizada. Editora Casa do Código.
- Saxena, P., Kumar, V., and Kumar, R. (2016). Optimization for animal diet formulation: Programming technique. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2255–2260. IEEE.
- Wikimedia (2016). A visual representation of the devops workflow. Acesso em 24 de Maio de 2021.

ANEXO A. Tabelas de nutrientes para cães e gatos segundo *NRC*

A Tabela 1 expõe as recomendações diárias de proteína e aminoácidos para cães considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com *NRC* (2006).

Tabela 1. Recomendações diárias de proteína e aminoácidos para cães

Energia / Nutrientes	Unidade	Desmame - 14 Sem.	14 Sem. - Adulto	Adulto
ARG	g	1,980	1,650	0,880
PB	g	56,30	43,80	25,00
HIS	g	0,980	0,630	0,480
ISO	g	1,630	1,250	0,950
MET	g	0,880	0,650	0,830
MET + CIS	g	1,750	1,330	1,630
LEU	g	3,220	2,050	1,700
LIS	g	2,200	1,750	0,880
FEN	g	1,630	1,250	1,130
FEN + TIR	g	3,250	2,500	1,850
TREO	g	2,030	1,580	1,080
TRI	g	0,580	0,450	0,350
VAL	g	1,700	1,400	1,230

A Tabela 2 expõe as recomendações lipídicas para cães considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 2. Recomendações lipídicas para cães em crescimento e adulto

Energia / Nutrientes	Unidade	Desmame - 14 Sem.	14 Sem. - Adulto	Adulto
Ác. Linoleico	g	3,30	3,30	2,80
Ác. Linolênico	g	0,200	0,200	0,110
Ác. Araquidônico	g	0,080	0,080	-
DHA + EPA	g	0,130	0,130	0,110
EE	g	21,30	21,30	13,80

A Tabela 3 expõe as recomendações macrominerais para cães em crescimento e adulto considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 3. Recomendações macrominerais para cães em crescimento e adulto

Energia / Nutrientes	Unidade	Desmame - 14 Sem.	14 Sem. - Adulto	Adulto
Na	g	0,55	0,55	0,20
Ca	g	3,00	3,00	1,00
P	g	2,50	2,50	0,75
Mg	g	0,10	0,10	0,15
K	g	1,10	1,10	1,00
Cl	g	0,72	0,72	0,30

A Tabela 4 expõe as recomendações microminerais para cães em crescimento e adulto considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 4. Recomendações microminerais para cães em crescimento e adulto

Energia / Nutrientes	Unidade	Desmame - 14 Sem.	14 Sem. - Adulto	Adulto
Cu	mg	2,70	2,70	1,50
Fe	mg	22,00	22,00	7,50
Zn	mg	25,00	25,00	15,00
Mn	mg	1,40	1,40	1,20
Se	mg	0,09	0,09	0,09
I	mg	0,22	0,22	0,22

A Tabela 5 expõe as recomendações de vitaminas para cães em crescimento e adulto considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 5. Recomendações de vitaminas para cães em crescimento e adulto

Energia / Nutrientes	Unidade	Desmame - 14 Sem.	14 Sem. - Adulto	Adulto
Vit. A	UI	1263	1263	1263
Vit. D (Colicalciferol)	UI	136	136	136
Vit. E	UI	7,50	7,50	7,50
Vit. K (Menadiona)	mg	0,410	0,410	0,410
Vit. B1 (Tiamina)	mg	0,340	0,340	0,560
Vit. B2 (Riboflavina)	mg	1,320	1,320	1,300
Vit. B3 (Niacina)	mg	4,250	4,250	4,250
Vit B5 (Ác. Pantotênico)	mg	3,750	3,750	3,750
Vit. B6 (Piridoxina)	mg	0,375	0,375	0,375
Vit. B7 (Biotina)	mg	-	-	-
Vit. B8 (Colina)	mg	425	425	425
Vit. B9 (Ác. Fólico)	mg	0,0680	0,0680	0,0675
Vit. B12 (Cobalamina)	mg	0,0088	0,0088	0,0088

A Tabela 6 expõe as recomendações lipídicas para gatos considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 6. Recomendações lipídicas para gatos em crescimento e adulto

Energia / Nutrientes	Unidade	Após Desmame	Adulto
Ác. Linoleico	g	1,400	1,400
Ác. Linolênico	g	0,050	-
Ác. Araquidônico	g	0,050	0,015
DHA + EPA	g	0,025	0,025
EE	g	22,50	22,50

A Tabela 7 expõe as recomendações diárias de proteína e aminoácidos para gatos considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 7. Recomendações diárias de proteína e aminoácidos para gatos

Energia / Nutrientes	Unidade	Após Desmame	Adulto
ARG	g	2,400	1,930
PB	g	56,30	50,00
HIS	g	0,830	0,650
ISO	g	1,400	1,080
MET	g	1,100	0,430
MET + CIS	g	2,200	0,850
LEU	g	3,200	2,550
LIS	g	2,100	0,850
FEN	g	1,300	1,000
FEN + TIR	g	4,800	3,830
TAURINA	g	1,600	1,280
TREO	g	1,600	1,300
TRI	g	0,400	0,330
VAL	g	1,600	1,280

A Tabela 8 expõe as recomendações macrominerais para gatos em crescimento e adulto considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 8. Recomendações macrominerais para gatos em crescimento e adulto

Energia / Nutrientes	Unidade	Após Desmame	Adulto
Na	g	0,35	0,17
Ca	g	2,00	0,72
P	g	1,80	0,64
Mg	g	0,10	0,10
K	g	0,10	1,30
Cl	g	0,23	0,24

A Tabela 9 expõe as recomendações microminerais para gatos em crescimento e adulto considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 9. Recomendações microminerais para gatos em crescimento e adulto

Energia / Nutrientes	Unidade	Após Desmame	Adulto
Cu	mg	2,10	1,20
Fe	mg	20,00	20,00
Zn	mg	18,50	18,50
Mn	mg	1,20	1,20
Se	mg	0,01	0,08
I	mg	0,45	0,35

A Tabela 10 expõe as recomendações de vitaminas para gatos em crescimento e adulto considerando Unidade/Dia para 1000 Kcal conforme fase de vida de acordo com NRC (2006).

Tabela 10. Recomendações de vitaminas para gatos em crescimento e adulto

Energia / Nutrientes	Unidade	Desmame	Adulto
Vit. A	UI	173	82
Vit. D (Colicalciferol)	UI	12	6,8
Vit. E	UI	2,00	0,94
Vit. K (Menadiona)	mg	0,050	0,025
Vit. B1 (Tiamina)	mg	0,290	0,140
Vit. B2 (Riboflavina)	mg	0,210	0,099
Vit. B3 (Niacina)	mg	2,100	0,990
Vit B5 (Ác. Pantotênico)	mg	300,00	0,140
Vit. B6 (Piridoxina)	mg	0,130	0,060
Vit. B7 (Biotina)	mg	0,004	1,900
Vit. B8 (Colina)	mg	133	63
Vit. B9 (Ác. Fólico)	mg	0,0390	19,000
Vit. B12 (Cobalamina)	mg	0,0012	0,5600

O gráfico na Tabela 11 expõe as recomendações de níveis de aminoácidos para cães e gatos.

ANEXO B. Restrições e resultados dos trabalhos correlatos

Tabela 12. As restrições aplicadas ao modelo [Santos et al. 2006, p.5]

Restrição	Sigla
Exigência do animal em energia líquida	ener
Exigência do animal em proteína metabolizável	prot
Exigência do animal em cálcio (Ca)	calc
Exigência do animal em fósforo (P)	fosf
Limitação do consumo de fibra pelo animal	fibr
Recomendação do consumo de fibra efetiva pelo animal	f_ef

Tabela 11. Recomendação de níveis de aminoácidos para cães e gatos [Couto and Real 2019, p.161]

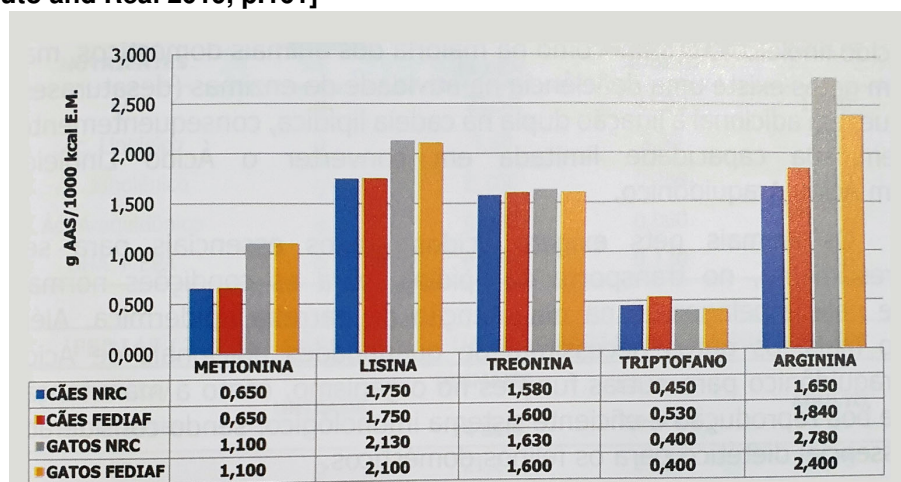


Tabela 13. Resultados para minimização de custo e conteúdo de água [Saxena et al. 2016, p.3820, tradução nossa]

Classe de peso	Custo (Rs/Kg.)	Teor de Água
0-680	261.64079822616407	2.604856512141289
200-680	59.83061330569353	0.5744060050122107
300-680	81.56982689611455	0.7842420640020565
450-680	126.72984349237079	1.247303726594053

ANEXO C. Azure DevOps

Com o Azure DevOps, é possível implementar o processo de desenvolvimento como um modelo obrigatório para todos os novos projetos. Ao criar um novo projeto, também é criada uma nova instância do modelo de processo. Também é possível implementar a maioria das partes do ciclo de DevOps no modelo, permitindo que se aproveite as vantagens do Azure DevOps ao longo do caminho. O modelo ajuda a visualizar e automatizar tarefas e etapas que o processo inclui. O Azure DevOps também ajuda a fornecer modelos de documentos para especificações de requisitos, casos de teste, cenários, transferências e outros artefatos que você produz [Rossberg 2019].

ANEXO D. Tabela de escore corporal de Laflamme

A Tabela 14 expõe uma forma de avaliação subjetiva para a condição nutricional de cães.

Tabela 14. Escore corporal para cães. Adaptado de [Laflamme 1997]

Condição	Escore	Características
Subalimentado	1	Costelas, vértebras lombares, ossos pélvicos e todas as saliências ósseas visíveis a distância. Não há gordura corporal evidente. Perda nítida de massa muscular
Subalimentado	2	Costelas, vértebras lombares, ossos e algumas outras saliências ósseas visíveis. Não há gordura corporal palpável. Perda mínima de massa muscular.
Subalimentado	3	Costelas facilmente palpáveis podendo ser vistas sem gordura palpável. Ossos pélvicos começam a ficar visíveis. O topo das vértebras lombares é visível. Cintura e reentrância abdominal evidentes.
Ideal	4	Costelas facilmente palpáveis com pouquíssima cobertura de gordura. Cintura facilmente observada vista de cima. Reentrância abdominal evidente.
Ideal	5	Costelas palpáveis sem excesso de cobertura de gordura. Abdome retraído visível de lado.
Ideal	6	Costelas palpáveis com um pouco de excesso de gordura. Cintura visível de cima, porém não acentuada. Reentrância abdominal aparente.
Sobrealimentado	7	Costelas palpáveis com dificuldade, intensa cobertura de gordura. Depósitos de gordura evidentes sobre a lombar e base da cauda. Cintura quase não visível. Reentrância abdominal pode estar presente.
Sobrealimentado	8	Costelas impalpáveis devida cobertura de gordura excessivamente densa ou costelas palpáveis apenas com pressão acentuada. Depósitos pesados de gordura sobre área lombar e base da cauda. Cintura inexistente. Sem reentrância abdominal. Pode existir distensão abdominal evidente.
Sobrealimentado	9	Grande quantidade de depósitos de gordura sobre o tórax, espinha, base da cauda, pescoço e membros. Distensão abdominal evidente

ANEXO E. *Model-view-viewmodel*

No padrão *Model-view-viewmodel* (MVVM), a *model* e a *view* têm as mesmas responsabilidades que no padrão *MVC*. O *ViewModel* está intimamente relacionado a uma visualização e é responsável por visualizações específicas coisas como vinculação de dados, manipulação de eventos da interface de usuário e notificações dessas interfaces. A *ViewModel* encapsula dados e comportamentos específicos da *view*. E também atualiza o modelo de dados subjacente sempre que necessário. [Joshi 2019] A Figura 11 apresenta a estrutura do MVVM.

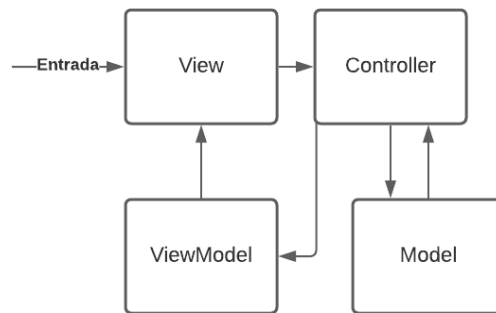


Figura 11. Estrutura MVVM

APÊNDICE A. Capturas de Tela do Produto

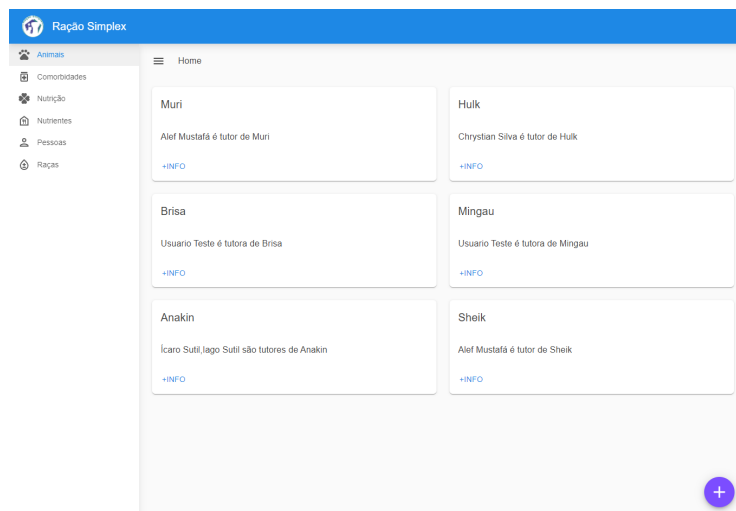


Figura 1. Captura de tela da listagem de animais

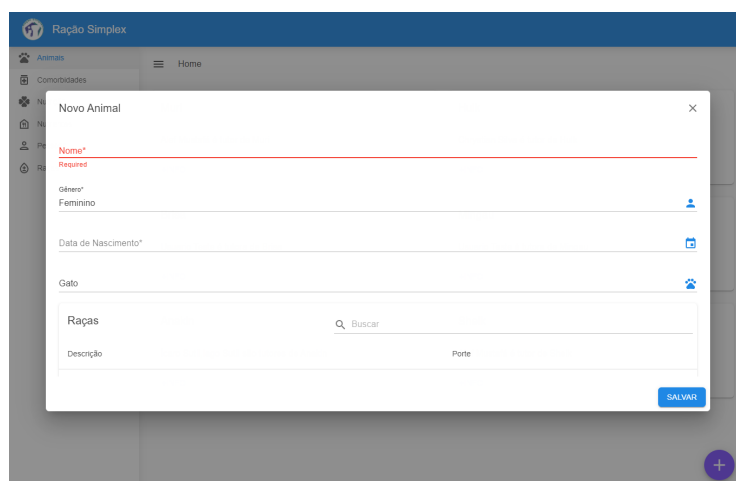


Figura 2. Captura de tela de cadastro de animal

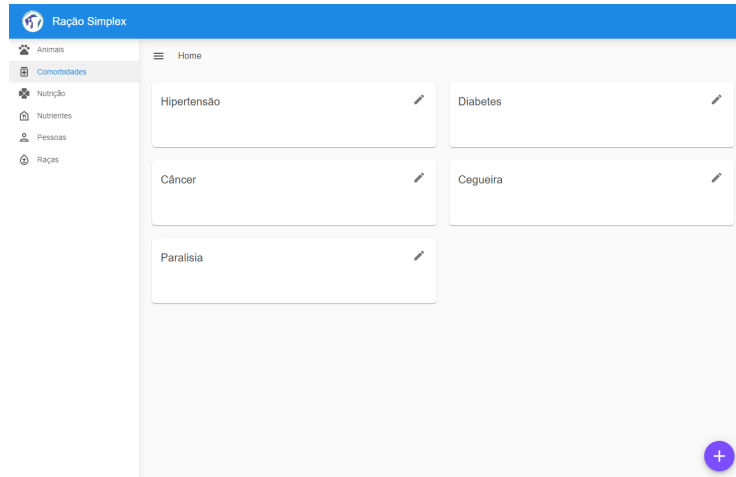


Figura 3. Captura de tela da listagem de comorbidades

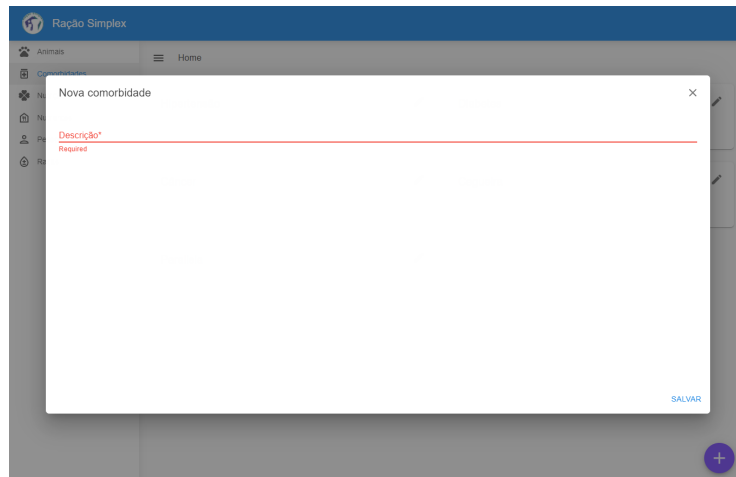


Figura 4. Captura de tela de cadastro de comorbidade

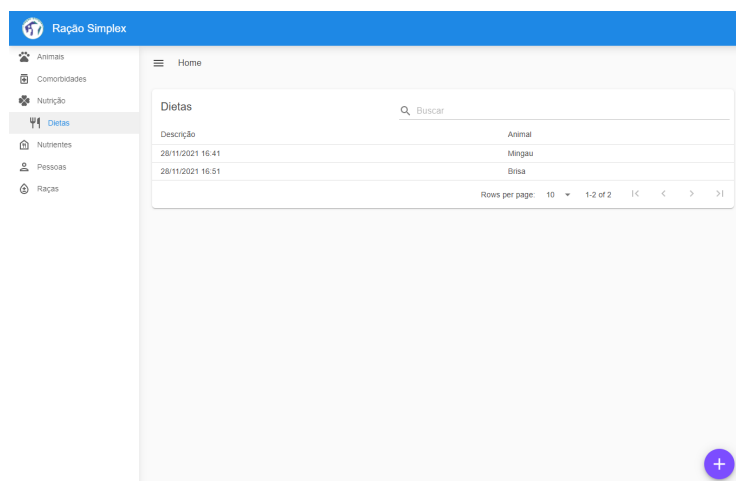


Figura 5. Captura de tela da listagem de dietas

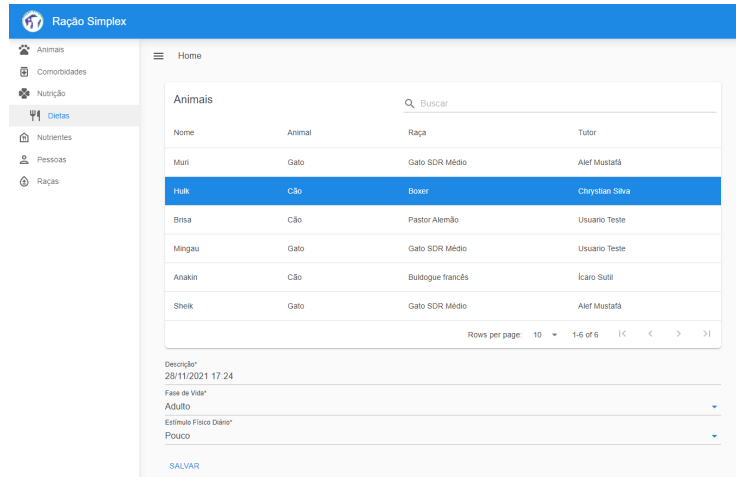


Figura 6. Captura de tela de cadastro de dieta

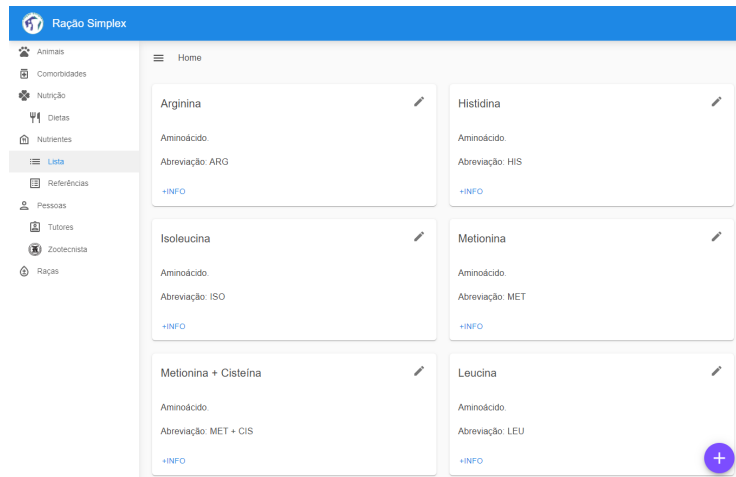


Figura 7. Captura de tela da listagem de nutrientes

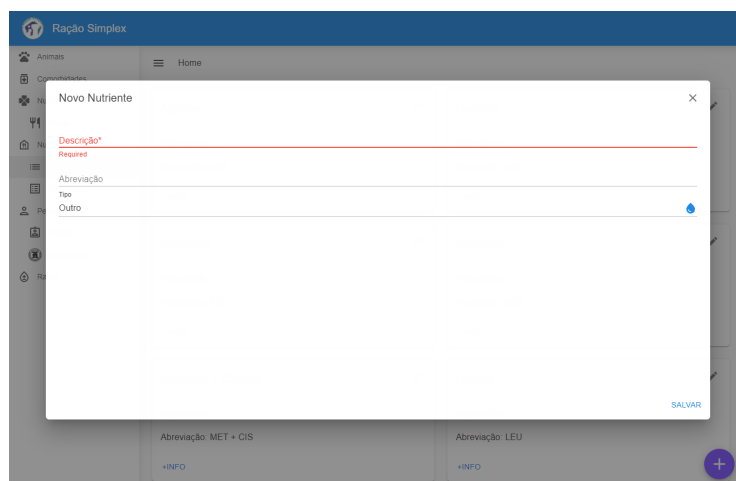


Figura 8. Captura de tela de cadastro de nutriente

Ração Simplex

Home

Referências

Buscar

Descrição	Abreviação	Quantidade	Unidade	Animal	Fase de Vida	Tipo
Arginina	ARG	0,880	grama	Cão	Adulto	Aminoácido
Arginina	ARG	1,650	grama	Cão	14 Sem. - Adulto	Aminoácido
Arginina	ARG	1,98	grama	Cão	Desmame - 14 Sem	Aminoácido
Histidina	HIS	0,48	grama	Cão	Adulto	Aminoácido
Histidina	HIS	0,630	grama	Cão	14 Sem. - Adulto	Aminoácido
Histidina	HIS	0,980	grama	Cão	Desmame - 14 Sem	Aminoácido
Isoleucina	ISO	0,950	grama	Cão	Adulto	Aminoácido
Isoleucina	ISO	1,250	grama	Cão	14 Sem. - Adulto	Aminoácido
Isoleucina	ISO	1,630	grama	Cão	Desmame - 14 Sem	Aminoácido
Metionina	MET	0,830	grama	Cão	Adulto	Aminoácido

Rows per page: 10 1-10 of 124

Figura 9. Captura de tela da listagem de referências de nutrientes

Ração Simplex

Home

Nova referência para nutriente

Lisina	LIS	Aminoácido
Fenilalanina	FEN	Aminoácido
Fenilalanina + Tirosina	FEN + TIR	Aminoácido
Treonina	TREO	Aminoácido

Rows per page: 10 1-10 of 44

Fase de Vida*
Adulto

Animal*
Cão

Unidade*
miligrama

Quantidade*
0,0000

SALVAR

Figura 10. Captura de tela de cadastro de referência de nutriente

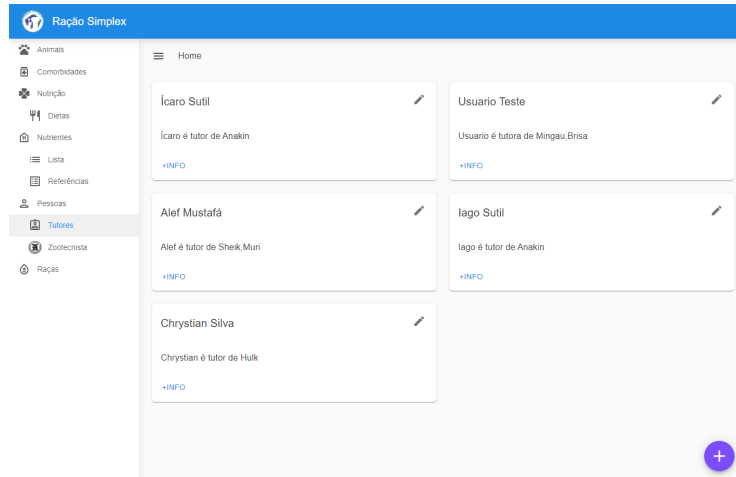


Figura 11. Captura de tela da listagem de tutores

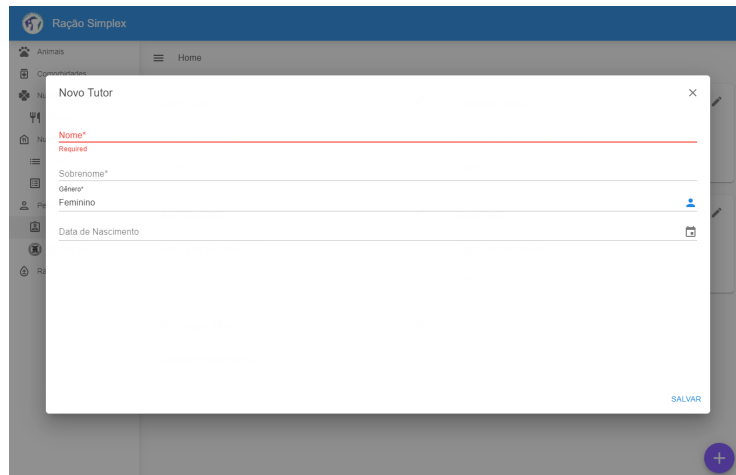


Figura 12. Captura de tela de cadastro de tutor

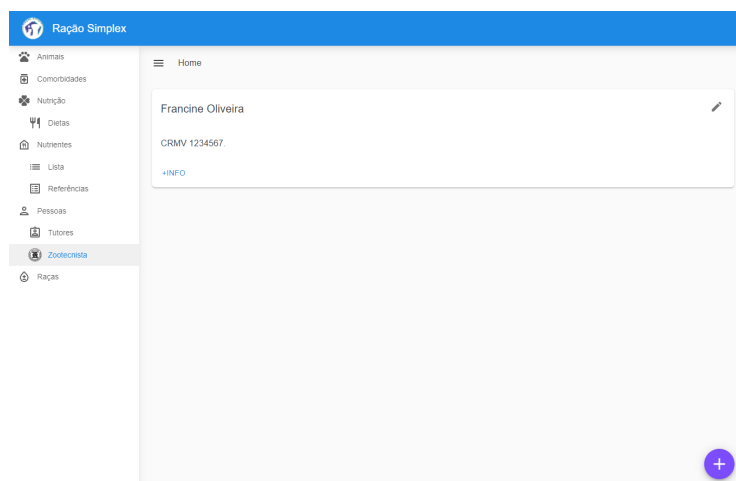


Figura 13. Captura de tela da listagem de zootecnistas

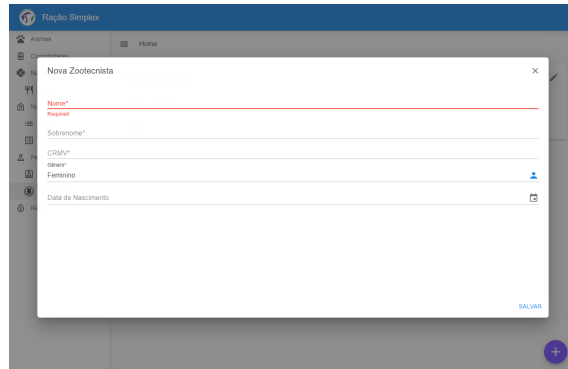


Figura 14. Captura de tela de cadastro de zootecnista

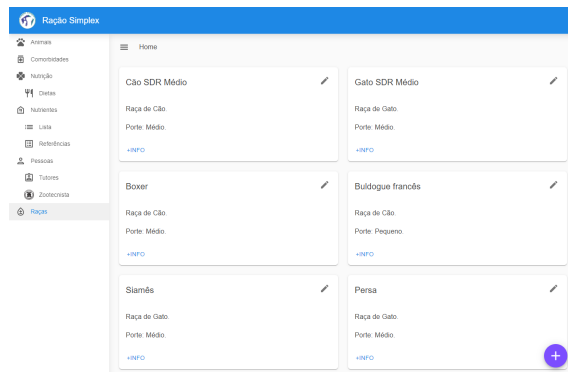


Figura 15. Captura de tela da listagem de raças

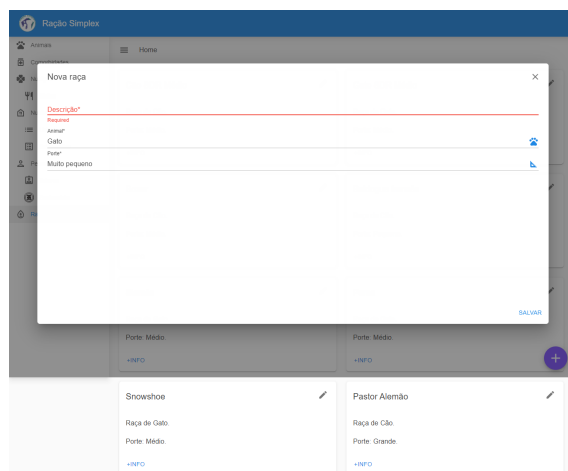


Figura 16. Captura de tela de cadastro de raça