INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS – *CAMPUS* BAMBUÍ
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Jean Gustavo Ferreira Rezende

DESENVOLVIMENTO DE UM APLICATIVO PARA RECOMENDAÇÃO DE CORRETIVOS E FERTILIZANTES COM BASE EM ANÁLISE DE SOLO

Bambuí - MG
2023

JEAN GUSTAVO FERREIRA REZENDE

DESENVOLVIMENTO DE UM APLICATIVO PARA RECOMENDAÇÃO DE CORRETIVOS E FERTILIZANTES COM BASE EM ANÁLISE DE SOLO

Trabalho de conclusão de curso apresentado ao Curso Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Bambuí para obtenção do grau de Bacharel em Engenharia de Computação.

Bambuí - MG
2023

Catalogação na Fonte Biblioteca IFMG - Campus Bambuí

R467d Rezende, Jean Gustavo Ferreira.
Desenvolvimento de um aplicativo para recomendação de corretivos e fertilizantes com base em análise de solo. / Jean Gustavo Ferreira Rezende. – 2023.
68 f. : il. ; color.

Orientador: Prof. Dr. Marcos Roberto Ribeiro.
Trabalho de Conclusão de Curso (graduação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Bambuí, MG, Curso Bacharelado em Engenharia de Computação, 2023.

1. Dispositivos móveis. 2. Cálculo de adubação. 3. Análise de solo. I. Ribeiro, Marcos Roberto. II. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Bambuí, MG. III. Título.

CDD 005.636

Elaborada por Douglas Bernardes de Castro- CRB-6/2802

Jean Gustavo Ferreira Rezende

DESENVOLVIMENTO DE UM APLICATIVO PARA RECOMENDAÇÃO DE CORRETIVOS E FERTILIZANTES COM BASE EM ANÁLISE DE SOLO

Trabalho de conclusão de curso apresentado ao Curso Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Bambuí para obtenção do grau de Bacharel em Engenharia de Computação.

Aprovado em 30 de novembro de 2023 pela banca examinadora:

Documento assinado eletronicamente por **Marcos Roberto Ribeiro**, **Professor**, em 30/11/2023, às 14:17, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Claudio Ribeiro de Sousa**, **Professor EBTT**, em 30/11/2023, às 14:17, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Itagildo Edmar Garbazza**, **Professor**, em 30/11/2023, às 14:18, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carlos Manoel de Oliveira**, **Professor**, em 30/11/2023, às 14:20, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **1750066** e o código CRC **6E1D39AD**.

Bambuí - MG
2023

RESUMO

Os gastos com fertilizantes impactam diretamente o custo de produção na agricultura. Assim, é importante utilizar tal insumo de forma racional para garantir a produtividade da lavoura sem aumentar o custo de produção. Considerando o crescente uso de ferramentas tecnológicas no setor rural, o objetivo do presente projeto foi desenvolver um aplicativo capaz de interpretar a análise de solo e recomendar as indicações de corretivos e fertilizantes para o cultivo das principais culturas anuais do Brasil. O desenvolvimento do aplicativo utilizou tecnologias multiplataforma, para que possa ser disponibilizado para diversos públicos. As metodologias utilizadas para as recomendações foram o método de saturação das bases, para a calagem; o teor de argila, para a gessagem; e as tabelas de extração, para a adubação. Como resultado, o presente trabalho criou um banco de dados com tabelas de extração de nutrientes, que possibilita a inclusão de outras culturas no futuro. O aplicativo gerado tem potencial para trazer benefícios significativos para a agricultura a um custo acessível.

Palavras-chave: Dispositivos Móveis. Cálculo de Adubação. Análise de Solo.

ABSTRACT

Fertilizer expenses directly impact the cost of production in agriculture. Therefore, it is important to use this input rationally to guarantee crop productivity without increasing production costs. Considering the growing use of technological tools in the rural sector, this project aimed to develop an application capable of interpreting soil analysis and recommending corrective and fertilizer recommendations for the cultivation of the main annual crops in Brazil. The development of the application used multiplatform technologies, so that it can be made available to several audiences. The methodologies used for the recommendations were the base saturation method for liming, the clay content for gypsum and the extraction tables for fertilization. As a result, this work created a database with nutrient extraction tables and the possibility of including other crops in the future. The generated application has the potential to bring significant benefits to agriculture at an affordable cost.

Keywords: Mobile devices. Fertilization Calculation. Soil Analysis.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Segundo a Companhia Nacional de Abastecimento (CONAB), a produção brasileira de grãos, safra 2022/23, teve volume estimado em 322,5 milhões de toneladas, com previsão de 317,5 milhões para a safra de 2023/24 (CONAB, 2023a). Além disso, o Brasil é o quarto maior produtor mundial de grãos (FAO, 2020).

Para o ganho em produtividade, é imprescindível a manutenção dos investimentos nas áreas de produção. Dentre os insumos considerados essenciais na prática agrícola, a utilização correta de corretivos e fertilizantes se torna cada vez mais importante. O consumo de fertilizantes no Brasil nunca se apresentou tão alto, e os dados de importação comprovam a demanda recorde. De janeiro a agosto de 2021, o Brasil importou o equivalente a US$7,46 bilhões em fertilizantes, 49,1% a mais quando comparado com as importações do mesmo período em 2020 (FORMIGONI, 2021).

Com a desvalorização da moeda nacional frente ao dólar, aliada à crise sanitária da COVID-19 e ao aumento da importação dos fertilizantes, o custo por hectare tem elevado ano após ano. De maneira geral, os gastos com fertilizantes ultrapassam os 20% do custo total por hectare. Merecem destaque os custos de 21,70% com fertilizantes para se produzir algodão em Barreiras/BA; de 24,90%, para a produção de milho em Sorriso/MT; e de 19,97%, para se produzir café em Guaxupé/MG (CONAB, 2023b).

Aliar as expectativas anuais de produtividade com a redução dos custos por hectare se torna, portanto, a premissa básica para a rentabilidade nas áreas de produção. O uso racional dos corretivos, gesso agrícola e fertilizantes se apresenta como o ponto de partida para este estudo. A amostragem, a análise do solo e o cálculo de correção, feitos incorretamente, podem gerar aumento nos custos, devido à utilização excessiva e elevação nos gastos, ou à subutilização, levando à redução de produtividade (CRAVO; VIEGAS; BRASIL, 2020).

A análise de solo possibilita avaliar o nível de nutrientes e grau de fertilidade do solo, e, por meio de interpretações e cálculos, profissionais como agrônomos e técnicos podem definir uma recomendação adequada de corretivos e fertilizantes (SILVA, 2009). Assim, acontecem a correção do pH, a neutralização do alumínio e o fornecimento balanceado de macro e micronutrientes para o desenvolvimento adequado da cultura (ALVARES *et al.* 1999).

O setor rural é favorável às tecnologias e tem cada vez mais adeptos. Isso inclui aplicativos, que são ferramentas úteis no ambiente rural para desempenhar inúmeras atividades no campo, com o propósito de facilitar e agilizar os processos de gestão das fazendas (AIBA, 2018).

O presente trabalho teve como objetivo desenvolver um aplicativo para auxiliar produtores e profissionais no cálculo de corretivos e adubação. O desenvolvimento levou em consideração as técnicas mais recomendadas de cálculo de corretivos e fertilizantes para as mais relevantes culturas anuais do País. Além disso, o aplicativo foi implementado com

tecnologias que permitem a execução multiplataforma nos principais sistemas operacionais e dispositivos móveis.

1.1 Objetivo geral

O objetivo geral do presente trabalho foi projetar e desenvolver um aplicativo que, por meio das análises de solo, automatize o cálculo de corretivos e adubação para as principais culturas anuais do País, para proporcionar mobilidade e praticidade ao produtor ou agrônomo.

1.2 Objetivos específicos

Os objetivos específicos do presente trabalho foram os seguintes:

- analisar e selecionar as principais culturas anuais cultivadas no País, bem como as principais técnicas de recomendação de corretivos e fertilizantes para elas;
- modelar e criar o banco de dados para cadastro das informações usadas nos cálculo;
- desenvolver um aplicativo que atenda às demandas do mercado em nutrição do solo, com implementação em multiplataformas para dispositivos móveis.

1.3 Justificativa

Apesar de haver vários métodos para se calcular esses insumos, em sua maioria, são demorados e exigem conhecimento técnico. A importância do dimensionamento correto de fertilizantes, por si só, já identifica um setor crítico, que pode ser auxiliado pela modernização, assim como as inúmeras ferramentas tecnológicas que beneficiam o ramo rural. Suprir essa demanda exige automatização e contribuição entre as áreas de computação e agronomia.

Este aplicativo tem o potencial de mitigar a ocorrência de deficiências nutricionais nas plantas e prover confiança de que os cálculos são baseados nas técnicas mais modernas.

Por meio deste aplicativo, os profissionais do setor agrícola e produtores têm à disposição uma ferramenta que lhes permite gerenciar a nutrição do solo das diferentes áreas de cultivo sob sua supervisão. Além disso, podem automatizar tarefas demoradas ao inserir dados de análise de solo e receber interpretação e recomendações apropriadas.

2 REFERENCIAL TEÓRICO

Este capítulo descreve os fundamentos teóricos e estudos correlatos que nortearam o trabalho. A Seção 2.1 apresenta conceitos que envolvem a análise de solo e a sua importância; a Seção 2.2 expõe as vantagens e como calcular os corretivos necessários ao solo. Em seguida, a Seção 2.3 explica como é o processo de adubação e formas de utilizá-la, e a Seção 2.4 aborda o desenvolvimento multiplataforma para dispositivos móveis. Por fim, a Seção 2.6 apresenta estudos ligados ao mesmo tema de pesquisa deste trabalho e que contribuíram para o seu desenvolvimento.

2.1 Análise de solo

A análise de solo é essencial para reconhecer e avaliar o grau de deficiência nutricional que este apresenta. De posse do resultado de uma análise de solo e de tabelas de interpretação de sua fertilidade, podem-se determinar as quantidades adequadas de corretivos e fertilizantes a serem aplicados no cultivo de uma cultura. Dessa forma, o produtor tem mais garantias de boa produtividade na agricultura, com melhor rendimento econômico. (SILVA, 2009; FREITAS *et al.* 2018).

Tomando a cultura do milho como exemplo, existem padrões de extração e exportação de nutrientes durante o seu ciclo de crescimento. Com a análise de solo, observando esses padrões, é possível dimensionar a produtividade esperada e quanto de fertilizante é necessário para alcançá-la (RESENDE; SILVA *et al.* 2016). Mesmo em solos com alta disponibilidade de nutrientes, a reposição destes se torna necessária para manutenção da qualidade do terreno (RESENDE; GUTIERREZ *et al.* 2016).

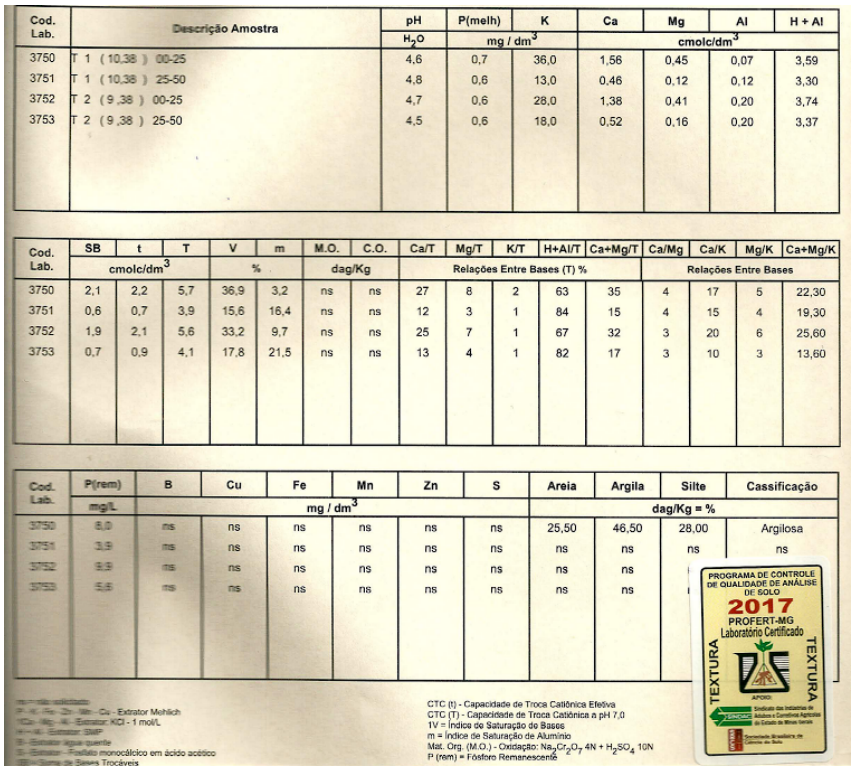
Para o cultivo em determinado local, é importante que haja a análise do solo. O primeiro passo é subdividir áreas maiores que 10 hectares (ha) ou ao se notar grande diferença nas propriedades físicas do solo. Essas subdivisões são chamadas de talhões ou glebas (ALVARES *et al.* 1999).

De cada talhão, são retiradas 20 a 30 amostras simples, que são unidas e misturadas para se formar uma amostra composta. A profundidade da retirada das amostras é definida pelo tipo de planta a ser cultivada. Essa informação é essencial, pois cada espécie faz a extração de nutrientes em diferentes camadas do solo. A coleta é enviada para um laboratório de solos, que determina as propriedades químicas e físicas (ALVARES *et al.* 1999). Um relatório, como mostrado na Figura 1, é emitido e retornado para se prosseguir com o processo de nutrição do solo.

2.2 Corretivos

Os processos de aplicação de corretivos têm como objetivo geral a neutralização ou diminuição de acidez no solo, por meio da aplicação de substâncias alcalinas. Há

Figura 1 – Relatório da análise de solo



Cod. Lab.	Descrição Amostra	pH	P(melh)	K	Ca	Mg	Al	H + Al
		H_2O	mg / dm^3		cmolc/dm^3			
3750	T 1 (10,38) 00-25	4,6	0,7	36,0	1,56	0,45	0,07	3,59
3751	T 1 (10,38) 25-50	4,8	0,6	13,0	0,46	0,12	0,12	3,30
3752	T 2 (9 ,38) 00-25	4,7	0,6	28,0	1,38	0,41	0,20	3,74
3753	T 2 (9 ,38) 25-50	4,5	0,6	18,0	0,52	0,16	0,20	3,37

Cod. Lab.	SB	t	T	V	m	M.O.	C.O.	Ca/T	Mg/T	K/T	H+Al/T	Ca+Mg/T	Ca/Mg	Ca/K	Mg/K	Ca+Mg/K
	cmolc/dm^3			%		dag/Kg		Relações Entre Bases (T) %					Relações Entre Bases			
3750	2,1	2,2	5,7	36,9	3,2	ns	ns	27	8	2	63	35	4	17	5	22,30
3751	0,6	0,7	3,9	15,6	16,4	ns	ns	12	3	1	84	15	4	15	4	19,30
3752	1,9	2,1	5,6	33,2	9,7	ns	ns	25	7	1	67	32	3	20	6	25,60
3753	0,7	0,9	4,1	17,8	21,5	ns	ns	13	4	1	82	17	3	10	3	13,60

Cod. Lab.	P(rem)	B	Cu	Fe	Mn	Zn	S	Areia	Argila	Silte	Cassificação
	mg/L	mg / dm^3						dag/Kg = %			
3750	8,0	ns	ns	ns	ns	ns	ns	25,50	46,50	28,00	Argilosa
3751	3,9	ns	ns	ns	ns	ns	ns	ns	ns	ns	ns
3752	9,9	ns	ns	ns	ns	ns	ns	ns	ns		
3753	5,6	ns	ns	ns	ns	ns	ns	ns	ns		

CTC (t) - Capacidade de Troca Catiônica Efetiva
CTC (T) - Capacidade de Troca Catiônica a pH 7,0
IV = Índice de Saturação de Bases
m = Índice de Saturação de Alumínio
Mat. Org. (M.O.) - Oxidação: $Na_2Cr_2O_7$ 4N + H_2SO_4 10N
P (rem) = Fósforo Remanescente

Fonte: LABORATÓRIO DE SOLOS, 2017.

variações na formulação de corretivos, sempre baseadas em cálcio e magnésio, que contam com diferentes poderes de neutralização e taxas de reatividade (ALCARDE, 1992).

As técnicas de correção comumente usadas na agricultura são a calagem e a gessagem. A principal diferença é a profundidade em que cada técnica reage: a calagem, mais superficial (até 30 cm), e a gessagem, mais profundamente (abaixo de 30 cm) (RAMOS *et al.* 2006). A Seção2.2.1 e a Seção 2.2.2 detalham a calagem e a gessagem, respectivamente.

2.2.1 Método de calagem

A calagem é adequada para corrigir a acidez do solo, reduzir ou neutralizar o alumínio, melhorar a fixação de nitrogênio, incentivar a atividade microbiana e aumentar a disponibilidade da maioria de nutrientes para a planta, além de fornecer cálcio (Ca) e magnésio (Mg). O produto utilizado para a calagem é o calcário (ALVARES *et al.* 1999; MARTHA JÚNIOR; VILELA; SOUSA, 2007).

O cálculo da calagem pode ser feito pelo Método da neutralização da acidez trocável e da elevação dos teores de Ca e de Mg trocáveis ou pelo Método da Saturação por Bases (ALVARES *et al.* 1999). De acordo com as orientações do profissional da área que acompanhou o presente trabalho, decidiu-se optar pelo uso do segundo método.

O método de saturação por bases considera a relação entre a acidez do solo, a saturação por bases (V), obtida na análise de solo, e o nível de pH adequado para a

cultura que será cultivada. Com a observação dos teores de Ca, Mg, K e Na e da acidez potencial H + Al, é possível utilizar o método da calagem para se alcançar o pH desejado (ALVARES *et al.* 1999).

A Equação (1) demonstra como o cálculo é feito. (T) representa o potencial de acidez (H + Al) em cmolc/dm³, (Va) é a saturação por bases atual do solo, e (Ve) é a saturação por bases desejada. O resultado da equação é a necessidade de calagem (NC) em toneladas por hectare (t/ha).

$$NC = \frac{T(Ve - Va)}{100} \tag{1}$$

Essa métrica (NC) indica a quantidade de $CaCO_3$ ou calcário com poder relativo neutralizante total (PRNT) de 100% utilizado por hectare. Esse índice corresponde à efetividade da reação do corretivo no solo (ALVARES *et al.* 1999). Na comercialização de corretivos, o PRNT é variável, com custos econômicos diferentes.

Portanto, para a aplicação e compra do calcário, deve ser considerada a porcentagem da superfície a ser coberta (SC) na calagem, a que profundidade (PF) de incorporação no solo (em cm) e o PRNT do calcário. A quantidade de calcário a ser utilizada (QC) em t/ha é obtida pela Equação (2).

$$QC = NC \times \frac{SC}{100} \times \frac{PF}{20} \times \frac{100}{PRNT} \tag{2}$$

Como exemplo, em um talhão de 10 hectares, a análise de até 25 cm retorna os valores 5,7 cmolc/dm³ e 36,9% de T e Va, respectivamente. Para o plantio de milho, Ve ideal é de 70%. A NC resulta em 1,89 t/ha.

$$NC = \frac{5,7(70 - 36,9)}{100} = 1,89t/ha \tag{3}$$

Para culturas anuais, a superfície coberta e a profundidade de incorporação no solo são consideradas totais. Portanto, apenas o PRNT do calcário contribui para o resultado. Considerando-o como 80%, obtemos, na Equação (4), que a quantidade necessária é de 2,36 t/ha.

$$QC = 1,89 \times \frac{100}{80} = 2,36t/ha \tag{4}$$

2.2.2 Método de gessagem

A gessagem é utilizada para corrigir a acidez subsuperficial, com potencial de reduzir a saturação de alumínio do solo, que tem efeitos tóxicos para as plantações. O gesso tem maior infiltração no solo por se dissolver na chuva, conseguindo alcançar camadas abaixo de 20 cm (ALVARES *et al.* 1999; MARTHA JÚNIOR; VILELA; SOUSA, 2007).

A aplicação de gesso na quantidade ideal pode melhorar o desenvolvimento radicular da planta em camadas mais profundas, proporcionando mais resistência a tempos de seca e crescimento por maior período de tempo. Também possibilita que os nutrientes sejam absorvidos em maiores quantidades e com maior eficiência (MARTHA JÚNIOR; VILELA; SOUSA, 2007).

A quantidade de gesso a ser aplicada pode ser calculada com base no teor de argila presente no solo, saturação de alumínio ou com base na determinação de fósforo remanescente (ALVARES *et al.* 1999; MARTHA JÚNIOR; VILELA; SOUSA, 2007).

No presente trabalho, decidiu-se, por orientações, calcular a necessidade de gessagem (NG) pelo método de teor de argila seguindo a Tabela 1.

Tabela 1 – Necessidade de gesso de acordo com o teor de argila da camada subsuperficial de 20 cm de espessura

Argila (%)	Necessidade de gessagem (t/ha)
0 a 15	0,0 a 0,4
15 a 35	0,4 a 0,8
35 a 60	0,8 a 1,2
60 a 100	1,2 a 1,6

Fonte: ALVARES *et al.* 1999.

A Equação (5) pode calcular a quantidade exata de gessagem NG, observando em qual categoria a porcentagem de argila se encontra. Kf corresponde à porcentagem final; Ki, à inicial; e K, à porcentagem de argila proveniente do resultado da análise de solo. NGf corresponde à quantidade de NG final, e NGi, à quantidade inicial em t/ha.

$$NG = NGi + \frac{(K - Ki) \times (NGf - NGi)}{(Kf - Ki)} \tag{5}$$

Por exemplo, a quantidade ideal de argila para o milho é de 35% K. Em um talhão com 46,5% de argila, os dados da tabela são 35 Ki, 60 Kf, 0,8 NGi, 1,2 NGf.

$$NG = 0,8 + \frac{(46,5 - 35) \times (1,2 - 0,8)}{(60 - 35)} = 0,984t/ha \tag{6}$$

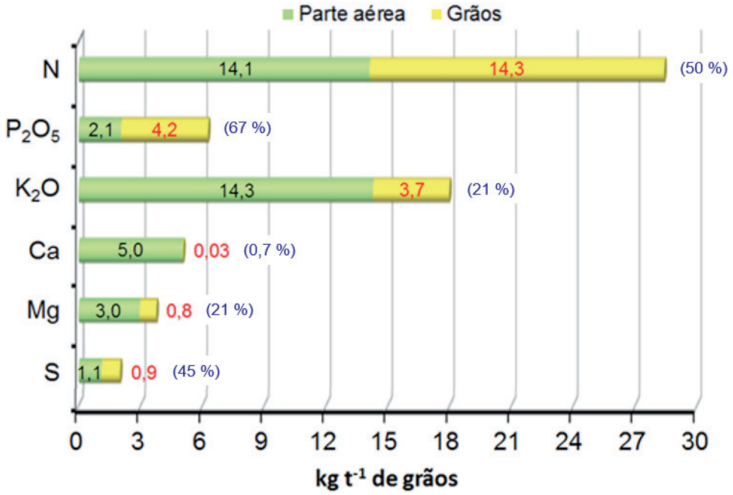
2.3 Adubação

A adubação tem como objetivo o aumento da produtividade da planta. Pode ser obtida por meio de fertilizantes nitrogenados, trazendo benefícios para todos os ciclos do cultivo (MARTHA JÚNIOR; VILELA; SOUSA, 2007).

Para quantificar o requerimento nutricional, foi usada como base a pesquisa de Resende, Silva *et al.* (2016), que realizou experimentos em diferentes híbridos modernos de milho e análises de solo. A Figura 2 mostra graficamente a quantidade de macronutrientes presentes na parte aérea e grãos no milho, referente à média desses híbridos; já a Figura 3 exibe a quantidade de micronutrientes.

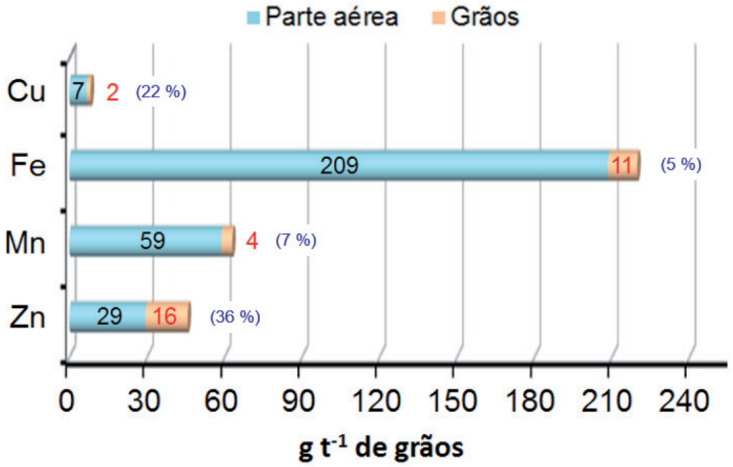
A produtividade retrata a máxima capacidade de produção destes híbridos em um cenário ideal. Esses dados são essenciais para o planejamento financeiro de uma lavoura, já que representam o resultado de uma plantação bem adubada antes mesmo do plantio (RESENDE; GUTIERREZ *et al.* 2016).

Figura 2 – Macronutrientes na parte aérea e grãos, para cada tonelada de milho produzida



Fonte: RESENDE; SILVA *et al.* 2016.

Figura 3 – Micronutrientes na parte aérea e grãos, para cada tonelada de milho produzida



Fonte: RESENDE; SILVA *et al.* 2016.

No aplicativo, para o cálculo de adubo requisitado pelo solo, foi utilizado o método por extração, por ser mais moderno. Esse método consiste em extensas análises feitas pela comunidade científica, que identificam o potencial de exportação de nutrientes pela planta. Para o milho, essas análises resultaram na Tabela 2, que especifica a quantidade de nutrientes extraídos do solo por 4 híbridos diferentes. Desses dados, determina-se sua média, a qual é empregada para representar a cultura do milho.

Unindo as informações de tamanho do talhão reservado para a cultura, a produção desejada e a quantidade de nutrientes disponíveis no solo, basta subtraí-las da média da tabela para se obter a recomendação de adubação. Essa metodologia garante

Tabela 2 – Exportação de nutrientes (parte aérea + grãos) por híbridos de milho. Média de cultivos em ambientes com médio e alto investimento em adubação

Híbrido	Nutriente									
	N	P_2O_5	K_2O	Ca	Mg	S	Cu	Fe	Mn	Zn
	kg/ha						g/ha			
AG 8088 PRO X	148	41	35	0,3	8	9	17	110	42	183
DKB 310 PRO 2	187	51	46	0,3	10	12	20	116	60	209
DKB 390 PRO	155	47	38	0,4	9	9	27	118	51	169
P 30F53 YH	132	43	36	0,2	8	9	13	123	37	167
BRS 1040	140	41	36	0,4	8	8	12	131	45	161
1| 873	125	37	36	0,4	7	8	11	91	33	122
Média	148	43	38	0,4	8	9	17	115	45	169

Fonte: RESENDE; SILVA *et al.* 2016.

que a planta terá a quantidade desejável de adubo para suprir a necessidade nutricional do cultivo.

A Tabela 3 apresenta as principais culturas do País, com as quantidades de nutrientes extraídas pela planta para uma produção de uma tonelada (t) por hectare (ha).

Tabela 3 – Extração de nutrientes do solo pelas principais culturas anuais do Brasil por tonelada de grãos produzida

Cultura	Nutriente										
	N	P_2O_5	K_2O	Ca	Mg	S	B	Cu	Fe	Mn	Zn
	kg/ha						g/ha				
Algodão	70,0	26	73,0	26,0	16,0	6,0	120,0	34,8	840,0	73,1	54,8
Feijão	87,0	17,9	94,8	65,0	26,0	66,0	56,0	17,0	366,1	149,0	42,0
Soja	91,2	21,8	52,0	20,5	10,5	8,2	64,8	34,9	450,4	131,4	74,7
Trigo	33,0	12,0	28,9	2,4	4,0	4,0	41,0	6,2	308,1	132,0	44,0
Arroz sequeiro	47,0	17,2	40,8	5,5	4,5	2,5	15,0	23,0	1043,0	377,0	96,0
Arroz irrigado	20,0	10,3	40,8	5,3	3,2	2,5	15,0	32,0	788,0	718,0	113,0
Milho grão	28,4	6,3	18,0	5,0	3,8	2,0	15,8	9,0	220,0	63,0	45,0
Milho silagem	21,5	8,9	20,5	2,4	2,8	2,6	19,3	6,8	161,0	37,4	37,5

Fonte: Compilado de MOSAIC FERTILIZANTES, 2023; BORIN; FERREIRA; CARVALHO, 2014; RESENDE; SILVA *et al.* 2016; FAGERIA *et al.* 1995.

Como exemplo, considere uma lavoura de milho para grãos(1) em um talhão com 10 hectares, cuja análise de solo informa 0,7 mg/dm^3 de fósforo (P), que equivale a 1,4 kg/ha de P. Como a tabela de extração indica a quantidade em P_2O_5, torna-se necessária a conversão de P para P_2O_5, multiplicando o valor de P por 2,29, totalizando 3,2 kg/ha de P_2O_5.

Considerando-se a necessidade de 6,3 kg/ha de P2O5, pela tabela 3 de extração, ao subtrair da quantidade presente no solo (3,2 kg/ha), serão necessários 3,1 kg/ha para

produção de uma tonelada de milho em grão, e, para uma produtividade de 9,0 t/ha, 27,9 kg de P2O5 por hectare. Caso o fertilizante adquirido pelo produtor seja o superfosfato simples, com 20% de P2O5, necessita-se da aquisição de 140 kg de superfosfato simples por hectare. Cada 10 ha cultivados resultam em cerca de 279 kg/ha de P2O5 na adubação de semeadura. Se o fertilizante comprado tiver 20% de P2O5 em sua composição, será necessária 1,39 tonelada.

2.4 Desenvolvimento multiplataforma

Os dispositivos móveis passaram a ter grande relevância no dia a dia das pessoas devido à sua versatilidade. O desenvolvimento para essas plataformas é um caso especial, sendo necessário considerar o ciclo de desenvolvimento rápido, a fragmentação de configurações dos dispositivos, como o tamanho da tela, interface de usuário de cada fabricante, navegação, segurança e privacidade (EL-KASSAS *et al.* 2015).

Mesmo com a existência de vários sistemas operacionais para dispositivos móveis, o Android e o iOS são os mais utilizados. De acordo com *International Data Corporation* IDC (2023), o sistema operacional Android está disponível em 84,2% dos dispositivos móveis mundiais, enquanto o iOS possui 15,8% de presença no mercado. No desenvolvimento de *software*, é importante considerar métodos capazes de criar aplicações para diferentes plataformas, atingindo, assim, o maior número possível de usuários.

O desenvolvimento nativo de aplicações para determinado sistema operacional, geralmente, possui desempenho superior, além de proporcionar uma experiência de interface nativa para a plataforma de destino. Entretanto, exigem-se conhecimentos específicos para cada plataforma. Assim, portanto, desenvolvedores optam por *frameworks* multiplataformas, com o propósito de criar a aplicação somente uma vez, permitindo executá-la nos diferentes sistemas operacionais através de uma única base de código, abrangendo um maior número de dispositivos (EL-KASSAS *et al.* 2015; BIØRN-HANSEN; GRØNLI; GHINEA, 2018).

Segundo Jardim (2021), existem várias técnicas para o desenvolvimento *mobile* multiplataforma. As principais abordagens são WebApp, híbrido e *Cross-Compiled*. Considerando-se a parcela de utilização das plataformas Android e iOS, é interessante disponibilizar a aplicação em ambas. Portanto, a definição da ferramenta a ser utilizada no desenvolvimento, assim como as características de interface de usuário e navegação, devem ser assertivas (EL-KASSAS *et al.* 2015).

O *framework* de desenvolvimento de *software* livre e de código aberto Flutter[1], criado pela empresa Google, possibilita a criação de aplicativos para dispositivos móveis que utilizam os sistemas operacionais Android e iOS, além do desenvolvimento de *softwares* para Windows, Mac, Linux e *Web*, a partir de uma única base de código. Seu objetivo é

[1] https://flutter.dev/

permitir que os desenvolvedores criem aplicativos de alta performance com uma experiência unificada (FLUTTER, 2022).

No Flutter, os *widgets* são os blocos de construção básicos da interface do usuário. Esses blocos são expansíveis e customizáveis, adequando-se a qualquer projeto. O *layout* é simplificado e fácil de construir, com alta performance e possibilidades de otimização. O *framework* também acelera a programação e a correção de erros com o recurso *Hot Reload*, que compila apenas a parte de código modificada quase instantaneamente (FAYZULLAEV, 2018).

O Flutter utiliza o Dart[2] como linguagem de programação, o qual é uma linguagem fortemente tipada, orientada a objetos, também desenvolvida pela Google. A utilização dessa linguagem permite que o mesmo código seja empregado para gerar uma aplicação para as plataformas citadas anteriormente (FLUTTER, 2022).

2.5 Interface

Segundo o livro de Barbosa *et al.* (2021), as tecnologias digitais estão presentes na vida das pessoas e afetam direta ou indiretamente as suas ações. Ao procurar entender e melhorar a concepção, construção e inserção dessas tecnologias, podemos desenvolver sistemas que trazem benefícios para as pessoas, ao aumentar a produtividade, trazer bem-estar e satisfazendo suas necessidades e desejos.

Essa qualidade de uso também é benéfica para redução do número e gravidade de erros, do custo de treinamento, do custo de suporte técnico e para o aumento de vendas e fidelidade do cliente (BARBOSA *et al.* 2021).

Rocha e Baranauskas (2003) definem que o usuário deve ser apresentado com possibilidades claras de interação, decifrando o que fazer somente olhando (*affordance*), com flexibilidade para o acesso à informação e interação sem barreiras (acessibilidade). É importante destacar que, ao conjunto de atributos relacionados ao esforço necessário para um grupo de usuários ao utilizar um sistema interativo, dá-se o nome de usabilidade.

A organização das informações da interface deve manter consistência dos dados sendo exibidos, facilitar a assimilação, minimizar a carga de memória, manter compatibilidade entre a entrada de dados e conteúdo exibido, e flexibilizar o acesso e uso das informações (MACHADO NETO, 2013).

Para facilitar a avaliação de interfaces, cada autor define heurísticas que determinam a necessidade de certas características, sendo que a falta delas compromete a usabilidade do sistema. As heurísticas definidas por Barbosa *et al.* (2021) são:

- visibilidade do estado do sistema;
- correspondência entre o sistema e o mundo real;

[2] https://dart.dev/

- controle e liberdade do usuário;
- consistência e padronização;
- reconhecimento em vez de memorização;
- flexibilidade e eficiência de uso;
- projeto estético e minimalista;
- prevenção de erros;
- ajuda para os usuários reconhecerem, diagnosticarem e se recuperarem de erros;
- ajuda e documentação.

Dentre as diretrizes de *design* focadas em aparelhos móveis, apresentadas por Machado Neto (2013), destaca-se que a interface deve ser preenchida de cima para baixo, deve ser disponibilizado um caminho lógico, tornar a interação fácil e óbvia, facilitar a entrada de dados, trazer elementos realistas e conscientizar o usuário de qualquer ação.

2.6 Estado-da-arte

O trabalho de Tanaka *et al.* (2020) desenvolveu um aplicativo para dispositivos móveis que faz o cálculo de adubo e calagem para plantas medicinais. Utilizando ferramentas para desenvolvimento em alto nível e por meio de blocos, o código-fonte foi montado. O aplicativo funciona em Android e salva os dados no dispositivo em um arquivo *commum separated values* (CSV). A abordagem utilizada para o cálculo da calagem foi o método de saturação por bases, ou via elevação dos teores de Ca e Mg. Para o adubo, sem a consideração da análise de solo, uma fórmula generalizada foi utilizada.

Delvaux e Silveira (2021) elaboraram um aplicativo para recomendação de fertilizantes e corretivos. Focados no cultivo de milho em Minas Gerais, os pesquisadores aplicaram as equações de saturação por bases para o adubo, e a neutralização do alumínio, para os corretivos. O aplicativo foi criado na linguagem Java e exclusivamente para o sistema operacional Android.

Para computadores, utilizando-se o Microsoft Excel e Visual Basic para Aplicações (VBA), o trabalho de Gubiani *et al.* (2007) usou um banco de dados para geração de relatórios referentes à adubação e correção de acidez. Esse cálculo engloba culturas produtoras de grãos, hortaliças e forrageiras. O método empregado para adubação foi o NPK, que consiste em suprir Nitrogênio, Potássio e Fósforo do solo, e, para corretivos, utilizou-se o teor de argila no solo.

Foi desenvolvido por Longui e Vitória (2011) um sistema para computadores através da linguagem de programação Delphi. Esse *software* é focado na calagem e adubação

de pastagens e culturas forrageiras, com o objetivo de manutenção desse campo, visando manter uma quantidade saudável de nutrientes na dieta de ruminantes. O programa possui uma interface simples e direta, que recebe os laudos técnicos e fornece, com base nessas informações, as necessidades de nutrientes. O usuário cadastra o produtor, a propriedade, a análise química e granulométrica do solo. Então, através do método de alumínio trocável ou de saturação por bases, é calculada a necessidade de calagem, e as classificações são descritas na quinta aproximação, para ser estimada a adubação.

No artigo de Paula (2007), foram desenvolvidas, por meio de uma planilha eletrônica, funcionalidades que automatizam o cálculo de adubação no cultivo de melancias e melões. De acordo com a análise de solo, é gerado o resultado de nutrientes calculados utilizando-se o método NPK.

Visando à produção de tomates, o trabalho de Silva (2012) empregou as tecnologias Java e MySQL para produzir um sistema *web*, implementando a arquitetura cliente-servidor para utilização em qualquer navegador. O sistema inclui funcionalidades como o cadastro do produtor, propriedade, área cultivada e os dados da análise do solo, para se realizar o cálculo de calagem e adubação. Também conta com o cálculo de calcário necessário, com base no PRNT, dicas para observação de sinais na planta e emissão de quatro tipos de relatório.

No Congresso Brasileiro de Agroinformática, foi apresentado um projeto em desenvolvimento de Nascimento, Salame e Tavares (2015) já no estágio de resultados, com a produção de um aplicativo para recomendação de calagem e adubação para produção de mandioca no Amazonas. O maior foco é na conscientização da importância de realização da correção do solo. O aplicativo utiliza o método de recomendação NPK e conta com o planejamento de divisão de doses, para evitar lixiviação dos adubos e maximizar a absorção do solo.

O presente trabalho foi norteado pelo Delvaux e Silveira (2021), que, como os demais trabalhos, considerou a análise de solo e usou as abordagens de cálculo para corretivos e fertilizantes, com o devido embasamento científico. Foi considerada a geração de relatórios e banco de dados, assim como Silva (2012), para auxiliar na distribuição e retenção desse conhecimento.

O diferencial implementado neste projeto foi o desenvolvimento multiplataforma, para se alcançar maior público, e integração ao banco de dados, que permite a adição de mais culturas e atualização das tabelas de extração, ofertando uma gama maior de culturas, com uma combinação única das técnicas de cálculo. Além disso, a recomendação retorna o resultado de múltiplos talhões da safra ao mesmo tempo.

3 METODOLOGIA

Este capítulo descreve a metodologia empregada para o desenvolvimento do presente trabalho. A Seção 3.1 trata da classificação da pesquisa. Em seguida, a Seção 3.2 explica o problema e a solução proposta. Os materiais e tecnologias utilizados são expostos na Seção 3.3, e a Seção 3.4 explica os métodos e procedimentos adotados.

3.1 Classificação da pesquisa

A classificação da pesquisa está intimamente ligada ao objetivo do trabalho, pois guia os procedimentos, de maneira sistemática e racional, assim como as atividades e passos a serem seguidos para se descobrir e interpretar os fatos estudados (GERHARDT; SILVEIRA, 2009). Sendo assim, o presente trabalho foi classificado quanto à natureza, ao objetivo, aos procedimentos e à abordagem.

Quanto à natureza, o trabalho é uma pesquisa aplicada, pois, segundo Gerhardt e Silveira (2009), "objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos". Com isso, o trabalho gerou conhecimentos sobre ferramentas multiplataforma, para uma aplicação prática que contribuirá para a automação do cálculo de adubação.

No que diz respeito ao objetivo, caracteriza-se como pesquisa exploratória. De acordo com Gerhardt e Silveira (2009), "a pesquisa exploratória tem como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito". Assim, buscou-se focar na maior familiaridade, com métodos de cálculos de adubação, para, então, automatizá-los por meio de um aplicativo.

No que se refere aos procedimentos, o presente trabalho é uma pesquisa-ação, pois pressupõe uma participação planejada do pesquisador na situação problemática a ser investigada (GERHARDT; SILVEIRA, 2009). Portanto, o objetivo foi transformar uma realidade observada, no caso, a automatização do cálculo de adubação.

Foi uma abordagem qualiquantitativa, sendo que a parte qualitativa se apresenta no projeto e desenvolvimento do aplicativo, e a parte quantitativa, na geração de dados. Conforme Gerhardt e Silveira (2009), "a pesquisa qualitativa não se preocupa com representatividade numérica, mas, sim, com o aprofundamento da compreensão de um grupo social, de uma organização, etc.". Portanto, buscou-se entender e atender às necessidades do produtor rural. A pesquisa quantitativa gera dados numéricos que descrevem causas de fenômenos, relações entre variáveis, precisão de cálculos, para garantir os resultados adequados.

No âmbito computacional, foi classificado como a apresentação de algo diferente. De acordo com Wazlawick (2009), "consiste na apresentação de uma forma diferente de resolver um problema". O resultado do aplicativo pode ser calculado manualmente, mas automatizá-lo através de um aplicativo pode trazer benefícios.

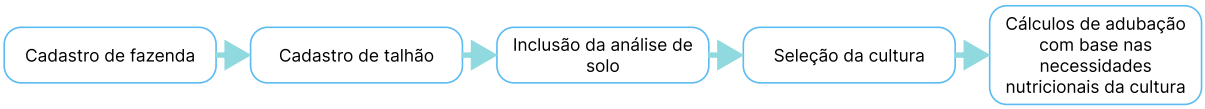
3.2 Solução

O cálculo de corretivos e fertilizantes pode ser feito de forma manual, com base na tabela nutricional da cultura. Contudo, assim como qualquer cálculo manual, podem ocorrer erros, além de ser um processo oneroso. Ademais, com a constante melhora de cultivares modificados geneticamente, há uma exigência cada vez maior para que o profissional se mantenha atualizado quanto às necessidades nutricionais das culturas.

O presente trabalho propôs o desenvolvimento de um aplicativo para o cálculo automático de adubação e corretivos das principais culturas cultivadas no Brasil, além de disponibilizar um banco de dados de tabelas nutricionais de culturas com a possibilidade de atualização e adição de novas cultivares.

Com o aplicativo, o usuário poderá cadastrar fazendas, talhões e análises de solos. O cálculo de adubação é realizado com base no resultado da análise de solo e cultura desejada. O diagrama da Figura 4 demonstra os passos da interação do usuário com a solução.

Figura 4 – Fases do aplicativo



Fonte: Elaborado pelo Autor, 2023.

3.3 Materiais e Tecnologias

Esta seção trata dos materiais, tecnologias e ferramentas usados no desenvolvimento do presente trabalho. O desenvolvimento do aplicativo foi realizado em um computador pessoal com as seguintes configurações:

Processador: AMD Ryzen™ 3 3200g @ 3.6GHz x 4.

Processador Gráfico: AMD Radeon™ RX 570, 4GB, GDDR5.

Memória RAM: 16 GB, 3000MHz, DDR4.

Unidade de Estado Sólido: 480GB.

Sistema operacional: Windows 10 Pro 64 bits.

O dispositivo móvel utilizado para testes foi:

Marca: Samsung.

Modelo: Galaxy S20 FE.

Sistema Operacional: Android 13.

Para o desenvolvimento, utilizaram-se as seguintes ferramentas:

- Visual Studio Code (https://code.visualstudio.com);
- Flutter (https://flutter.dev);
- Dart (https://dart.dev);
- MySQL Workbench (https://www.mysql.com/products/workbench);
- SQLite (https://sqlite.org);
- GitHub (https://github.com);
- Figma (https://figma.com).

O *design* foi concebido através da plataforma gratuita Figma, com uso direto no navegador, o que facilita a montagem de uma interface, contando com várias ferramentas para praticidade do uso, sem necessidade de linguagem de programação. A ferramenta possui, também, recursos de prototipagem, incluindo navegação entre telas através de botões e animações.

Para o desenvolvimento do aplicativo, foi utilizado o *framework* Flutter, que conta com a linguagem de programação orientada a objetos Dart.

O esquema lógico do banco de dados foi feito com o auxílio do programa MySQL Workbench, o qual conta com uma interface gráfica e funcionalidades que facilitam a concepção dos esquemas, como a ligação entre tabelas, suas cardinalidades, chaves de busca e tipo de cada campo, se adaptando a quaisquer projetos de bancos relacionais.

Para o armazenamento de dados, foi utilizado o SQLite Consortium (2023), que permite o desenvolvimento de bancos portáteis, proporcionando a persistência dos dados localmente. O SQLite tem integração com o Flutter, que providencia a entrada e a saída de informações.

Os códigos gerados foram incluídos em um repositório do GitHub, que é uma plataforma gratuita com funções de versionamento e hospedagem de repositórios, além de ferramentas para gerenciamento do projeto com quadro *Kanban*. Com esse recurso, foi possível controlar melhor o gerenciamento do fluxo de desenvolvimento do projeto e alterações no progresso das atividades propostas.

O Visual Studio Code foi utilizado como editor de código-fonte, por contar com ferramentas que aceleram o processo de programação. Além disso, o editor conta com extensões Flutter e Dart que suportam a edição, refatoração, execução e depuração do projeto.

3.4 Métodos e procedimentos

A presente seção explica o desenvolvimento do trabalho, descrevendo os passos seguidos, bem como o detalhamento de como foram feitos. A Seção 3.4.1 revela a metodologia utilizada, e a Seção 3.4.2 descreve as etapas seguidas. Por fim, a Seção 3.4.4 explica a escolha da arquitetura do processo de desenvolvimento e como foi implementada.

3.4.1 Scrum

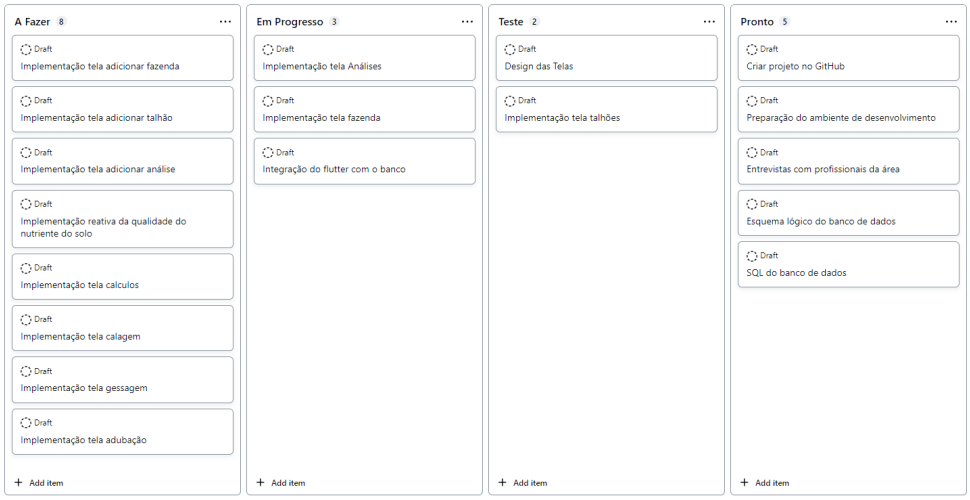
Para gerenciar a produção deste trabalho, foi empregada a metodologia *Scrum* de desenvolvimento ágil. Segundo Sabbagh (2013), o *Scrum* se tornou a forma mais comum de se trabalhar em projetos de *software* e *hardware*, sendo utilizado também em diferentes mercados, como empresas de *marketing*. Além disso, o autor evidencia, como benefícios da utilização desta metodologia, a redução dos riscos do projeto, maior qualidade no produto gerado, visibilidade do progresso, redução do desperdício, aumento de produtividade, entre outros.

O *Scrum* determina o uso de *Sprints*, que são ciclos de trabalho, com entregas menores, que compõem o produto final. No caso do presente projeto, foram definidas reuniões semanais para o fechamento do ciclo e planejamento das próximas demandas.

Todas as tarefas foram listadas no *product backlog*, em forma de texto, no Apêndice D. Aproximadamente, foram 16 *Sprints* para conclusão de todas as tarefas planejadas para os projetos práticos.

O controle das atividades foi realizado através de um quadro *Kanban*, o qual é uma forma visual de se observar prazos, metas e estado das atividades planejadas. O projeto fez uso do quadro *Kanban* disponível na plataforma GitHub.

Figura 5 – Quadro Kanban



Fonte: Elaborado pelo Autor, 2023.

O quadro *Kanban*, retratado na Figura 5, apresenta as colunas a fazer, em progresso, teste e pronto. As tarefas incluídas podiam ser movidas a qualquer momento

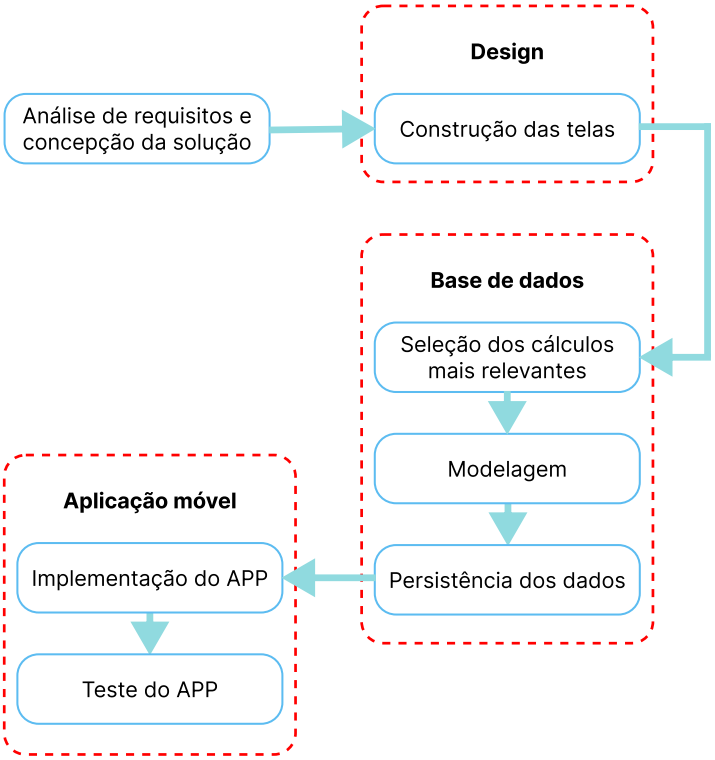
para outras colunas, alterando seu estado. Quando todas as tarefas chegaram à coluna das tarefas prontas, o trabalho foi concluído. A evolução do quadro *Kanban* a cada *Sprint* se encontra no Apêndice A.

3.4.2 Etapas de desenvolvimento

As principais etapas do desenvolvimento são apresentadas na Figura 6. O primeiro passo foi a análise de requisitos baseada em entrevista com profissional da área em dados de análises de solo. Com base nos requisitos levantados, foi elaborado o projeto de interfaces contendo as telas da solução.

Utilizando-se as diretrizes de usabilidade e heurísticas de avaliação propostas na Seção 2.5, foi construído o projeto das telas, e a interface gerada guiou a produção do aplicativo, empregando-se a metodologia ágil.

Figura 6 – Principais etapas do desenvolvimento



Fonte: Elaborado pelo Autor, 2023.

Após a definição das funcionalidades determinadas no projeto de interfaces do Apêndice C, foram feitas a revisão e a avaliação de fontes de pesquisa científica da área agronômica, para levantamento das principais culturas anuais cultivadas no Brasil, considerando-se os dados disponibilizados pela CONAB.

Ao se designar as culturas, foram definidos os métodos de cálculo de calagem, gessagem e adubação, por meio de pesquisa bibliográfica, considerando-se publicações recentes e recomendações mais relevantes no âmbito da agronomia.

Unindo-se as informações do projeto de interface e das necessidades nutricionais das culturas, foi gerada a modelagem, que envolveu a construção do esquema lógico com o *MySQL Workbench*, a partir do qual foi criado o banco de dados no *SQLite*.

Com as etapas de projeto prontas, o próximo passo foi a construção do aplicativo. Para isso, foi utilizado o *framework* Flutter, sendo que, após a implementação, foram feitos testes com a intenção de garantir as funcionalidades requisitadas e solucionar erros.

3.4.3 Projeto e prototipagem

A utilização do Figma para projetar o aplicativo foi um facilitador, contando com várias ferramentas e aceleradores do processo, possibilitando, logo no início, confirmar o levantamento nas reuniões, em funcionalidade, organização das telas, formulários e identidade visual. O resultado desse projeto encontra-se no Apêndice B.

O Figma conta com uma área de trabalho infinita, podendo ampliar ou reduzir seu tamanho a qualquer momento, uma régua para ajustes, alinhamento, biblioteca de projetos gratuitos e exportação de criações.

Ao abrir o projeto criado, são apresentados dois menus laterais. A Figura 7(a) mostra as camadas desenvolvidas, divididas em *frames*, que se referem a cada tela. Já a Figura 7(b) contém as ferramentas para cada elemento adicionado, podendo controlar tamanho, raio do canto, transparência, cor, bordas, elevação e fonte.

Na Figura 8(b), há um exemplo de como um *frame* foi utilizado para montar a tela de logo do aplicativo. O próprio *frame* foi colorido em verde e faz parte da camada mais profunda; acima dele, estão o `Tab Bar` e `Status Bar`, que são a simulação das barras de sistema em um smartphone. E mais acima, estão o texto estilizado, com o nome do aplicativo, e um grupo de elementos que compõem a planta, que engloba grupos como folhas e talo.

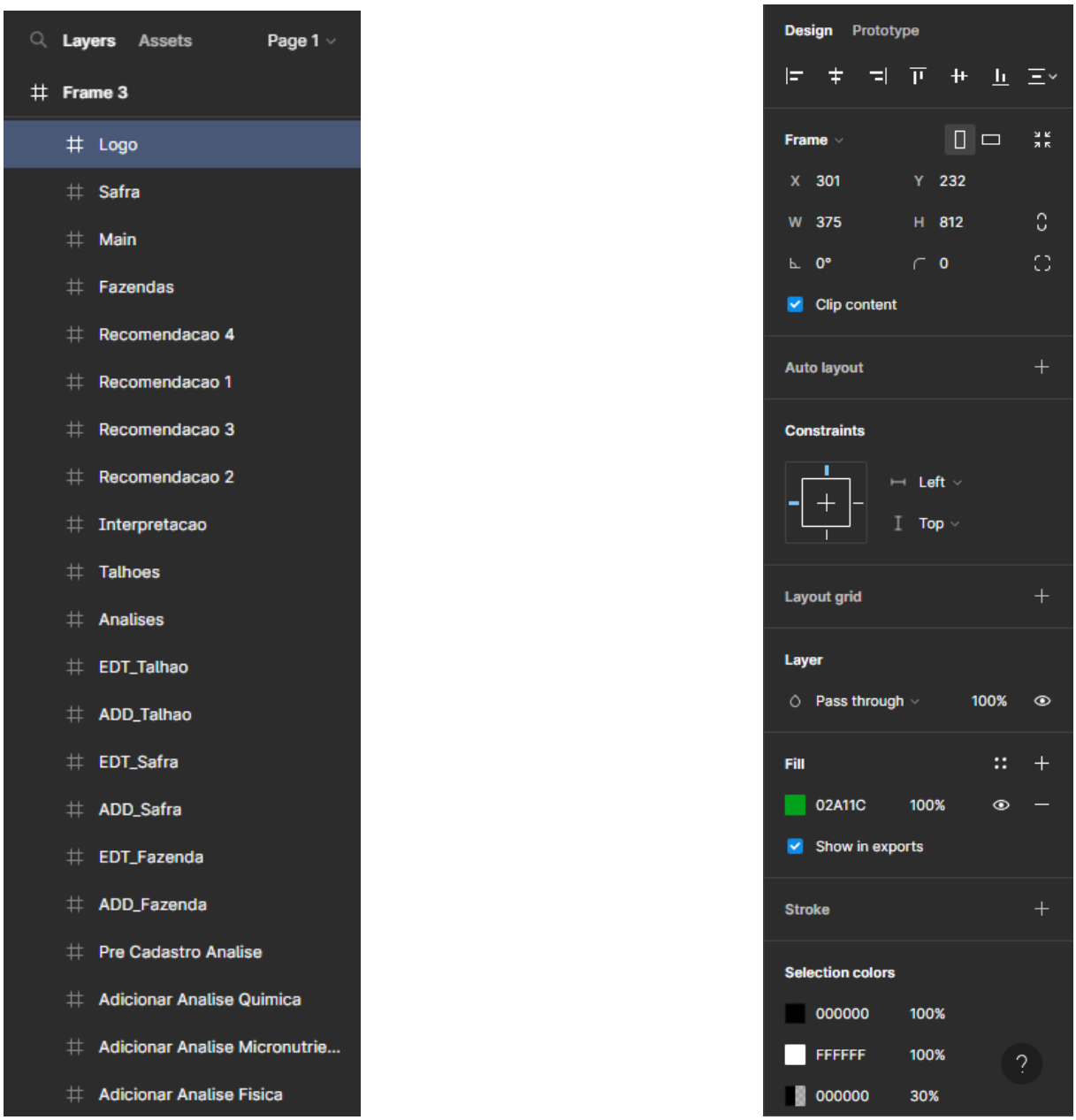
Após a criação das telas, foi feita a prototipagem, mostrada no Apêndice C, que faz o endereçamento de algum elemento ou botão para outra página, o que permitiu recolher opiniões de profissionais da área, mesmo sem se desenvolver o código.

3.4.4 Arquitetura do projeto de desenvolvimento

Pela familiaridade, a organização do código seguiu a arquitetura de projeto *model, view, presenter* (MVP) ou modelo, visão, apresentador, mostrada da Figura 9. O objetivo é separar as funcionalidades em diferentes arquivos, pela afinidade de cada operação. Essa separação, além de prover maior controle, proporciona o encapsulamento, para evitar o acesso direto aos dados, que pode comprometer questões de segurança.

Os *models* são responsáveis por representar as tabelas do banco de dados. Essa representação é utilizada para se criar uma camada de abstração, possibilitando a

Figura 7 – Menus do Figma



(a) Camadas (b) Ferramentas

Fonte: Elaborado pelo Autor, 2023.

manipulação do banco em memória principal e a utilização de coleção de dados como listas.

Cada tabela corresponde a uma nova classe no código, que, por sua vez, deve ter os mesmos atributos da tabela, com métodos adicionais para haver uma conversão integrada. Esses métodos foram nomeados como `toMap()` e `fromMap()`, os quais definem chaves e valores específicos para cada classe, para convertê-los em mapas ou objetos.

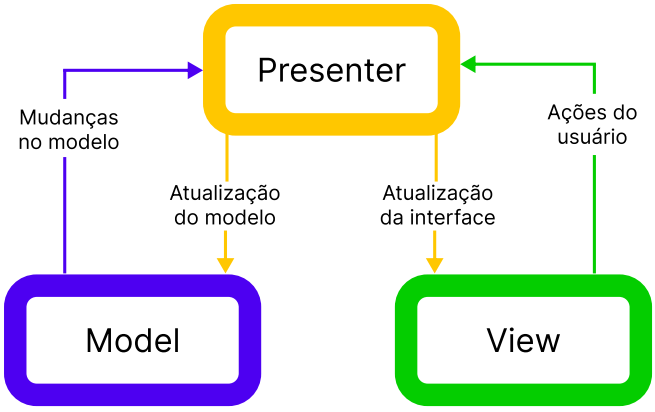
Cada módulo é composto por um único *presenter* conectado com *model* e *view*, fazendo a ponte entre eles. O *presenter* se encarrega de operações sobre as tabelas do banco e manutenção das listas em memória principal, além de fornecer os dados ajustados

Figura 8 – Logo no Figma

(a) Frame (b) Logo

Fonte: Elaborado pelo Autor, 2023.

Figura 9 – MVP - *Model, View, Presenter*



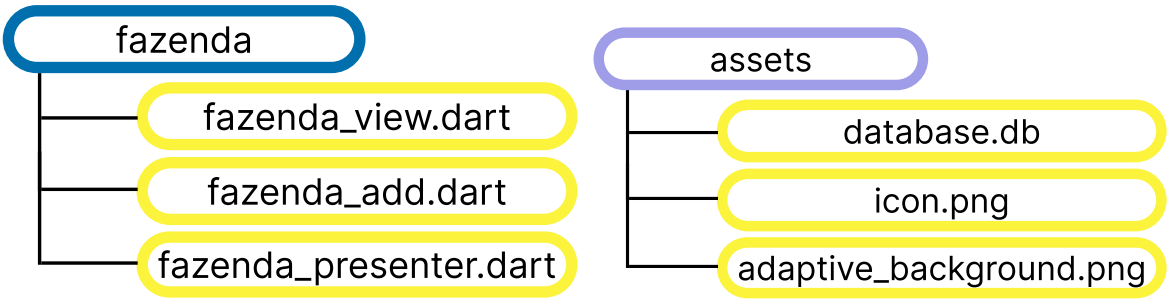
Fonte: Elaborado pelo Autor, 2023.

para a *view*, que os materializa na tela.

É comum haver várias *views* para um único módulo, pois elas contêm operações distintas, como a seleção, adição, edição. A Figura 10(a) apresenta como exemplo o módulo `fazenda`. As telas `fazenda_view.dart` e `fazenda_add.dart` recebem dados de `fazenda_presenter.dart`, convertendo as respostas do banco de dados utilizando o arquivo `fazenda_model.dart`, que se encontra na pasta `models`.

Nos *assets*, conforme mostrado na Figura 10(b), estão o banco de dados e as

Figura 10 – Módulos e pastas do desenvolvimento



(a) Módulo fazenda (b) Recursos do projeto

Fonte: Elaborado pelo Autor, 2023.

imagens como o ícone do aplicativo. Esses recursos foram agrupados em uma única pasta para facilitar a importação.

Um exemplo de como o MVP foi implementado está no Apêndice E, descrito a seguir. O `model` é criado com suas variáveis e métodos para converter esse objeto em map, e vice-versa. Estes são utilizados na classe `TalhaoPresenter`, que busca, no banco, uma lista de talhões, através do método privado `_getTalhoes()`, que recebe a identificação da fazenda.

Já no `TalhaoView`, ao se criar uma instância do `presenter`, há apenas uma lista visível a ser consultada. Essa lista é consumida em um `ListView`, que ordena a criação de cada item da lista. Assim, os dados são mostrados na tela, no modelo definido pelo `CustomCard`.

4 DESENVOLVIMENTO

O presente capítulo expõe o desenvolvimento, os resultados e as discussões obtidos no decorrer do projeto. A Seção 4.1 explica o esquema lógico do banco de dados; os detalhes da implementação se encontram na Seção 4.2. Em seguida, é exposto como foi feito e os resultados do teste de funcionamento, na Seção 4.3. Já a Seção 4.4 apresenta soluções, com projetos de interface e organização do código-fonte no ambiente de programação.

4.1 Construção do banco de dados

A Figura 11 apresenta o esquema lógico do banco de dados desenvolvido para o aplicativo, implementado de forma embarcada com o SQLite. O banco de dados contém as tabelas utilizadas para cadastros de dados do usuário e também para análise e cálculo da necessidade de nutrientes no solo.

As tabelas `fazenda`, `talhao`, `safra`, `analise` e `recomendacao` estão relacionadas com os cadastros de dados do usuário. Elas têm o objetivo de facilitar o uso recorrente, possibilitando gerenciamento de múltiplas fazendas, talhões e safras.

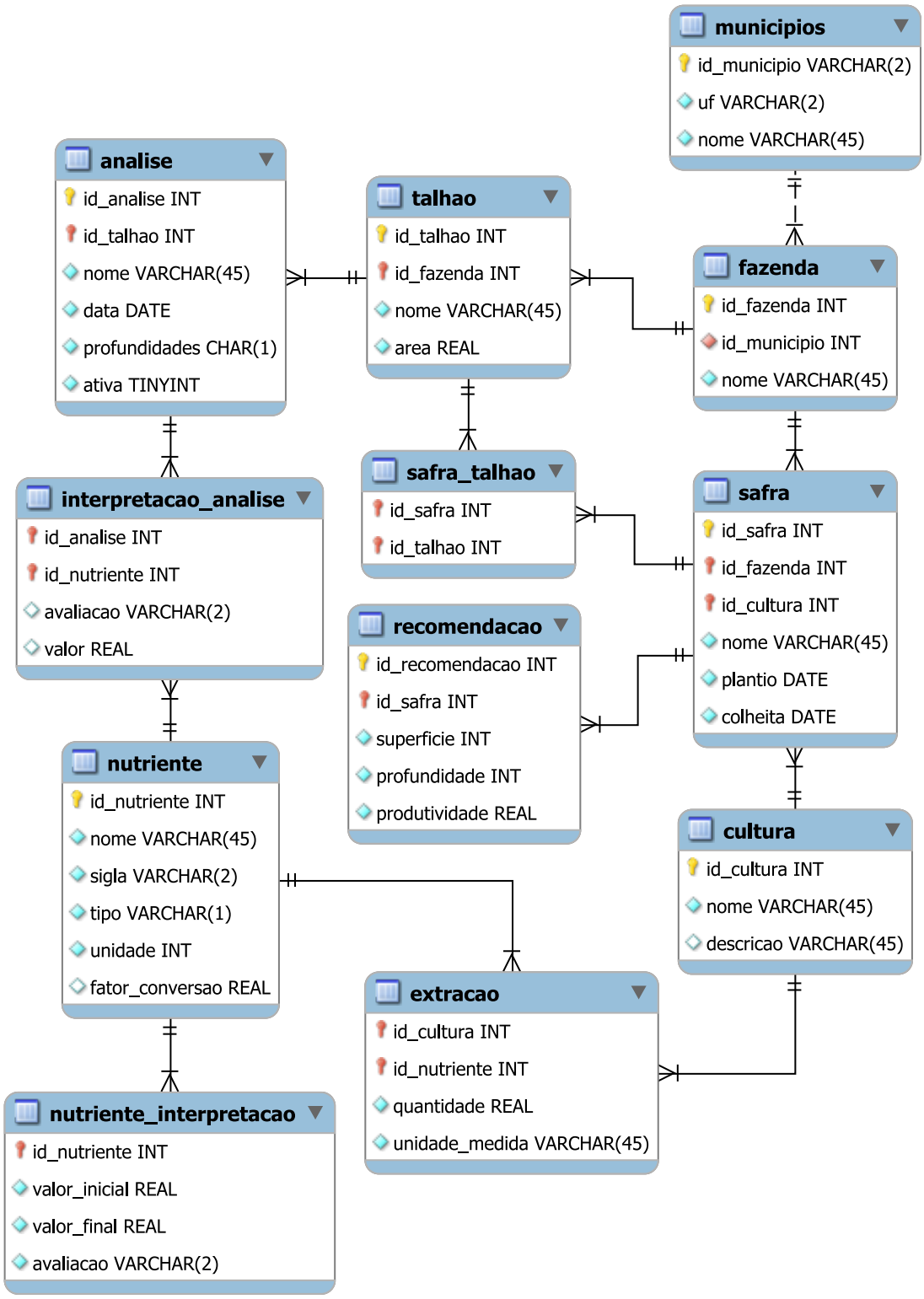
As tabelas `safra`, `talhao` e `safra_talhao` permitem controlar as safras de uma fazenda, assim como os talhões que compõem essa safra. Basicamente, os registros da tabela `safra` estão relacionados com a cultura a ser cultivada e as datas de início e fim do cultivo. Cada safra pode ter múltiplas `recomendações`, que armazenam os valores, definidos pelo usuário, de variáveis necessárias para o cálculo de corretivos e fertilizantes.

As tabelas `cultura`, `extracao` e `municipios` possuem dados pré-cadastrados, contendo, respectivamente, as principais culturas anuais do Brasil, tabelas de extração destas e todos os municípios brasileiros. Além disso, em versões futuras do aplicativo, os dados podem ser atualizados para a inclusão de novas culturas.

Os dados das análises de solo são armazenados nas tabelas `analise` (dados gerais de uma amostra de solo analisada) e `interpretacao_analise` (itens específicos de cada amostra). A tabela `nutriente` tem dados pré-cadastrados com os possíveis itens de uma análise de solo, tipo (físico, químico ou micronutriente) e unidade mais comum. Já a tabela `nutriente_interpretacao` possui as faixas de valores que são usadas para interpretar cada item de uma amostra.

As análises podem mudar de estado para ativas ou inativas, para facilitar o gerenciamento do histórico do talhão. Assim, durante os cálculos de recomendações, são consideradas todas as análises definidas como ativas pelo usuário. Essa característica tem a intenção de proporcionar uma visão geral sobre a adubação, apresentando as necessidades de cada talhão incluído na safra.

Figura 11 – Esquema lógico das tabelas de cálculo de adubação



Fonte: Elaborado pelo Autor, 2023.

4.2 Detalhes do processo de implementação

Nesta seção, são abordados os passos mais relevantes no processo de implementação. Os detalhes de como o ambiente de programação foi preparado encontram-se na

Seção 4.2.1. O projeto foi estruturado segundo as diretrizes da Seção 4.2.2. A integração do banco de dados com o Flutter foi esmiuçada na Seção 4.2.3. As vantagens de se utilizar um arquivo-tema foram apresentadas na Seção 4.2.4, e a Seção 4.2.5 relatou o uso de *widgets* para construção do aplicativo. Os nutrientes foram interpretados de acordo com a Seção 4.2.6, e o modo como foi feita a navegação entre os temas encontra-se na Seção 4.2.7. As telas com formulários seguiram o modelo explicado na Seção 4.2.8, e, por fim, como atualizar mudanças na interface está revelado na Seção 4.2.9.

4.2.1 Preparação do ambiente de programação

O processo de implementação começou com a preparação do ambiente de programação no sistema operacional Windows. O *framework* Flutter foi baixado no site oficial, na versão 3.13.6. O arquivo compactado foi extraído para a `C:\flutter\bin`, cujo caminho foi adicionado à variável de ambiente do sistema `Path`.

Após a instalação do ambiente Flutter, por meio do utilitário de linha de comando, foi executado `flutter doctor` para confirmar se a configuração do ambiente foi bem-sucedida. A Figura 12 apresenta o resultado retornado pelo comando, mostrando que as dependências necessárias para o desenvolvimento para Android foram atendidas.

Figura 12 – Resultado do comando `flutter doctor`

```
C:\Users\jean_.JEAN>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.13.6, on Microsoft Windows [versão 10.0.22621.2506], locale pt-BR)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run `flutter doctor --android-licenses` to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2022.3)
[√] VS Code, 64-bit edition (version 1.82.2)
[√] Connected device (4 available)
[√] Network resources
```

Fonte: Elaborado pelo Autor, 2023.

O projeto foi criado utilizando-se o comando `flutter create Solofert`. Como resultado, o comando cria uma pasta com vários arquivos de configuração e um exemplo inicial.

A inclusão de bibliotecas é feita através da edição do arquivo `pubspec.yaml`, que controla a inclusão de recursos. Parte desse arquivo se encontra no Apêndice F. No setor de dependências, iniciado na linha 7, foram adicionados o provider, sqflite e path, com a versão especificada. No VSCode, o comando `flutter pub get` ativa ao salvar o arquivo, que baixa e inclui as bibliotecas no projeto.

Como se trata de um aplicativo para o sistema Android, foi necessário baixar o Android Studio, que possui a ferramenta Virtual Device Manager. Tal ferramenta possibilita baixar e gerenciar vários emuladores, dos mais diversos dispositivos que utilizam o sistema Android. O Virtual Device Manager permite editar ou criar um dispositivo, além de executar e interromper a execução de dispositivos existentes.

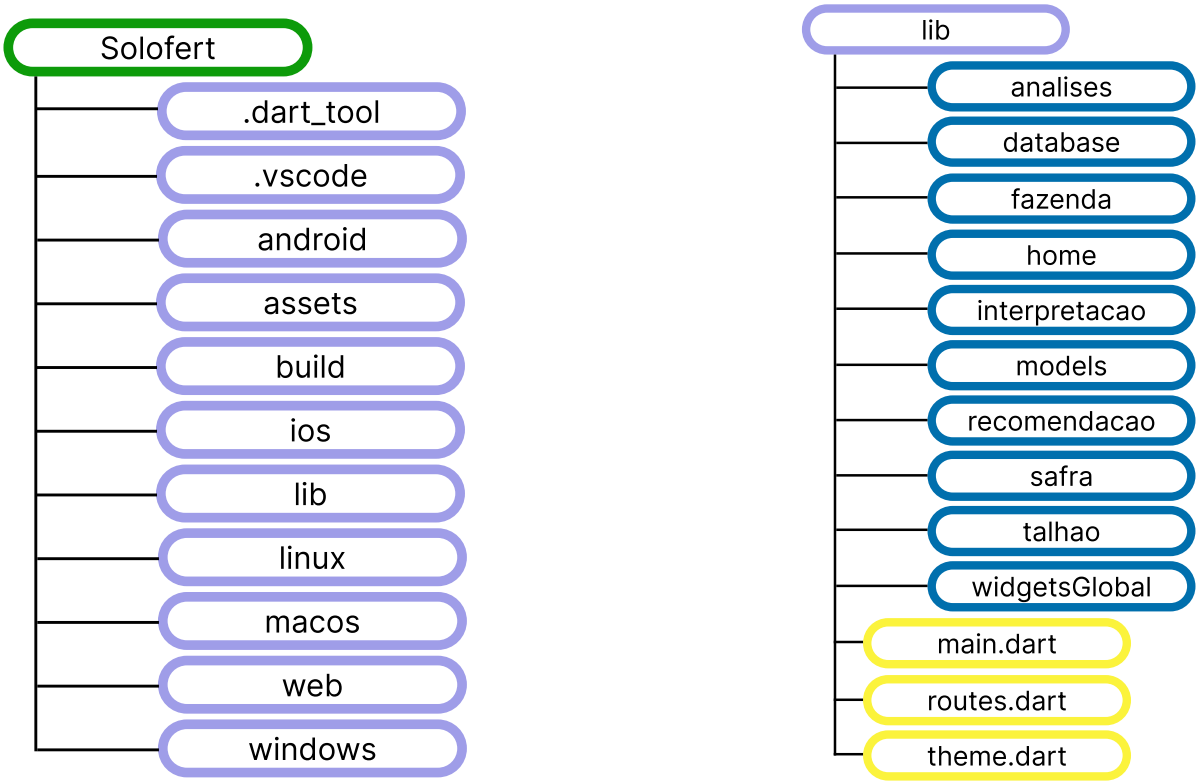
Um detalhe importante sobre o Virtual Device Manager é que o computador precisa ter o recurso de virtualização ativado.

O telefone Pixel 3 foi o dispositivo escolhido para emulação pelo fato de ter uma proporção de tela comumente usada no mercado. Ao abrir o projeto no Visual Studio Code, as extensões *dart* e *flutter* são sugeridas. Após instalação dessas, a linguagem *framework* e emulador são reconhecidos e integrados de forma automática.

4.2.2 Estrutura e organização do projeto

A criação do projeto gera pastas, conforme exposto na Figura 13(a). Algumas delas contam com configurações para plataformas suportadas pelo Flutter. O desenvolvimento do aplicativo concentrou-se na pasta `lib`, que é a reservada para inclusão dos arquivos em Dart, e a pasta `assets`, usada para inclusão de recursos como imagens.

Figura 13 – Estrutura de pastas do projeto



(a) Pastas padrão

(b) Módulos do projeto

Fonte: Elaborado pelo Autor, 2023.

A Figura 13(b) apresenta o conteúdo da pasta `lib`. Ela foi dividida em novas pastas (cor azul), agrupadas de acordo com sua finalidade, nomeadas como módulos daqui em diante. A pasta `lib` possui também (em amarelo) arquivos de controle principal do aplicativo. O `main.dart` é o ponto de entrada para execução do *software*, o `routes.dart` controla a navegação entre telas, e o `theme.dart` define cores e estilos de texto utilizados.

4.2.3 Integração ao banco de dados

A Figura 14 apresenta o código usado para a conexão com o banco de dados através da classe `DBProvider`, que é iniciada juntamente com o aplicativo. A função `get database` impede a criação de múltiplas instâncias. Já o método `_initDatabase()` recebe o caminho e o nome para o banco de dados. Caso o arquivo do banco de dados não exista, o método `onCreate` é usado para criar as tabelas e inserir os dados iniciais.

Figura 14 – Código para criação do banco

```dart
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class DBProvider {
  DBProvider._(); //cria uma instancia do Database Provider
  static final DBProvider instance = DBProvider._();
  static Database? _database; //instancia do SQlite
  get database async {
    if (_database != null) return _database;
    return await _initDatabase();
  }

  _initDatabase() async {
    return await openDatabase(
      join(await getDatabasesPath(), 'database.db'),
      version: 2,
      onCreate: _onCreate, //procedimentos ao criar
    );
  }
}
```

Fonte: Elaborado pelo Autor, 2023.

Após a conexão com o banco de dados, uma única instância é referenciada para efetuar as operações de criar, ler, atualizar e deletar, além de trazer os dados do banco para a memória principal. Essas operações sempre recebem uma resposta do banco, com conteúdo, confirmação ou identificação.

A Figura 15 apresenta um exemplo de operação de leitura. Essas operações são feitas através de consultas SQL ou funções do `SQFLite`, como *query*, retornando os valores requeridos. As consultas retornam tabelas em formato `map`, com a nomeação do campo seguida de seu respectivo valor.

Figura 15 – Operação de leitura no banco

```
final db = await DBProvider.instance.database;
final List<Map<String, Object?>> response =
await db.rawQuery('''
  SELECT DISTINCT uf
  FROM municipio'''
);
response =
await db.query('fazenda',
  where: 'idFazenda=?',
  whereArgs: [id]
);
```

Fonte: Elaborado pelo Autor, 2023.

4.2.4 Tema

O tema é a identidade do aplicativo; portanto, a definição de elementos visuais é parte essencial do processo de desenvolvimento. O tema pode ser alterado ao longo do projeto, de acordo com fatores como acessibilidade, aceitação e estilo.

Pensando nisso, um arquivo foi criado para centralização dessas configurações, de forma a possibilitar que as alterações sejam reproduzidas em todos os elementos da aplicação, incluindo definições de cores, fonte, tamanho e estilo de texto referentes a todas as variações utilizadas na interface. Posteriormente, os *widgets* com parâmetros customizáveis utilizam uma função no Flutter para buscar o estilo desejado.

Um exemplo de código de tema pode ser visto no Apêndice G. As linhas 2 a 7 definem três das cores mais utilizadas, nomeadas com palavras predefinidas, usando o formato de cor ARGB (*Alfa, Red, Green, Blue*). Entre as linhas 8 e 26, estão os estilos de texto, com configurações de cor, tamanho e grossura do traçado.

O ícone e a animação de abertura também fazem parte do processo de tematização do aplicativo. Para esses recursos, foram utilizadas as dependências *flutter launcher icons* e *flutter native splash*.

Ambas são *scripts* automatizados; a primeira recebe uma imagem, que é convertida em várias versões, de tamanhos diferentes, para se adaptar a cada situação em que o ícone é utilizado no sistema. Já a segunda recebe duas imagens: a do logotipo e a do fundo desejado na abertura do aplicativo, que são processadas e colocadas na devida pasta.

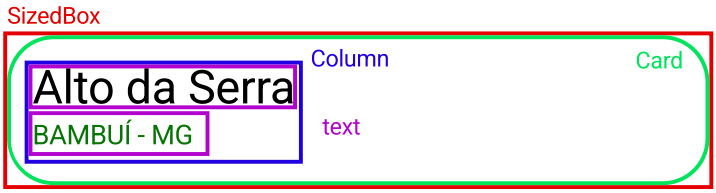
4.2.5 Widgets

Os *widgets* são os blocos de construção da parte visual do aplicativo. Cada tipo de bloco tem uma funcionalidade e pode ter um ou mais filhos, os quais são aninhados combinando suas funcionalidades para montar a tela.

Os *widgets* mais utilizados foram `SizedBox`, `Card`, `Column`, `Row`, `Text`, `Padding` e `ListView`. O *widget* `SizedBox` é o definidor de tamanho, e o `Card` dá a profundidade e arredonda as bordas. Os *widgets* `Column` e `Row` são blocos com múltiplos filhos organizados na vertical e horizontal, respectivamente. O `Text` escreve o texto na tela, e o `Padding` faz o espaçamento. Por fim, o `ListView` renderiza uma lista de *widgets* na tela.

A Figura 16 representa, visualmente, cada bloco construído, da parte mais externa para a mais interna. Esses blocos aninhados foram definidos como um *widget* personalizado. Esse *widget* personalizado foi importado e utilizado para compor uma lista de cartões, que recebe o título e subtítulo como parâmetro, de dados anteriormente cadastrados.

Figura 16 – Cartão utilizado no app com identificação de cada *widgets*



Fonte: Elaborado pelo Autor, 2023.

Cada cartão conta com uma identificação que serve para realizar operações sobre os dados e navegação entre telas. Esses *widgets* personalizados, por serem utilizados em diversas telas do aplicativo, foram montados para se adaptarem ao conteúdo recebido. Isso contribui para a manutenibilidade do código-fonte.

O cartão da Figura 16 foi construído usando-se o código mostrado no Apêndice H. Cada *widget* tem parâmetros customizáveis, que foram utilizados para definir cor, sombra, tamanho, alinhamento e espaçamento.

Outra dupla de *widgets* a ser destacada são o `ListView` e `ScrollView`, que são utilizados em conjunto devido às suas características. O primeiro cria uma lista ilimitada, e o segundo permite deslizar a tela para ver esses itens. Apesar das possibilidades, o `ListView` é limitado pelo tamanho da tela oferecido pelo `ScrollView`.

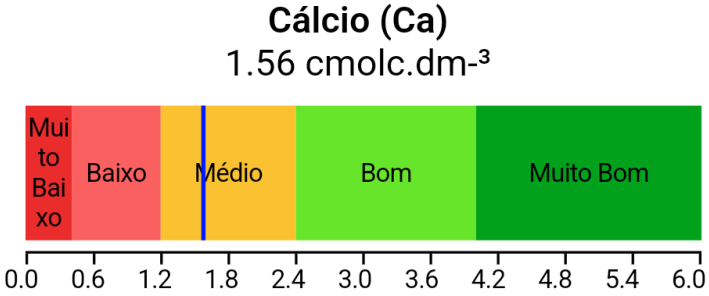
Assim como a consulta de tema, o tamanho da tela também pode ser buscado, e é uma função poderosa para adaptabilidade do sistema. Com isso, é possível definir uma relação percentual que cada bloco irá ocupar, possibilitando responsividade do sistema em todas as proporções da tela. No código, essa função é utilizada na altura, que é controlada pelo *height* na linha 2. A largura, que, nesse caso, não foi definida pelo *width*, determina que o cartão deve ocupar todo espaço horizontal livre.

4.2.6 Interpretação de nutrientes

Mais um aspecto que foi possível alcançar através do uso do aplicativo foi a melhoria na interpretação de nutrientes da análise de solo, conforme representado na

Figura 17. A parte superior do cartão mostra o item, sigla e valor obtido na amostra de solo. Em seguida, o cartão mostra um gráfico com intervalos coloridos e legendados. Abaixo, há uma régua para complementar com a referência numérica, sendo que a linha em azul indica a quantidade atual do item.

Figura 17 – Gráficos para interpretação de nutrientes



Fonte: Elaborado pelo Autor, 2023.

Esse gráfico foi implementado utilizando-se a tabela `nutriente_interpretacao`, na qual foram cadastrados, para cada nutriente, intervalos numéricos e a avaliação destes. Esses dados são processados atribuindo-se cor, nome, proporção relativa ao total e valores inicial e final, sendo armazenados na classe`DataGraphModel`.

O primeiro e último valores são os guias, e a diferença entre o valor final e o valor inicial é transformada em uma porcentagem. Esse porcentual é o fator incluído no modelo e pode ser multiplicado pela largura de tela para se definir o tamanho que cada intervalo terá.

Para montagem do gráfico, foi utilizada uma característica interessante do Flutter. O código cria uma lista de *widgets*, que define um retângulo de tamanho, cor e nome estabelecidos pelo modelo personalizado recebido. Posteriormente, essa lista é unida ao título e régua para finalizar a montagem do bloco.

O Apêndice I demonstra um exemplo da lista de intervalos e lista de *widgets*. A régua teve construção semelhante à empregada no gráfico. A quantidade de marcações e o tamanho da fonte são definidos pela quantidade de numerais, a fim de encaixar as divisões em uma linha, sem comprometer a legibilidade.

4.2.7 Navegação entre telas

A navegação entre as telas é auxiliada pelo arquivo `routes`, que guarda as rotas do aplicativo. As rotas recebem um nome, utilizado para identificá-las ao serem solicitadas. Os dados são limitados a apenas um objeto, sendo enviados pelo cartão selecionado; se o tipo corresponder, a navegação é feita. O objeto recebido deve ser inserido em uma variável para disponibilizar o uso na tela atual.

A Figura 18 exemplifica a implementação desse arquivo. A função `generateRoute` recebe um argumento como parâmetro na linha dois, o qual passa por

`switch case` para tentar achar uma correspondência, que, se não identificada, cai no caso `default` (linha 8), que retorna um erro padrão. Porém, caso o argumento passado (linha 7) condisser com alguma das opções, o objeto é convertido para o tipo especificado, e a tela é chamada.

Figura 18 – Rotas para navegação entre telas

```
1  class RouteGenerator {
2    Route<dynamic> generateRoute(RouteSettings settings) {
3      final arguments = settings.arguments;
4      switch (settings.name) {
5        case '/talhao':
6          return MaterialPageRoute(
7              builder: (_) => Talhao(arguments as FazendaModel));
8        default:
9          return MaterialPageRoute(
10           builder: (_) => Scaffold(
11             body: Center(child: Text("Sem rota definida
                 ↪  ${settings.name}")),
12           ),);}}}
```

Fonte: Elaborado pelo Autor, 2023.

Ao ser chamada, ela é carregada e toma o lugar da passada através de uma animação de transição, que pode ser customizada. O *framework* cria o botão de voltar para todas as telas provenientes da primeira, podendo desfazer todo o caminho navegado.

4.2.8 Telas com formulários

Os módulos que recebem dados cadastrados pelo usuário são os formulários. Essas telas contêm campos que aceitam determinados dados.

As funções de adicionar e editar foram unidas em uma única tela, e são customizadas dependendo da ação desejada. Isso é definido pelo objeto da navegação entre telas; se o identificador for inválido, significa que é uma adição, mas, caso seja válido, os dados dele são buscados, completando os campos, que podem ser editados e salvos.

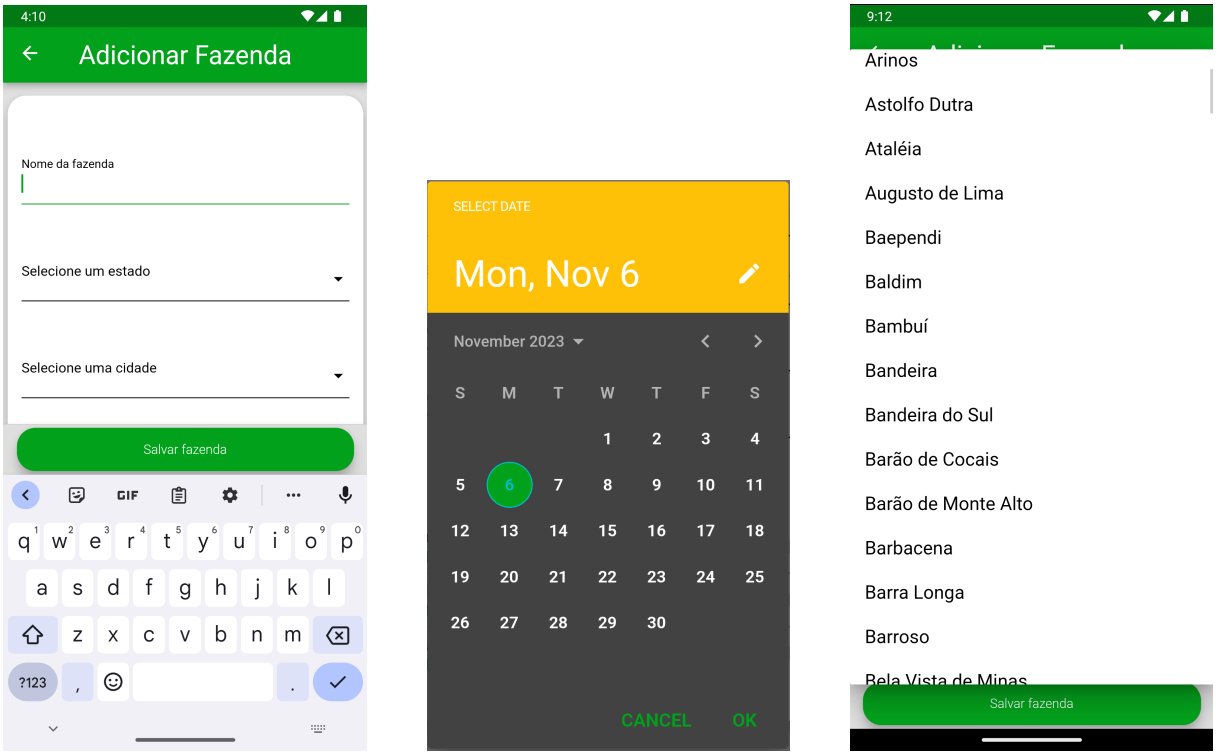
O Apêndice J mostra o código para um formulário em texto da Figura 19(a), sendo um `controller` e uma `key` que servem para armazenar e manter o estado da sessão. É possível definir título para caixa de texto e o tipo de teclado que abrirá caso o campo seja selecionado.

Também se implementa o método `validator`, que tem o papel de verificar o valor digitado, impedir valores nulos ou vazios e uma lista de regras. Se o usuário inserir um valor inválido, uma mensagem definida para cada tipo de regra aparecerá abaixo do campo, facilitando a identificação do erro cometido.

Além do formulário de texto, foram utilizados dois outros tipos. O primeiro foi o `datePicker`, da Figura 19(b), que abre um calendário do sistema em uma janela

sobreposta, no dia atual, e botões para cancelar ou confirmar a seleção. Já o segundo é o `dropDownButton`, na Figura 19(c), que, ao interagir, mostra uma lista com todas as possibilidades predeterminadas para esse campo.

Figura 19 – Formulários



(a) TextFormField (b) Datepicker (c) DropDownButton

Fonte: Elaborado pelo Autor, 2023.

4.2.9 Gerenciamento de estados

Um aspecto inerente ao Flutter é o desempenho; portanto, a interface não é atualizada mesmo ao se adicionar ou alterar uma variável que esteja na tela. Então, é necessário ordenar uma atualização de estado.

O gerenciamento de estados pode ser feito com o método `SetState` em um botão controlado pelo usuário. Porém, esse método não é funcional quando há informações transitando entre telas. Para isso, foi utilizada a biblioteca `provider`, que permite colocar um observador em uma classe e forçar a tela a se redesenhar, caso haja alteração.

Esses `providers` são simples de implementar, pois basta incluir uma biblioteca na classe. Nesse caso, foram colocados nos `presenters`, por ter o controle dos dados enviados para tela. E, a cada alteração nesses dados, chama-se a função `notifyListeners`, que notifica as telas que estão consumindo essas informações.

Porém, esse recurso deve ser utilizado com cautela, pois essas listas e variáveis são mantidas em memória principal por todo o tempo de execução, podendo afetar o desempenho e uso de bateria dos dispositivos.

4.3 Teste de funcionamento

Os testes de funcionamento do aplicativo foram conduzidos apenas em dispositivos Android virtual e físico, sendo que o virtual foi o Pixel 3, com Android versão 11, e o físico foi o dispositivo pessoal de modelo Galaxy S20FE, com Android versão 13.

Os testes compararam a consistência do resultado entre esses dispositivos e foram executados de duas formas diferentes. A primeira foi com a versão de desenvolvimento, na qual se têm várias ferramentas que fornecem métricas e execução de alterações quase instantaneamente. Essa característica auxiliou na identificação e correção de erros.

O segundo modo foi através da versão de lançamento, que vai para o usuário e conta com um instalador Android Application Pack (APK) contendo um executável. Essa etapa foi feita nos estágios mais avançados de desenvolvimento, para garantir que o comportamento de funcionalidades e telas se mantivesse.

Os erros referentes à aplicação eram, em sua maioria, relacionados à conexão ao banco ou bibliotecas e a elementos visuais que não se comportavam como deveriam. Para detectá-los, utilizamos os formulários, por meio da adição de caracteres inválidos, fora de intervalo e com alto número de caracteres. Isso os revelou, possibilitando a tomada de ações para que não acontecessem mais.

Também foi verificado, em um teste sem usuários, possíveis problemas de usabilidade, como botões que não funcionam, problemas na arquitetura, layout, organização e fluxo, com objetivo de facilitar a navegação pelos recursos, realizar tarefas-chaves e encontrar informações.

Para o desempenho, o *framework* Flutter fornece uma ferramenta que analisa a consistência do aplicativo, demonstrando, visualmente, métricas que auxiliam no reconhecimento de engasgos e uso de memória. A cada adição, essa ferramenta foi utilizada.

Portanto, para melhorar a experiência de uso, foram utilizados os erros apontados pelas ferramentas e inspeção visual, visando corrigir as apresentações defeituosas e cálculos incorretos.

Na parte dos cálculos, os resultados de calagem, gessagem e adubação foram testados extensivamente através de testes de mesa. Esses testes incluíram análises obtidas no Laboratórios de Solos do IFMG *Campus* Bambuí e confirmação das fórmulas por meio de valores unitários. Adicionalmente, esses resultados foram conferidos em reunião com o professor coorientador do presente trabalho.

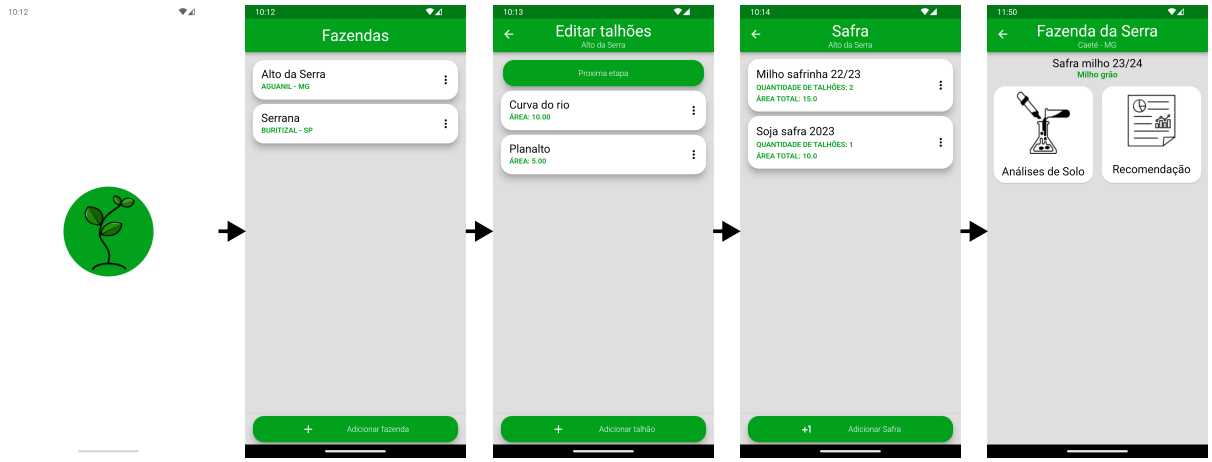
4.4 Aplicação móvel

A presente seção tem como objetivo mostrar os frutos da união de cada etapa realizada nas Seções 4.1 e 4.2, destacando-se os principais módulos e a aplicação de funcionalidades anteriormente apresentadas.

Por ser um sistema com grande quantidade de telas, o projeto utilizou diretrizes e heurísticas de usabilidade da Seção 2.5, para facilitar a utilização e satisfazer as necessidades do usuário. Alguns desses facilitadores são o agrupamento de funções por afinidade, caminho guiado, ícones com legenda e similaridades a aplicativos mais baixados nas lojas de aplicativo, como redes sociais e compras online. Tais diretrizes tendem a diminuir a curva de aprendizado de um novo sistema, diminuindo, consequentemente, a frustração do usuário.

O usuário é apresentado, inicialmente, aos cadastros dos módulos fazenda, talhões e safras. A Figura 20 apresenta o fluxo entre as telas.

Figura 20 – Fluxo inicial de navegação entre telas



Fonte: Elaborado pelo Autor, 2023.

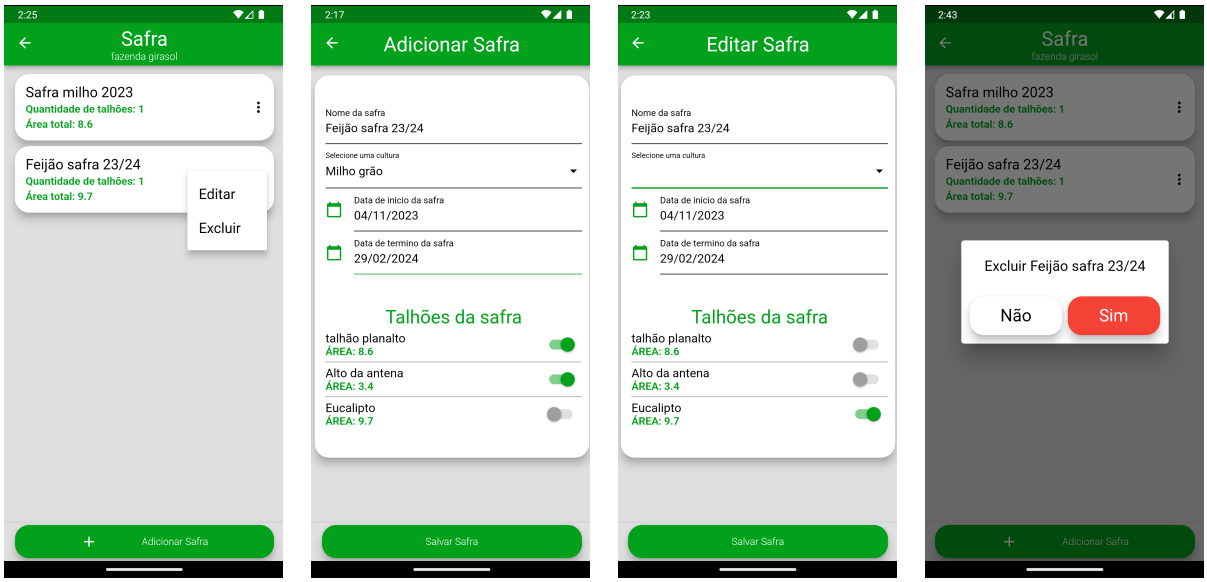
Os dados cadastrados são divididos em cartões que apresentam as informações mais relevantes para o usuário. A ação de seleção de um cartão redireciona a navegação para a próxima tela do fluxo. A cada nova tela navegada, a escolha anterior é relembrada, e a cada ação, há uma mensagem de confirmação, para o usuário reconhecer o estado de suas ações.

Após essa etapa linear de cadastros e escolhas, uma tela com duas possibilidades aparecem, sendo elas a análise de solo e as recomendações. Elas foram organizadas dessa forma, apesar de uma ser requisito da outra, para melhorar a usabilidade em caso de uso recorrente, e preparar para a adição de novas funcionalidades no futuro.

As operações possíveis para cada módulo são exemplificadas na Figura 21, e a Figura 21(a) lista todas as safras já cadastradas, informando quantos talhões fazem parte dessa safra, a área total combinada de todos e as ações editar, excluir e adicionar. As ações de adicionar e editar levam a mesma tela, que se adapta para cada circunstância.

A Figura 21(b) exibe os campos necessários para o cadastro da safra e a seleção de talhões. Os dados na tela de edição da Figura 21(c) são automaticamente completados com o cadastro anterior. Para a exclusão, uma caixa de confirmação ocupa a tela; para evitar perdas acidentais, o nome da safra compõe a caixa, havendo botões de sim e não, como demonstrado na Figura 21(d).

Figura 21 – Módulo safra



(a) Tela safra (b) Adicionar (c) Editar (d) Editar

Fonte: Elaborado pelo Autor, 2023.

O módulo de análise, apesar de compartilhar várias características do anterior, tem diferenças a serem apresentadas. A Figura 22(a) lista as análises, separando-as em ativas e inativas, que são alteradas através do menu do cartão, servindo para controle de quais serão válidas para a recomendação. A ação de adicionar encontra-se na Figura 22(b), em que o campo de seleção de talhão apenas apresenta os incluídos na safra atual.

Após o pré-cadastro, vem a etapa de inclusão dos nutrientes analisados. Como exibido na 22(c), há uma segmentação entre nutrientes químicos, físicos e micronutrientes. Cada um inclui o nome e a unidade de medida, assim como na Figura 1, e os valores apenas serão inseridos caso sejam válidos, permitindo prosseguir para próxima divisão.

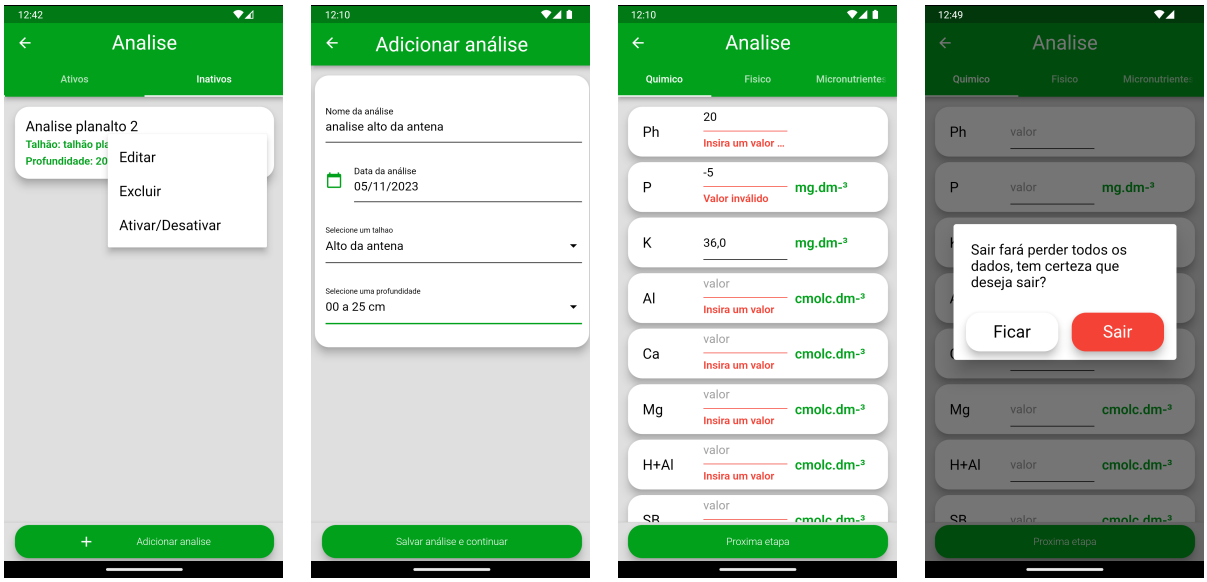
Caso seja apertado o botão de voltar antes de ser completada a inserção dos dados, a mensagem da Figura 22(d) oferece a opção de ficar ou sair e perder os dados já inseridos. Caso esteja na adição, tudo é perdido, mas, na edição, os dados antigos são mantidos.

Ao finalizar o cadastro da análise, ela é disposta na lista de análises, sendo que, ao se abrir uma análise, o aplicativo redireciona para a interpretação de nutrientes exposta na Figura 23. No topo da tela, há as informações de identificação de safra, análise e talhão, e, após, uma lista de 20 blocos, com o nome do nutriente, valor inserido, representação visual por cores e a régua de medidas.

Cada bloco conta com um gráfico que pode ter vários intervalos, os quais têm cor, tamanho e nomeação referentes às avaliações do nutriente. Dentro de cada gráfico, está uma linha azul, que representa o estado atual desse nutriente no solo.

As recomendações são adicionadas, assim como na Figura 24(a), e levam em consideração a superfície coberta por plantas, profundidade de correção da calagem, PRNT

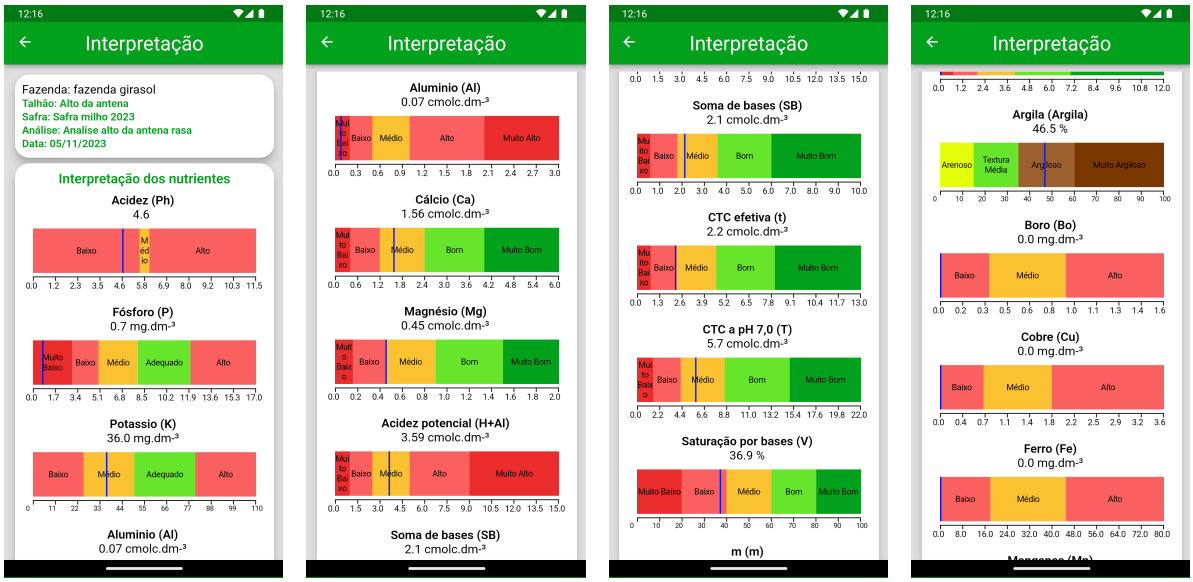
Figura 22 – Módulo análise



(a) Tela análise (b) Adicionar (c) Análise química (d) Confirmação

Fonte: Elaborado pelo Autor, 2023.

Figura 23 – Tela de interpretação



Fonte: Elaborado pelo Autor, 2023.

do calcário e produtividade desejada por hectare. Essa abordagem busca facilitar a comparação de diferentes parâmetros.

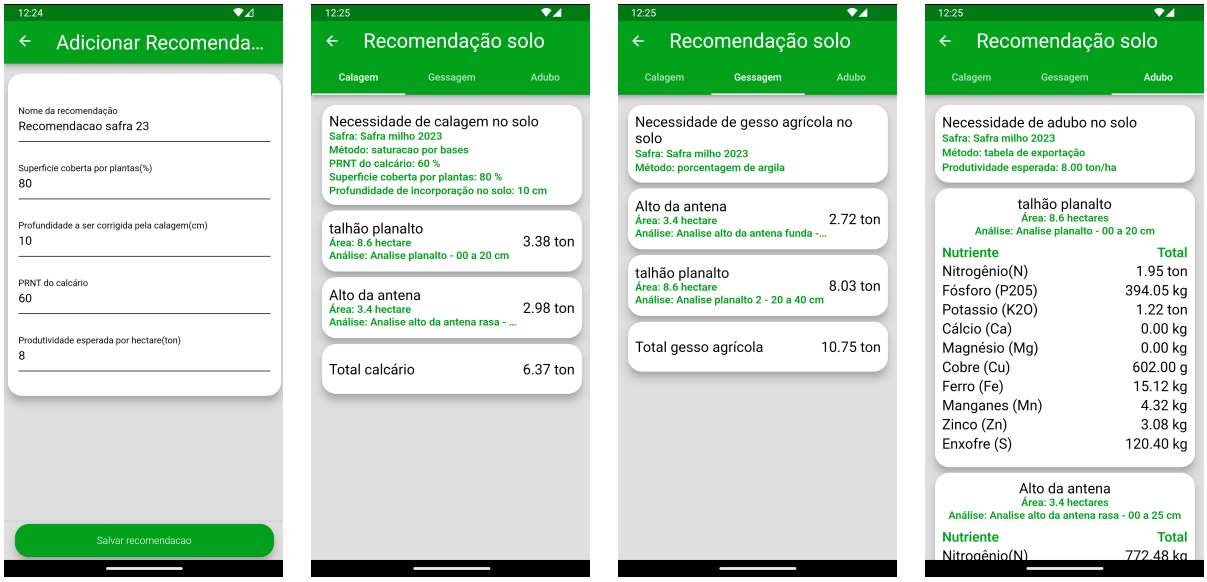
A tela de recomendação é dividida em três abas: calagem, gessagem e adubo. Cada aba conta com um cartão de informações, que indica o método utilizado, safra e dados inseridos relevantes para cada tipo de cálculo. Em seguida, é apresentado o resultado da recomendação, que se encontra em cartões referentes a cada análise ativa, relatando o talhão e sua área, assim como a análise e sua profundidade.

Os resultados de calagem e gessagem, que se encontram, respectivamente,

nas Figuras 24(b) e 24(c), consideram a área do talhão; portanto, o número resultante corresponde à quantidade a ser adquirida desses corretivos. Além disso, a unidade de medida é convertida, para facilitar a leitura.

A recomendação de adubação herda características das abas anteriores, adaptando o tamanho do cartão à quantidade de nutrientes resultantes. Apresentada na Figura 24(d), a lista de nutrientes é convertida para as variantes (entre parênteses) mais encontradas nos adubos do mercado. Esse cálculo é baseado na tabela de extração, produtividade esperada e tamanho do talhão.

Figura 24 – Módulo recomendação



(a) Adicionar (b) Calagem (c) Gessagem (d) Adubação

Fonte: Elaborado pelo Autor, 2023.

Nesta seção, foram informadas as características dos módulos mais relevantes desenvolvidas na aplicação móvel. Dentre elas, a organização de telas e funcionalidades, que foram definidas desde o projeto de interface e contribuíram para o resultado final. As considerações finais e conclusões do presente trabalho se encontram na próxima seção.

5 CONSIDERAÇÕES FINAIS

O consumo racional de fertilizantes pode trazer muitos benefícios em uma safra, por exemplo, a garantia da produtividade e custo condizente com a quantidade de produto desejada.

Considerando o crescente uso de ferramentas tecnológicas no setor rural, o presente projeto teve como objetivo desenvolver um aplicativo capaz de interpretar a análise de solo e recomendar as indicações de corretivos e fertilizantes para o cultivo das principais culturas do Brasil.

O banco de dados e aplicativo desenvolvidos podem trazer benefícios significativos à agricultura, principalmente para pequenos produtores, pela possibilidade de reduzir o custo do cálculo de recomendação de corretivos e fertilizantes.

Voltando aos objetivos específicos, foi feita a seleção das principais técnicas de corretivos e fertilizantes, e a inclusão das seis principais culturas anuais do Brasil.

O banco de dados foi definido desde o início do projeto e contribuiu para reconhecer os dados essenciais para cada módulo do aplicativo. Também possibilitará a adição de futuras tabelas de extração para novos híbridos e técnicas de cálculo, a fim de manter métodos modernos e relevantes.

O aplicativo foi desenvolvido para Android, mas, utilizando-se um *framework* multiplataforma, o mesmo código funciona para iOS. O *software* gerado tem capacidade de atender à demanda do mercado, com recomendações adequadas para a nutrição do solo.

5.1 Publicações realizadas

O trabalho, originado das metodologias propostas, foi apresentado em duas situações:

1. REZENDE, J.G.F., RIBEIRO, M.R., OLIVEIRA, C.M. **Desenvolvimento de um aplicativo para recomendação de corretivos e fertilizantes**. 4º lugar na IV Olimpíada de Inovação do IFMG. dez. 2022.

2. REZENDE, J.G.F., RIBEIRO, M.R., OLIVEIRA, C.M. **Desenvolvimento de um aplicativo para recomendação de corretivos e fertilizantes**.XV Jornada Científica do IFMG. out. 2023.

5.2 Trabalhos futuros

Alguns objetivos que estão planejados para o mais breve possível são a disponibilização na loja de aplicativos Android; a geração de um arquivo da recomendação, para facilitar compartilhamento e impressão dos dados gerados; e o suporte a mais sistemas operacionais.

Além disso, o presente projeto tem capacidade de se transformar em um produto comercial e continuará em andamento. Além disso, o projeto está, atualmente, competindo como finalista na quinta edição da Olimpíada de Inovação do IFMG, apresentando perspectiva de registro de propriedade intelectual.

Assim como os fertilizantes, os agrotóxicos correspondem a grande parte do custo de uma safra. Portanto, fazer esse cálculo pode ser uma adição valiosa ao aplicativo. Desse modo, com o resultado dos cálculos, permitir que o usuário adquira esses insumos, através desse sistema, incluindo as ofertas dos fabricantes de fertilizante locais, ressignifica o aplicativo em produto.

O Optical Character Recognition (OCR) é um método de leitura de dados por meio de uma câmera, que também está dentre as metas de expansão, pela facilitação da inclusão de análises de solo, já que é um processo que demanda grande tempo.

REFERÊNCIAS

ALCARDE, J. C. **Corretivos da acidez dos solos**: características e interpretações técnicas. 6. ed. São Paulo: ANDA, 1992. Disponível em: https://anda.org.br/wp-content/uploads/2019/03/boletim_06.pdf. Acesso em: 24 jun. 2023.

ALVARES, V. V. H. *et al.* Interpretação dos resultados das análises de solos. In: **Recomendação para o uso de corretivos e fertilizantes em Minas Gerais**: 5.ª Aproximação. Edição: A. C. Ribeiro, P. T. G. Guimaraes e V. V. H. Alvarez. 5. ed. Viçosa: Comissão de Fertilidade do Solo do Estado de Minas Gerais, 1999.

ASSOCIAÇÃO DE AGRICULTORES E IRRIGANTES DA BAHIA (AIBA). **Aplicativos rurais**: conheça apps gratuitos para a Agricultura e a Pecuária. 2018. Disponível em: https://www.agrolink.com.br/noticias/aplicativos-rurais%E2%80%94conheca-apps-gratuitos-para-a-agricultura-e-a-pecuaria_406555.html. Acesso em: 28 mar. 2023.

BARBOSA, S. D. J. *et al.* **Interação Humano-Computador e Experiência do Usuário**. Autopublicação, 2021.

BIØRN-HANSEN, A.; GRØNLI, T.-M.; GHINEA, G. A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development. **ACM Computing Surveys (CSUR)**, ACM, New York, v. 51, n. 5, p. 1–34, 2018.

BORIN, A. L. D. C.; FERREIRA, G. B.; CARVALHO, M. d. C. S. Adubação do algodoeiro no ambiente de Cerrado. **Comunicado Técnico, Embrapa**, Campina Grande, 2014. Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1012109. Acesso em: 1 dez. 2023.

COMPANHIA NACIONAL DE ABASTECIMENTO (CONAB). **Acompanhamento da safra brasileira**. Grãos. 2023a. Disponível em: https://www.conab.gov.br/info-agro/safras/graos/boletim-da-safra-de-graos. Acesso em: 1 dez. 2023.

_____. **Planilhas de custo de produção**. 2023b. Disponível em: https://www.conab.gov.br/info-agro/custos-de-producao/planilhas-de-custo-de-producao. Acesso em: 1 dez. 2023.

CRAVO, M. d. S.; VIEGAS, I. d. J.; BRASIL, E. C. **Recomendações de calagem e adubação para o estado do Pará**. 2. ed. Brasília: Embrapa, 2020.

DELVAUX, J. C.; SILVEIRA, L. F. V. Desenvolvimento de aplicativo móvel para recomendação de corretivos e fertilizantes para o cultivo do milho no Estado de Minas Gerais. **Revista Inova:** Ciência & Tecnologia / Innovative Science & Technology Journal, Uberaba, v. 7, p. 29–33, 2021. Disponível em: https://periodicos.iftm.edu.br/index.php/inova/issue/view/35. Acesso em: 25 mar. 2023.

FAGERIA, N. K. *et al.* Seja o doutor do seu arroz. **Potafos**, Piracicaba, v. 9, 1995. Disponível em: https://www.npct.com.br/npctweb/npct.nsf/article/BRS-3145. Acesso em: 1 dez. 2023.

FAYZULLAEV, J. **Native-like Cross-Platform Mobile Development**: Multi-OS Engine & Kotlin Native vs Flutter. 2018. Tese (Doutorado em Tecnologia da Informação) – South-Eastern Finland University of Applied Sciences, Kouvola, Finland, 2018. Disponível em: https://urn.fi/URN:NBN:fi:amk-2018053011229. Acesso em: 4 abr. 2023.

FLUTTER. **Framework multiplataforma de código aberto**. 2022. Disponível em: https://flutter.dev/. Acesso em: 24 jun. 2023.

FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS (FAO). **Production indices**. 2020. Disponível em: https://www.fao.org/faostat/en/#data. Acesso em: 21 jun. 2023.

FORMIGONI, I. **Consumo de fertilizantes no Brasil em 2021 nunca foi tão alto**: confira dados. 2021. Disponível em: https://www.farmnews.com.br/mercado/consumo-de-fertilizantes-no-brasil/. Acesso em: 4 abr. 2023.

FREITAS, R. H. M. D. *et al.* Adubação fosfatada na producão de mudas de Cassia ferruginea e Cassia Grandis. **Nucleus**, Viçosa, v. 15, n. 1, p. 41–49, 2018.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. 1. ed. Porto Alegre: Plageder, 2009.

GUBIANI, P. I. *et al.* CADUB GHF: um programa computacional para cálculo da quantidade de fertilizantes e corretivos da acidez do solo para culturas produtoras de grãos, hortaliças e forrageiras. **Ciência Rural**, Santa Maria, v. 37, n. 4, p. 1161–1165, 2007.

INTERNATIONAL DATA CORPORATION (IDC). **Smartphone Market Share**. 2023. Disponível em: https://www.idc.com/promo/smartphone-market-share. Acesso em: 11 nov. 2023.

JARDIM, R. L. **Apresentação e análise do ecossistema para desenvolvimento mobile em multiplataforma**. 2021. Tese (Doutorado em Engenharia Informática) – Universidade da Madeira, Madeira, Portugal, 2021.

EL-KASSAS, W. S. *et al.* Taxonomy of Cross-Platform Mobile Applications Development Approaches. **Ain Shams Engineering Journal**, v. 8, n. 2, p. 163–190, 2015. Disponível em: https://www.sciencedirect.com/science/article/pii/S2090447915001276. Acesso em: 25 mar. 2023.

LABORATÓRIO DE SOLOS. **Relatório da análise de solo**. IFMG - Campus Bambuí. 2017.

LONGUI, F.; VITÓRIA, E. L. Desenvolvimento de um software para recomendação de calagem e adubação de pastagens. **Enciclopédia Biosfera**, v. 7, n. 13, p. 362–370, 2011. Disponível em: https://www.conhecer.org.br/ojs/index.php/biosfera/article/view/4124. Acesso em: 24 jun. 2023.

MACHADO NETO, O. J. **Usabilidade da interface de dispositivos móveis**: heurísticas e diretrizes para o design. 2013. Dissertação (Mestrado em Ciências de Computação) – Universidade de São Paulo (USP), São Carlos, 2013.

MARTHA JÚNIOR, G. B.; VILELA, L.; SOUSA, D. M. G. de. **Cerrado**: uso eficiente de corretivos e fertilizantes em pastagens. 1. ed. Planaltina: Embrapa Cerrados, 2007. Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1113533. Acesso em: 25 mar. 2023.

MOSAIC FERTILIZANTES. **Nutrição de Safras**. 2023. Disponível em: https://nutricaodesafras.com.br/extracao-e-exportacao-de-nutrientes. Acesso em: 7 nov. 2023.

NASCIMENTO, R. d. S. d.; SALAME, M. F. A.; TAVARES, F. d. A. Aplicativo móvel para recomendação de adubação e calagem para produção de mandioca no Amazonas. In: CONGRESSO BRASILEIRO DE AGROINFORMÁTICA (SBIAGRO), X., Ponta Grossa. Disponível em: https://www.alice.cnptia.embrapa.br/alice/handle/doc/1027836. Acesso em: 1 dez. 2023.

PAULA, J. A. d. A. **Desenvolvimento e verificação de um sistema computacional para cálculo da adubação/fertirrigação de melão e melancia**. 2007. Dissertação (Mestrado em Agricultura Tropical) – Universidade Federal Rural do Semi-Árido, Mossoró, 2007.

RAMOS, L. A. *et al.* Reatividade de corretivos da acidez e condicionadores de solo em colunas de lixiviação. **Revista Brasileira de Ciência do Solo**, v. 30, p. 849–857, 2006.

RESENDE, A. V. d.; GUTIERREZ, A. M. *et al.* Requerimentos nutricionais do milho para a produção de silagem. **Circular Técnica**, Embrapa Milho e Sorgo, Sete Lagoas, v. 221, 2016. Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1063399. Acesso em: 25 mar. 2023.

RESENDE, A. V. d.; SILVA, C. G. M. *et al.* Indicadores de demanda de macro e micronutrientes por híbridos modernos de milho. **Circular Técnica**, Embrapa Milho e Sorgo, Sete Lagoas, v. 220, 2016. Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1063622. Acesso em: 25 mar. 2023.

ROCHA, H. V. d.; BARANAUSKAS, M. C. C. **Design e avaliação de interfaces humano-computador**. Campinas: Unicamp, 2003.

SABBAGH, R. **Scrum**: Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013.

SILVA, F. C. da. **Manual de análises químicas de solos, plantas e fertilizantes**. 2. ed. Brasília: Embrapa Informação Tecnológica, 2009. Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/330496. Acesso em: 25 mar. 2023.

SILVA, T. M. D. da. **Software para recomendação de calagem e adubação do tomateiro no estado de Minas Gerais**. 2012. Dissertação (Mestrado em Ciências Agrárias) – Universidade José do Rosário Vellano (UNIFENAS), Alfenas, 2012. Disponível em: http://tede2.unifenas.br:8080/jspui/handle/jspui/53. Acesso em: 25 mar. 2023.

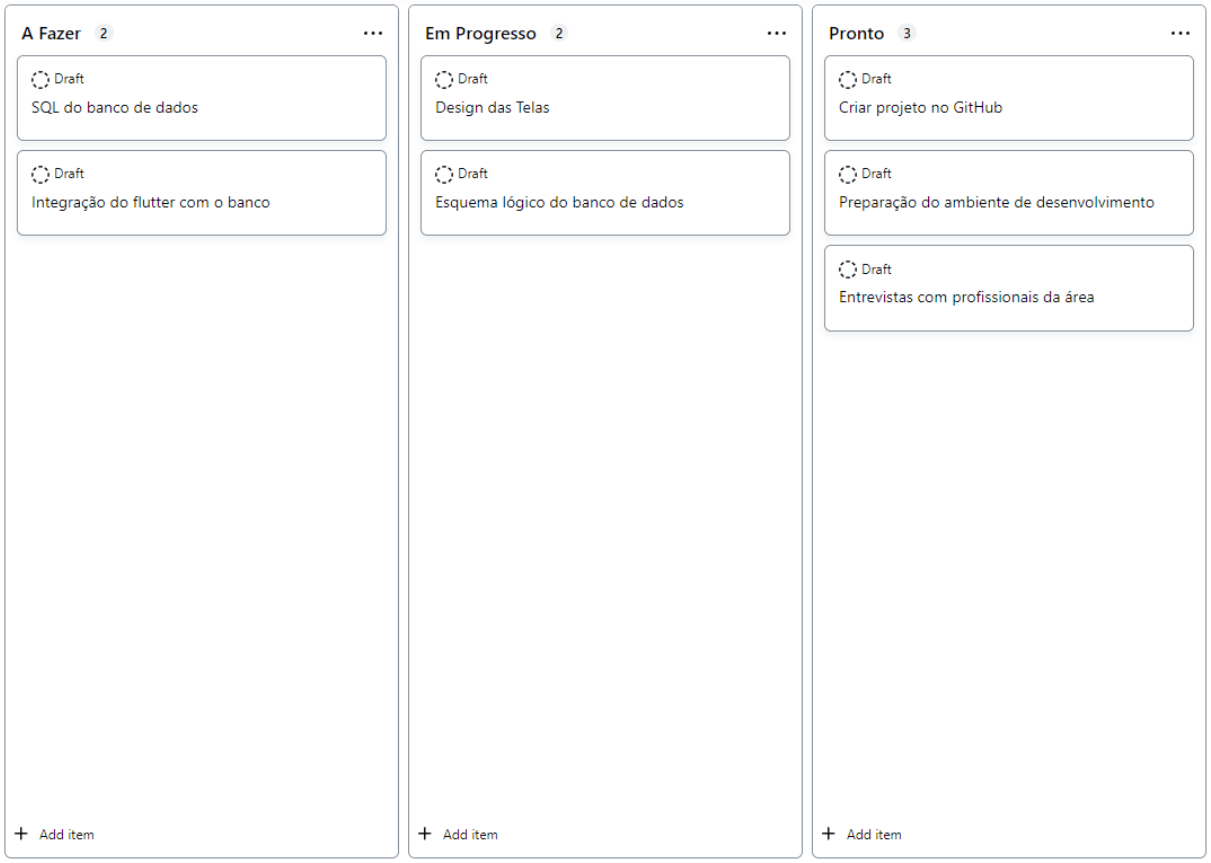
SQLITE CONSORTIUM. **SQLite home page**. 2023. Disponível em: https://sqlite.org/. Acesso em: 6 set. 2023.

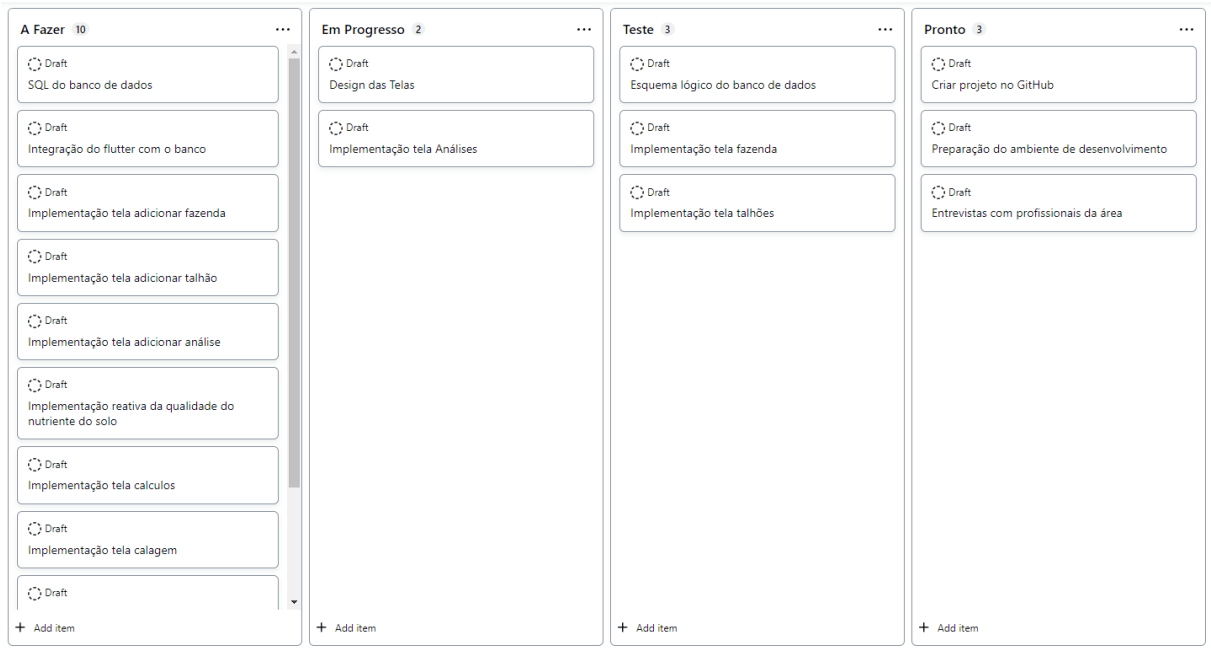
TANAKA, A. H. A. *et al.* Fertiup! aplicativo de recomendações de adubação e calagem para plantas medicinais. **Brazilian Journal of Development**, Curitiba, v. 6, n. 1, p. 430–440, 2020. Disponível em: https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/5855. Acesso em: 6 set. 2023.

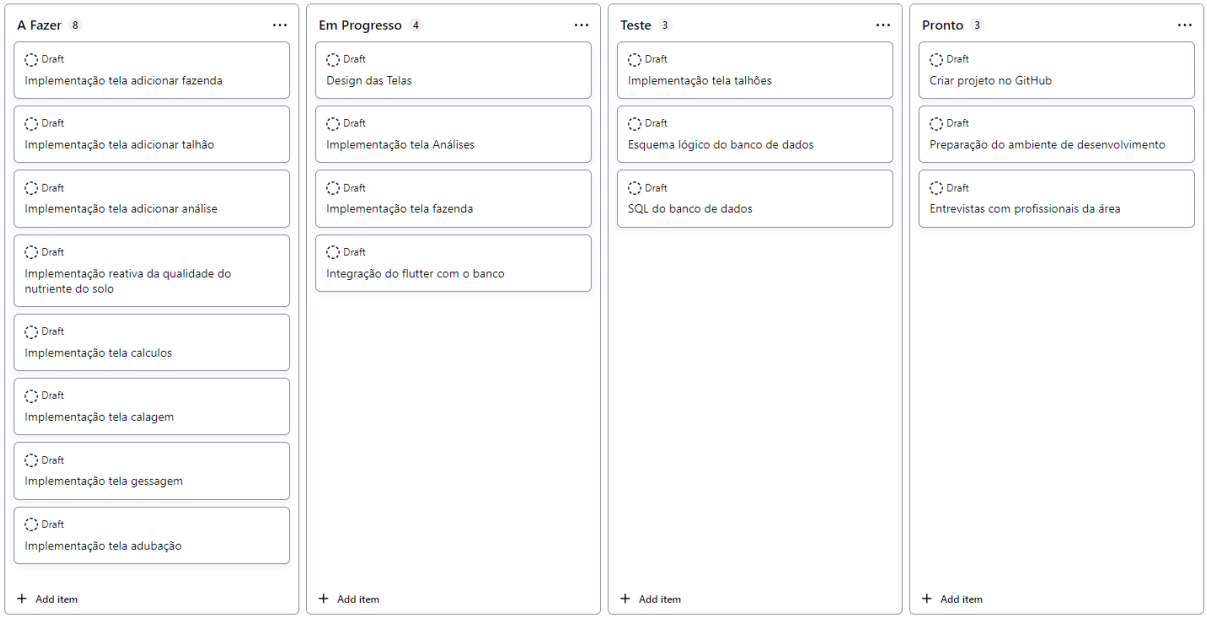
WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. 2. ed. Rio de Janeiro: Elsevier, 2009.

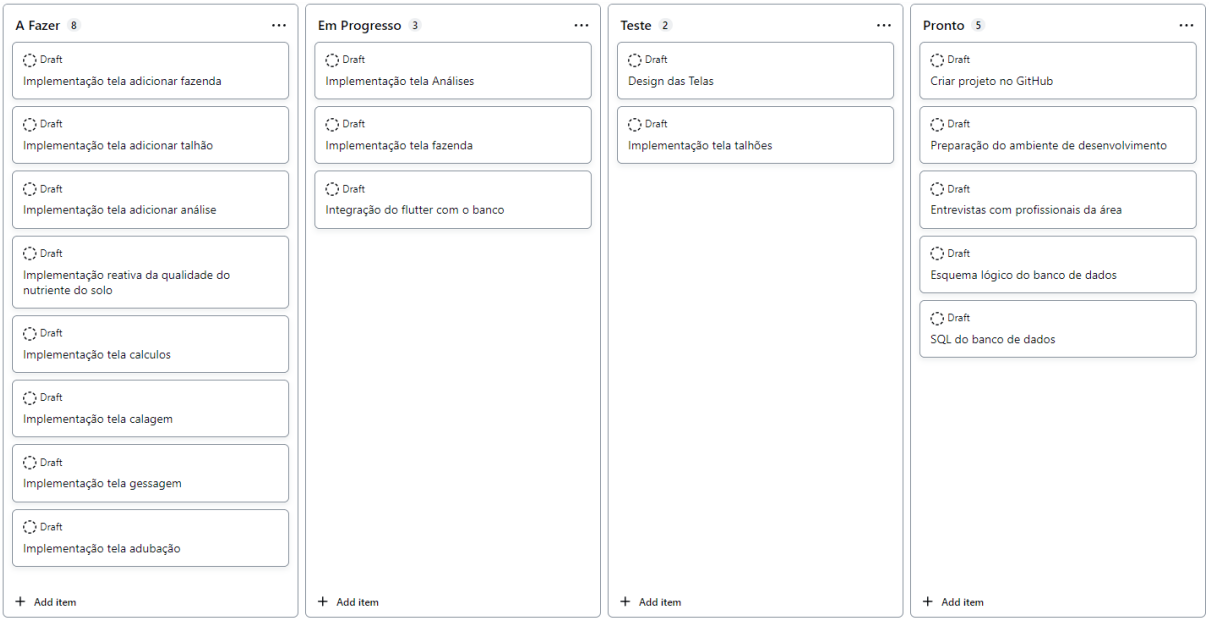
APÊNDICES

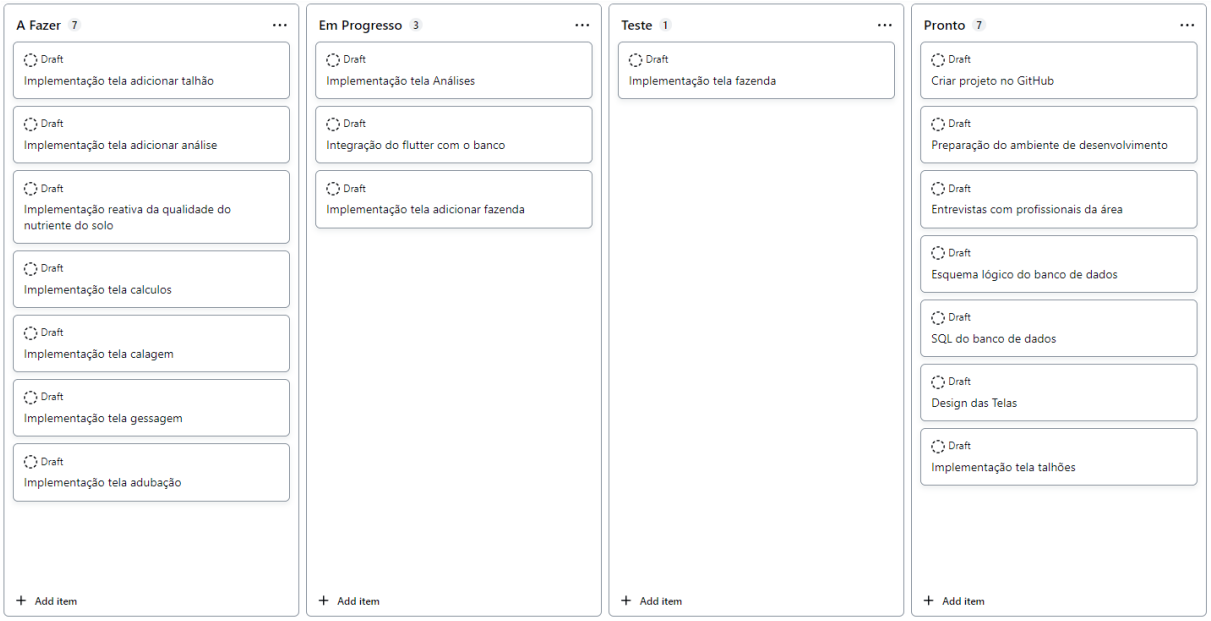
APÊNDICE A – QUADRO *KANBAN*

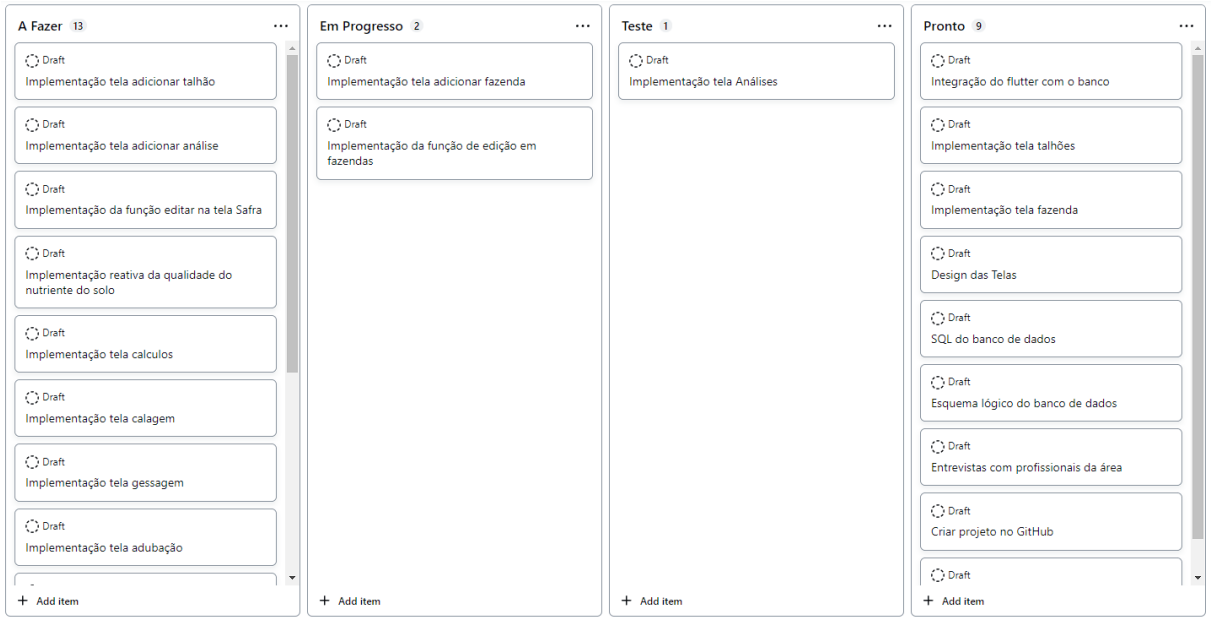


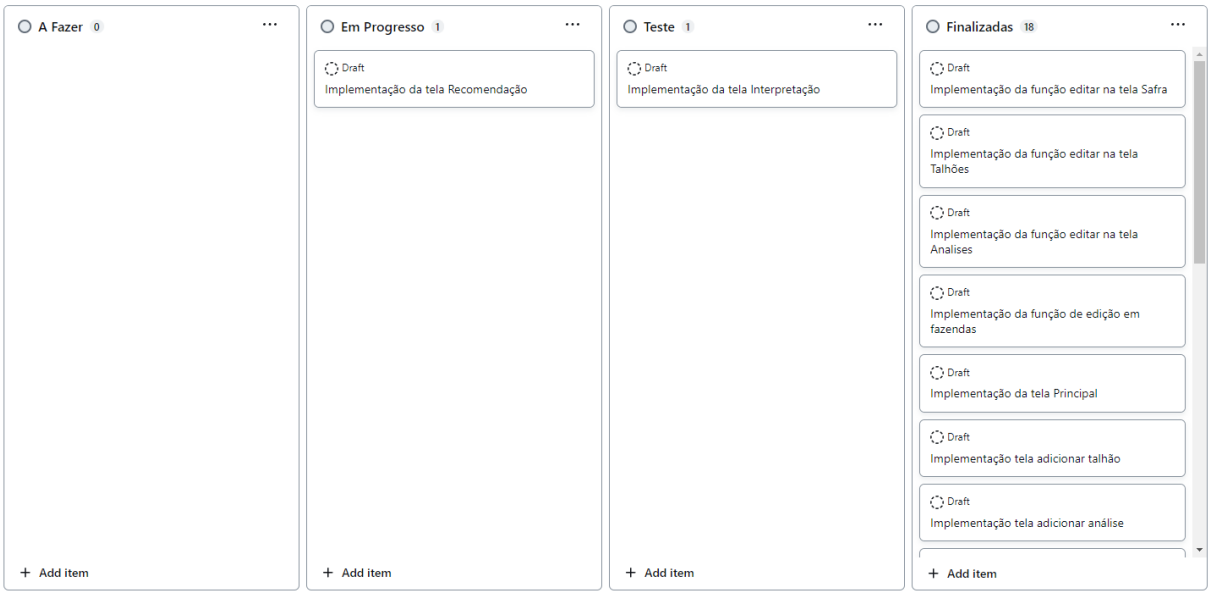


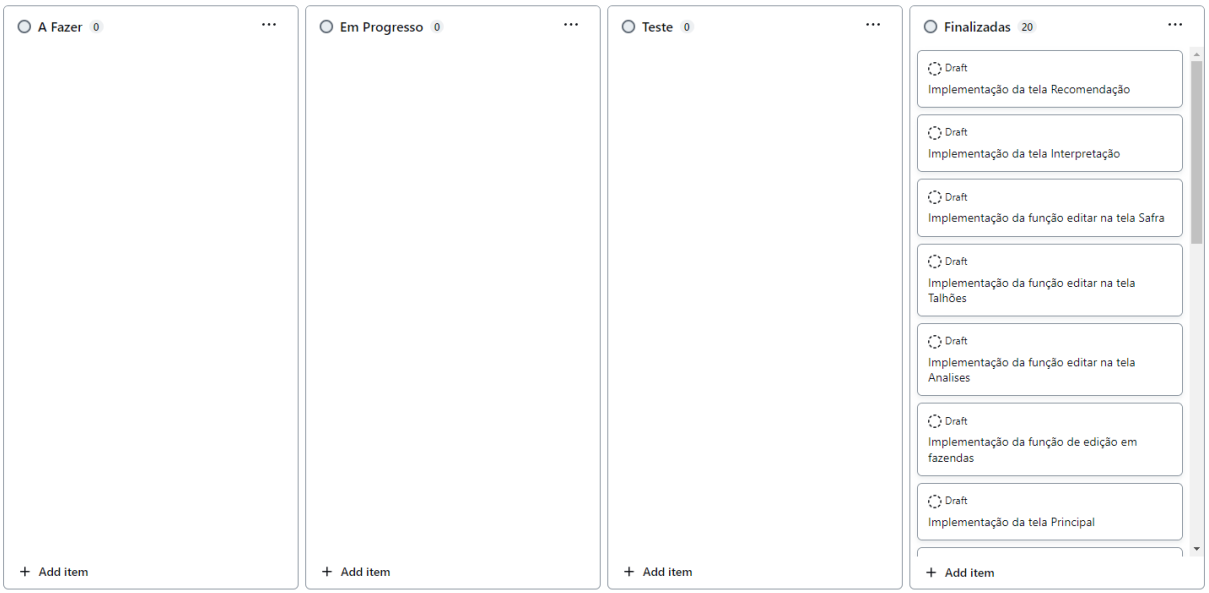

A Fazer 8
Draft
Implementação tela adicionar fazenda
Draft
Implementação tela adicionar talhão
Draft
Implementação tela adicionar análise
Draft
Implementação reativa da qualidade do nutriente do solo
Draft
Implementação tela calculos
Draft
Implementação tela calagem
Draft
Implementação tela gessagem
Draft
Implementação tela adubação
Add item
Em Progresso 4
Draft
Design das Telas
Draft
Implementação tela Análises
Draft
Implementação tela fazenda
Draft
Integração do flutter com o banco
Add item
Teste 3
Draft
Implementação tela talhões
Draft
Esquema lógico do banco de dados
Draft
SQL do banco de dados
Add item
Pronto 3
Draft
Criar projeto no GitHub
Draft
Preparação do ambiente de desenvolvimento
Draft
Entrevistas com profissionais da área
Add item


A Fazer 8
Draft
Implementação tela adicionar fazenda
Draft
Implementação tela adicionar talhão
Draft
Implementação tela adicionar análise
Draft
Implementação reativa da qualidade do nutriente do solo
Draft
Implementação tela calculos
Draft
Implementação tela calagem
Draft
Implementação tela gessagem
Draft
Implementação tela adubação
Add item
Em Progresso 3
Draft
Implementação tela Análises
Draft
Implementação tela fazenda
Draft
Integração do flutter com o banco
Add item
Teste 2
Draft
Design das Telas
Draft
Implementação tela talhões
Add item
Pronto 5
Draft
Criar projeto no GitHub
Draft
Preparação do ambiente de desenvolvimento
Draft
Entrevistas com profissionais da área
Draft
Esquema lógico do banco de dados
Draft
SQL do banco de dados
Add item

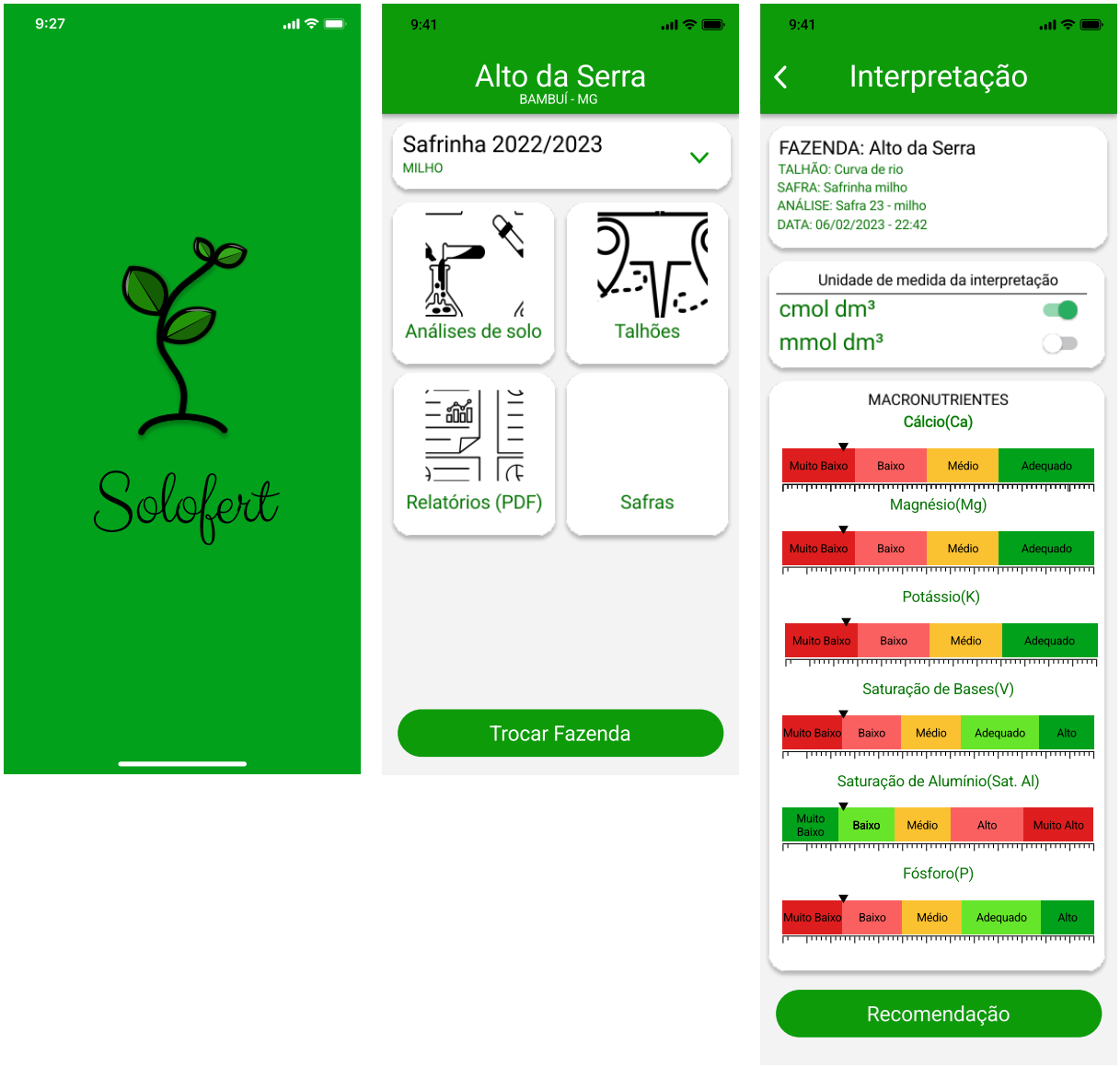

A Fazer 7
Draft
Implementação tela adicionar talhão
Draft
Implementação tela adicionar análise
Draft
Implementação reativa da qualidade do nutriente do solo
Draft
Implementação tela calculos
Draft
Implementação tela calagem
Draft
Implementação tela gessagem
Draft
Implementação tela adubação
Add item
Em Progresso 3
Draft
Implementação tela Análises
Draft
Integração do flutter com o banco
Draft
Implementação tela adicionar fazenda
Add item
Teste 1
Draft
Implementação tela fazenda
Add item
Pronto 7
Draft
Criar projeto no GitHub
Draft
Preparação do ambiente de desenvolvimento
Draft
Entrevistas com profissionais da área
Draft
Esquema lógico do banco de dados
Draft
SQL do banco de dados
Draft
Design das Telas
Draft
Implementação tela talhões
Add item

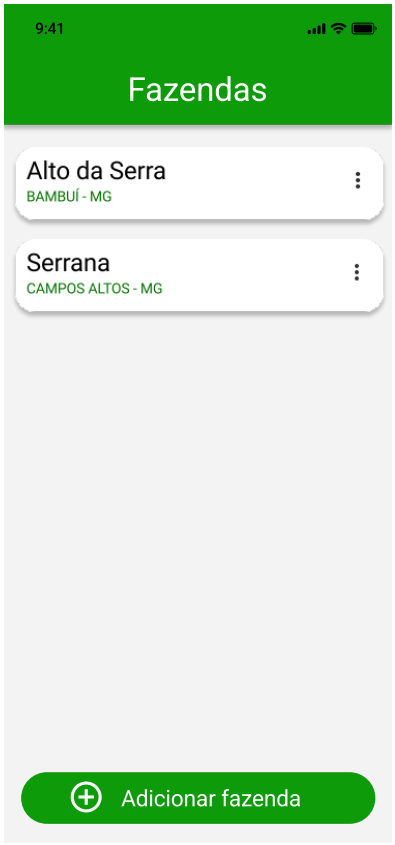

A Fazer 13
Draft
Implementação tela adicionar talhão
Draft
Implementação tela adicionar análise
Draft
Implementação da função editar na tela Safra
Draft
Implementação reativa da qualidade do nutriente do solo
Draft
Implementação tela calculos
Draft
Implementação tela calagem
Draft
Implementação tela gessagem
Draft
Implementação tela adubação
Add item
Em Progresso 2
Draft
Implementação tela adicionar fazenda
Draft
Implementação da função de edição em fazendas
Add item
Teste 1
Draft
Implementação tela Análises
Add item
Pronto 9
Draft
Integração do flutter com o banco
Draft
Implementação tela talhões
Draft
Implementação tela fazenda
Draft
Design das Telas
Draft
SQL do banco de dados
Draft
Esquema lógico do banco de dados
Draft
Entrevistas com profissionais da área
Draft
Criar projeto no GitHub
Draft
Add item

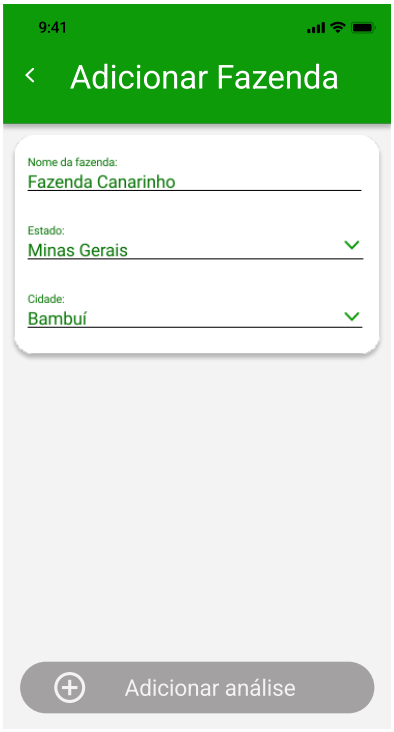

A Fazer 0
Add item
Em Progresso 1
Draft
Implementação da tela Recomendação
Add item
Teste 1
Draft
Implementação da tela Interpretação
Add item
Finalizadas 18
Draft
Implementação da função editar na tela Safra
Draft
Implementação da função editar na tela Talhões
Draft
Implementação da função editar na tela Analises
Draft
Implementação da função de edição em fazendas
Draft
Implementação da tela Principal
Draft
Implementação tela adicionar talhão
Draft
Implementação tela adicionar análise
Add item

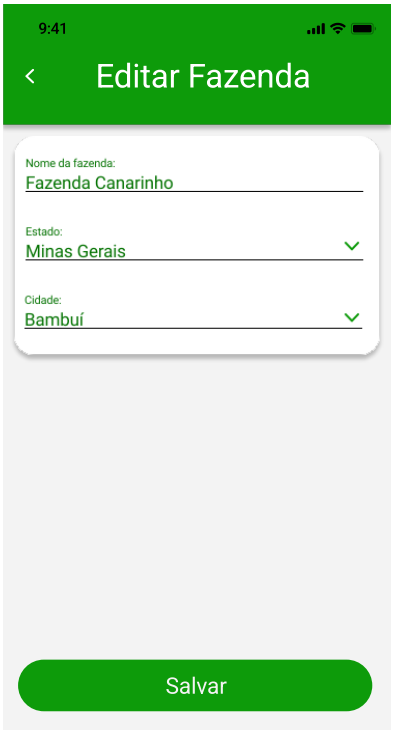

A Fazer 0
Add item
Em Progresso 0
Add item
Teste 0
Add item
Finalizadas 20
Draft
Implementação da tela Recomendação
Draft
Implementação da tela Interpretação
Draft
Implementação da função editar na tela Safra
Draft
Implementação da função editar na tela Talhões
Draft
Implementação da função editar na tela Analises
Draft
Implementação da função de edição em fazendas
Draft
Implementação da tela Principal
Add item

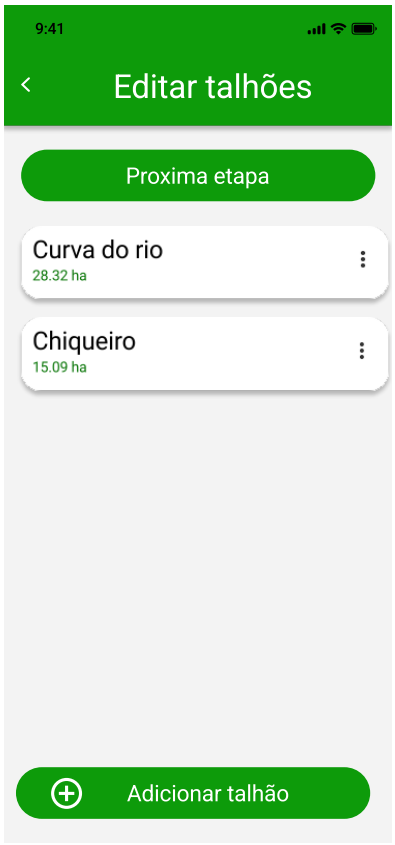
APÊNDICE B – PROJETO DE INTERFACE

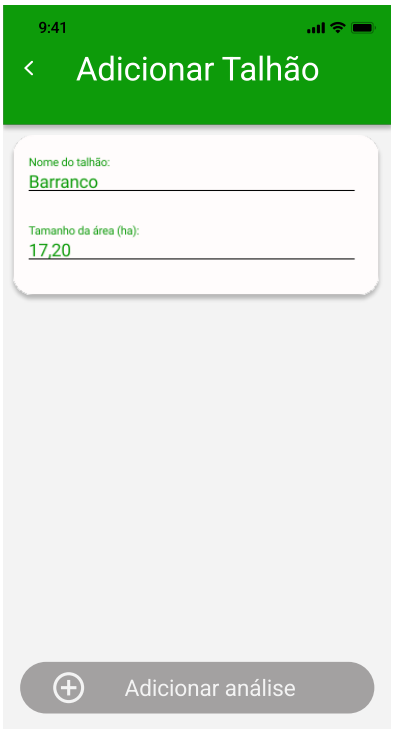


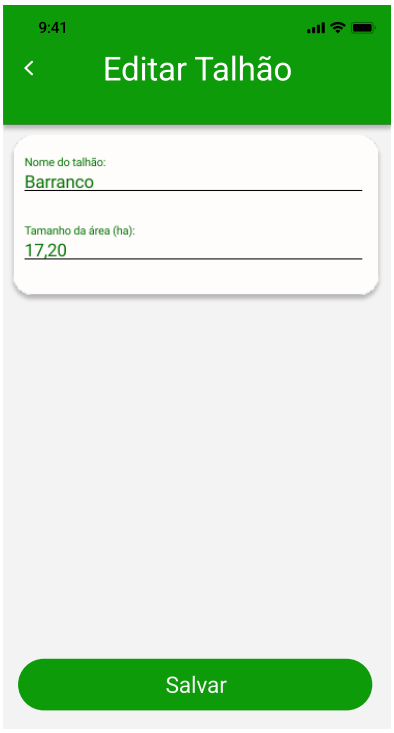

9:41
Fazendas
Alto da Serra
BAMBUÍ - MG
Serrana
CAMPOS ALTOS - MG
Adicionar fazenda

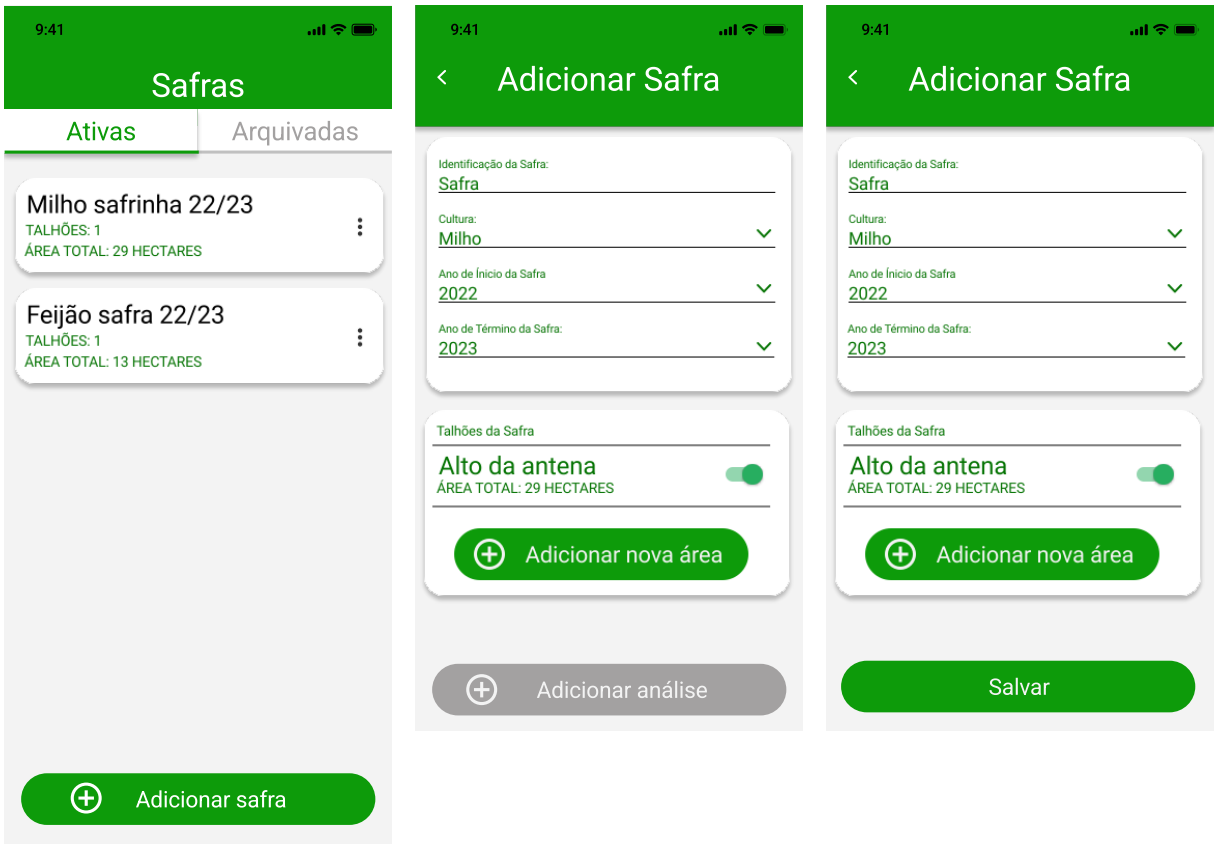

9:41
Adicionar Fazenda
Nome da fazenda:
Fazenda Canarinho
Estado:
Minas Gerais
Cidade:
Bambuí
Adicionar análise

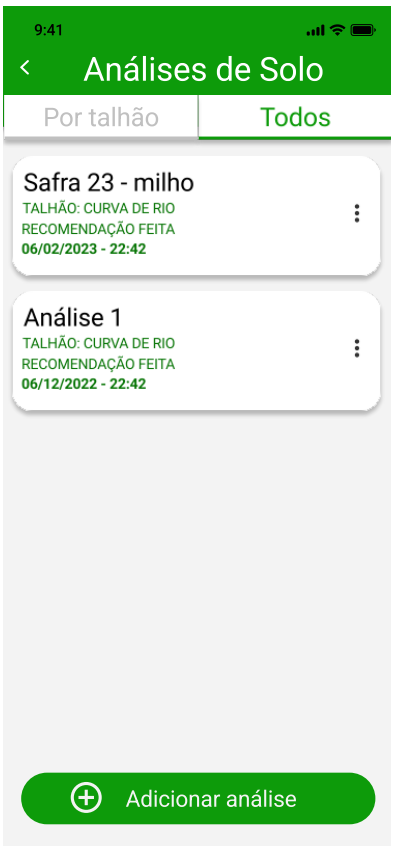

9:41
Editar Fazenda
Nome da fazenda:
Fazenda Canarinho
Estado:
Minas Gerais
Cidade:
Bambuí
Salvar

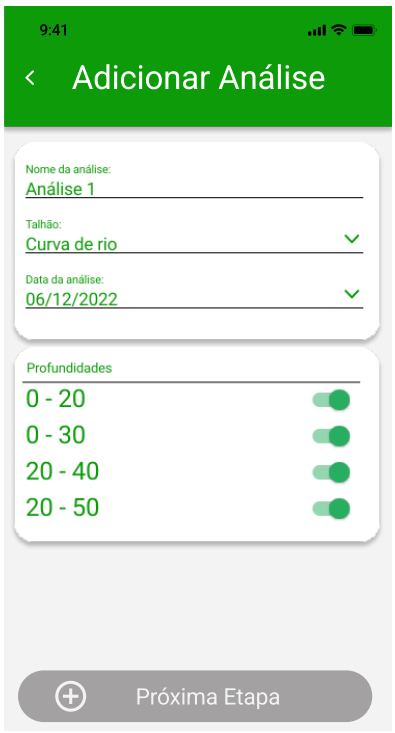

9:41
Editar talhões
Proxima etapa
Curva do rio
28.32 ha
Chiqueiro
15.09 ha
Adicionar talhão

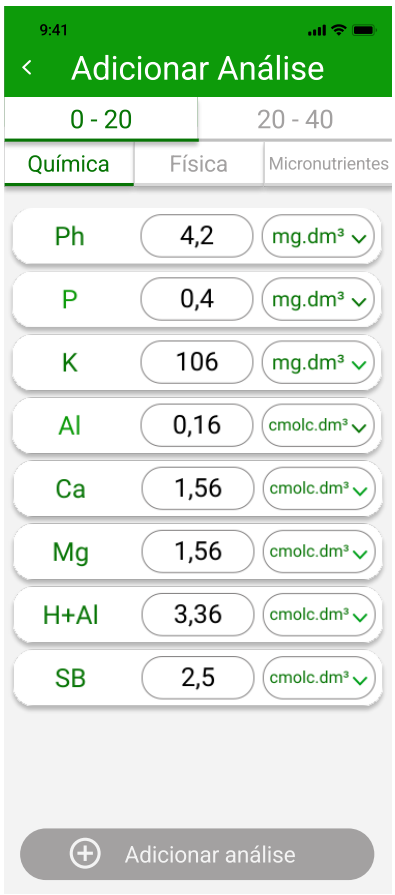

9:41
Adicionar Talhão
Nome do talhão:
Barranco
Tamanho da área (ha):
17,20
Adicionar análise

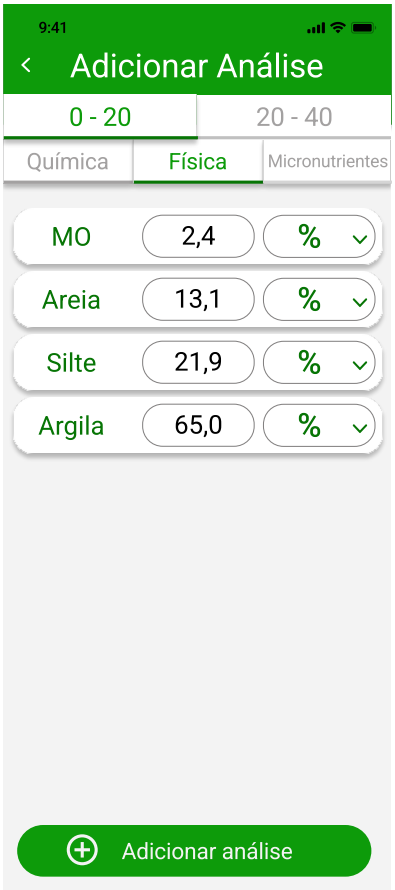

9:41
Editar Talhão
Nome do talhão:
Barranco
Tamanho da área (ha):
17,20
Salvar

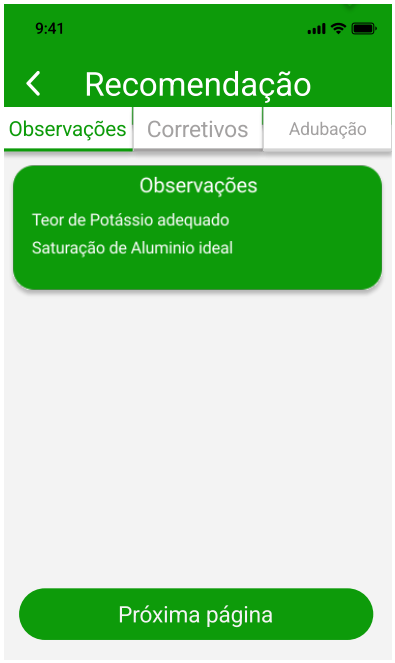

9:41
Safras
Ativas
Arquivadas
Milho safrinha 22/23
TALHÕES: 1
ÁREA TOTAL: 29 HECTARES
Feijão safra 22/23
TALHÕES: 1
ÁREA TOTAL: 13 HECTARES
Adicionar safra
9:41
Adicionar Safra
Identificação da Safra:
Safra
Cultura:
Milho
Ano de Início da Safra
2022
Ano de Término da Safra:
2023
Talhões da Safra
Alto da antena
ÁREA TOTAL: 29 HECTARES
Adicionar nova área
Adicionar análise
9:41
Adicionar Safra
Identificação da Safra:
Safra
Cultura:
Milho
Ano de Início da Safra
2022
Ano de Término da Safra:
2023
Talhões da Safra
Alto da antena
ÁREA TOTAL: 29 HECTARES
Adicionar nova área
Salvar

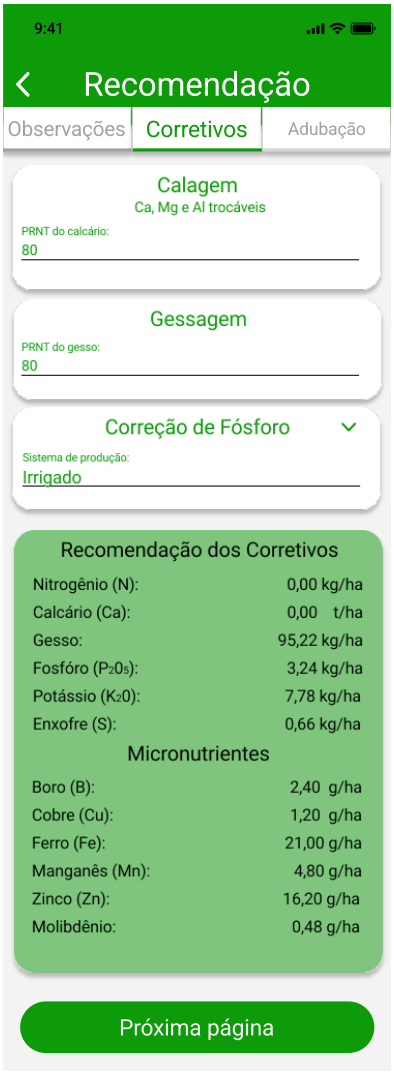

9:41
Análises de Solo
Por talhão
Todos
Safra 23 - milho
TALHÃO: CURVA DE RIO
RECOMENDAÇÃO FEITA
06/02/2023 - 22:42
Análise 1
TALHÃO: CURVA DE RIO
RECOMENDAÇÃO FEITA
06/12/2022 - 22:42
Adicionar análise

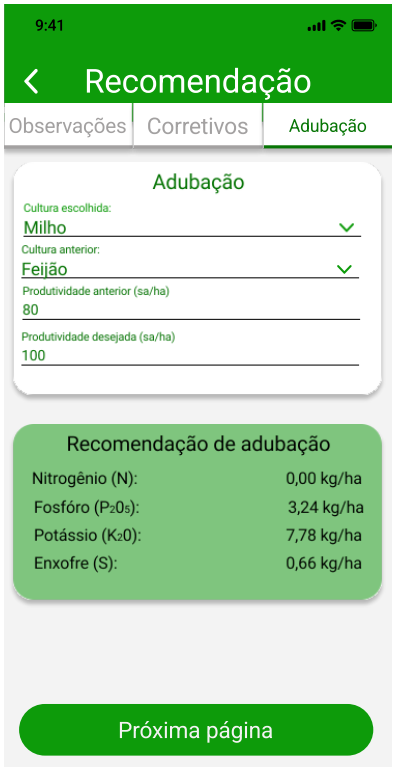

9:41
Adicionar Análise
Nome da análise:
Análise 1
Talhão:
Curva de rio
Data da análise:
06/12/2022
Profundidades
0 - 20
0 - 30
20 - 40
20 - 50
Próxima Etapa

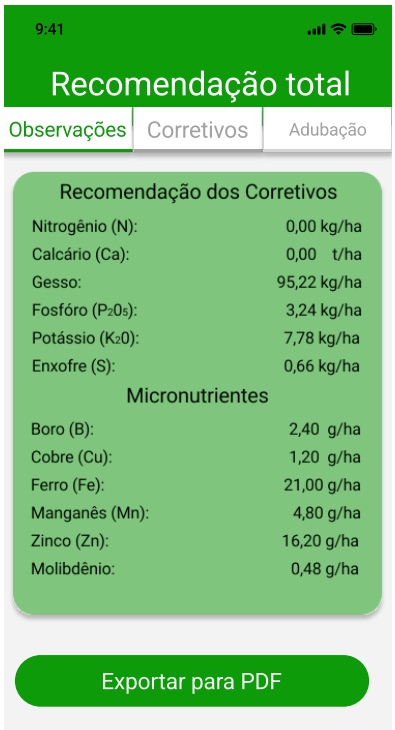

9:41
Adicionar Análise
0 - 20
20 - 40
Química
Física
Micronutrientes
Ph 4,2 mg.dm³
P 0,4 mg.dm³
K 106 mg.dm³
Al 0,16 cmolc.dm³
Ca 1,56 cmolc.dm³
Mg 1,56 cmolc.dm³
H+Al 3,36 cmolc.dm³
SB 2,5 cmolc.dm³
Adicionar análise


9:41
Adicionar Análise
0 - 20
20 - 40
Química
Física
Micronutrientes
MO 2,4 %
Areia 13,1 %
Silte 21,9 %
Argila 65,0 %
Adicionar análise

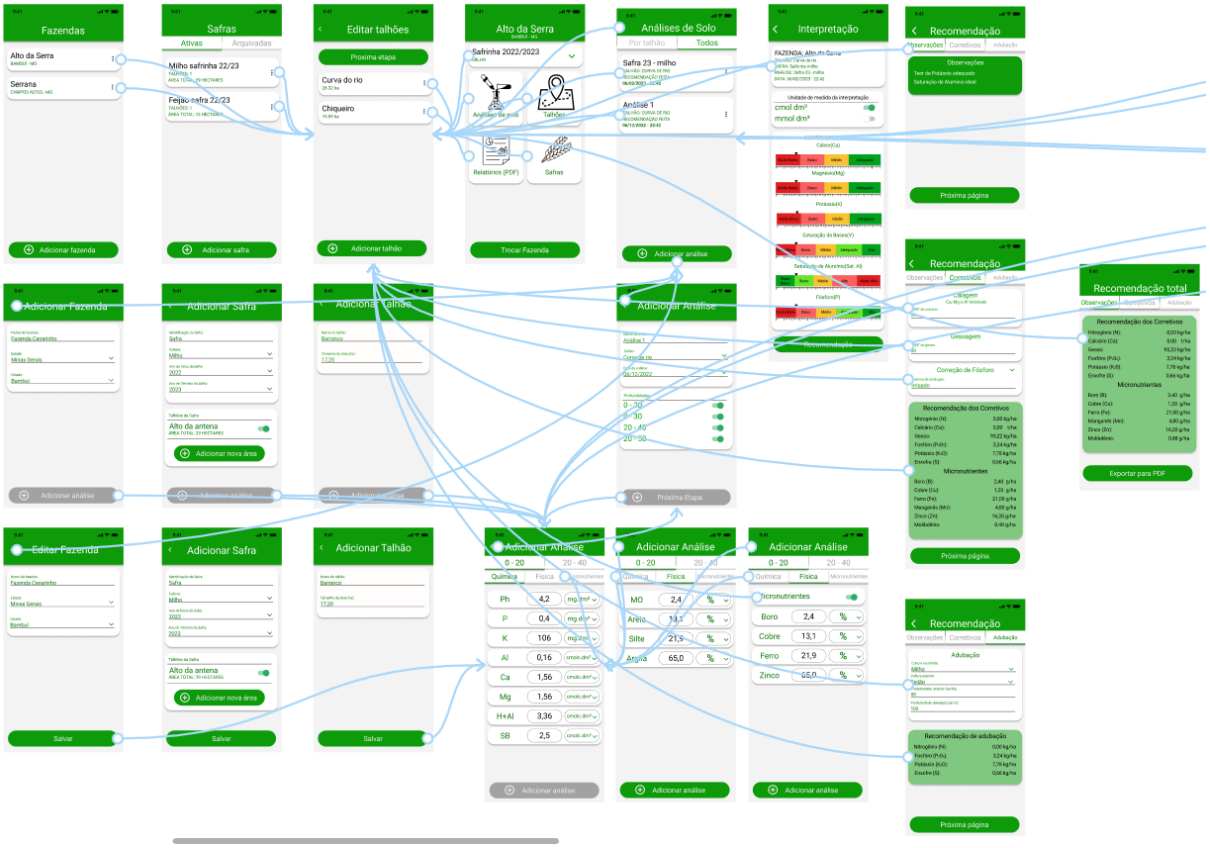

9:41
Recomendação
Observações
Corretivos
Adubação
Observações
Teor de Potássio adequado
Saturação de Aluminio ideal
Próxima página


9:41
Recomendação
Observações
Corretivos
Adubação
Calagem
Ca, Mg e Al trocáveis
PRNT do calcário:
80
Gessagem
PRNT do gesso:
80
Correção de Fósforo
Sistema de produção:
Irrigado
Recomendação dos Corretivos
Nitrogênio (N): 0,00 kg/ha
Calcário (Ca): 0,00 t/ha
Gesso: 95,22 kg/ha
Fosfóro (P2O5): 3,24 kg/ha
Potássio (K20): 7,78 kg/ha
Enxofre (S): 0,66 kg/ha
Micronutrientes
Boro (B): 2,40 g/ha
Cobre (Cu): 1,20 g/ha
Ferro (Fe): 21,00 g/ha
Manganês (Mn): 4,80 g/ha
Zinco (Zn): 16,20 g/ha
Molibdênio: 0,48 g/ha
Próxima página


9:41
Recomendação
Observações
Corretivos
Adubação
Adubação
Cultura escolhida:
Milho
Cultura anterior:
Feijão
Produtividade anterior (sa/ha)
80
Produtividade desejada (sa/ha)
100
Recomendação de adubação
Nitrogênio (N): 0,00 kg/ha
Fosfóro (P2O5): 3,24 kg/ha
Potássio (K20): 7,78 kg/ha
Enxofre (S): 0,66 kg/ha
Próxima página


9:41
Recomendação total
Observações
Corretivos
Adubação
Recomendação dos Corretivos
Nitrogênio (N): 0,00 kg/ha
Calcário (Ca): 0,00 t/ha
Gesso: 95,22 kg/ha
Fosfóro (P2O5): 3,24 kg/ha
Potássio (K20): 7,78 kg/ha
Enxofre (S): 0,66 kg/ha
Micronutrientes
Boro (B): 2,40 g/ha
Cobre (Cu): 1,20 g/ha
Ferro (Fe): 21,00 g/ha
Manganês (Mn): 4,80 g/ha
Zinco (Zn): 16,20 g/ha
Molibdênio: 0,48 g/ha
Exportar para PDF

APÊNDICE C – PROTOTIPAGEM NO FIGMA



APÊNDICE D – *PRODUCT BACKLOG*

- Esquema lógico do banco de dados
- SQL a partir do esquema lógico
- Projeto de telas
- Prototipagem de telas
- Implementação das rotas
- Implementação dos temas
- Implementação do widget global
- Implementação da biblioteca provider
- Integração ao banco de dados
- Implementação tela fazenda
- Implementação tela fazenda_add
- Implementação tela talhao
- Implementação tela talhao_add
- Implementação tela safra
- Implementação tela safra_add
- Implementação tela home
- Implementação tela analise
- Implementação tela analise_add
- Implementação tela analise_add_quimico
- Implementação tela analise_add_fisico
- Implementação tela analise_add_micronutriente
- Implementação do widget interpretacao_nutriente
- Implementação da tela interpretacao
- Implementação tela recomendacao
- Implementação tela recomendacao_add

- Implementação do cálculo de calagem
- Implementação do cálculo de gessagem
- Implementação do cálculo de adubo
- Testes dos cálculos
- Implementação resultado recomendacao
- Testes de funcionamento
- Gerar *release*

APÊNDICE E – IMPLEMENTAÇÃO MVP

Figura 25 – Código *Model* em Flutter

```dart
class TalhaoModel {
  late int? idTalhao;
  late int idFazenda;
  late String nome;
  Map<String, dynamic> toMap() {
    var map = <String, dynamic>{
      'idTalhao': idTalhao,'idFazenda': idFazenda,'nome': nome,
    };
    return map;
}
  TalhaoModel.fromMap(Map<String, dynamic> map) {
    idTalhao = map['idTalhao'];
    idFazenda = map['idFazenda'];
    nome = map['nome'];
  }
}
```

Fonte: Elaborado pelo Autor, 2023.

Figura 26 – Código *Presenter* em Flutter

```dart
class TalhaoPresenter{
  late Database db;
  List<TalhaoModel> _talhoes = _getTalhoes();
  List<TalhaoModel> get talhoes => (_talhoes);
  Future<List<TalhaoModel>> _getTalhoes(int idFazenda) async {
    final db = await DBProvider.instance.database;
    final List<Map<String, dynamic>> response
    response = await db.rawQuery('''SELECT * FROM talhoes
    WHERE idFazenda = $idFazenda''');
    return TalhaoModel.fromMap(response.toList());
  }
}
```

Fonte: Elaborado pelo Autor, 2023.

Figura 27 – Código *View* em Flutter

```
class TalhaoView extends StatelessWidget {
  const TalhaoView({super.key});
  TalhaoPresenter _presenter = TalhaoPresenter();
  Widget build(BuildContext context) {
    return Consumer<TalhaoPresenter>(
      builder: (context, repositorio, child) {
        return ListView.builder(
          scrollDirection: Axis.vertical,
          itemCount: repositorio.talhoes.length,
          itemBuilder: (context, index) {
            return CustomCard(talhoes[index]);
          },),},);}
}
```

Fonte: Elaborado pelo Autor, 2023.

APÊNDICE F – IMPLEMENTAÇÃO CONFIGURAÇÃO INICIAL

Figura 28 – Arquivo de configuração de dependências

```
1  name: solofert
2  description: A new Flutter project.
3
4  environment:
5    sdk: ">=2.17.0 <3.0.0"
6
7  dependencies:
8    flutter:
9      sdk: flutter
10   provider: ^6.0.3
11   sqflite: ^2.0.3+1
12   path: ^1.8.1
13
14 flutter:
15   assets:
16     - assets/database.db
```

Fonte: Elaborado pelo Autor, 2023.

APÊNDICE G – IMPLEMENTAÇÃO ARQUIVO TEMA

Figura 29 – Definição de temas de cor e texto

```
class CommonMethod {
  ThemeData themedata = ThemeData(
    colorScheme: const ColorScheme(
      background: Color.fromARGB(255, 223, 223, 223),
      onBackground: Colors.white,
      primary: Color.fromARGB(255, 2, 161, 28),
    ),
    textTheme: const TextTheme(
      titleLarge: TextStyle(
        color: Colors.white,
        fontSize: 28,
        fontWeight: FontWeight.w400
      ),
      displayMedium: TextStyle(
        color: Color.fromARGB(255, 2, 161, 28),
        fontSize: 13,
        fontWeight: FontWeight.w600,
      ),
      bodyMedium: TextStyle(
        color: Colors.black,
        fontSize: 20,
        fontWeight: FontWeight.w400,
      ),
    ),
  );
}
```

Fonte: Elaborado pelo Autor, 2023.

APÊNDICE H – IMPLEMENTAÇÃO *WIDGET*

Figura 30 – Exemplo do cartão construído no app em Flutter

```
SizedBox(
  height: MediaQuery.of(context).size.height * 0.14,
  child: Card(
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.all(Radius.circular(20))
    ),
    elevation: 10,
    color: Theme.of(context).colorScheme.onBackground, //cor do card
    child: Padding(//padding dentro do card
      padding: const EdgeInsets.fromLTRB(15, 10, 10, 10),
      Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.start,
        children:[
          Text(
            title, //titulo do card
            style: Theme.of(context).textTheme.bodyMedium,
          ),
          Text(
            subtitle.toUpperCase(), //subtitulo do card em caixa alta
            style: Theme.of(context).textTheme.bodySmall,
          ),
        ],
      ),
    ),
  ),
);
```

Fonte: Elaborado pelo Autor, 2023.

APÊNDICE I – IMPLEMENTAÇÃO GRÁFICO INTERPRETAÇÃO

Figura 31 – Modelo personalizado

```
DataGraphModel(
  idNutriente: 17,
  {
  List<Intervalo>(
    nome: "Muito Baixo",
    valorInicial: calcio.valorInicial,
    valorFinal: calcio.valorFinal,
    fator: 0.2,
    color: const Color.fromARGB(255, 234, 44, 44),
    opcional: calcio.avaliacao.substring(0, 2),
    ),
  }
);
```

Fonte: Elaborado pelo Autor, 2023.

Figura 32 – Lista de *widgets*

```
listaCores.add(
  SizedBox(
    height: MediaQuery.of(context).size.height * 0.08,
    width: MediaQuery.of(context).size.width * DataGraphModel.fator,
    child: DecoratedBox(
      decoration:BoxDecoration(color: DataGraphModel.color),
      child: Text(
        DataGraphModel.nome,
        style: Theme.of(context).textTheme.labelSmall,
      ),
    ),
  ),
);
```

Fonte: Elaborado pelo Autor, 2023.

APÊNDICE J – IMPLEMENTAÇÃO DE UM FORMULÁRIO

Figura 33 – Implementação de um formulário

```
final TextEditingController _controller = TextEditingController();
Form(
  key: _formKey,
  TextFormField( //Formulario em texto
    controller: _controller,
    keyboardType: TextInputType.text, //tipo do teclado
    labelText: "Nome da safra", //titulo do formulário
    validator: (value) {
      if (value == null || value.isEmpty) {
        return "Insira um nome para a safra";
      }
     return null;
    },
  ),
);
```

Fonte: Elaborado pelo Autor, 2023.