UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FÁBIO VINÍCIUS IANTAS

HENRIQUE MATEUS TEIXEIRA EIFERT

FERRAMENTA PARA CÁLCULO DE BALANCEAMENTO DE MACRONUTRIENTES DO SOLO

PONTA GROSSA

2021

FÁBIO VINÍCIUS IANTAS

HENRIQUE MATEUS TEIXEIRA EIFERT

FERRAMENTA PARA CÁLCULO DE BALANCEAMENTO DE MACRONUTRIENTES DO SOLO

TOOL FOR BALANCING CALCULATION OF SOIL MACRONUTRIENTS

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas o da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. M.Sc. Luiz Rafael Schmitke.

PONTA GROSSA

2021

FÁBIO VINÍCIUS IANTAS

HENRIQUE MATEUS TEIXEIRA EIFERT

FERRAMENTA PARA CÁLCULO DE BALANCEAMENTO DE MACRONUTRIENTES DO SOLO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 01 de setembro de 2021

Luiz Rafael Schmitke
Mestrado
Universidade Tecnológica Federal do Paraná

Richard Duarte Ribeiro
Doutorado
Universidade Tecnológica Federal do Paraná

Vinícius Camargo Andrade
Mestrado
Universidade Tecnológica Federal do Paraná

PONTA GROSSA
2021

RESUMO

A produção pecuária intensiva gera como subproduto grandes quantidades de dejetos. Uma das opções disponíveis para o uso dos resíduos de animais é a aplicação como complemento na fertilização do solo devido ao fato de que esses resíduos possuem teores consideráveis de nutrientes, como por exemplo, nitrogênio, fósforo e potássio. A aplicação deve ocorrer de forma controlada para evitar contaminação no solo ou impactos ambientais, para isto, deve ser calculado a proporção de aplicação em função da concentração de nutrientes no dejeto, índice de eficiência do dejeto, análise do solo e da recomendação de adubação para a cultura que será aplicada. Este trabalho apresenta o desenvolvimento de um aplicativo que indicará a quantidade recomendada de aplicação de resíduos no solo. O cálculo será realizado em tela específica para este fim onde será selecionado o resíduo que é pretendido aplicar em determinada cultura conforme cadastros previamente feitos e mantidos no aplicativo. O processo de cálculo apresentará na tela de resultado o resumo das especificações usadas e indicará a dose orgânica e a complementação mineral por nutriente que será atendida para a dosagem indicada. Para isso, utilizou-se o framework Ionic, o qual usa tecnologias web HTML, CSS e Javascript.

Palavras-chave: Resíduos de animais, Nutrientes, Aplicativo, IONIC.

ABSTRACT

The intensive livestock production generates as by-product large amounts of manures. One of the options available for using wastes of animals is the application as a complement in the soil fertilization due to the fact that these residuals have considerable levels of nutrients for example N, P and K (nytrogen, phosphorus and potassium). The application should occur in a controlled way in order to avoid soil contamination or environmental impacts such as soil and rivers contamination. This work has as its main purpose develop an app that will recommend the ideal amount of organic waste which will be applied on soil.The calculation will be made from the concentration of nutirents on the manure, rate of efficency, soil analysis and the fertilization's recommendation for cultivation used.The user will have as a result the ideal quantity of manures used for hectare and the supplementing in which minerals fertilizants should be used.In addition to this, the app will allow the user to change the manures recommended quantity according to what is available. For this, it was used framework Ionic, which uses web html, css and Javascript technology.

Keywords: Wastes of animals, Nutrients, Applicative. IONIC.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

K	Potássio
N	Nitrogênio
P	Fósforo

LISTA DE SIGLAS

API	*Application Programming Interface*
CASE	*Computer-Aided Software Engineering*
CSS	*Cascading Style Sheets*
HTML	*Hyper Text Markup Language*
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	*Integrated Development Environment*
SDK	Software Development Kit
SO	Sistema Operacional
XP	*Extreme Programing*

SUMÁRIO

1. INTRODUÇÃO

Atualmente o Brasil é considerado um dos maiores produtores de proteínas animais e grãos do mundo. Fato que faz a agropecuária brasileira ser um importante pilar econômico. Dados do IBGE mostraram em 2019 um efetivo de 41,44 milhões de suínos abatidos, quarta maior produção mundial. A Região Sul do Brasil lidera o ranking nacional com 49,7% do efetivo total. A produção bovina coloca o Brasil na segunda posição mundial, com um plantel efetivo de 231,5 milhões de cabeças de gado, com 34,2% a região sul lidera a produção de leite nacional. No que diz respeito a carne de frango, o Brasil lidera o *ranking* em exportação mundial, foram contabilizados um efetivo de 1,47 bilhão de galináceos, o Paraná detém 26,5% da produção nacional (IBGE, 2019).

Embora a produção pecuária intensiva proporcione um grande valor econômico, como subproduto gera grandes quantidades de dejetos que se não tratados corretamente possuem alta carga poluidora para o solo, ar e água (KONZEN, 2005; BARILLI, 2005). A quantidade total de dejetos líquidos produzidos varia de acordo com o tamanho do animal, fase de desenvolvimento, peso, sexo, alimentação entre outras características. Em média um suíno excreta 7% do seu peso por dia em fezes e urina, ou seja, um suíno de 100kg irá produzir diariamente 7 litros de dejetos (DIESEL, MIRANDA e PERDOMO, 2002).

Entre as opções disponíveis para o uso dos resíduos de animais, a aplicação agrícola como complemento na fertilização do solo é uma opção interessante sob o aspecto ambiental e econômico. No entanto, a aplicação em quantidades excessivas e contínuas em uma mesma área pode causar problemas no solo, contaminação das águas superficiais e subsuperficiais, devido ao acúmulo de nutrientes e sua posterior movimentação por meio da erosão e lixiviação (SEGANFREDO, 2002).

Sendo assim, incorporar o uso de softwares para integração lavoura-pecuária por meio de informações precisas para o uso de resíduos de animais pode minimizar os impactos ambientais decorrente desta atividade, maximizar a utilização dos insumos agrícolas com a reciclagem de nutrientes reduzindo os custos de produção, proporcionando uma agricultura de precisão e cada vez mais sustentável.

1.1 Objetivos

Neste capítulo será abordado o objeto principal do trabalho, bem como os objetivos específicos.

1.1.1 Objetivo Geral

Desenvolver um aplicativo móvel para auxiliar profissionais do campo com cálculo de balanceamento de nitrogênio, fósforo e potássio oriundos de dejetos de animais, a fim de minimizar os impactos ambientais causados pelo uso descontrolado de fertilizantes.

1.1.2 Objetivos Específicos

- Desenvolver um aplicativo para auxiliar profissionais do campo;
- Aplicar o cálculo de balanceamento de NPK, considerando fertilizantes e dejetos animais;

1.2 Justificativa

Os dejetos oriundos da suinocultura, avicultura e bovinocultura podem ser disponíveis para uso na agricultura devido ao fato de que esses resíduos possuem teores consideráveis de nutrientes como por exemplo N, P e K, constituindo excelentes fertilizantes para as plantas (PALHARES, 2019).

Segundo Palhares (2019), umas das maiores dificuldades para do uso racional de dejetos é o ajuste das dosagens conforme a necessidade de nutrientes da cultura, enquanto os fertilizantes industrializados fornecem diversas formulações com diversas proporções de nutrientes, os dejetos possuem formulações únicas e podem variar drasticamente dependendo das condições de geração e armazenamento.

Neste contexto, surge a necessidade de criar uma ferramenta que auxilie o produtor no cálculo da proporção ideal de nitrogênio, fósforo e potássio utilizando como complemento dejetos orgânicos de animais. Tendo em vista que existe

aplicativos similares disponíveis no mercado como por exemplo, NutriSolo[1] e Calibra[2] que realizam a recomendação de adubação e calagem do solo, mas não possibilitam que seja considerado o uso de dejetos como complementação na aplicação agrícola.

1.3 Metodologia

Para desenvolvimento da aplicação foi escolhida a metodologia *Extreme Programing (*XP*)*, por se tratar de uma metodologia ágil que proporciona o desenvolvimento de sistemas de qualidade em um menor período, reduzindo os custos do projeto (CASTRO, 2006). O desenvolvimento do referido trabalho foi dividido em algumas etapas: embasamento teórico sobre os critérios técnicos e legais para adubação de manutenção da lavoura com a ciclagem de nutrientes, levantamento dos requisitos do sistema baseado na primeira etapa, apresentação das tecnologias necessárias para o desenvolvimento da aplicação e por final a documentação e diagramação com base nos requisitos levantados nas etapas anteriores.

A primeira etapa possibilitou conhecer os processos que envolvem a geração e reciclagem de dejetos, bem como os problemas gerados pela utilização indiscriminada sem os mínimos critérios para aplicação. Esta etapa foi realizada por meio de consultas a legislação, literatura da área agronômica e por consulta a profissionais da área ambiental, com intuito de extrair a melhor forma de minimizar o problema com o uso da tecnologia. Com o embasamento da primeira etapa deu início a segunda, o levantamento dos requisitos com a estrutura necessária e as funcionalidades utilizadas no aplicativo para o cálculo de balanceamento de nutrientes do solo. Essas informações irão direcionar o desenvolvimento da documentação e diagramas UML (Linguagem de Modelagem Unificada).

Com base na documentação inicia-se a próxima etapa, a apresentação das tecnologias que foram utilizadas para o desenvolvimento do aplicativo, onde foi utilizado o *framework* Ionic versão 6.16.1, devido sua característica de utilizar

[1]https://www.embrapa.br/busca-de-publicacoes/-/publicacao/1090682/aplicativo-nutrisolo-ao-alcance-das-maos-em-poucos-cliques-e-em-todo-lugar
[2]https://www.suino.com.br/empresa-lanca-app-para-calcular-quantidade-de-nutrientes-necessaria-aos-solos/

tecnologias *web* nativas, podendo ser utilizados em diferentes plataformas além de ser umas das ferramentas de desenvolvimento híbrido mais utilizadas no mundo (SILVA; SOTTO, 2018).

Para que o *framework* Ionic possa gerenciar suas aplicações e acesso aos recursos nativo de um dispositivo móvel, utilizou-se o Capacitor versão 1.0.0, um substituto melhorado do Apache Cordova, ambas ferramentas possuem o mesmo objetivo, que é fornecer uma maneira estruturada de expor as funcionalidades nativas dos dispositivos para seu código na web, permitindo que desenvolvedores usem seu código *Hyper Text Markup Language* (HTML), Cascade Style Sheets (CSS) e Java Script para criar aplicativos nativos para uma variedade de plataformas móveis e de desktop. (CAPACITOR, 2021).

A aplicação irá utilizar como armazenamento o banco de dados relacional SQLite versão 3.3.3, por se tratar de um sistema embarcado e usado de forma nativa, não necessita de configurações específicas. Seu mecanismo é seguro através de transações ACID, que garante que nenhum dado seja alterado sem necessidade aparente (SQLITE.ORG, 2021). A aplicação conta com uma base estruturada de dados parametrizados de acordo com os requisitos levantados nas primeiras etapas. O uso do SQLite ajudará a efetuar consultas complexas e manipular os dados da base de dados do aplicativo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Métodos Ágeis

De acordo com Sommerville (2007), metodologias ágeis de desenvolvimento ou métodos ágeis definem um conjunto de processos que intercalam as atividades de especificação, projeto, desenvolvimento e testes com o intuito de proporcionar o desenvolvimento rápido de softwares.

Havia uma ideia que para garantir a qualidade de um *software*, era necessário um planejamento minucioso, métodos de análise e desenvolvimento apoiado por ferramentas *Computer-Aided Software Engineering* (*CASE)* e gerenciado por processos rigorosos de desenvolvimento. Essas técnicas tornavam-se inadequadas para o desenvolvimento de pequenos projetos, em que o objetivo era satisfazer o cliente por meio de entrega adiantada e contínua do software (PRESSMAN, 2011).

De acordo com Sommerville (2007), metodologias ágeis surgiram devido a insatisfação que os desenvolvedores possuíam em utilizar os processos inadequados para o desenvolvimento de pequenos projetos. Sommerville (2007) ainda cita que a burocracia, o tempo exigido com a elaboração do projeto e documentação eram maiores que o tempo gasto com o desenvolvimento da aplicação.

Sommerville (2007) descreve que podem existir diferentes abordagem para o desenvolvimento ágil, no entanto algumas características fundamentais são compartilhadas entre elas, os quais são mostrados no Quadro 1 abaixo:

Quadro 1 - Princípios compartilhados por metodologias ágeis

Princípio	Descrição
Envolvimento do Cliente	Fornecendo e especificando novos requisitos à medida que o software é implementado.
Entrega incremental	O software é desenvolvido incremental e o cliente especifica os requisitos a serem incluídos em cada uma das interações.
Pessoa, não processo	Desenvolver sem os processos prescritivos, focar nas habilidades das pessoas.
Aceite as mudanças	Projetar o sistema de forma que seja fácil implementar mudanças e incrementos.
Mantenha simplificado	Simplicidade na hora de desenvolver, eliminando as complexidades sempre que possível.

Fonte: Sommerville (2007, p.263). Adaptados pelos autores.

2.1.1 Extreme Programming (XP)

Extreme Programming é uma metodologia baseada nos princípios ágeis de desenvolvimento, adequado para pequenas e médias equipes que desenvolvem softwares com requisitos vagos e em constante mudança. A metodologia XP é uma abordagem que ajuda a reduzir o tempo de projeto sem perder a qualidade de entrega, além de reduzir os custos e transtornos causados às pessoas envolvidas (CASTRO, 2016).

Segunda Beck (2004), a metodologia XP é definida por um conjunto de cinco valores que servem como embasamento para o trabalho realizado, são eles:

Quadro 2 – Valores da XP

Valores	Descrição
1. Feedback	- Reavaliação das necessidades ao decorrer do projeto pelo Cliente;
2. Comunicação	- Compartilhar aprendizado. **-** Comunicação direta e eficaz;
3. Simplicidade	-Implementar apenas o suficiente para atender cada necessidade.
4. Coragem	- Disciplina para aprender a projetar para o “hoje”; **-** Reconhecimento de que as necessidades futuras podem mudar drasticamente;

Fonte: Beck (2004) e Pressman (2011, p.87). Adaptado pelos autores.

Segundo Teles (2004) a metodologia XP é composta por um conjunto de regras e práticas constantes que devem ser aplicadas ao longo do projeto:

- Cliente presente: O cliente conduz o desenvolvimento através da troca de *feedback,* sua presença viabiliza a simplicidade do processo de desenvolvimento, especialmente na comunicação;
- Jogo de planejamento: Gerar valores bem definidos ao cliente através de *releases*, pequenos módulos funcionais do sistema, para que o cliente tenha diversas oportunidades de se reunir para revisar o planejamento.
- *Stand up meeting:* Reuniões diárias da equipe para avaliar o trabalho feito no dia anterior e priorizar aquilo que será implementado no dia que se inicia;
- Programação em par: código revisado permanentemente, dois desenvolvedores em um código.

- Desenvolvimento guiado pelos testes: plano de testes para cada funcionalidade antes de desenvolvê-la.
- *Refactoring:* Consiste em alterar um código sem afetar a funcionalidade que ele implementa.
- Código coletivo: Código compartilhado entre todos os desenvolvedores.
- Código padronizado: padrões de nomenclatura, tornar o código familiar.
- *Design* simples: agilidade para desenvolver, facilitar possíveis alterações futuras e melhorias;
- Metáfora: Linguagem comum entre a equipe.
- Ritmo sustentável: manter um ritmo de horas de desenvolvimento adequado durante o dia.
- Integração contínua: incorporar as novas funcionalidades diariamente, para garantir que todos os módulos estejam funcionando de forma harmoniosa.
- *Releases* curtos: disponibilizar rapidamente em produção.

2.2 Desenvolvimento de Aplicativos Móveis

O desenvolvimento de aplicativos móveis contém características diferentes do desenvolvimento de aplicações desktop. Os desenvolvedores devem considerar aspectos importantes como limitação de recursos, ecossistema heterogêneo, experiência de uso, manutenção frequentes e ciclo de vida de desenvolvimento curto (KASSAS, 2015). De acordo com CORRAL (2012), aplicações móveis costumam ser de pequeno porte, desenvolvida de forma dinâmica e lançadas no mercado em pequenos ciclos.

Segundo Bernardes e Miake (2016), o desafio dos profissionais de tecnologia da informação e fabricantes de aplicativos ainda são as diferentes linguagens de programação, padrões de desenvolvimento e plataformas variadas. Pois, desenvolver um aplicativo nativo que atenda os mais diversos sistemas operacionais exige projetos diferentes, ou seja, réplica de todo o processo de desenvolvimento, pois cada plataforma apresenta arquitetura distinta e específica ao sistema operacional atribuído.

Cada fornecedor de plataforma disponibiliza aos desenvolvedores diferentes ambientes de desenvolvimento integrado (IDEs), linguagens de programação, APIs e

mercado de distribuição de aplicativos, as chamadas lojas de apps (KASSAS, 2015). Em um cenário que haja a necessidade de entregar o aplicativo de forma mais rápida e econômica, desenvolvedores recorrem ao conceito de desenvolvimento multiplataforma ou desenvolvimento de plataforma cruzada. Porém, segundo BERNARDES E MIYAKE (2016), essa pode não ser a melhor solução de acordo com as características de determinadas aplicações, como por exemplo, jogos, aplicativos de processamento de vídeo e imagens.

2.2.1 Aplicações Nativas

Aplicações nativas são desenvolvidas com linguagem de programação definida pelo seu SO, ou seja, são executadas apenas para a plataforma específica a qual ela foi desenvolvida. Neste modelo móvel de desenvolvimento o código base muda completamente de uma plataforma para outra. As aplicações nativas fazem chamada de Interface de Programação de Aplicação (API) diretamente ao SO, garantindo desempenho e maior personalização (MORE, CHANDRAN, 2016).

A abordagem para desenvolvimento de aplicações nativas oferece algumas vantagens como exploração da capacidade máxima do dispositivo, maior performance, experiência nativa de usabilidade e maior controle e personalização dos recursos do dispositivo. Por outro lado, é necessário desenvolver um aplicativo por plataforma, a dificuldade de manutenção aumenta e o desenvolvimento requer domínio de linguagens de programação de diferentes plataformas (BERNARDER E MIYAKE, 2016).

O quadro abaixo mostra algumas das principais plataformas disponíveis no mercado, suas respectivas linguagens nativas de programação e IDE de desenvolvimento.

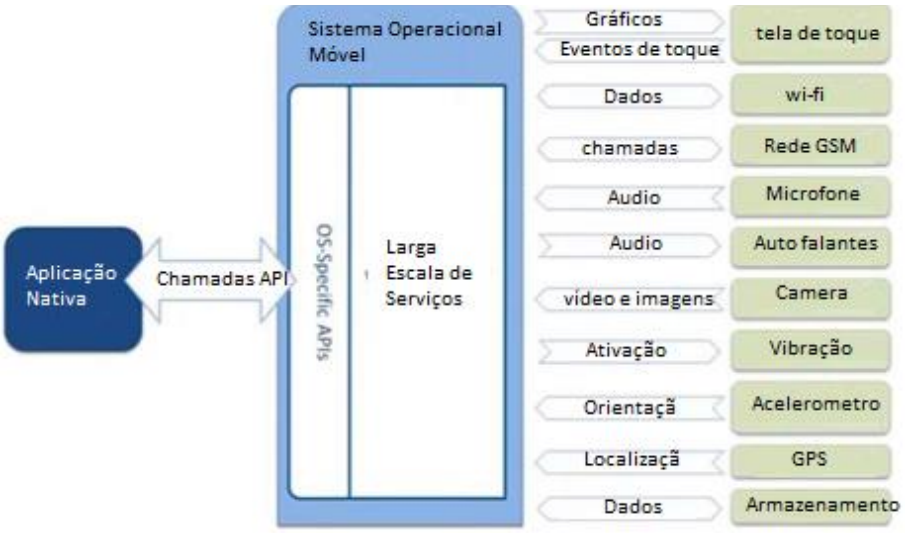
Quadro 3 - Principais Plataformas de desenvolvimento, linguagens e IDEs

Plataforma	Linguagem	IDE
Google Android	Java	Android Studio, Eclipse, NetBeans
Apple IOS	C, Objective C	XCode
Windows Phone	C#, .NET, Visual Basic	Visual Studio

Fonte: Autoria Própria (2021).

A figura 1 demonstra o funcionamento das aplicações nativas, fazendo chamadas diretamente as APIs do SO:

Figura 1 - Funcionamento das aplicações nativas



Fonte: Vendors e Resources (2018).

2.2.2 Desenvolvimento híbrido

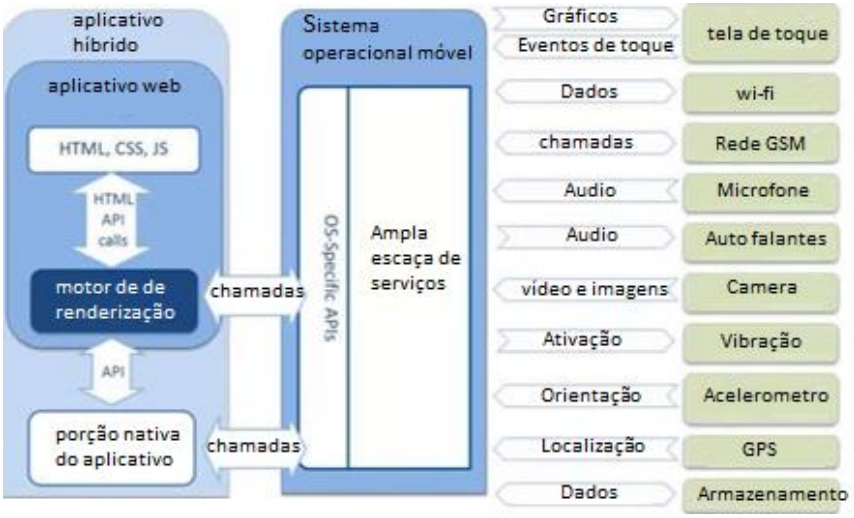
As aplicações híbridas, ao contrário das nativas, têm como principal característica o funcionamento em diferentes plataformas com apenas o desenvolvimento de um código fonte. Os elementos que fazem parte de uma aplicação híbrida são basicamente constituídos de um código *Hyper Text Markup Language* 5 (HTML5), *Cascading Style Sheets* (CSS) e Javacript, executando dentro de um container nativo que, por sua vez, integra as funcionalidades que o dispositivo oferece através de APIs do SO (BEZERRA, 2016).

A abordagem híbrida surgiu como uma combinação do uso das tecnologias *web* com possibilidade de acesso a funcionalidades nativas (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013). De acordo com Vandecandelare (2015), é importante saber quais as opções quando desenvolver uma aplicação móvel, a escolha irá depender dos recursos disponíveis, tempo de desenvolvimento e objetivo da aplicação. Vandecandelare (2015) ainda cita que o uso de um *framework* para o desenvolvimento de aplicações híbridas fornecerá possibilidade de manter a interface do usuário consistente em todas as plataformas, como por exemplo, Android e IOS.

No entanto, a abordagem híbrida pode não ser a melhor escolha quando o principal requisito da aplicação é a *performance*, como por exemplo, em jogos.

A figura 2 demostra como o funcionamento das aplicações híbridas, fazendo chamadas ao SO através de uma camada adicional (*container* nativo). Essa camada é chamada WebView no Android e UIWebView no IOS. A aplicação acessa os recursos de *hardware* do dispositivo por bibliotecas disponibilizadas pela camada adicional.

Figura 2 - Funcionamento dos aplicativos híbridos.



Fonte: Vendors e Resources (2018). Adaptado pelos autores.

2.2.3 Web App

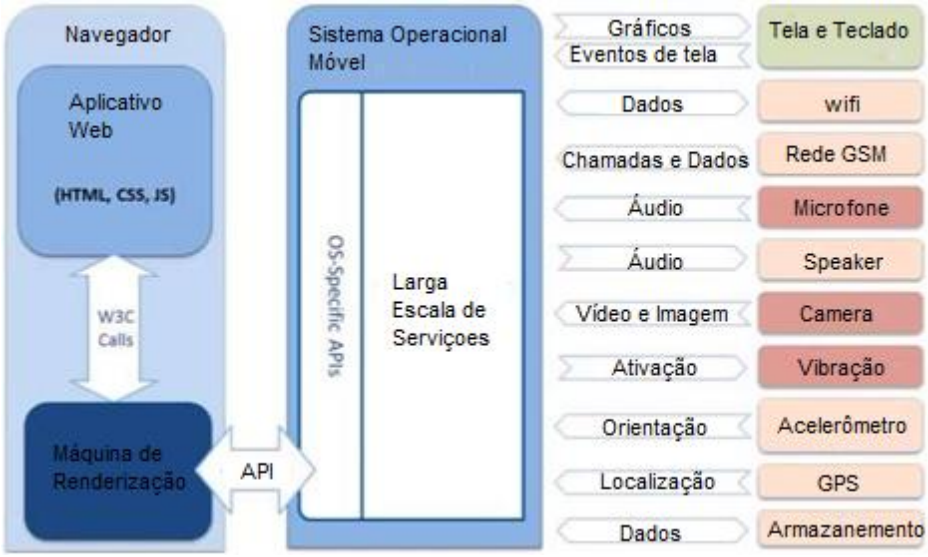
São aplicações desenvolvidas com linguagens de programação e ferramentas *web* como HTML, CSS e Javascript, com interface e funcionalidades otimizadas para as limitações dos dispositivos móveis, pois são executadas pelo *browser* fornecidos por cada plataforma e seu comportamento irá depender da maneira como o *browser* processa a aplicação. Aplicações *web* não são capazes de acessar recursos nativos dos dispositivos, como gps, câmera, sensores, o que os tornam mais limitadas (XANTHOPOULOS E XINOGALOS, 2013).

Aplicações web são executadas em navegadores, podem ter conteúdo estático e/ou dinâmicos e são acessados através de uma *Uniform Resource Locator* (URL) e não há necessidade de instalação no dispositivo. De acordo com Prezotto e Boniati (2014), aplicações são na verdade *sites* com um *layout* responsivo que permite adequar as telas aos dispositivos móveis e podem apresentar funcionalidades

semelhantes. Aplicações *web* apresentam uma desvantagem que é depender de acesso à internet.

A figura 3 demostra a interação da aplicação *web* com o dispositivo móvel:

Figura 3 - Funcionamento de aplicativos web.



Fonte: Vendors e Resources (2018).

2.3 Manejo de Dejetos

De acordo com Alves (2007), com a chegada das empresas integradoras, a criação de cooperativas e o apoio do governo disponibilizando financiamento a baixos juros, a produção de proteína animal aumentou de forma quantitativa e proporcionou que pequenos produtores pudessem praticar esta atividade. Esta expansão vem com a necessidade da correta destinação dos dejetos e resíduos. Segundo Rizzoni (2012), a suinocultura é considerada pelos órgãos de controle ambiental, a atividade agropecuária que ocasiona maior impacto ambiental.

Os compostos orgânicos provenientes dos dejetos de animais em sistemas de produção pecuária, podem ser utilizados como fertilizantes de qualidade em lavouras e pastagens, diminuindo os custos com adubos químicos e não comprometendo as características e tipologia do solo e do meio ambiente. Para isso, o produtor rural deve seguir critérios no que diz respeito a composição do solo, composição dos dejetos, quantidade, local onde será aplicado e a cultura plantada (PERDOMO; MIRANDA, 2002).

Para fins de preservação dos recursos naturais e controle sobre qualidade do solo que recebem aplicações de fertilizantes orgânicos de animais visando sua reciclagem na adubação de culturas agrícolas, florestais e outras, a Resolução SEDEST Nº 052, de 15 de julho de 2019, cita que é necessário seguir as recomendações agronômicas vigentes e estabelecidas pelo Manual de Adubação e Calagem do Estado do Paraná.

2.3.1 Cálculo para Recomendação de Fertilizantes Orgânicos

A taxa de aplicação deve ser calculada em função da concentração de nutrientes no dejeto, do índice de eficiência do dejeto da análise do solo e da recomendação de adubação para as culturas utilizadas de acordo com o Manual de adubação e calagem para o estado do Paraná (SBCS-NEPAR, 2017). Considera-se os elementos limitantes para o uso agrícola dos dejetos, o nitrogênio, fósforo e potássio, efetuando-se uma adubação baseada no princípio de equilíbrio, ou seja, a taxa de aplicação deverá ser em função do elemento que exigir menor quantidade de dejeto, realizando a complementação quando necessário (SEDEST N°052, 2019).

Os resíduos orgânicos não possuem uma proporção fixa de N, P e K, então é necessária uma análise dos resíduos para determinar sua composição. A análise dos dejetos, por parte do empreendimento gerador, é exigida por lei e deve ser feita no mínimo a cada dois anos (RESOLUÇÃO SEDEST, 2019).

O aplicativo irá efetuar o cálculo da proporção de fertilizantes orgânicos a partir da formulação já calculada da recomendação de adubação para uma dada cultura, que é obtida através da análise do solo, da tabela de interpretação dos níveis de fósforo juntamente com a recomendação para adubação do Manual de Adubação e Calagem do Estado do Paraná.

Após obter a quantidade de N, P e K recomendada para uma dada cultura, obtém-se o índice de liberação de nutrientes de acordo com o tipo de resíduo a ser aplicado, conforme Tabela 1:

Tabela 1 - Índice de mineralização de nutrientes

Origem	Resíduo orgânico	N		P_2O_5		K_2O	
animal		1º	2º	1º	2º	1º	2º

Aves	Cama de frangos	50	20	80	20	100	0
	Cama de peru	50	20	70	20	100	0
	Cama de poedeiras	50	20	70	20	100	0
Suínos	Esterco sólido de suínos	60	20	90	20	100	0
	Dejeto líquido de suínos	80	0	70	10	100	0
	Cama sobreposta de suínos	20	0	70	30	100	0
	Composto de dejeto de suínos	20	0	70	30	100	0
Bovinos	Esterco sólido de bovinos	30	20	80	20	100	0
	Dejeto líquido de bovinos	50	20	80	20	100	0
	Composto de esterco de bovinos	50	20	70	20	100	0

Fonte: Palhares, 2019, adaptado pelo autor.

Após obtenção das variáveis aplica-se as seguintes equações de acordo com o estado físico do resíduo que pode ser sólido (equação 1) ou líquido (equação 2):

$$\textbf{Dose (t ha-1) = QRN / [(MS/100) * C * (IE/100)]} \quad (1)$$

$$\textbf{Dose (m}^3\textbf{/ha-1) = QRN / [C * (IE/100)]} \quad (2)$$

Em que:

- Dose: quantidade de adubo orgânico a ser aplicado no solo (toneladas por hectare para sólidos ou metros cúbicos para líquidos).
- QRN: quantidade recomendada do nutriente.
- MS: Percentagem de matéria seca do fertilizante orgânico sólido obtida através da análise do resíduo.
- C: concentração de N, P ou K no fertilizante orgânico em quilogramas por toneladas para sólidos e quilogramas por metros cúbicos para líquidos.
- IE: índice de eficiência agronômica do fertilizante (obtido na figura 5);

A equação deve ser aplicada para cada macronutriente obtido pela recomendação de adubação do manual conforme o cultivar selecionado, assim obtém-se a dosagem total de resíduos para suprir cem por cento da necessidade de cada macronutriente.

Considerando o exemplo a seguir para lavoura de trigo, de acordo com o Manual de Adubação e Calagem do Estado do Paraná, para produzir 4,2 t/ha em um solo classificado com níveis muito alto de fósforo, são necessários uma fertilização total de 80 kg/ha de Nitrogênio, 50 kg/ha de Fósforo e 40 kg/ha de Potássio.

O resíduo empregado neste caso será o dejeto líquido de suínos, que possui uma taxa de mineralização de 80% de Nitrogênio, 90% para o Fósforo e 100% para o Potássio no primeiro cultivo, ou seja, nem todo nutriente presente no material orgânico será imediatamente disponibilizado para a planta (NICOLOSO, 2016). Em seguida, obtém-se a concentração média de nutrientes no dejeto, no entanto os valores apresentados na Tabela 2 pode variar drasticamente, de acordo com Palhares (2019), é recomendado a análise química dos dejetos para um ajuste de dosagem mais precisa.

Tabela 2 - Teores médios de nutrientes em resíduos (kg/m³)

Origem	Tipo Resíduo	Nitrogênio	Fósforo	Potássio
Suínos	Dejeto líquido de suínos	2,8	2,4	1,5

Fonte: Palhares, 2019. Adaptado pelos autores.

Aplicando a para dejetos líquidos, obtém-se a dose total de dejetos necessário para suprir cem por cento da demanda nutricional de cada macronutrientes:

a) Para atender a demanda de N:

$$35,7\ m^3 = 80 / [2,8 * (80 / 100)]$$

b) Para atender a demanda e P:

$$23,1\ m^3 = 50 / [2,4 * (90 / 100)]$$

c) Para atender a demande de K:

$$26,6\ m^3 = 40 / [1,5 * (100 / 100)]$$

Neste caso obtém-se os valores conforme a Tabela 3.

Tabela 3 - Total de dejetos necessários por macronutriente em m³

Nitrogênio	Fósforo	Potássio
35,7	23,1	26,6

Fonte: Autoria própria (2021).

Para esta situação, de acordo com Nicoloso (2016), a recomendação é usar a menor dose de dejetos obtida, neste caso 23,1 m³ para suprir a demanda de Fósforo. Caso a opção mais alta fosse utilizada de 35,7 m³ para suprir a demanda de Nitrogênio, seria depositado um excessivo de 27 kg/ha de Fósforo e 13,6 kg/ha de Potássio, o que deve ser evitado a fim de mitigar possíveis impactos ambientais do excesso de P (PALHARES, 2019).

Utilizando a menor dose obtida, que é de 23,1 m³, é calculado a proporção em relação ao Nitrogênio e o Potássio, ou seja, 23,1 m³ de dejetos contém 51 kg dos 80 necessários de N e 34,7 kg dos 40 recomendados de Potássio. E por final, calcula-se a diferença entre a formulação recomendada para produção estimada e a proporção obtida com a menor dose, conforme Tabela 4.

Tabela 4 - Cálculo de complementação proporcional pela menor dosagem (Kg/ha)

Recomendação total	**Nitrogênio**	**Fósforo**	**Potássio**
	80	50	40
Proporção pela menor dose (23,1 m³)	51	50	34,7
Formulação de Complementação	26	0	5,3

Fonte: Autoria Própria (2021).

Ao final obtém-se o resultado de que para produção de trigo, com uma estimativa de 4,2 t/ha são necessários um total de 26 kg/ha de N, 0 kg/ha de P e 5,3 kg/ha de K, utilizando uma dosagem de 26,1 m³ de dejetos líquido de suínos, com uma concentração média de 2,8 kg/m³ de N, 2,4 kg/m³ de P, 1,5 kg/m³ de K.

3 TECNOLOGIAS UTILIZADAS

Este capítulo tem como finalidade apresentar as ferramentas que serão utilizadas para construção do aplicativo. Devido ao fato de que o tempo e recursos para desenvolvimento do aplicativo são reduzidos, optou-se pela utilização de um *framework* com abordagem híbrida. Além de proporcionar um desenvolvimento multiplataforma, ele atende um requisito básico do aplicativo que é acesso aos recursos de armazenamento do dispositivo, através da utilização de um banco de dados relacional embarcado nas plataformas alvo Android e IOS sem a necessidade de instalações e configurações adicionais.

Frameworks no geral são constituídos de classes que contém uma estrutura base como um esqueleto e é usado para desenvolvimento de uma aplicação. Com o uso de um *framework* é possível reduzir o tempo e recursos utilizados, por meio de um esquema conceitual possui uma representação com alto nível de abstração que modela os fatos do mundo real, além de poder maximizar a utilização de código feitos e já testados por outros desenvolvedores (LIMA, 2014).

3.1 Ionic

O Ionic é um *framework* para desenvolvimento de aplicativos móveis com uma abordagem híbrida, que possibilita o uso de elementos do desenvolvimento *web* como, HTML, CSS e Javascript juntamente com funções nativas dos dispositivos. O Ionic emula as diretrizes de interface do usuário do aplicativo *mobile* utilizando a SDK (Software Development Kit) nativa. Ionic é uma ferramenta *open source* sob licença do MIT[3], código fonte aberto e apoiada por uma enorme comunidade mundial (IONIC, 2021).

O framework tem como principal característica a interface com usuário, pois fornece elementos e *layouts* com estilo muito semelhantes àqueles que seriam obtidos com desenvolvimento em uma SDK nativa. De acordo com Ionic (2021) o framework

[3] Permissão concedida gratuitamente, a qualquer pessoa que obtenha uma cópia deste software e arquivos de documentação associados (o "Software"), para lidar com o Software sem restrição, incluindo, sem limitação, os direitos de usar, copiar, modificar, mesclar, publicar, distribuir, sublicenciar e/ou vender cópias do Software.

foi otimizado com intenção de ser executado em navegadores de baixo nível, como por exemplo o WebView que é utilizado como container nativo para execução de aplicativos híbridos nos sistemas operacionais móveis. O *framework* foi desenvolvido de modo que a criação de aplicativos seja fácil de aprender e acessível a praticamente qualquer pessoa com habilidades de desenvolvimento *web* IONIC (2021).

Aplicativos construídos com Ionic podem ser baixados diretamente das principais lojas de aplicativos, como por exemplo, a App Store da Apple e/ou a Play Store da Google. Os Kits de desenvolvimento (SDKs) do IOS e Android fornecem visualizações da web que renderizam qualquer aplicativo Ionic, ao mesmo tempo que permite acesso aos recursos nativos (IONIC, 2021).

Projetos como Capacitor e Cordova são comumente usados para dar aos aplicativos Ionic esse acesso a SDKs nativas. Isso significa que os desenvolvedores podem criar rapidamente um aplicativo usando ferramentas comuns de desenvolvimento da *Web* e ter acesso a recursos nativos, como acelerômetro, câmera GPS e muito mais (IONIC, 2021).

3.2 Angular

Versões anteriores a 4.X do Ionic eram fortemente acopladas ao Angular, de acordo com Ionic (2021), o uso dos padrões de projeto Angular foi o que tornou o Ionic uma excelente ferramenta. As versões mais recentes foram projetadas para funcionar como uma biblioteca de componentes *Web* autônoma possibilitando a integração com outros *frameworks* para front-end, incluindo o React[4] e Vue[5].

Angular é uma plataforma de desenvolvimento de aplicações web, construído em TypeScritp que contém uma estrutura baseada em componentes para construção de aplicativos escalonáveis. O angular fornece uma coleção de bibliotecas bem integradas que cobrem uma ampla variedade de recursos, incluindo roteamento de páginas, gerenciamento de formulários, comunicação cliente servidor e muito mais (ANGULAR, 2021).

O Angular possui uma estrutura básica que possui ao menos um componente que, conecta uma hierarquia de outros componentes que são responsáveis pela lógica

[4] **React** é uma biblioteca JavaScript para construção de interfaces de usuário
[5] Vue é uma **estrutura progressiva** para construir interfaces de usuário.

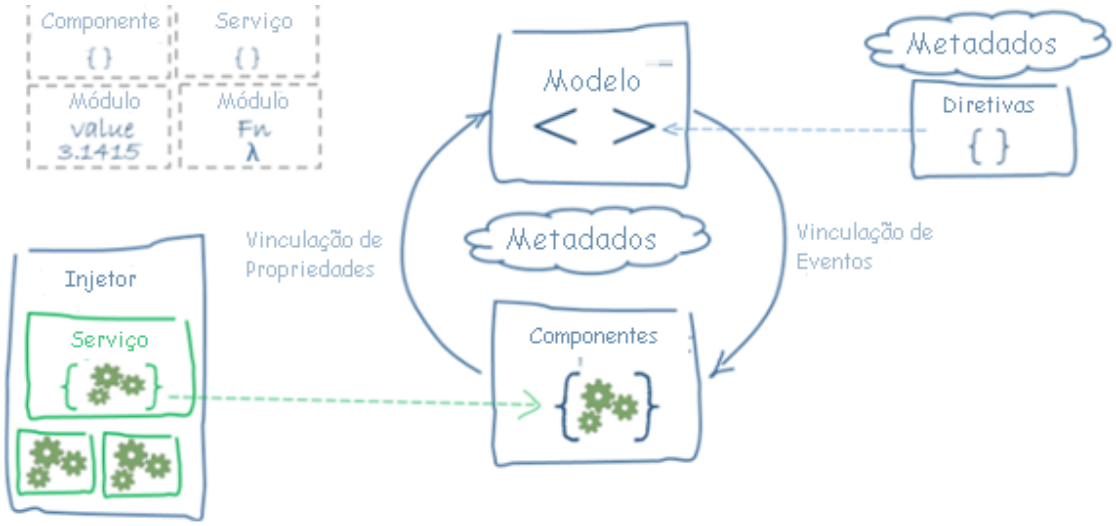
da aplicação, classe de dados e associados a um modelo HTML de visualização (ANGULAR IO, 2021). A ligação hierárquica entre componentes é feita através dos chamados decoradores, que fornecem os metadados específicos entre os componentes.

Os componentes de dados e lógica são alimentados pelas chamadas classes de serviço do angular, que tem com função por exemplo, buscar dados de um servidor ou validar entradas de dados do usuário. O angular diferencia componentes de serviços para aumentar a modularidade e capacidade de reutilização de código (ANGULAR. 2021).

A classes de serviços não estão associadas a nenhuma visualização HTML específica e podem ser compartilhadas entre os componentes com a chamada injeção de dependências (DI). A DI permite manter as classes de componentes enxutas e eficientes. Ao definir uma classe de serviço como injetável, ela será disponibilizada para qualquer componente da aplicação (ANGULAR, 2021).

A figura 4 representa a ligação entre os componentes hierárquicos que compõem uma aplicação angular:

Figura 4 - Relacionamento das peças que compõe o angular



Fonte: Angular IO (2021).

3.3 TypeScript

TypeScript (ts) é uma linguagem de código aberto baseadas em JavaScript, que é uma das ferramentas de plataforma cruzada mais difundidas do mundo

(TypeScript, 2021). De acordo com TypeScript (2021), o ts começou sua vida como uma tentativa de trazer os tipos tradicionais orientados a objeto para o JavaScript, com intuito de que os programadores pudessem trazer os programas tradicionais orientados a objetos para a *web.*

TypeScript é a linguagem primária para desenvolvimento de aplicações Angular. No entanto, os navegadores não podem executar diretamente o código ts, antes o código deve ser transformado para JavaScript usando o compilador TSC (TypeScript Compiler), gerando um código que roda em qualquer lugar que o JavaScript seja executado (ANGULAR IO e TYPESCRIPT, 2021).

A linguagem ts tem como característica ser um verificador do tipo estático para programas JavaScript, ou seja, diferentemente do código JavaScript, a verificação de erros é feita antes do programa ser executado. Por exemplo, o JavaScript oferece tipos primitivos como *string* e *number*, mas não verifica se os valores atribuídos estão condizentes com o tipo declarado da variável, mas o ts sim, e informa o erro antes da execução do programa, minimizando os erros indesejados durante a execução das aplicações *web (*IONIC IO, 2021*)*.

3.4 Capacitor

O Capacitor é um projeto de plataforma cruzada que facilita a construção de aplicativos modernos que são executados nativamente no IOS, Android e na *web.* O capacitor funciona como um container nativo para empacotar seus aplicativos JavaScript e implementar nas plataformas móveis e *desktops*. Ele fornece um conjunto de APIs de *plug-in* que permitem que um aplicativo fique o mais próximo possível dos padrões da *web* acessando recursos nativos das plataformas que o suportam. O Capacitor permite que aplicações funcionem igualmente bem no IOS, Android e como Apps Progressivos da *web.* Com ele é possível acessar SDKs nativos completos em cada plataforma e implantar diretamente seus aplicativos nas Apps *Stores* (CAPACITOR, 2021).

O Capacitor oferece uma variedade de *plug-ins* que permite que o JavaScript faça interface diretamente com as APIs nativas, fazendo tudo o que um aplicativo nativo tradicional pode fazer. Com os plug-ins do capacitor é possível acessar

recursos como câmera, armazenamento, GPS, sensores de movimento, acelerômetro, entre outros. (CAPACITORJS, 2021).

4 DESENVOLVIMENTO

Este capítulo mostra como a etapa de desenvolvimento foi realizada, iniciando pelo levantamento de requisitos do aplicativo, seguido da elaboração da documentação de caso de uso, diagrama de classe e modelo de entidade relacionamento para representar a estrutura lógica do banco de dados. Para o desenvolvimento da aplicação optou-se por utilizar o *framework* Ionic versão 6.16.3, com a linguagem *TypeScript* versão 4.2.4 e padrões de projeto Angular. O banco de dados utilizado foi o SQLite versão 3.12.1. O GitHub foi a plataforma usada para armazenamento, controle de versão e compartilhamento do código-fonte da aplicação. A plataforma inicialmente utilizada para o desenvolvimento foi exclusivamente o Android.

4.1 Definição dos Requisitos

A definição dos requisitos foi feita inicialmente com um especialista na área ambiental que tem experiência com o manejo, tratamento e disposição final de resíduos orgânicos. Neste caso, foi realizada uma análise do problema e identificado os seguintes requisitos:

- O sistema deve possuir um cadastro de culturas, no qual é possível informar os parâmetros necessários para o cálculo, sendo eles os valores recomendados de nitrogênio (N), fósforo (P) e potássio (K).
- O sistema deve permitir cadastrar a mesma cultura com diferentes níveis de interpretação de fósforo do solo.
- O sistema deve possuir um cadastro de tipo de resíduos no qual é possível informar os parâmetros necessários para o cálculo, sendo ele a concentração de nitrogênio, fósforo e potássio em quilos por metros cúbicos e a taxa de mineralização de cada nutriente.
- O sistema deve armazenar todos os parâmetros usados para efetuar o cálculo referentes à cultura e resíduo. No resultado deve ser incluído o campo ano para representar a safra.

- O sistema deve apresentar a quantidade necessária de resíduos orgânicos em metros cúbicos por hectares para líquidos e toneladas por hectare para sólidos.
- O sistema deve apresentar a diferença de N, P e K em relação a quantidade inicial exigida pela cultura com o resultado da recomendação de fertilizante orgânico obtido pelo cálculo.

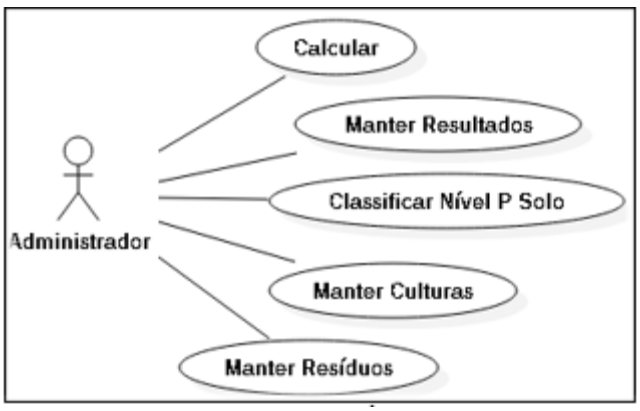
O sistema deve permitir alterar a quantidade recomendada de fertilizante orgânico, recalculando a diferença de N, P e K com a quantidade exigida pela cultura.

4.2 Diagrama de Casos de Uso

A Figura 5 demonstra o Diagrama de Casos de Uso do sistema. Neste existe um único ator principal nomeado como Administrador, que é o responsável por todas as operações do aplicativo. Os itens na figura são:

- Calcular: Efetuar o cálculo da quantidade de dejetos recomendada.
- Manter resultados: Cadastrar, atualizar e remover resultados do cálculo.
- Classificar nível fósforo do solo: Intepretação do nível de fósforo do solo.
- Manter cultura: Cadastrar, atualizar e remover informações sobre culturas.
- Manter Resíduos: Cadastrar, atualizar e remover informações sobre Resíduos.

Figura 5 – Diagrama de Caso de Uso

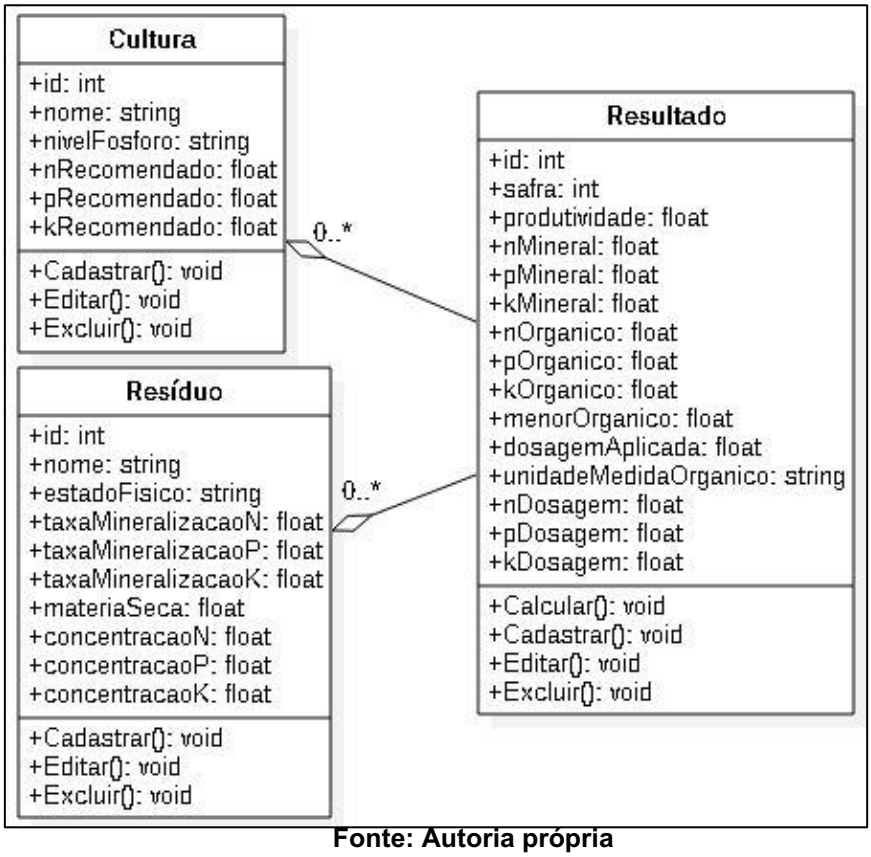


Fonte: Autoria própria (2021).

4.3 Diagrama de Classe

A Figura 6 representa o modelo estrutural do aplicativo através de um diagrama de classe, onde existem apenas três classes no sistema: Cultura, Resíduo e Resultado. A classe Resultado está associada com a classe Cultura e Resíduo, que são necessárias para efetuar o cálculo da recomendação de adubação orgânica e da complementação mineral.

Figura 6 - Diagrama de Classes



Fonte: Autoria própria

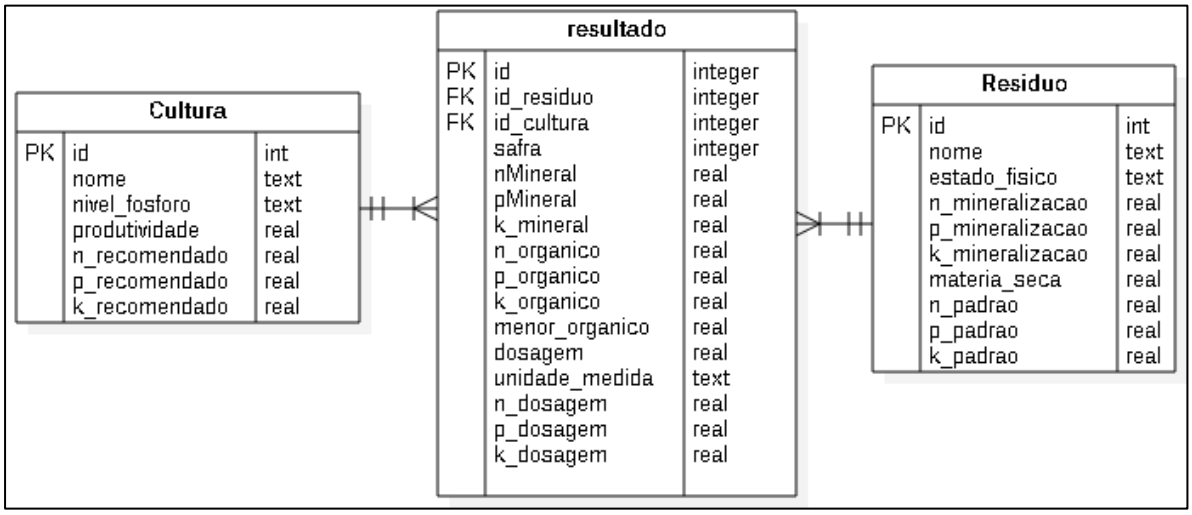
4.4 Modelo de banco de dados

O banco de dados do aplicativo possui três tabelas:

- Cultura: concentra as informações da exigência nutricional da cultura. É possível cadastrar a mesma cultura com diferentes interpretações de nível de fósforo do solo e diferentes exigências nutricionais de N, P e K.
- Resíduo: Armazena as informações do estado físico do resíduo, que é necessário para determinar qual fórmula será aplicada no cálculo.

- Resultado: centraliza todas as informações relevantes aos resultados do cálculo e demais informações relevantes ao cenário como a safra e produtividade esperada.

Figura 7 - Modelo entidade relacional



Fonte: Autoria própria

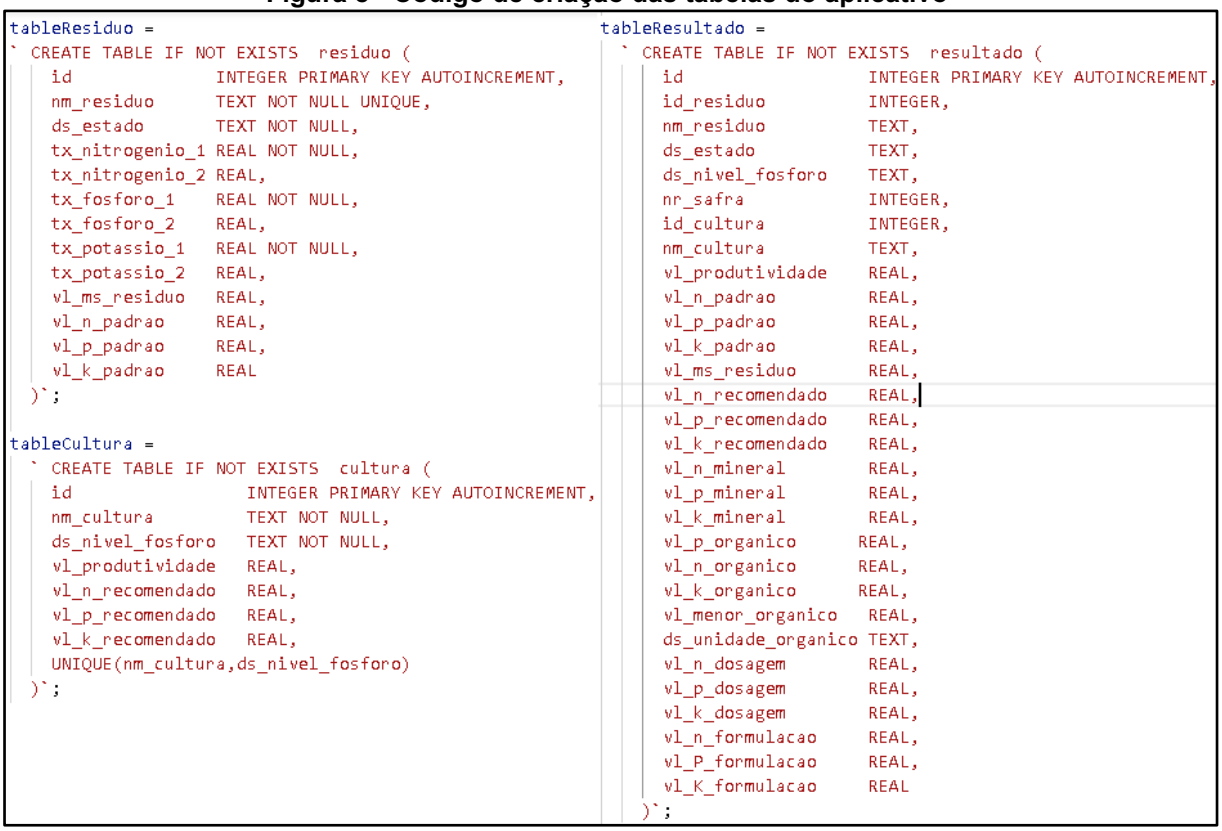
4.5 Configuração do banco de dados

Após a definição dos requisitos e a estrutura do banco de dados arquitetada, é iniciada a etapa de implementação do código fonte. Nesta seção serão exploradas as funcionalidades do aplicativo, demonstrando partes do código fonte e imagens das telas.

4.5.1 Banco de dados

Para gerenciar as operações de banco de dados foi criado um serviço chamado *database.service.ts,* que é responsável pela criação do esquema, tabelas e operações de inserção, atualização e remoção dos registros. O código de criação das tabelas é demonstrado na Figura 8, seguindo o modelo inicial proposto conforme o diagrama do Modelo de Entidade Relacional demonstrado na Figura 7, seção 4.4 do capítulo 4.

Figura 8 - Código de criação das tabelas do aplicativo



```
tableResiduo =
` CREATE TABLE IF NOT EXISTS  residuo (
    id              INTEGER PRIMARY KEY AUTOINCREMENT,
    nm_residuo      TEXT NOT NULL UNIQUE,
    ds_estado       TEXT NOT NULL,
    tx_nitrogenio_1 REAL NOT NULL,
    tx_nitrogenio_2 REAL,
    tx_fosforo_1    REAL NOT NULL,
    tx_fosforo_2    REAL,
    tx_potassio_1   REAL NOT NULL,
    tx_potassio_2   REAL,
    vl_ms_residuo   REAL,
    vl_n_padrao     REAL,
    vl_p_padrao     REAL,
    vl_k_padrao     REAL
  )`;

tableCultura =
  ` CREATE TABLE IF NOT EXISTS  cultura (
    id                  INTEGER PRIMARY KEY AUTOINCREMENT,
    nm_cultura          TEXT NOT NULL,
    ds_nivel_fosforo    TEXT NOT NULL,
    vl_produtividade    REAL,
    vl_n_recomendado    REAL,
    vl_p_recomendado    REAL,
    vl_k_recomendado    REAL,
    UNIQUE(nm_cultura,ds_nivel_fosforo)
  )`;

tableResultado =
  ` CREATE TABLE IF NOT EXISTS  resultado (
    id                  INTEGER PRIMARY KEY AUTOINCREMENT,
    id_residuo          INTEGER,
    nm_residuo          TEXT,
    ds_estado           TEXT,
    ds_nivel_fosforo    TEXT,
    nr_safra            INTEGER,
    id_cultura          INTEGER,
    nm_cultura          TEXT,
    vl_produtividade    REAL,
    vl_n_padrao         REAL,
    vl_p_padrao         REAL,
    vl_k_padrao         REAL,
    vl_ms_residuo       REAL,
    vl_n_recomendado    REAL,
    vl_p_recomendado    REAL,
    vl_k_recomendado    REAL,
    vl_n_mineral        REAL,
    vl_p_mineral        REAL,
    vl_k_mineral        REAL,
    vl_p_organico      REAL,
    vl_n_organico      REAL,
    vl_k_organico      REAL,
    vl_menor_organico   REAL,
    ds_unidade_organico TEXT,
    vl_n_dosagem        REAL,
    vl_p_dosagem        REAL,
    vl_k_dosagem        REAL,
    vl_n_formulacao     REAL,
    vl_P_formulacao     REAL,
    vl_K_formulacao     REAL
  )`;
```

Fonte: Autoria própria

Para utilização do *plug-in* do SQLite é necessário efetuar a importação da biblioteca *SQLite* que é responsável por fornecer o método de criação do banco de dados ejk da biblioteca SQLiteObject que fornece o método para execução dos comandos SQL de definição e manipulação de dados. Para utilizar os métodos é necessário efetuar a injeção da biblioteca no construtor do serviço, conforme Figura 9, linha 200. Em seguida é chamado o método *create()* passando dois parâmetros, nome do banco e local, conforme Figura 9, linhas de 202 a 204. Caso o banco seja criado com sucesso, é retornado um objeto do tipo SQLiteObject denominado db (linha 205), que será usado para chamar os métodos de criação de tabelas, inserção, atualização e remoção de registros.

Figura 9 - Criação do Banco de Dados

```
195   constructor(private plataform: Platform,
196                 private sqlite: SQLite) {
197     this.plataform.ready().then(() => {
198       this.sqlite.create({
199         name: 'dejetos.db',
200         location: 'default'
201       }).then((db: SQLiteObject) => {
202         this.database = db;
203         this.createTables(this.database);
204         this.dbReady.next(true);
205       });
206     });
207   }
```

Fonte: Autoria própria

A Figura 10 demonstra a criação das tabelas usando a instância do objeto do tipo SQLiteObject, resultante da criação bem-sucedida do esquema de dados do aplicativo. Em seguida é efetuada uma carga inicial de dados na base para fins de testes da aplicação.

Figura 10 - Comando de criação da tabela Resultado

```
213   createTables(db: SQLiteObject) {
214
215     db.executeSql(this.tableResiduo).then(() => {
216       console.log('tabela residuo criada');
217     });
218     db.executeSql(this.seedResiduo).then(() => {
219       console.log('insert residuo');
220     });
221     db.executeSql(this.tableCultura).then(() => {
222       console.log('tabela cultura criada');
223     });
224     db.executeSql(this.seedtCultura).then(() => {
225       console.log('insert cultura');
226     });
227     db.executeSql(this.tableResultado).then(() => {
228       console.log('insert Resultado');
229     });
230     db.executeSql(this.tableNivelP).then(() => {
231       console.log('table nivel p');
232     });
233     db.executeSql(this.insertNivelP).then(() => {
234       console.log('insert nivel p');
235     });
236   }
```

Fonte: Autoria própria

Após a criação da base de dados e carga inicial de registro foram criados os métodos de manipulação dos dados, a Figura 11 demonstra as operações de seleção, inserção, atualização e remoção da tabela Cultura. As operações com as demais tabelas (Resultado, Resíduo) seguem o mesmo padrão de código, cada uma com seus respectivos atributos.

Figura 11 - Métodos de operação da tabela Cultura

```
async selectCulturaById(row: any): Promise<any> {
  const data = await this.database.executeSql('select * from cultura where id = ? ', row);
  const cultura = [];
  if (data.rows.length > 0) {
    for (let i = 0; i < data.rows.length; i++) {
      cultura.push(data.rows.item(i));
    }
  }
  return cultura;
}

insertCultura(row: any) {
  const insert = `insert into cultura (
    nm_cultura,
    ds_nivel_fosforo,
    vl_produtividade,
    vl_n_recomendado,
    vl_p_recomendado,
    vl_k_recomendado
    ) values (?,?,?,?,?,?)`;
  return this.database.executeSql(insert, row);
}

updateCultura(row: any) {
  const update = `
  update cultura
      set nm_cultura = ?,
          ds_nivel_fosforo = ?,
          vl_produtividade = ?,
          vl_n_recomendado = ?,
          vl_p_recomendado = ?,
          vl_k_recomendado = ?
    where id = ?`;
  return this.database.executeSql(update, row);
}

deleteCulturaById(id: any): Promise<any> {
  return this.database.executeSql(`delete from cultura where id = ?`, [id]);
}
```

Fonte: autoria própria

4.6 IMPLEMENTAÇÃO DAS TELAS DO SISTEMA

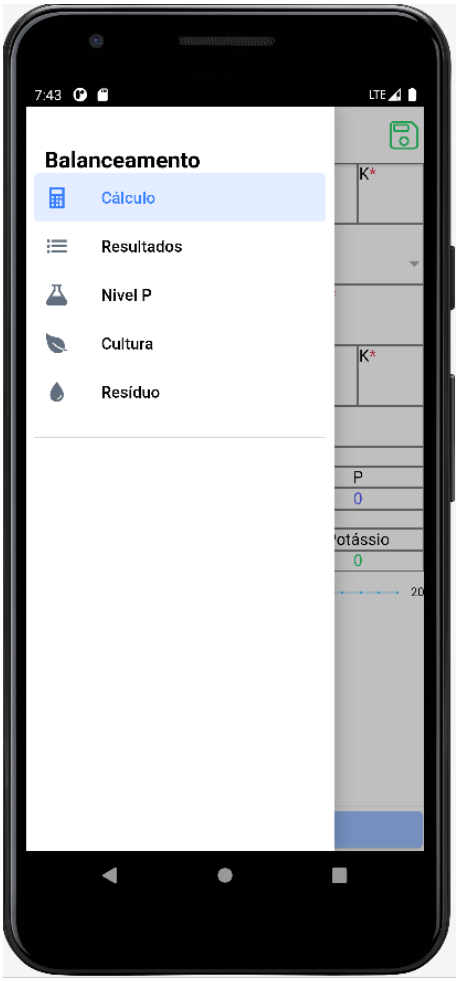
Como resultado do trabalho desenvolvido obteve- se um sistema composto por 7 (sete) telas:

- Tela para cálculo da dosagem recomendada;
- Tela para interpretação do nível de fósforo no solo;
- Tela de Resultados;
- Tela para listagem e remoção das culturas cadastradas por nível de fósforo;
- Tela para manutenção das Culturas;
- Tela para listagem e remoção dos resíduos cadastrados;
- Tela para manutenção de Resíduos;

4.6.1 Menu

O *laytout* da tela de menu selecionada para o sistema foi o Menu lateral, conforme demonstrado na Figura 12. Por padrão, o menu lateral é posicionado na esquerda, podendo ser alterado caso necessário. O menu permanece oculto e é ativado por um botão presente em todas as *interfaces,* sempre posicionado no cabeçalho da tela ao lado esquerdo.

Figura 12 - Menu Lateral



Fonte: Autoria própria

4.6.2 Tela de Interpretação de Nível de Fósforo

A tela de Interpretação do Nível de fósforo do solo é formada por dois parâmetros de entrada. O primeiro deles é a taxa de argila no solo, medida em gramas por quilogramas (g/kg) e possui três opções disponíveis:

- >400;
- 200-400;
- <250;

O segundo parâmetro é a quantidade de fósforo disponível em miligramas por decímetro cúbico (mg/dm³) e possui 5 opções disponíveis:

- <4;
- 4-8;
- 9-12;
- 13-18;
- >18;

A terceira informação é o resultado da interpretação dos parâmetros anteriores sendo elas:

- Muito baixo;
- Baixo;
- Médio;
- Alto;
- Muito Alto;

As informações da tela são baseadas na tabela de interpretação para fósforo disponível no solo para o estado do Paraná (SBCS-NEPAR, 2017). A Figura 13 apresenta a tela de interpretação de nível de fósforo do solo.

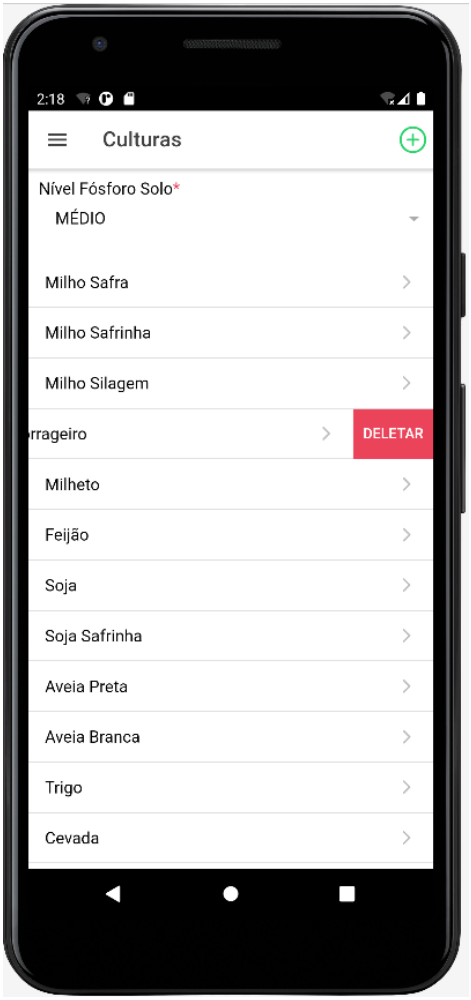
Figura 13 - Tela interpretação nível de fósforo

Fonte: Autoria própria (2021).

4.6.3 Listagem de Culturas

As culturas são classificadas pelo nível de fósforo no solo, podendo haver a mesma cultura cadastrada com diferentes níveis de fósforo. A tela mostra as culturas conforme o nível de fósforo selecionado pelo componente de seleção no topo da lista. A tela contém a opção para adicionar uma nova cultura por meio do botão posicionado no cabeçalho canto direito, editar uma cultura pressionando no item desejado e excluir uma cultura deslizando o item da direita para esquerda, revelando a opção deletar. A Figura 14 ilustra a tela responsável por listar as culturas cadastradas.

Figura 14 - Listagem de Culturas



Fonte: Autoria própria (2021).

4.6.4 Manutenção de Cultura

A tela de cadastro de culturas tem a finalidade de armazenar os parâmetros que serão usados na etapa do cálculo da recomendação, sedo que no momento que o usuário selecionar a cultura, os parâmetros serão automaticamente adicionados a fórmula da equação. A informação da produtividade é usada apenas para questões de referência, pois a exigência nutricional da cultura varia de acordo com o nível de fósforo no solo e a produtividade esperada. A Figura 15 apresenta a tela de cadastro de novas culturas.

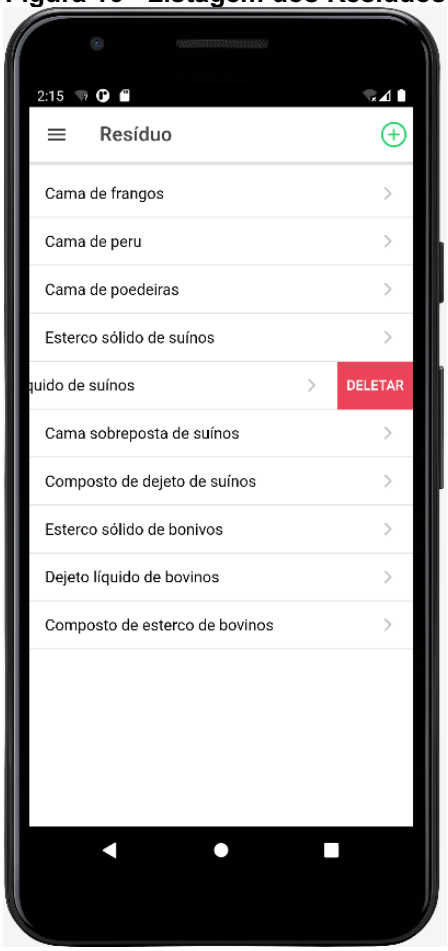
Figura 15 - Manutenção de Culturas

Fonte: Autoria própria (2021).

A tela de listagem mostra todos os resíduos cadastrados no sistema e permite editar os resíduos pressionando no item desejado, adicionar um novo resíduo pelo botão superior direito no cabeçalho da tela e excluir um resíduo deslizando o item da direta para a esquerda, revelando a opção de deletar o item, conforme mostra a Figura 16.

Fonte: Autoria própria (2021).

Figura 16 - Listagem dos Resíduos



4.6.5 Manter Resíduos

A tela de cadastro de resíduos tem a finalidade de armazenar os parâmetros que serão usados na etapa do cálculo da recomendação, sendo eles o estado físico do resíduo, necessário para determinar qual fórmula será aplicada no cálculo, a taxa de mineralização, a quantidade de N, P e K e a quantidade de matéria seca no caso de resíduos sólidos. Os parâmetros serão automaticamente adicionados à fórmula na tela de cálculo após o usuário selecionar o resíduo desejado. A Figura 17 apresenta a tela de cadastro de resíduo.

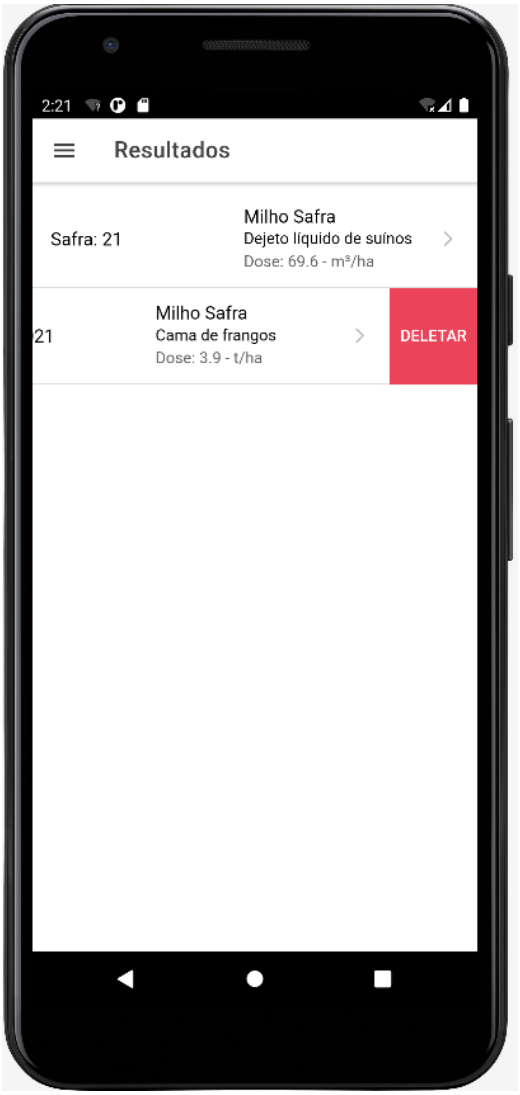
Figura 17 - Manter Resíduo

Fonte: Autoria própria (2021).

4.6.6 Lista de resultados

A lista apresenta todos os resultados que foram armazenados após o cálculo da recomendação de dosagem orgânica. É possível editar as informações clicando no item desejado, escolhendo para remover o resultado da mesma forma que as listagem de cultura e resíduo. A tela de resultados apresenta as informações da safra, cultura, tipo de resíduo e a dose aplicada. A Figura 18 exibe a lista dos resultados armazenados.

Figura 18 – Lista de Resultados



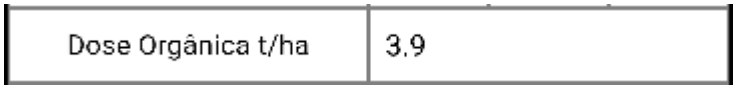
Fonte: Autoria própria (2021).

4.6.7 Cálculo

Para iniciar o cálculo da dosagem recomendada é necessário selecionar o resíduo que será utilizado como fertilizante orgânico previamente cadastrados na tela de manutenção de resíduos. Ao selecionar o resíduo as informações do estado físico, taxa de mineralização e composição química são automaticamente adicionadas à fórmula. A safra deve ser informada por questões de rastreabilidade, em seguida deve ser selecionado o nível de P do solo, que servirá de parâmetro para filtrar as culturas. Após a escolha do nível de fósforo selecionado, seguimos com a escolha da cultura, e da mesma forma que o resíduo, os parâmetros referentes a exigência nutricional de N, P e K são automaticamente adicionados à fórmula.

Após todos os parâmetros selecionados o botão calcular é habilitado e o cálculo pode ser efetuado. O primeiro resultado é a Dose Orgânica (baseada no princípio da menor dosagem, como visto no capítulo 2) que é representada em toneladas por hectares para resíduos sólidos e metros cúbicos por hectare para líquidos conforme exemplo da Figura 19.

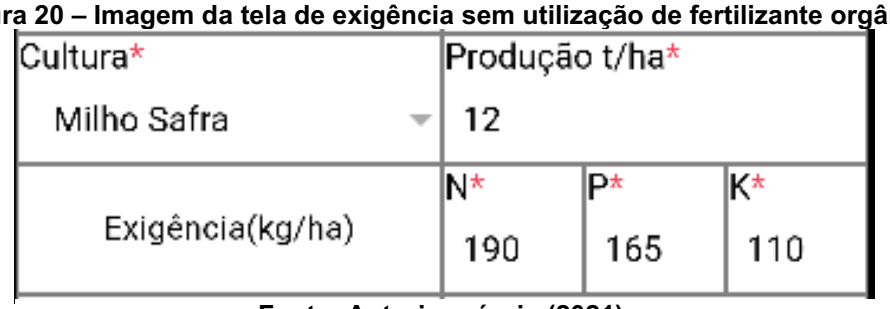
Figura 19 – Imagem da tela de resultado da recomendação orgânica para sólidos



Fonte: Autoria própria (2021).

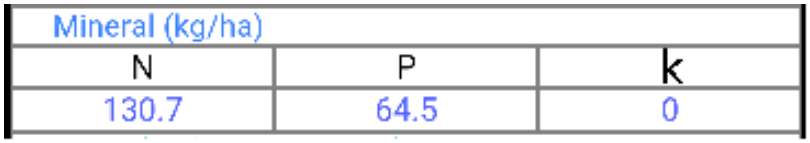
O segundo resultado apresentado é a complementação mineral necessária para suprir a demanda nutricional da cultura selecionada, que é a diferença da exigência inicial (Figura 20) com a proporção de N, P e K contida na menor dosagem recomendada pelo sistema, ou seja, 3.9 t/há, conforme exemplo da Figura 19. A Figura 21 demonstra o resultado da diferença.

Figura 20 – Imagem da tela de exigência sem utilização de fertilizante orgânico



Fonte: Autoria própria (2021).

Figura 21 – Imagem da tela de exigência final utilizando a dose recomendada

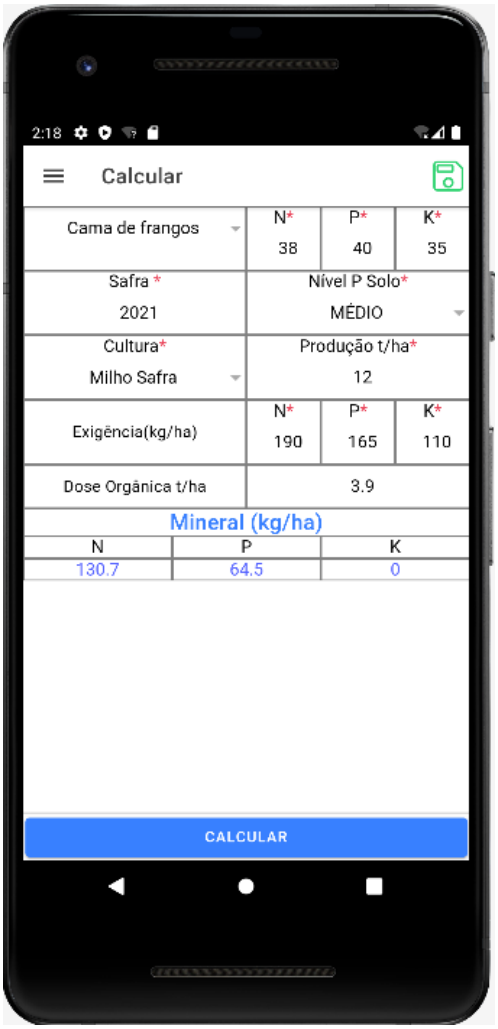


Fonte: Autoria própria (2021).

A Figura 19 mostra que ao utilizar a dose recomendada de 3.9 t/ha de resíduos é atendido toda demanda nutricional de potássio (K), 59.3 dos 190 kg/ha de nitrogênio (N) e 100.5 dos 165 kg/ha de fósforo (P) exigidos pela cultura, resultando em uma complementação mineral de N, P e K de 130.7 – 64.5 – 0 kg/ha.

A Figura 22 mostra a tela completa de cálculo com os resultados obtidos.

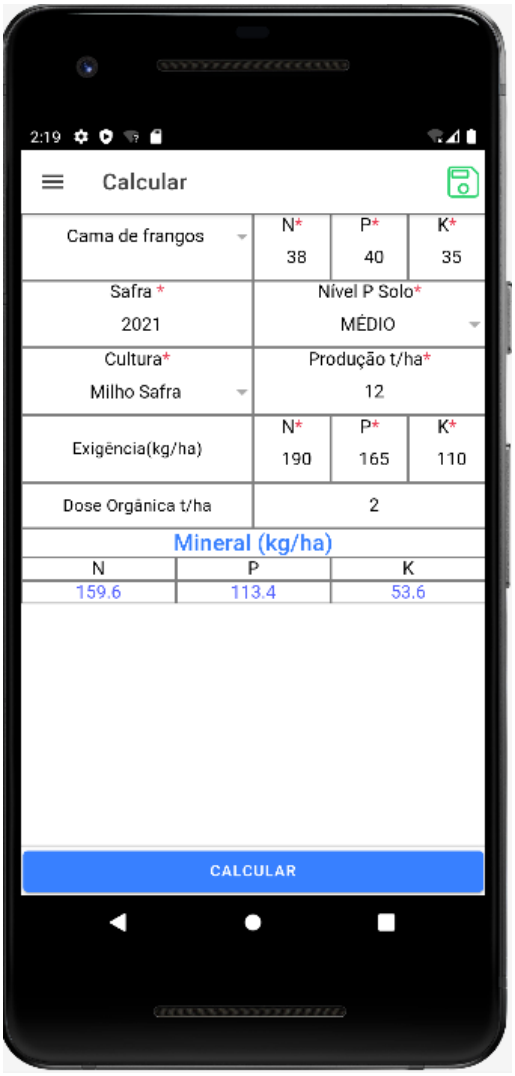
Figura 22 - Tela de cálculo



Fonte: Autoria Própria (2021).

Após obter o resultado o aplicativo permite variar a dosagem orgânica recomendada, recalculando a diferença de complementação mineral sugerida. A Figura 23 mostra a variação da dosagem recomendada de 3.9 t/ha para 2 t/ha resultado em uma maior complementação mineral pela diminuição da dose de resíduo orgânico aplicado. Nota-se que o potássio (K) não foi completamente atendido com a diminuição da dosagem recomendada, sendo necessário uma complementação mineral.

Figura 23 - Variação da dosagem orgânica



Fonte: Autoria Própria (2021).

5 CONCLUSÃO

A vida do homem do campo pode ser auxiliada pela tecnologia. O desenvolvimento do aplicativo apresentado neste trabalho possibilitou a integração lavoura-pecuária por meio do auxílio no cálculo necessário para a aplicação de resíduos de animais no complemento na fertilização do solo. Permitindo assim a maximização na utilização dos insumos agrícolas com a reciclagem de nutrientes e reduzindo os custos de produção, proporcionando uma agricultura de precisão e cada vez mais sustentável. Isso também permite auxiliar o cumprimento das recomendações agronômicas vigentes e estabelecidas pelo Manual de Adubação e Calagem do Estado do Paraná

Os requisitos levantados para o aplicativo foram cumpridos, sendo o esmo capaz de cadastrar, atualizar e excluir culturas e dejetos. Permite ainda que os cálculos sejam realizados e mantidos no aplicativo para consultas posteriores.

O uso da metodologia ágil eXtreme Programming (XP) auxiliou a realização deste trabalho no prazo pois a concentração de esforços ocorreu no que é relevante e as funcionalidades foram implementadas de forma incremental, tornando o processo mais ágil e a utilização de *frameworks* contribuiu na produtividade pois permitiu manter o foco nas funcionalidades que o aplicativo é especialista.

O aplicativo desenvolvido permitiu acesso fácil as informações das culturas, resíduos e histórico dos cálculos realizados. Novos cálculos podem ser realizados auxiliando a análise e tomada de decisão sobre a utilização dos resíduos de animais levando em consideração todas as questões envolvidas no manejo de dejetos apresentadas durante o trabalho.

5.1 Trabalhos Futuros

Como trabalho futuros, destaca-se algumas funcionalidades que poderão ser desenvolvidas:

- Realização de *backup* das informações;
- Permitir que seja especificado o local, fazenda, talhão ou outra especificação que determine o local de aplicação dos resíduos.
- Registro das amostras de solo e de resíduos.

- Disponibilizar o aplicativo para uso em sistema operacional iOS.

REFERÊNCIAS

ALVES, R. G. C. de M. **Tratamento e valorização de dejetos da suinocultura através de processos anaeróbicos – operação e avaliação de diversos reatores em escala real**. Tese de Doutorado – Universidade Federal de Santa Catarina. Programa de Pós-graduação em Engenharia Ambiental. Florianópolis, 2007.

APPLE DEVELOPER, **Develop.** Disponível em https://developer.apple.com/develop/. Acesso em: 06 mai. 2021.

BECK, K. **Programação Extrema (XP) explicada: acolha as mudanças**. Porto Alegre: Bookman, 2004. 182 p.

BERNARDES, T. F; MIYAKE, M. Y. **Cross-platform Mobile Development Approaches: A Systematic Review.** 2016. Disponível em: http://www.revistaieeela.pea.usp.br/issues/vol14issue4April2016/14TLA4_51FreitasBernardes.pdf. Acesso em: 05 mai. 2021.

BEZERRA, P. T.; SCHIMIGUEL, J. **Desenvolvimento de aplicações mobile crossplatform utilizando phonegap**. 2016. Disponível em: https://www.eumed.net/cursecon/ecolat/br/16/phonegap.html. Acesso em: 13 jun. 2021.

CAPACITOR. Capacitor: Native progressive web Apps. 2021. Disponível em: https://capacitor.ionicframework.com/docs/. Acesso em: 06 mai. 2021.

DEVELOPER MICROSOFT, **Documentação.** Disponível em https://docs.microsoft.com/pt-br/. Acesso em: 06 mai. 2021.

DIÁRIO OFICIAL DO ESTADO DO PARANÁ. **Resolução SEDEST N° 52, de 15 de julho de 2019**, 2019.

DIESEL, R; MIRANDA, C. R; PERDOMO, C. C. **Coletânea de tecnologias sobre dejetos suínos.** Concórdia: Embrapa Suínos e Aves, 30 p., 2002.

EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA - EMBRAPA. **Sistema brasileiro de classificação dos solos.** Brasília, 1999. P. 412.

HEITKOTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. **A. Evaluating crossplatform development approaches for mobile applications.** 2013. p. 120–138. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-36608-6_8. Acesso em: 07 mai. 2021.

IBGE. Instituto Brasileiro de Geografia e Estatística. **Pesquisa da Pecuária Municipal, PPM: tabelas 2017.** Disponível em: https://www.ibge.gov.br/estatisticas/economicas/agricultura-e-pecuaria/9107-producao-da-pecuariamunicipal.html?edicao=22651&t=resultados. Acessado em: 05 abr. 2021.

IBGE. Instituto Brasileiro de Geografia e Estatística. **PIB cai 4,1% em 2020 e fecha o ano em R$ 7,4 trilhões.** Disponível em: https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/30165-pib-cai-4-1-em-2020-e-fecha-o-ano-em-r-7-4-trilhoes. Acessado em: 03 abr. 2021.

KONZEN, E. A. **Manejo e utilização dos dejetos de suínos.** Concórdia: EMBRAPA-CNPSA. Circular Técnica, 6, 1983.

KUNZ, A.; HIGARASHI, M. M.; OLIVEIRA P. A. **Tecnologias de Manejo e Tratamento de Dejetos de Suínos Estudadas no Brasil.** Cadernos de Ciência & Tecnologia, Brasília, 2005. v. 22, n. 3, p. 651-665.

KUNZ, A.; OLIVEIRA, P. A.; HIGARASHI, M. M.; SANGOI, V. **Recomendações técnicas para uso de esterqueiras para a armazenagem de dejetos de suínos.** Comunicado Técnico, Concórdia: Embrapa Suínos e Aves, n. 361, p. 1-4, 2004.

LIMA, W.A.; ARANHA, R.V.; RAIMANN, E.; CAMARGO, C.A.X.; INOCENCIO, A.C.G.; RIBEIRO, M.W.S. (2014). **Uma Proposta de Framework para o Auxílio na Criação de Serious Games.** In: XI WRVA. p. 126-131.

MORE, K. A.; CHANDRAN, M. P. **Native vs hybrid apps**. Proceeding of International Journal of Current Trends in Engineering & Research, p. 563–572, 2016. 7, 8, 9

NUNES, M. L. A. **Avaliação de procedimentos operacionais na compostagem de dejetos de suínos.** 2003. 117 f. Dissertação (Mestrado em Engenharia Ambiental) – Programa de Pós-Graduação em Engenharia Ambiental, Universidade Federal de Santa Catarina, Santa Catarina.

OLIVEIRA, P. A. V. **Manual de manejo e utilização dos dejetos de suínos.** EMBRAPA-CNPSA. Documentos, 27, Concordia SC, 1993. p. 188.
ANDROID DEVELOPERS, **Documentos.** Disponível em https:// https://developer.android.com/docs. Acesso em: 06 mai. 2021.

PRESSMAN, R. S. **Engenharia de Software: Uma abordagem Profissional.** 7. ed. Porto Alegre: Mcgraw-Hill, 2011. 780 p.

PREZOTTO, E. D.; BONIATI, B. B. **Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas.** IN: ENCONTRO ANUAL DE TECNOLOGIA DA INFORMAÇÃO (EATI), 2014, Frederico Westphalen. Anais... Frederico Westphalen, 2014. p. 72-79.

SQLITE.ORG **About SQLite.** Disponível em https://www.sqlite.org/about.html. Acesso em: 02 mai. 2021.

RAMOS, E. A. **Metodologias Ágeis: Extreme Programming.** Maringá: Universidade Estadual de Maringá, 2013. 42 p.

RIZZONI, L.B.; TOBIAS, A.C.T.; DEL BIANCHI, M.; GARCIA, J. A. D. **Biodigestão anaeróbia no tratamento de dejetos de suínos.** Revista Científica Eletrônica de Medicina Veterinária, v.9, n.18, p.1-20, 2012.

SBCS-NEPAR. Sociedade Brasileira de Ciência do Solo. Núcleo Estadual do paraná. **Manual de adubação e calagem para o estado do Paraná.** Curitiba: SBCS/NEPAR, 2017.

SCHERER, E. E.; AITA, C.; BALDISSERA, I. T. **Avaliação da qualidade do esterco líquido de suínos da região Oeste Catarinense para fins de utilização como fertilizante.** EPAGRI,1996.

SEGANFREDO, M. A. **A questão ambiental na utilização de dejetos de suínos como fertilizante do solo**. 2002. Concórdia: Embrapa/SC. Ministério da Agricultura e do abastecimento.

SILVA, E. P. A. da; SOTTO, E. C. S. A **Utilização do Ionic Framework no Desenvolvimento de Aplicações Híbridas em Arquitetura Orientada a Service.** Revista Interface Tecnológica, [S. l.], v. 15, n. 1, p. 97-108, 2018. DOI: 10.31510/infa.v15i1.333. Disponível em: https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/333. Acesso em: 2 mai. 2021.

SOMMERVILLE, Ian. **Engenharia de Software.** 8. ed. São Paulo: Pearson, 2007. 568 p.

TELES, Vinícius M. **Extreme Programming**. São Paulo: Novatec Editora, 2004.

TYPESCRIPT. **The JavaScript that scales**.Disponível em: https://www.typescriptlang.org. Acesso em: 06 mai. 2021.

VENDORS, R.. W.; RESOURCES. **Difference between native app, web app and hybrid app**. 2018. Disponível em: https://www.rfwireless-world.com/Terminology/Native-App-vs-Web-App-vs-Hybrid-App.html Acesso em: 06 mai. 2021.

VANDECANDELAERE, B. **Developing the UDUBS-IT platform as a hybrid app with the ionic framework.** Western Cape: [s.n.], 2015. Bacharelado em Novas Mídias e Tecnologia de Comunicação.

XANTHOPOULOS, S.; XINOGALOS, S. **A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications.** Proceedings of the 6th Balkan Conference in Informatics. ACM, 2013, p. 213-220.