



Universidade de Brasília

FCTE

Engenharia de Software

**Seletividade alimentar em pessoas
diagnosticadas com Transtorno do
Espectro Autista: Apoio tecnológico por
software**

Caio Mesquita Vieira

Gabriel Marques de Souza

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA DE SOFTWARE

Brasília

2025

Universidade de Brasília

FCTE

Engenharia de Software

**Seletividade alimentar em pessoas
diagnosticadas com Transtorno do
Espectro Autista: Apoio tecnológico por
software**

Caio Mesquita Vieira

Gabriel Marques de Souza

Trabalho de Conclusão de Curso submetido
como requisito parcial para obtenção do grau
de Engenheiro de Software.

Orientador: Prof. Dr. Ricardo Ajax Dias Kosloski

Coorientadora: Prof. Dra. Marília Miranda Forte Gomes

Brasília

2025

FICHA CATALOGRÁFICA

Vieira, Caio Mesquita; Souza, Gabriel Marques de.

Seletividade alimentar em pessoas diagnosticadas com Transtorno do Espectro Autista: Apoio tecnológico por software / Caio Mesquita Vieira; Gabriel Marques de Souza; orientador Ricardo Ajax Dias Kosloski; coorientadora Marília Miranda Forte Gomes. -- Brasília, 2025.

83 p.

Trabalho de Conclusão de Curso (Engenharia de Software) -- Universidade de Brasília, 2025.

1. Transtorno do Espectro Autista. 2. Seletividade Alimentar. 3. Aplicativo Mobile. 4. Palavra-chave 4. I. Souza, Gabriel Marques de. II. Kosloski, Ricardo Ajax Dias , orient. III. Gomes, Marília Miranda Forte, coorient. IV. Título.

**Universidade de Brasília
FCTE
Engenharia de Software**

**Seletividade alimentar em pessoas diagnosticadas com
Transtorno do Espectro Autista: Apoio tecnológico por
software**

Caio Mesquita Vieira
Gabriel Marques de Souza

Trabalho de Conclusão de Curso submetido
como requisito parcial para obtenção do grau
de Engenheiro de Software.

Trabalho apresentado. Brasília, 02 de dezembro de 2025

**Prof. Dr. Ricardo Ajax Dias Kosloski ,
UnB/FCTE
Orientador**

**Prof. Dra. Marília Miranda Forte Gomes ,
UnB/FCTE
Coorientadora**

**Prof. MSc. Paula Uessugue ,
UnB/Darcy Ribeiro
Examinador interno**

**Prof. MSc. Cristiane Soares Ramos,
UnB/FCTE
Examinador interno**

Dedico este trabalho a todos que, de alguma forma, fizeram parte da minha trajetória. Aos meus familiares, pelo amor incondicional e apoio constante. Aos amigos, pela companhia nos momentos difíceis e pelas risadas nos dias leves. E a todos que acreditaram em mim mesmo quando eu duvidei. Este trabalho é resultado de muitos esforços compartilhados.

Caio Mesquita Vieira

Dedico este trabalho à minha família, pelo amor incondicional, incentivo constante e por sempre acreditarem em meu potencial ao longo de toda a minha trajetória acadêmica. Agradeço, especialmente, a Deus, por ter me concedido força, sabedoria e perseverança para chegar até aqui.

Gabriel Marques de Souza

Agradecimentos

Agradeço primeiramente à minha família, pelo amor, apoio incondicional e por acreditar em mim em cada etapa dessa jornada acadêmica. Vocês foram meu alicerce nos momentos difíceis e minha inspiração para seguir em frente.

À minha companheira, por todo amor, paciência e compreensão nos dias em que precisei me ausentar para estudar. Sua presença e incentivo foram fundamentais para que eu pudesse chegar até aqui.

Aos meus professores e orientadores, por toda dedicação, paciência e pelos ensinamentos compartilhados ao longo do curso e durante a construção deste trabalho. Cada orientação contribuiu diretamente para meu crescimento pessoal e profissional.

Ao meu parceiro de TCC, pela parceria, comprometimento e companheirismo em todas as fases do projeto. Sua dedicação tornou esta caminhada mais leve, produtiva e motivadora.

Por fim, Agradeço a Deus, que sem ele não seríamos nada.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho, meu muito obrigado.

Caio Mesquita Vieira

Agradeço, em primeiro lugar, à minha família, pelo apoio incondicional, pelo incentivo constante aos meus estudos e por toda a assistência prestada ao longo desta caminhada. Sem o amor, o cuidado e os sacrifícios de vocês, não seria possível chegar tão longe. Foi graças à força e aos esforços de cada um que este trabalho se tornou realidade.

Aos professores que contribuíram para a minha formação acadêmica na Universidade, deixo meus mais sinceros agradecimentos. Em especial, aos professores orientadores deste trabalho, pela dedicação, paciência e orientação valiosa em cada etapa deste percurso.

Por fim, deixo um agradecimento especial ao meu amigo e parceiro de TCC, cuja colaboração foi essencial para a concretização deste trabalho. Sua amizade, companheirismo e apoio nos desafios da vida fizeram toda a diferença para que pudéssemos chegar até aqui.

Gabriel Marques de Souza

*“Se você quer ser bem-sucedido,
precisa ter dedicação total,
buscar seu último limite e dar o melhor de si.”*
(Ayrton Senna)

Resumo

O Transtorno do Espectro Autista (TEA) é uma condição do neurodesenvolvimento que frequentemente impõe desafios à rotina diária, sendo a seletividade alimentar uma das comorbidades mais comuns e impactantes. Esse comportamento pode comprometer a saúde nutricional do indivíduo e gerar sobrecarga aos cuidadores, contudo, nota-se no mercado atual uma escassez significativa de ferramentas tecnológicas projetadas para auxiliar no manejo prático dessa questão. Visando preencher essa lacuna, este trabalho desenvolveu um aplicativo móvel para apoiar a expansão do repertório alimentar de pessoas com TEA, oferecendo sugestões de trocas baseadas em compatibilidade sensorial. O projeto seguiu uma abordagem de pesquisa aplicada com metodologias ágeis, adaptando práticas do Scrum e Extreme Programming (XP), e foi construído utilizando React Native para o front-end, Node.js para o back-end e PostgreSQL para a persistência de dados. A solução implementa um algoritmo fundamentado na estratégia de Encadeamento Alimentar (Food Chaining), processando dados de instrumentos como o Questionário de Frequência Alimentar (QFA) para recomendar substituições seguras e personalizadas.

Palavras-chave: Transtorno do Espectro Autista; Seletividade Alimentar; Aplicativo Mobile.

Abstract

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition that frequently imposes challenges on daily routines, with selective eating being one of the most common and impactful comorbidities. This behavior can compromise the individual's nutritional health and burden caregivers; however, there is a significant scarcity of technological tools designed to assist in the practical management of this issue. Aiming to fill this gap, this work developed a mobile application to support the expansion of the dietary repertoire of people with ASD, offering suggestions for substitutions based on sensory compatibility. The project followed an applied research approach with agile methodologies, adapting practices from Scrum and Extreme Programming (XP), and was built using React Native for the front-end, Node.js for the back-end, and PostgreSQL for data persistence. The solution implements an algorithm based on the Food Chaining strategy, processing data from instruments such as the Food Frequency Questionnaire (FFQ) to recommend safe and personalized substitutions.

Keywords: Autism Spectrum Disorder; Food Selectivity; Mobile Application.

Lista de figuras

Figura 3.1	Metodologia da Pesquisa	28
Figura 3.2	Ciclo de vida	30
Figura 3.3	Estrutura da arquitetura baseada em componentes	34
Figura 3.4	Organização das branches - GitHub	45
Figura 4.1	Logo do Aplicativo	50
Figura 4.2	Paleta de Cores	50
Figura 4.3	Diagrama Entidade-Relacionamento Banco de Dados	58
Figura 4.4	Cobertura de testes do Back-End	61
Figura 4.5	Cobertura de testes do Front-End	61
Figura A.1	Tela de Login	73
Figura A.2	Tela de Cadastro de Usuário	74
Figura A.3	Home do Cuidador	75
Figura A.4	Perfil do Cuidador	76
Figura A.5	Cadastro de Assistido	77
Figura A.6	Preenchimento do Questionário	78
Figura A.7	Refeições Disponíveis	79
Figura A.8	Sugestão de Troca Alimentar	80
Figura A.9	Tela de Feedback	81
Figura B.1	Protótipo de Média Fidelidade	83

Lista de tabelas

Tabela 3.1 Objetivos, questões e métricas definidas pelo método GQM	41
Tabela 4.1 Resultados das Métricas GQM	62
Tabela 4.2 Cronograma Executado de Desenvolvimento e Entregas (TCC 2)	63

Sumário

1	Introdução	14
1.1	Contextualização	14
1.2	Justificativa	14
1.3	Questões de Pesquisa e Desenvolvimento	15
1.4	Objetivos	16
1.4.1	Objetivo Geral	16
1.4.2	Objetivos Específicos	16
1.5	Organização da Monografia	16
2	Referencial Teórico	18
2.1	Seletividade Alimentar e TEA	18
2.1.1	Estabelecimento de conceitos teóricos	18
2.1.2	Transtorno do Espectro Autista	19
2.1.3	Seletividade Alimentar no Contexto do TEA	19
2.1.4	Tratamentos e Intervenções Comportamentais para Seletividade Alimentar no TEA	20
2.1.5	Uso de Tecnologia no Contexto	21
2.2	Engenharia de <i>Software</i>	21
2.3	Metodologias Ágeis	22
2.4	Desenvolvimento <i>Mobile</i>	23
2.5	Tecnologias de Desenvolvimento	23
2.5.1	React Native	23
2.5.2	Node.js	24
2.5.3	Expo	24
2.5.4	NativeWind	25
2.5.5	PostgreSQL	25
2.5.6	Visual Studio Code	26
2.5.7	GitHub	26
2.5.8	Microsoft Teams	27
2.5.9	Docker	27
3	Metodologias	28
3.1	Metodologia de pesquisa	28
3.2	Metodologia de Desenvolvimento de Software	29
3.2.1	Ciclo de Vida	29
3.2.2	Artefatos	30

3.2.3	Fases de Desenvolvimento	30
3.2.4	Requisitos	31
3.2.5	Arquitetura	33
3.2.6	Estilo Arquitetural Baseado em Componentes	33
3.2.7	Arquitetura Em 3 Camadas	35
3.2.8	Desenvolvimento de Software	35
3.2.9	Gerência de configuração de software	36
3.2.10	Repositório	37
3.2.11	Política de Branchs	37
3.2.12	Política de commits	38
3.3	Métricas	39
3.3.1	GQM	39
3.4	Testes	42
3.4.1	Testes unitários	42
3.4.2	Testes de integração	42
3.4.3	Testes Funcionais e de Interface (UI)	43
3.5	Tecnologias	44
3.5.1	Gestão do projeto	44
3.5.2	Ferramentas de comunicação	45
3.5.3	Persistência de dados	45
3.5.4	PostgreSQL	46
3.5.5	Ferramentas de Apoio à Escrita	46
4	Solução Implementada	47
4.1	Sobre o Aplicativo	47
4.2	<i>Backlog</i> do produto	47
4.3	Identidade Visual	49
4.3.1	Logotipo	49
4.3.2	Interface do Aplicativo	50
4.3.3	Interface do Aplicativo	51
4.4	Estrutura do Código da Aplicação	51
4.4.1	Camada de Apresentação (Front-end)	52
4.4.2	Camada de Negócio e Dados (Back-end)	52
4.4.3	Ambiente e Orquestração	53
4.5	Organização dos Dados	53
4.5.1	Estrutura de Usuários e assistidos	53
4.5.2	Armazenamento de Dados de Avaliação (Questionários)	54
4.5.3	Base de Conhecimento Alimentar e Perfil Sensorial	54
4.5.4	Modelo de Sugestão, Opções e Feedback	54

4.5.5	Modelo Relacional e DER	55
4.6	Regra de negócio	59
4.6.1	Fundamentos e Modelo de Dados de Suporte	59
4.6.2	O Fluxo Operacional do Algoritmo de Sugestão	59
4.6.3	Dinamismo e Adaptação: O Ciclo de Feedback	60
4.7	Testes	60
4.7.1	Testes Unitários	60
4.7.2	Testes de Integração	60
4.7.3	Teste Funcional	60
4.8	Resultados das Métricas (GQM)	62
4.9	Cronograma de Desenvolvimento	62
5	Conclusões	64
5.1	Introdução	64
5.1.1	Objetivos Específicos	64
5.1.2	Processo de desenvolvimento	64
5.2	Fragilidades e Propostas de Evolução	65
5.2.1	Fragilidades Internas	65
5.2.2	Fragilidades Externas	66
5.2.3	Proposta de Evolução	66
5.2.4	Conclusões finais	67
Referências	69	
Apêndices	72	
Apêndice A Telas do Aplicativo	73	
Apêndice B Protótipo de Média Fidelidade	82	

1 Introdução

Este capítulo apresenta a contextualização do trabalho, focado no desenvolvimento de uma aplicação móvel para apoiar cuidadores de pessoas com Transtorno do Espectro Autista (TEA) que possuem seletividade alimentar e os próprios indivíduos com TEA. O objetivo é desenvolver um software que auxilie na sugestão de trocas alimentares. A aplicação analisará informações fornecidas pelo usuário para oferecer alternativas adequadas às preferências e necessidades da pessoa com TEA. As seções seguintes detalham o problema, a justificativa do projeto, as questões de pesquisa e os objetivos que norteiam o desenvolvimento.

1.1 Contextualização

A seletividade alimentar é um desafio prevalente entre crianças e adolescentes com Transtorno do Espectro Autista (TEA), com impacto na qualidade de vida dos indivíduos e de seus cuidadores. Estudos indicam que até 84% das crianças com TEA apresentam padrões alimentares restritivos, como recusa de grupos alimentares, rigidez a texturas e cores, e consumo nutricional limitado (Sharp *et al.*, 2018). Tais comportamentos elevam o risco de deficiências nutricionais, como baixos níveis de cálcio e proteínas, e associam-se a comorbidades gastrointestinais e transtornos de comportamento (Leader *et al.*, 2020).

Embora existam abordagens terapêuticas eficazes, como as baseadas em princípios analítico-comportamentais (Taylor; DeQuinzio; Hao, 2017; Ulloa; Tereshkov, 2020), sua aplicação no cotidiano representa um desafio para os cuidadores.

Os desafios alimentares no TEA também geram repercussões emocionais nos cuidadores, como ansiedade, estresse e sentimentos de impotência, derivados do manejo de recusas alimentares e da preocupação com a nutrição dos assistidos (Guller; Yaylaci, 2024).

Neste contexto, o trabalho propõe o desenvolvimento de um aplicativo móvel como ferramenta de apoio aos cuidadores e aos próprios indivíduos com TEA. A aplicação coletará informações por meio de formulários, cujo método será descrito no capítulo de Metodologia, para sugerir alternativas alimentares compatíveis. O objetivo é oferecer um recurso funcional e baseado em evidências científicas, apresentadas no Referencial Teórico, que amplie a autonomia dos usuários e contribua para a melhoria da rotina alimentar.

1.2 Justificativa

A pesquisa bibliográfica que fundamenta este trabalho revela uma lacuna no uso de tecnologias de software para apoiar os usuários na rotina alimentar de pessoas com TEA.

Intervenções, como estratégias de reforço ou orientações nutricionais personalizadas, são, geralmente, restritas a ambientes clínicos e nem sempre disponíveis à maioria das famílias (Vazquez; Fitt; Rich, 2019; Ulloa; Tereshkov, 2020).

Ainda que haja acompanhamento profissional, a aplicação das recomendações no dia a dia enfrenta barreiras. Fatores como falta de tempo, excesso de informações não sistematizadas e o estresse associado às recusas alimentares dificultam a implementação eficaz das orientações (Guller; Yaylaci, 2024).

A tecnologia móvel, já integrada à rotina de muitas famílias, oferece um meio para disponibilizar conhecimento e sistematizar o manejo alimentar. A ferramenta proposta busca promover a autonomia dos cuidadores e incentivar a diversificação alimentar de forma gradual.

Portanto, o projeto se justifica pela proposta de uma ferramenta digital que busca ampliar o conjunto de alimentos que são consumidos pelos indivíduos com TEA com seletividade alimentar, tudo isso através de uma contribuição tecnológica para auxiliar na área social.

1.3 Questões de Pesquisa e Desenvolvimento

Este Trabalho de Conclusão de Curso tem como objetivo principal desenvolver um aplicativo móvel que ofereça suporte aos seus usuários, auxiliando na identificação de alternativas alimentares mais adequadas e promovendo a diversificação da dieta dos indivíduos que possuem TEA e seletividade alimentar. A proposta visa proporcionar uma solução tecnológica funcional, gratuita e disponível para os sistemas Android e iOS, baseada em dados coletados para subsidiar decisões alimentares no ambiente domiciliar.

Durante o processo de desenvolvimento do aplicativo, pretende-se responder à seguinte questão de pesquisa principal:

- Quais são as características necessárias em um aplicativo móvel que possa apoiar cuidadores de pessoas com TEA no manejo da seletividade alimentar, por meio da sugestão de trocas alimentares viáveis e personalizadas?

Além da questão central, este trabalho busca explorar as seguintes questões secundárias, que darão suporte teórico e prático à concepção da solução proposta:

- O que é seletividade alimentar no contexto de TEA?
- Quais as características de tratamentos usados neste contexto?
- Quais tecnologias envolvendo aplicativos de software têm sido usadas neste contexto?

Essas questões nortearão tanto a fundamentação teórica quanto as decisões de projeto envolvidas no desenvolvimento do aplicativo, assegurando que a solução proposta

esteja alinhada às necessidades reais dos cuidadores e fundamentada em evidências técnico-científicas.

1.4 Objetivos

Esta seção apresenta o objetivo geral e os objetivos específicos deste trabalho de TCC.

1.4.1 Objetivo Geral

Desenvolver um aplicativo móvel como ferramenta de apoio tecnológico voltada a cuidadores de pessoas com Transtorno do Espectro Autista (TEA) que apresentam seletividade alimentar e também aos próprios indivíduos com TEA e seletividade alimentar.

1.4.2 Objetivos Específicos

1. Identificar os conceitos necessários ao desenvolvimento do aplicativo de software almejado.
2. Estabelecer uma ferramenta de software para apoiar os usuários com a questão da seletividade alimentar no contexto de TEA.
3. Estabelecer estudos de caso para avaliar a adequação do produto de software aos fins a que se dedica.

1.5 Organização da Monografia

Este Trabalho de Conclusão de Curso está organizado nos seguintes capítulos:

- **Capítulo 1 - Introdução:** contextualiza o trabalho, suas questões de pesquisa, apresenta as justificativas para o desenvolvimento e descreve seus objetivos (geral e específicos);
- **Capítulo 2 - Referencial Teórico:** apresenta os fundamentos teóricos do trabalho, especialmente em relação à seletividade alimentar e ao Transtorno do Espectro Autista, além de tópicos mais técnicos de Engenharia de Software (ex. engenharia de requisitos, arquitetura de software, metodologias de desenvolvimento, testes e qualidade);
- **Capítulo 3 - Metodologias:** apresenta os aspectos metodológicos sobre o levantamento bibliográfico, o desenvolvimento do software e a análise de resultados;
- **Capítulo 4 - Solução Implementada:** descreve em detalhes a solução que foi implementada por meio do desenvolvimento de um software dedicado;
- **Capítulo 5 - Conclusão:** apresenta os resultados alcançados na primeira etapa do TCC, bem como retoma questionamentos e objetivos para conferir uma visão geral

de como os mesmos foram tratados até o momento. Por fim, aborda detalhes sobre os próximos passos desse trabalho de software;

2 Referencial Teórico

Este capítulo apresenta o referencial teórico dividido em dois grupos principais. O primeiro aborda os conceitos de TEA, seletividade alimentar, tratamentos e o uso de tecnologias de *software* neste contexto, enquanto o segundo grupo foca nos conceitos de engenharia de *software* que nortearam o desenvolvimento da aplicação. Para a definição do primeiro grupo, foram utilizados mapeamentos sistemáticos da literatura, detalhados nas seções seguintes.

Este trabalho foi realizado com o apoio da Paula Uessugue, nutricionista de referência em nutrição materno-infantil no DF, com 20 anos de atuação acadêmica e clínica e mais de mil crianças atendidas. Possui reconhecimento público por sua contribuição em introdução alimentar, TEA e educação nutricional, além de ser professora e mentora de projetos acadêmicos.

2.1 Seletividade Alimentar e TEA

2.1.1 Estabelecimento de conceitos teóricos

Para embasar este trabalho, foi realizado uma pesquisa bibliográfica exploratória sobre seletividade alimentar em indivíduos com Transtorno do Espectro Autista (TEA).

A consulta foi feita na base Scopus (via portal *Periódicos CAPES*), onde foram usadas três estratégias de busca com operadores booleanos e descritores combinados, como “*eating disorder*”, “*food selectivity*”, “*autism spectrum disorder*”, “*technology*”, “*software*” e “*application*”.

- Expressão de busca final:

```
TITLE-ABS-KEY((Technology OR "Software application development" OR app)
AND ("Food selectivity"OR "food refusal") AND ("autism spectrum disorder"OR
tea OR "food exchange"))
```

Como critérios de seleção, foram considerados artigos publicados entre 2018 e 2024, escritos em inglês ou português, revisados por pares e com foco nas áreas de Medicina, Psicologia, Enfermagem, Neurociência e Sociologia. Foram excluídos trabalhos duplicados, artigos de revisão, resumos de conferência e publicações sem texto completo disponível.

A expressão de busca identificou 44 artigos, dos quais foram extraídas contribuições teóricas sobre:

- Caracterização de seletividade alimentar em pessoas com TEA;
- Causas e implicações da seletividade alimentar;

-
- Lacunas na utilização de recursos tecnológicos no suporte ao tratamento alimentar;

A busca por ferramentas tecnológicas retornou poucas publicações, indicando uma lacuna na literatura sobre o uso de *software* para seletividade alimentar no TEA. Os resultados são apresentados nas seções seguintes e fundamentam o desenvolvimento da aplicação.

2.1.2 Transtorno do Espectro Autista

O Transtorno do Espectro Autista (TEA) é uma condição do neurodesenvolvimento caracterizada por déficits na comunicação social, padrões repetitivos de comportamento e particularidades sensoriais. Tais características se manifestam em graus variados, o que justifica o termo “espectro” ([American Psychiatric Association, 2013](#)). Indivíduos com TEA frequentemente demonstram hipersensibilidade ou hipossensibilidade a estímulos (sons, luzes, texturas, cheiros, sabores), impactando diretamente a vida cotidiana, incluindo a alimentação.

Diversos estudos destacam a relação entre essas sensibilidades e a seletividade alimentar, indicando que estímulos sensoriais específicos, como texturas, podem provocar reações de rejeição alimentar em crianças com TEA ([Ulloa; Tereshkov; Aranki, 2022](#)). Essa aversão não é uma preferência comum, mas sim uma resposta sensorial intensa que pode desencadear comportamentos como choro, gritos ou evasão. Esses padrões tornam o processo de alimentação um desafio para familiares e profissionais.

Além disso, as dificuldades alimentares associadas ao TEA não estão relacionadas à fome, mas sim à dificuldade de processar estímulos sensoriais dos alimentos ([Bicer; Alsaffar, 2020](#)). Dessa forma, a alimentação torna-se uma experiência estressante, que precisa ser gerida com estratégias de apoio baseadas no perfil sensorial do indivíduo.

2.1.3 Seletividade Alimentar no Contexto do TEA

A seletividade alimentar é uma característica comum em pessoas com Transtorno do Espectro Autista (TEA) e representa um desafio para familiares, cuidadores e profissionais. Indivíduos no espectro tendem a apresentar padrões alimentares restritivos, marcados pela recusa de alimentos, preferência por marcas ou preparos específicos e escolhas limitadas por cor, textura ou temperatura.

Essa condição está relacionada a alterações no processamento sensorial, fatores que podem gerar desconforto ou aversão a alimentos. Além disso, aspectos comportamentais, como rigidez em rotinas e dificuldade com mudanças, contribuem para tornar a alimentação uma fonte de estresse.

A consequência da seletividade alimentar é o risco de uma dieta desequilibrada, podendo resultar em deficiências de nutrientes. A limitação alimentar persistente também

impacta o convívio social, pois interfere em refeições familiares ou atividades externas.

2.1.4 Tratamentos e Intervenções Comportamentais para Seletividade Alimentar no TEA

O manejo da seletividade alimentar em indivíduos com TEA exige uma abordagem estruturada, que considere os níveis de suporte, o grau de seletividade e o uso de instrumentos de avaliação para guiar o processo terapêutico.

Segundo o Manual Diagnóstico e Estatístico de Transtornos Mentais (DSM-5) ([American Psychiatric Association, 2013](#)), o TEA pode ser classificado em três níveis de suporte — variando de apoio mínimo até apoio muito substancial — de acordo com a intensidade dos déficits de comunicação social e a presença de comportamentos restritivos.

Essa classificação impacta o planejamento das intervenções. Crianças que demandam maior suporte tendem a apresentar maior rigidez comportamental e menor tolerância a novos alimentos. Assim, quanto mais elevado o nível de suporte, mais complexas devem ser as estratégias de introdução de alimentos e reeducação nutricional, envolvendo acompanhamento contínuo de equipes multidisciplinares.

Ademais, o grau de seletividade varia amplamente. Há casos em que a restrição é moderada, limitando-se a alguns grupos de alimentos, enquanto em quadros mais severos é comum a recusa de categorias alimentares, com aceitação restrita a pouquíssimos itens. Tal comportamento compromete a variedade nutricional, podendo gerar carências de nutrientes ou obesidade associada à desnutrição, devido ao consumo de alimentos ultraprocessados.

Para avaliar essas condições, são utilizadas escalas e instrumentos, como *checklists*, entrevistas com cuidadores e ferramentas padronizadas, a exemplo da *Brief Autism Mealtime Behavior Inventory* (BAMBI). O BAMBI ajuda a identificar padrões de recusa, preferências e intensidade de comportamentos durante as refeições. O uso dessas ferramentas possibilita monitorar a evolução da intervenção e planejar ajustes.

No que se refere às estratégias, as abordagens baseadas na Análise do Comportamento Aplicada (ABA) têm mostrado resultados na ampliação do repertório alimentar. Técnicas como reforço positivo, modelagem, dessensibilização sistemática, extinção de fuga e exposição gradual são aplicadas de forma individualizada. O reforço positivo, por exemplo, utiliza estímulos para encorajar comportamentos (como aceitar um novo alimento), enquanto a exposição repetida e gradual contribui para reduzir a rejeição inicial.

O envolvimento da família é outro fator nas intervenções. O trabalho conjunto entre profissionais e cuidadores possibilita criar um ambiente alimentar de reforço, dá continuidade às práticas fora do ambiente clínico e reduz o estresse cotidiano. Assim, a cooperação entre todos os envolvidos garante uma intervenção adaptada e sustentável.

2.1.5 Uso de Tecnologia no Contexto

Durante a pesquisa por tecnologias existentes, a solução que mais se aproximou da proposta foi o aplicativo "Garfinho". Ele se posiciona como uma ferramenta de apoio à alimentação infantil, oferecendo planejamento de cardápios e receitas. Contudo, uma análise revela que seu propósito e público são distintos. O "Garfinho" atende a um público geral, sem especialização nas complexidades do Transtorno do Espectro Autista (TEA), e sua abordagem não considera as questões sensoriais centrais na seletividade alimentar. Adicionalmente, seu modelo de negócio é baseado em assinatura paga, o que pode limitar o acesso.

Deste modo, para preencher essa lacuna que este aplicativo foi concebido, focado nas necessidades de pessoas com TEA e seus cuidadores. Contrapondo-se à imposição de dietas, o sistema opera com base em trocas alimentares personalizadas, partindo dos alimentos já aceitos pelo indivíduo para sugerir substituições graduais. Este método respeita o perfil de cada usuário, tornando o processo mais adaptado. O objetivo é entregar uma ferramenta especializada e direcionada para este contexto.

Como parte da fundamentação deste trabalho, foi conduzida uma revisão da literatura focada nos avanços científicos e tecnológicos para o manejo da seletividade alimentar no Transtorno do Espectro Autista (TEA). Este estudo, desenvolvido em coautoria pelos autores desta monografia e submetido à Revista Políticas Públicas & Cidades ([Uessugue et al., 2025](#)), analisou a literatura recente para identificar lacunas e tendências de pesquisa, destacando a necessidade de desenvolver novas ferramentas tecnológicas que auxiliem cuidadores e pessoas diagnosticadas, uma vez que a literatura ainda é incipiente na aplicação de *software* para este problema.

2.2 Engenharia de Software

A Engenharia de *Software* aplica uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de *software*. Ela abrange um conjunto de métodos, ferramentas e procedimentos que visam produzir *software* que atenda aos requisitos, dentro do prazo e orçamento.

Segundo Pressman e Maxim ([Pressman; Maxim, 2021](#)), a Engenharia de *Software* é um campo em evolução que busca soluções para a construção de sistemas complexos. Suas principais áreas de atuação incluem:

- **Processo de Software:** Define as atividades, tarefas e produtos de trabalho necessários. Modelos de processo (cascata, ágeis) orientam o desenvolvimento.
- **Engenharia de Requisitos:** Foca na elicitação, análise, especificação e validação das necessidades dos usuários e das restrições do sistema.
- **Projeto de Software:** Envolve a criação da arquitetura, estrutura de dados e interfaces

para os componentes do sistema.

- **Construção de Software:** Refere-se à codificação, testes unitários e depuração.
- **Teste de Software:** Garante que o *software* funcione e atenda aos requisitos, identificando defeitos.
- **Manutenção de Software:** Lida com as modificações necessárias após a entrega, incluindo correção de erros, melhorias e adaptações.
- **Gerência de Configuração de Software (GCS):** Controla as mudanças nos artefatos do projeto ao longo do tempo.
- **Gerência de Projetos de Software:** Planeja, organiza e controla os recursos para garantir que o projeto seja concluído.

A aplicação dos princípios da Engenharia de *Software* permite gerenciar a complexidade, garantir a qualidade, otimizar recursos e entregar valor aos usuários.

2.3 Metodologias Ágeis

As metodologias ágeis representam uma abordagem para o desenvolvimento de *software* que prioriza a flexibilidade, a colaboração, a entrega de valor e a resposta rápida a mudanças. Elas surgiram como uma alternativa aos métodos tradicionais, que se mostravam rígidos e pouco adaptáveis a projetos com requisitos em evolução.

O Manifesto Ágil, publicado em 2001 ([Beck et al., 2001](#)), estabeleceu os quatro valores que norteiam essas metodologias:

- Indivíduos e interações mais que processos e ferramentas.
- *Software* em funcionamento mais que documentação abrangente.
- Colaboração com o cliente mais que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

Esses valores são complementados por doze princípios que detalham a forma de trabalho ágil, como a entrega frequente de *software* funcional, a colaboração diária, a simplicidade e a auto-organização das equipes.

Dentre as diversas metodologias ágeis, algumas das mais conhecidas incluem:

- **Scrum:** Um *framework* iterativo e incremental que organiza o desenvolvimento em ciclos curtos (*sprints*), com papéis e eventos bem definidos ([Schwaber; Sutherland, 2020](#)).
- **Kanban:** Um método visual para gerenciar o fluxo de trabalho, utilizando quadros com colunas que representam os estágios das tarefas, com foco na limitação do trabalho em progresso.
- **Extreme Programming (XP):** Uma metodologia que enfatiza a entrega contínua, testes frequentes, programação em pares, refatoração e *feedback* constante.

- **Lean Software Development:** Baseado nos princípios do *Lean Manufacturing*, foca na eliminação de desperdícios, na construção de qualidade e na entrega rápida.

As metodologias ágeis são amplamente adotadas na indústria devido à sua capacidade de promover maior adaptabilidade, reduzir riscos e melhorar a qualidade do produto final.

2.4 Desenvolvimento *Mobile*

O desenvolvimento *mobile* refere-se ao processo de criação de aplicativos para dispositivos móveis, como *smartphones* e *tablets*. Com a crescente ubiquidade desses aparelhos, a demanda por aplicações tem impulsionado a evolução de tecnologias e abordagens para esse segmento.

Existem três principais abordagens para o desenvolvimento *mobile*:

- **Desenvolvimento Nativo:** Envolve a criação de aplicativos usando as linguagens e ferramentas específicas de cada plataforma (por exemplo, Swift para iOS; Java/Kotlin para Android). Aplicativos nativos oferecem o melhor desempenho e acesso total aos recursos do dispositivo, mas exigem o desenvolvimento de bases de código separadas, o que pode aumentar o tempo e o custo.
- **Desenvolvimento Híbrido:** Permite a criação de aplicativos que funcionam em múltiplas plataformas usando uma única base de código (geralmente tecnologias *web* como HTML, CSS, JavaScript) encapsuladas em um *container* nativo. *Frameworks* como Cordova ou Ionic são exemplos. Embora ofereçam agilidade, podem ter limitações de desempenho e acesso a recursos nativos.
- **Desenvolvimento Multiplataforma (*Cross-Platform*):** Aborda o desenvolvimento com uma única base de código que compila para código nativo ou se comunica com componentes nativos. *Frameworks* como React Native e Flutter se enquadram nesta categoria. Eles buscam combinar a eficiência de uma única base de código com o desempenho nativo.

A escolha da abordagem depende de fatores como o orçamento, o prazo, a complexidade do aplicativo e a necessidade de acesso a recursos nativos.

2.5 Tecnologias de Desenvolvimento

2.5.1 React Native

React Native é um *framework* de código aberto criado pelo Facebook para o desenvolvimento de aplicativos móveis nativos utilizando JavaScript e React ([Meta Platforms Inc., 2024](#)). Ele permite que desenvolvedores construam interfaces para iOS e Android a partir de uma única base de código.

A principal característica do React Native é a capacidade de renderizar componentes da interface do usuário que são, de fato, componentes nativos da plataforma, e não *webviews*. Isso resulta em aplicativos com desempenho e aparência nativos.

Benefícios do React Native incluem:

- **Reuso de Código:** Grande parte do código JavaScript pode ser compartilhada entre as plataformas iOS e Android.
- **Hot Reloading e Fast Refresh:** Ferramentas que permitem visualizar as mudanças no código quase instantaneamente.
- **Comunidade Ativa:** Vasta comunidade de desenvolvedores que contribui com bibliotecas e suporte.
- **Acesso a Recursos Nativos:** Possibilita acessar APIs nativas do dispositivo quando necessário.

2.5.2 Node.js

Node.js é um ambiente de execução JavaScript de código aberto e multiplataforma, construído sobre o motor V8 do Google Chrome ([OpenJS Foundation, 2024](#)). Ele permite que desenvolvedores usem JavaScript para criar aplicações do lado do servidor (*back-end*).

A característica do Node.js é seu modelo de I/O não bloqueante e orientado a eventos, o que o torna eficiente para aplicações que lidam com muitas requisições simultâneas, como APIs *RESTful* e microsserviços.

Vantagens do Node.js:

- **Performance:** Graças ao motor V8 e ao modelo assíncrono, o Node.js é rápido na execução de código JavaScript.
- **Ecossistema NPM:** Possui o maior ecossistema de bibliotecas de código aberto (npm), facilitando o desenvolvimento.
- **JavaScript Full-Stack:** Permite que desenvolvedores utilizem a mesma linguagem (JavaScript) tanto no *front-end* quanto no *back-end*.
- **Escalabilidade:** Ideal para construir aplicações escaláveis e de alta concorrência.

2.5.3 Expo

Expo é um *framework* e plataforma de código aberto que simplifica o desenvolvimento de aplicativos React Native ([650 Industries, Inc., 2024](#)). Ele fornece um conjunto de ferramentas e serviços que abstraem complexidades do desenvolvimento nativo, permitindo que os desenvolvedores se concentrem na lógica de negócios usando apenas JavaScript.

Com o Expo, é possível:

- **Desenvolvimento Rápido:** Iniciar um projeto React Native sem a necessidade de configurar ambientes nativos (Xcode, Android Studio).
- **Testes Simplificados:** Testar aplicativos diretamente no dispositivo móvel escaneando um QR code.
- **Acesso a APIs Nativas:** Oferece uma vasta coleção de APIs nativas (câmera, localização, etc.) prontas para uso via JavaScript.
- **Over-the-Air (OTA) Updates:** Possibilita o envio de atualizações para o aplicativo sem a necessidade de submeter novas versões para as lojas.

O Expo é indicado para projetos que precisam de um desenvolvimento ágil e que não exigem acesso a módulos nativos muito específicos que não são suportados pelo *framework*.

2.5.4 NativeWind

NativeWind é uma biblioteca que traz a sintaxe do Tailwind CSS para o desenvolvimento React Native ([Lawlor, 2024](#)). O Tailwind CSS é um *framework* "utility-first" que oferece classes utilitárias para construir interfaces diretamente no JSX, sem a necessidade de escrever CSS personalizado.

Com o NativeWind, os desenvolvedores podem aplicar estilos usando classes utilitárias do Tailwind (como 'flex', 'pt-4', 'text-lg'). Isso resulta em:

- **Estilização Rápida:** Agiliza o processo de estilização, eliminando a necessidade de alternar entre arquivos JSX e folhas de estilo.
- **Consistência Visual:** Promove a consistência no *design*, utilizando um sistema baseado em *tokens*.
- **Manutenibilidade:** Facilita a manutenção, pois os estilos são aplicados diretamente onde são usados.
- **Otimização de Tamanho:** O NativeWind pode ser configurado para "purificar" o CSS, removendo classes não utilizadas.

2.5.5 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados relacional (SGBDR) de código aberto conhecido por sua confiabilidade, integridade de dados e conformidade com o padrão SQL ([The PostgreSQL Global Development Group, 2024](#)).

Características e vantagens do PostgreSQL:

- **Confiabilidade e Integridade:** Suporta transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade).
- **Extensibilidade:** Permite aos usuários definir seus próprios tipos de dados, operadores e funções.

- **Supporte a Dados Complexos:** Lida eficientemente com dados estruturados e não estruturados, incluindo JSON, XML e *arrays*.
- **Comunidade Ativa:** Possui uma grande comunidade que contribui para seu desenvolvimento.
- **Licença Permissiva:** Sua licença permite o uso e modificação sem restrições significativas.

2.5.6 Visual Studio Code

Visual Studio Code (VS Code) é um editor de código-fonte leve, gratuito e multiplataforma desenvolvido pela Microsoft ([Microsoft Corporation, 2024b](#)). Tornou-se um dos editores mais populares entre desenvolvedores de diversas linguagens.

Principais características do VS Code:

- **Extensibilidade:** Possui um vasto *marketplace* de extensões que adicionam funcionalidades, suporte a linguagens e ferramentas.
- **IntelliSense:** Oferece autocompletar inteligente baseado em tipos de variáveis e definições de funções.
- **Depuração Integrada:** Permite depurar o código diretamente no editor.
- **Controle de Versão Integrado:** Suporte nativo para Git.
- **Terminal Integrado:** Um terminal de linha de comando embutido no editor.
- **Leve e Rápido:** Conhecido por ser rápido e responsivo.

2.5.7 GitHub

GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando Git ([GitHub Inc., 2024](#)). É a maior plataforma para desenvolvimento colaborativo de *software*.

Funcionalidades chave do GitHub:

- **Controle de Versão (Git):** Permite rastrear e gerenciar todas as alterações no código.
- **Repositórios:** Cada projeto é um repositório, que pode ser público ou privado.
- **Pull Requests:** Mecanismo para propor e revisar alterações no código.
- **Issues:** Ferramenta para rastrear bugs, funcionalidades e tarefas.
- **Projetos (Kanban Boards):** Quadros estilo *Kanban* para gerenciamento visual de tarefas.
- **Actions (CI/CD):** Ferramenta para automação de fluxos de trabalho, como integração e entrega contínua.
- **Wiki e Páginas:** Para documentação do projeto.

2.5.8 Microsoft Teams

Microsoft Teams é uma plataforma unificada de comunicação e colaboração desenvolvida pela Microsoft ([Microsoft Corporation, 2024a](#)). Ela integra *chat*, reuniões de vídeo, armazenamento de arquivos e integração de aplicativos.

Recursos do Microsoft Teams:

- **Chat e Canais:** Permite comunicação em tempo real em conversas individuais ou em canais de equipe.
- **Reuniões Online:** Funcionalidades completas para reuniões de vídeo e áudio, com compartilhamento de tela e gravação.
- **Compartilhamento de Arquivos:** Facilita o compartilhamento e a coautoria de documentos.
- **Integrações:** Compatível com uma vasta gama de aplicativos, além de toda a suíte Microsoft 365.
- **Segurança e Conformidade:** Oferece recursos avançados de segurança e privacidade para dados corporativos.

2.5.9 Docker

Docker é uma plataforma de código aberto que automatiza a implantação, o dimensionamento e o gerenciamento de aplicações dentro de *containers* ([Docker Inc., 2024](#)). A tecnologia permite empacotar uma aplicação com todas as suas dependências — como bibliotecas, código e ambiente de execução — em uma unidade padronizada.

O objetivo principal do Docker é garantir a consistência entre múltiplos ambientes. Ele resolve o problema de "funciona na minha máquina" ao assegurar que o *software* se comporte da mesma maneira em desenvolvimento, testes e produção.

Os principais benefícios do Docker são:

- **Portabilidade:** Um *container* pode ser executado em qualquer máquina que tenha o Docker instalado, independentemente do sistema operacional subjacente.
- **Isolamento:** Aplicações em *containers* rodam de forma isolada, impedindo que uma aplicação interfira em outra.
- **Eficiência:** São mais leves que máquinas virtuais tradicionais, pois compartilham o *kernel* do sistema operacional *host*, resultando em inicialização rápida e menor consumo de recursos.
- **Reprodutibilidade:** O uso de um *Dockerfile* permite definir a construção do ambiente de forma programática e versionável.

3 Metodologias

3.1 Metodologia de pesquisa

Para o desenvolvimento desta pesquisa, foram adotadas abordagens metodológicas que estruturam de forma sistemática os métodos de coleta, análise e interpretação dos dados, garantindo consistência e clareza em todas as etapas do estudo.

Em relação à abordagem de pesquisa, adota-se o método **qualitativo**, uma vez que ele possibilita compreender em profundidade as percepções e necessidades dos usuários diretos, contribuindo para a construção de um backlog que reflita fielmente essas demandas.

Quanto à natureza, caracteriza-se como uma **pesquisa aplicada**, pois visa não apenas gerar conhecimento teórico, mas também utilizá-lo para resolver um problema prático por meio do desenvolvimento de um software funcional, baseado em requisitos reais identificados durante o processo.

No que se refere aos objetivos, trata-se de uma **pesquisa exploratória**, já que busca aprofundar a compreensão do fenômeno investigado — as necessidades específicas do público-alvo — de modo a embasar a proposição de uma solução tecnológica adequada.

No que tange aos procedimentos técnicos, optou-se por uma combinação de **pesquisa bibliográfica, pesquisa de campo e estudo de caso**. A pesquisa bibliográfica fornecerá o embasamento teórico necessário para sustentar as decisões de desenvolvimento, enquanto a pesquisa de campo e o estudo de caso permitirão o contato direto com a realidade dos usuários, ampliando o entendimento de suas demandas específicas.

Por fim, como procedimento de coleta de dados, será utilizada a **observação**, ferramenta essencial para registrar comportamentos, interações e contextos de uso do sistema, possibilitando um diagnóstico mais preciso das necessidades que o software deverá atender.

Abordagem	Pesquisa qualitativa	Pesquisa quantitativa
Natureza	Pesquisa básica	Pesquisa aplicada
Objetivos	Pesquisa exploratória	Pesquisa descritiva
	Pesquisa experimental	Pesquisa explicativa
Procedimentos	Pesquisa documental	
	Pesquisa de campo	Pesquisa ex-post-facto
	Pesquisa com survey	Pesquisa de levantamento
	Pesquisa -ação	Pesquisa participante
Procedimentos de coleta de dados	Pesquisa documental	Pesquisa eletrônica
	Pesquisa bibliográfica	Pesquisa eletrônica
	Questionário	Formulário
	Observação	Entrevista
		Diário de campo

Figura 3.1 – Metodologia da Pesquisa

Fonte: Autor, 2025.

3.2 Metodologia de Desenvolvimento de Software

3.2.1 Ciclo de Vida

No desenvolvimento deste software, tornou-se essencial, antes de qualquer implementação, compreender profundamente o contexto do problema e, a partir disso, selecionar um ciclo de vida adequado, que orientasse as etapas e atividades necessárias para a construção do sistema. A abordagem escolhida foi fundamentada nos princípios das metodologias ágeis, com ênfase no Scrum, devido à sua capacidade de estruturar o trabalho de forma iterativa, incremental e centrada na entrega contínua de valor.

Contudo, algumas adaptações foram realizadas para adequar o framework às características específicas do projeto. Uma dessas adaptações foi a não realização das daily meetings, reuniões diárias propostas pelo Scrum para promover alinhamento constante da equipe. Tal decisão ocorreu considerando a dinâmica enxuta do desenvolvimento, na qual a comunicação se deu predominantemente de forma assíncrona ou sob demanda, sempre que surgirem necessidades específicas.

Já a estrutura tradicional de papéis definida pelo Scrum — como Scrum Master, Development Team e Product Owner — foi adaptada para a realidade deste projeto. Em função do tamanho reduzido da equipe de desenvolvimento, optou-se por não designar um Scrum Master. Por outro lado, o papel de Product Owner (PO) foi mantido, sendo desempenhado pela Professora Paula Uessugue, que assume a responsabilidade de representar as necessidades dos usuários e priorizar os requisitos do produto.

Além das práticas do Scrum, este projeto incorporou também técnicas derivadas do Extreme Programming (XP), com o objetivo de aprimorar a qualidade do código e aumentar a produtividade. Dentre as práticas adotadas do XP, destacam-se a programação em pares (pair programming), que favorece a troca constante de conhecimento e a redução de erros; a refatoração contínua, voltada à melhoria incremental da estrutura do código; e a ênfase na simplicidade, buscando sempre soluções diretas e eficazes para os problemas encontrados. A adoção dessas práticas contribui para um desenvolvimento mais sustentável, com foco na manutenibilidade, qualidade e agilidade.

A figura 3.2 representa todo o ciclo de vida do projeto, estruturado com base nos artefatos e eventos definidos. A formação do backlog da sprint parte do backlog do produto e ocorre mediante uma priorização que considera o valor de negócio atribuído na etapa de levantamento e organização dos requisitos.

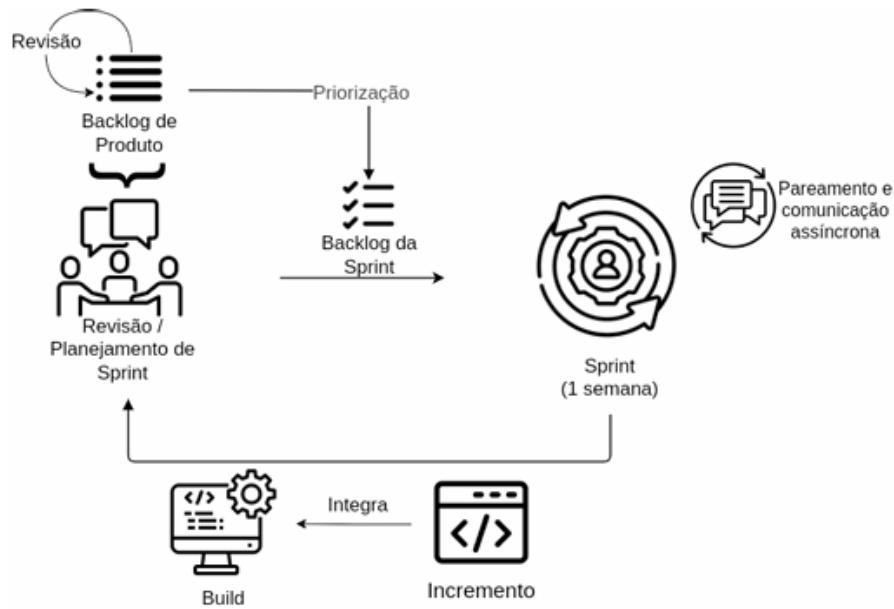


Figura 3.2 – Ciclo de vida

Fonte: Autor, 2025.

3.2.2 Artefatos

Ao longo do desenvolvimento do projeto, fez-se indispensável a utilização de determinados artefatos derivados tanto do framework Scrum quanto da metodologia visual Kanban, com o objetivo de organizar, acompanhar e gerir o progresso das atividades de maneira eficiente e estruturada.

Entre os principais artefatos empregados, destaca-se o **Backlog do Produto**, que consiste em uma lista priorizada contendo todas as funcionalidades, melhorias e requisitos identificados para o sistema. Este backlog representa a visão global do produto, servindo como fonte principal para seleção das demandas que seriam trabalhadas em cada ciclo de desenvolvimento (Sprint).

Outro artefato essencial foi o **Backlog da Sprint**, que corresponde a um subconjunto do backlog do produto, contendo exclusivamente as tarefas e funcionalidades selecionadas para serem desenvolvidas durante uma sprint específica. Este documento é atualizado de forma contínua, refletindo o avanço das tarefas, as mudanças nas prioridades e eventuais ajustes identificados durante a execução.

3.2.3 Fases de Desenvolvimento

- **Revisão e Planejamento da Sprint:** Ao final de cada ciclo semanal, foram discutidos os principais acontecimentos do ciclo, identificando os pontos positivos, os desafios enfrentados e as oportunidades de melhoria. Com base nessa análise, foi possível apri-

morar as estimativas de esforço e a organização das tarefas futuras. Simultaneamente, foram definidos os itens do Backlog do Produto que foram priorizados para compor o Backlog da Sprint subsequente, sempre considerando a ordem de prioridade estabelecida. Quando necessário, ajustes também foram feitos no próprio backlog do produto, refletindo aprendizados e necessidades observadas nos incrementos anteriores.

- **Execução da Sprint:** Durante a sprint, o desenvolvimento foi concentrado nas funcionalidades planejadas no backlog da sprint. A comunicação entre os integrantes foi realizada predominantemente de maneira assíncrona, permitindo que eventuais dúvidas ou problemas fossem resolvidos conforme surgindo.
- **Programação em Pares (Pair Programming):** Inspirada nas práticas do Extreme Programming (XP), a técnica de programação em pares foi adotada sempre que se mostrou pertinente ao longo das semanas de desenvolvimento. Essa prática consiste em dois desenvolvedores trabalhando simultaneamente sobre o mesmo trecho de código, colaborando na resolução de problemas, no compartilhamento de conhecimentos e na melhoria da qualidade do software. Esse método demonstrou-se especialmente útil na superação de dificuldades técnicas e na aceleração do desenvolvimento.
- **Testes:** Paralelamente à implementação das funcionalidades, foi conduzida a criação dos testes correspondentes, alinhados às boas práticas de desenvolvimento. Dependendo da natureza e da criticidade de cada funcionalidade, foram aplicados testes unitários, de integração ou de sistema. Essa estratégia visou assegurar que cada parte do software esteja funcionando corretamente, bem como garantir a robustez e a confiabilidade do aplicativo como um todo.

3.2.4 Requisitos

Para que o projeto de software fosse desenvolvido de forma eficiente e alinhada às expectativas dos usuários, tornou-se fundamental, em um primeiro momento, realizar a identificação dos seus requisitos, ou seja, das especificações que definem as funcionalidades, comportamentos e restrições do sistema. Esses requisitos serviram como base para orientar todo o processo de desenvolvimento, garantindo que o produto final atendesse às necessidades para as quais foi concebido. Abaixo é detalhado como cada fase da engenharia de requisitos foi executada no contexto deste TCC.

Elicitação e análise

A etapa de elicitação dos requisitos foi conduzida por meio de uma entrevista estruturada com a professora Paula Uessugue, docente do curso de Nutrição da Universidade de Brasília (UnB), que também atua como Product Owner (PO) para o desenvolvimento do

aplicativo. O principal objetivo dessa entrevista foi compreender, de maneira aprofundada, quais são as necessidades dos usuários no contexto de utilização do sistema, bem como identificar quais instrumentos e questionários seriam aplicados para o registro das informações dentro da aplicação.

Além disso, buscou-se entender de forma clara quais seriam os resultados esperados a partir do preenchimento desses questionários, tanto no que se refere ao processamento dos dados quanto à geração das sugestões alimentares. Também foi discutida a maneira mais adequada de apresentar esses resultados aos usuários na interface do aplicativo, garantindo clareza e alinhamento com as demandas práticas dos usuários.

Especificação

Após a realização das entrevistas, todas as informações coletadas foram organizadas e consolidadas na forma de requisitos, que foram devidamente documentados em um arquivo no formato Markdown, hospedado no repositório do projeto no GitHub. Esse documento serviu como base inicial para a formalização dos requisitos, sendo continuamente atualizado conforme surgiam novos entendimentos e validações ao longo do desenvolvimento. Assim sendo, este material foi estruturado como Backlog do Produto, que passou a desempenhar o papel central na gestão das funcionalidades, priorização de tarefas e acompanhamento do progresso do projeto.

Validação

Na fase de validação dos requisitos, foram aplicadas práticas recomendadas na engenharia de software, conforme proposto por (Sommerville; Sawyer, 1997). Esse processo teve como objetivo assegurar que os requisitos definidos estivessem corretos, completos e alinhados com as necessidades do projeto. Dentre as atividades realizadas, destacam-se:

- Análise de Consistência: Procedimento destinado a verificar se os requisitos estavam livres de conflitos, contradições, ambiguidades ou duplicidades que pudessem comprometer o desenvolvimento.
- Análise de Completude: Avaliação cujo foco foi assegurar que todos os comportamentos esperados do sistema, bem como suas funcionalidades essenciais, estivessem devidamente representados nos requisitos especificados.

Durante esse processo, eventuais inconsistências ou lacunas identificadas foram imediatamente corrigidas, evitando assim que erros se propagassem para as etapas posteriores de desenvolvimento. Essa abordagem preventiva foi fundamental para reduzir retrabalho e garantir maior eficiência no ciclo de desenvolvimento do aplicativo.

3.2.5 Arquitetura

A presente seção tem como objetivo descrever a arquitetura adotada na solução proposta para o desenvolvimento deste aplicativo. De acordo com (Pressman; Maxim, 2021), a escolha do padrão arquitetural exerce influência direta sobre todo o ciclo de vida do software, uma vez que define a organização dos artefatos, os fluxos de dados, a modularização e a estrutura geral do sistema. A arquitetura escolhida orienta não apenas como os componentes são desenvolvidos, mas também como eles interagem entre si e evoluem ao longo do tempo.

Dentre os diversos estilos arquiteturais existentes, optou-se pela utilização do estilo Arquitetural Baseado em Componentes, devido às suas características que promovem uma alta coesão, baixo acoplamento, facilidade de manutenção, escalabilidade e forte aderência a projetos desenvolvidos com frameworks baseados em componentes, como é o caso do React Native. Para o modelo arquitetural, foi escolhida a arquitetura em 3 camadas. Essa estrutura ofereceu clareza na organização e no gerenciamento do projeto, além de ter facilitado o desenvolvimento e a manutenção dos componentes do software.

3.2.6 Estilo Arquitetural Baseado em Componentes

A Arquitetura Componentizada consiste na divisão do sistema em unidades menores chamadas de componentes, que são independentes, reutilizáveis e responsáveis por funcionalidades bem definidas dentro da aplicação. Cada componente encapsula sua lógica, sua interface e seu estado, comunicando-se com os demais por meio de interfaces e propriedades bem definidas.

De acordo com (Pressman; Maxim, 2021), esse modelo arquitetural favorece a construção de sistemas mais robustos, modulares e de fácil manutenção, uma vez que qualquer alteração realizada em um componente não impacta diretamente os demais, desde que a interface de comunicação seja preservada.

No contexto do aplicativo proposto, os componentes serão organizados da seguinte maneira:

- **Componentes de Interface (UI Components):** São responsáveis pela construção da interface gráfica do aplicativo, incluindo botões, formulários, listas, cards, menus e demais elementos visuais. Estes componentes são desenvolvidos utilizando React Native em conjunto com o framework Expo, além do auxílio do NativeWind para estilização baseada em Tailwind CSS.
- **Componentes de Página (Screens):** Representam cada uma das telas do aplicativo, sendo compostos pela combinação de múltiplos componentes de interface e pela integração com os dados e funcionalidades necessárias. Por exemplo, a tela de cadastro de preferências alimentares ou a tela de exibição das sugestões geradas.
- **Componentes Funcionais (Hooks e Serviços):** Abrangem funções responsáveis

por tratar regras específicas, manipulação de estado, requisições assíncronas e conexão com serviços de dados. Incluem também hooks personalizados que encapsulam lógicas como validações de formulários, controle de estado e consumo de APIs.

Essa organização permitiu que cada parte do sistema fosse desenvolvida, testada e evoluída de maneira independente, promovendo maior agilidade no desenvolvimento, facilidade de manutenção e possibilidade de reutilização de código em diferentes contextos do aplicativo. Além disso, a arquitetura componentizada é altamente compatível com metodologias ágeis, como Scrum e práticas do Extreme Programming (XP), pois favorece entregas incrementais e contínuas.

Dessa forma, a arquitetura adotada neste projeto buscou equilibrar simplicidade estrutural, desempenho e escalabilidade, além de proporcionar uma base sólida para a evolução contínua do sistema. Abaixo, na Figura 3.3 há uma representação esquemática da estrutura de diretórios adotada para o projeto, seguindo o modelo do estilo arquitetural componentizado.

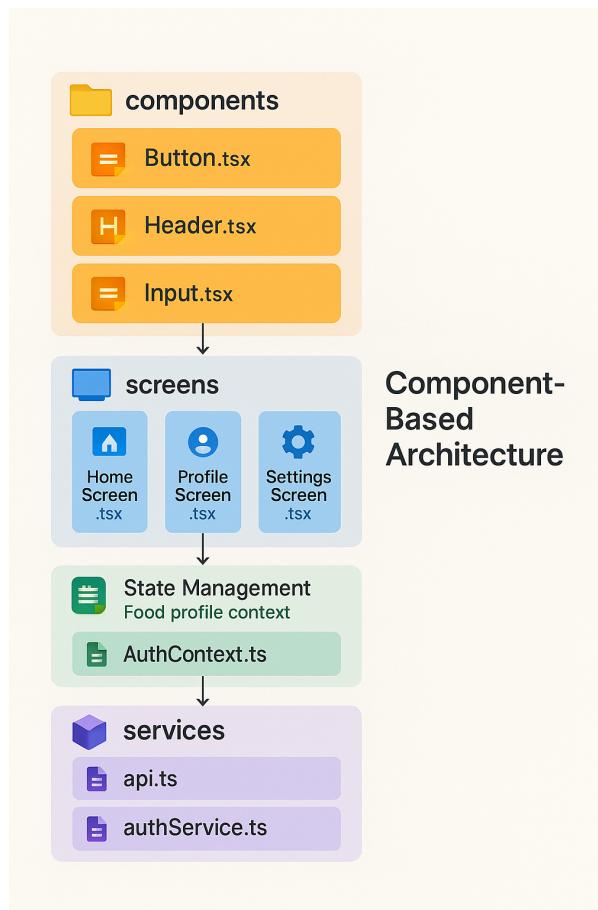


Figura 3.3 – Estrutura da arquitetura baseada em componentes

Fonte: Autor, 2025.

Essa estrutura favoreceu o isolamento de responsabilidades, o reaproveitamento de

componentes e a escalabilidade do projeto, alinhando-se às boas práticas de desenvolvimento de aplicações móveis com React Native e Expo. Além disso, facilitou o trabalho colaborativo da equipe, uma vez que cada parte do sistema pode ser desenvolvida, testada e mantida de forma independente, sem comprometer a integridade do restante do código.

3.2.7 Arquitetura Em 3 Camadas

A Arquitetura em 3 Camadas fundamenta-se na segmentação da aplicação em três níveis lógicos e independentes: a camada de apresentação, a camada de lógica de negócios e a camada de dados. Essa estrutura promove o desacoplamento das responsabilidades, permitindo que cada componente seja desenvolvido, mantido e escalado de forma autônoma, sem criar dependências rígidas entre a interface e o armazenamento. De acordo com (Pressman; Maxim, 2016), esse modelo favorece a manutenibilidade e a flexibilidade do software, pois possibilita que alterações em uma camada específica, como a atualização da interface do usuário, ocorram sem impactar necessariamente às regras de negócio ou a estrutura do banco de dados.

3.2.8 Desenvolvimento de Software

O processo de desenvolvimento da aplicação foi sustentado por um conjunto de tecnologias modernas e amplamente utilizadas na indústria de software, escolhidas por sua eficiência, ampla documentação e alinhamento com os objetivos do projeto. A seguir, são descritas as principais ferramentas e ambientes adotados durante a construção do sistema.

- **React Native:** O React Native é um framework criado pelo Facebook que permite o desenvolvimento de aplicativos móveis nativos utilizando a linguagem JavaScript e o paradigma de componentes do React. Sua principal vantagem está na possibilidade de desenvolver aplicações para Android e iOS a partir de uma única base de código, o que reduz custos e acelera o tempo de entrega. Segundo (Kuederle, 2018), o React Native oferece uma arquitetura flexível, baseada em componentes reutilizáveis, o que promove maior manutenibilidade e escalabilidade dos projetos.
- **Node.js:** O Node.js é um ambiente de execução para código JavaScript no lado do servidor, baseado no motor V8 do Google Chrome. Destaca-se por sua natureza event-driven e assíncrona, o que o torna altamente eficiente para aplicações que demandam escalabilidade e manipulação intensiva de I/O (entrada e saída). Além disso, trata-se de uma tecnologia de código aberto, amplamente suportada pela comunidade e adotada em larga escala por grandes empresas. Conforme a documentação oficial (OpenJS Foundation, 2024), o modelo orientado a eventos e não bloqueante do Node.js o torna leve e eficiente, sendo ideal para aplicações de dados intensivos em tempo real que rodam em dispositivos distribuídos.

- **NativeWind:** O NativeWind é uma biblioteca que permite a utilização do utilitário Tailwind CSS dentro de projetos desenvolvidos com React Native. Essa abordagem facilita a criação de interfaces coesas e responsivas por meio da aplicação de classes utilitárias diretamente nos componentes da interface. A principal vantagem do NativeWind é a redução da complexidade na estilização, além da maior legibilidade e reuso do código, o que contribui para a agilidade no desenvolvimento e a consistência visual da aplicação.
- **Visual Studio Code:** O Visual Studio Code (VS Code) é um editor de código-fonte leve, gratuito e multiplataforma, com suporte nativo a diversas linguagens de programação. Sua extensibilidade por meio de plugins torna-o uma ferramenta versátil, adequada para o desenvolvimento com React Native, Node.js e outras tecnologias modernas. De acordo com ([Spinellis, 2021](#)), o VS Code representa uma solução moderna e eficiente para ambientes de desenvolvimento integrados em equipes ágeis.
- **Docker:** O Docker é uma plataforma de containerização que permite isolar processos em ambientes independentes. Neste projeto, a ferramenta foi utilizada especificamente para a sustentação e o gerenciamento da instância do banco de dados. Essa abordagem permitiu a configuração de um ambiente de dados padronizado e isolado, dispensando a instalação direta do SGBD na máquina local dos desenvolvedores. O uso de contêineres para serviços específicos otimizou o consumo de recursos e garantiu a consistência do ambiente de desenvolvimento, evitando conflitos de versões e dependências.
- **PostgreSQL:** O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional (SGBD-OR) de código aberto, reconhecido por sua robustez, estabilidade e conformidade com os padrões ACID (Atomicidade, Consistência, Isolamento e Durabilidade). No projeto, ele atuou na camada de dados, sendo responsável pelo armazenamento seguro e estruturado das informações dos usuários, perfis sensoriais e histórico alimentar.

3.2.9 Gerência de configuração de software

A gerência de configuração de software (SCM) é uma disciplina fundamental dentro da engenharia de software, responsável por estabelecer processos, diretrizes e práticas voltadas ao controle sistemático das modificações realizadas no sistema ao longo do seu ciclo de desenvolvimento. Sua aplicação visa assegurar que todas as alterações sejam devidamente rastreadas, documentadas e integradas de forma controlada.

Essa prática torna-se essencial para evitar problemas como a perda de informações sobre o histórico de mudanças, a ocorrência de retrabalho e a geração de inconsistências entre diferentes versões do sistema desenvolvidas simultaneamente pela equipe. Segundo ([Sommerville, 2007](#)), a ausência de um controle eficiente sobre a configuração do software

pode comprometer significativamente a integridade do projeto e dificultar sua manutenção e evolução.

3.2.10 Repositório

Para o armazenamento e controle do código-fonte do projeto, foi utilizado um repositório na plataforma GitHub, que centraliza todo o desenvolvimento da aplicação. O repositório foi configurado como privado, garantindo que apenas os membros da equipe de desenvolvimento e o orientador tivessem acesso aos artefatos do projeto. A escolha pelo GitHub se deu, principalmente, devido à familiaridade da equipe com a ferramenta, além de seus recursos robustos de controle de versão, colaboração e rastreamento de alterações, os quais são fundamentais para manter a integridade e a organização do desenvolvimento ao longo de todo o ciclo de vida do software.

3.2.11 Política de Branchs

A estratégia adotada para a organização das branches no repositório foi inspirada no modelo Git Flow, proposto por Vincent Driessen ([Driessen, 2010](#)), amplamente utilizado por equipes de desenvolvimento devido à sua robustez na gestão de versões e controle de mudanças. Contudo, para atender às necessidades específicas deste projeto, foram realizadas algumas adaptações na estrutura tradicional do Git Flow, incluindo a adição das branches release e docs, as quais não fazem parte da proposta original, mas foram fundamentais para o fluxo de trabalho adotado.

A definição e a função de cada branch ficaram estabelecidas da seguinte maneira:

- **Main:** Contém a versão mais estável e consolidada do projeto, pronta para deploy ou entrega. Qualquer alteração nesta branch reflete uma versão oficialmente liberada do software.
- **Develop:** Serve como branch de integração, onde são reunidas todas as novas funcionalidades e correções desenvolvidas antes de serem promovidas à versão estável. É, portanto, uma versão de desenvolvimento que reflete o estado atual da construção do sistema.
- **Feature:** Branch criada a partir da develop, destinada ao desenvolvimento de uma nova funcionalidade, melhoria específica ou implementação de algum item do backlog. Após finalizada e testada, é integrada novamente à develop.
- **Hotfix:** Utilizada exclusivamente para correções emergenciais em produção. Quando um problema crítico é identificado na branch main, essa branch é criada para resolver rapidamente o problema e, após a correção, é integrada tanto na main quanto na develop.

- **Release:** Branch criada a partir da develop quando uma versão do sistema está próxima de ser finalizada. Nela são feitas correções menores, ajustes finais e preparação para o lançamento. Após a finalização, ela é mesclada tanto na main quanto na develop.
- **Docs:** Branch dedicada exclusivamente à documentação do projeto. Permite que a documentação evolua de forma independente do desenvolvimento do código, facilitando atualizações, correções e melhorias contínuas nos materiais de apoio, como o README, Wiki ou documentos técnicos.

Essa abordagem híbrida proporcionou uma organização eficiente do fluxo de trabalho, assegurando maior controle sobre os diferentes estágios de desenvolvimento, manutenção da qualidade do código e facilidade na gestão de entregas e documentação. A introdução das branches release e docs contribuirá significativamente para atender às demandas específicas deste projeto acadêmico, tornando o processo mais organizado, rastreável e alinhado às boas práticas de desenvolvimento de software.

3.2.12 Política de commits

Para garantir clareza, rastreabilidade e organização no histórico de desenvolvimento do projeto, foi adotada uma política de padronização nas mensagens de commit. Essa prática visa facilitar a compreensão das alterações realizadas, tanto para os desenvolvedores envolvidos quanto para eventuais colaboradores futuros, além de contribuir para a manutenção, revisão e evolução do código de forma eficiente.

O formato utilizado segue o seguinte padrão: **<tipo> <Descrição breve e objetiva>**. Nessa estrutura, o tipo indica a natureza da alteração realizada no código, enquanto a descrição resume, de forma clara, o que foi modificado. A seguir, são descritos os tipos definidos e suas respectivas finalidades:

- **feat:** Indica a implementação de uma nova funcionalidade no sistema.
- **fix:** Refere-se à correção de bugs ou problemas identificados.
- **docs:** Usado para alterações relacionadas à documentação, como atualizações no README ou em arquivos de suporte.
- **style:** Alterações puramente estéticas ou de formatação, como ajustes de identação, remoção de espaços, alterações de estilo, que não afetam a lógica do código.
- **refactor:** Aplicado em mudanças na estrutura do código que não alteram seu comportamento externo, como melhorias internas, reorganização de funções ou métodos.
- **test:** Relacionado à criação, modificação ou melhoria de testes automatizados.
- **chore:** Utilizado para tarefas de manutenção e configuração do projeto, que não estão diretamente ligadas ao desenvolvimento de funcionalidades ou correção de erros. Exemplos incluem atualizações de dependências, ajustes em scripts ou arquivos de

configuração.

Exemplos de commits seguindo essa convenção:

- **feat:** adicionar tela de cadastro de usuários
- **fix:** corrigir erro no comportamento do botão de login
- **docs:** atualizar README com instruções de execução
- **style:** ajustar espaçamentos e identação nos componentes de tela
- **refactor:** reorganizar funções do serviço de autenticação
- **test:** adicionar testes unitários para o componente de sugestões
- **chore:** atualizar dependências do projeto no package.json

A adoção desta política contribui significativamente para a manutenção de um histórico limpo, organizado e semântico, alinhando-se às boas práticas de desenvolvimento de software recomendadas por autores como (Pressman; Maxim, 2021). Além disso, essa abordagem favorece a utilização futura de ferramentas de integração contínua, geração de changelogs automáticos e auditoria de modificações no código-fonte.

3.3 Métricas

Esta seção tem como finalidade descrever o processo de coleta e análise de métricas, com ênfase em fornecer subsídios para o acompanhamento administrativo e o gerenciamento do cumprimento do cronograma de entregas planejadas para o projeto.

3.3.1 GQM

Para apoiar esse processo, adotou-se a estratégia GQM (Goal-Question-Metric), proposta por (Basili; Caldiera; Rombach, 1994), que estabelece um método sistemático para definir e organizar a medição em projetos de software. O GQM estrutura o monitoramento em três níveis: objetivos, questões e métricas.

A aplicação do método ocorreu por meio das seguintes etapas:

- Definição dos objetivos principais, que destacam os focos prioritários da medição dentro do contexto do projeto.
- Formulação de perguntas, que detalham cada objetivo e norteiam a investigação sobre o progresso e o desempenho das atividades.
- Determinação das métricas, especificando os dados a serem coletados para responder de forma clara às questões formuladas.
- Coleta, verificação e análise dos dados, garantindo que os registros sejam válidos, confiáveis e úteis para apoiar a tomada de decisão.

A utilização dessa abordagem possibilitou uma avaliação estruturada do andamento do desenvolvimento, identificando pontos de conformidade com as metas traçadas, além de sinalizar oportunidades de ajustes e melhorias nos processos.

No âmbito desta pesquisa, o GQM foi especialmente direcionado para o acompanhamento das atividades de desenvolvimento de software, uma etapa central e de alta criticidade no projeto. Assim, foi realizada uma declaração formal dos objetivos de medição, sintetizada na Tabela 3.1, que detalha cada meta definida e as métricas correspondentes, permitindo um monitoramento claro e objetivo de todos os aspectos relevantes para a gestão do progresso do trabalho.

Objetivos de Medição	Descrição do Objetivo	Questões e Métricas
Objetivo 1: Gerenciar e acompanhar o progresso do cumprimento dos requisitos do sistema a partir do backlog	Analisar o cumprimento dos requisitos do sistema com o propósito de entender o progresso do desenvolvimento em relação ao que foi planejado. Em relação ao backlog do produto. Do ponto de vista do gerente de projeto e do cliente. No contexto do desenvolvimento do sistema.	<p>Questão 1.1: O backlog atende às necessidades do usuário?</p> <p>Métrica 1.1.1: Número de tarefas concluídas do backlog</p> <p>Questão 1.2: O backlog foi cumprido?</p> <p>Métrica 1.2.1: Número de tarefas concluídas do backlog</p>
Objetivo 2: Gerenciar e acompanhar o cronograma do desenvolvimento do sistema	Analisar o andamento do cronograma com o propósito de verificar a aderência aos prazos estabelecidos. Em relação ao escopo do projeto. Do ponto de vista do gerente de projetos e dos stakeholders. No contexto do planejamento e execução do projeto.	<p>Questão 2.1: O cronograma está realista para o escopo do projeto?</p> <p>Métrica 2.1.1: Porcentagem de funcionalidades mínimas implementadas no MVP.</p>
Objetivo 3: Gerenciar e acompanhar o progresso do desenvolvimento do sistema	Analisar o processo de desenvolvimento com o propósito de avaliar a produtividade da equipe e a qualidade do produto. Em relação à entrega de valor e à saúde do código. Do ponto de vista da equipe de desenvolvimento e do líder técnico. No contexto das sprints e do ciclo de vida do software.	<p>Questão 3.1: Qual a produtividade média da equipe?</p> <p>Métrica 3.1.1: Taxa de bugs corrigidos (bugs corrigidos / total de bugs encontrados).</p> <p>Métrica 3.1.2: Número de tarefas concluídas</p> <p>Questão 3.2: Quantas versões do produto foram desenvolvidas?</p> <p>Métrica 3.2.1: Total de releases entregues</p> <p>Questão 3.3: Quantas tarefas estão relacionadas a débitos técnicos?</p> <p>Métrica 3.3.1: Número de débitos técnicos registrados</p>

Tabela 3.1 – Objetivos, questões e métricas definidas pelo método GQM

Fonte: Autor, 2025.

3.4 Testes

A realização de testes durante o processo de desenvolvimento foi um fator essencial para assegurar que o sistema estivesse funcionando conforme os requisitos definidos, de maneira confiável, eficiente e com qualidade. Com base nisso, foram adotadas estratégias de verificação sistemática por meio da aplicação de três tipos principais de testes: testes unitários, testes de integração e testes de sistema.

3.4.1 Testes unitários

Os testes unitários foram utilizados devido à sua praticidade e agilidade na execução. Esse tipo de teste é focado na validação de unidades isoladas do sistema — como funções e componentes — permitindo detectar falhas de forma precoce e com baixo custo. Por suas particularidades, os testes unitários contribuíram para garantir que pequenas porções da lógica de negócio estejam funcionando corretamente e que os requisitos estejam sendo devidamente atendidos.

Para a implementação dos testes unitários, foi utilizada a ferramenta Jest, um framework amplamente consolidado na comunidade JavaScript, conhecido por sua facilidade de configuração, performance otimizada e abrangente suporte à análise de cobertura de código.

Segundo a documentação oficial, o Jest foi inicialmente desenvolvido pelo Facebook com foco em testes de aplicações React, mas atualmente é utilizado em projetos variados de front-end e back-end, tornando-se uma solução robusta e versátil. Além disso, o Jest se destaca por executar testes de forma paralela, o que proporciona ganhos significativos de desempenho em projetos com grande volume de código.

A escolha pelo Jest se justifica, portanto, não apenas pela sua compatibilidade com o ecossistema do React Native, mas também por sua ampla adoção pela comunidade, extensa documentação e facilidade na identificação de problemas. Tais características o tornam uma ferramenta adequada para o contexto deste projeto, que exige confiabilidade, agilidade e suporte técnico durante as fases de desenvolvimento e validação.

3.4.2 Testes de integração

Os testes de integração desempenharam um papel essencial na verificação da interação entre diferentes módulos do sistema, assegurando que as funcionalidades implementadas de forma isolada operem corretamente quando combinadas. Esse tipo de teste visa identificar falhas na comunicação entre componentes, como inconsistências de dados, falhas de autenticação ou problemas na persistência das informações.

No contexto deste projeto, os testes de integração foram empregados principalmente para verificar operações críticas que envolvem múltiplos elementos do sistema, como, por exemplo, o fluxo de cadastro de um usuário, no qual a comunicação entre o front-end, o back-end e o banco de dados é fundamental para o correto funcionamento da funcionalidade.

Para a realização dos testes, foi utilizado o Insomnia, uma ferramenta open-source amplamente utilizada para a realização de requisições HTTP no padrão REST. O Insomnia permitiu a simulação de chamadas a endpoints da API da aplicação, oferecendo uma interface intuitiva para o envio de dados, análise das respostas e verificação dos comportamentos esperados.

A escolha pelo Insomnia se deu por sua facilidade de uso, suporte a diferentes métodos HTTP e recursos como histórico de requisições, organização por coleções e visualização clara das respostas da API. Tais características o tornam especialmente útil durante a fase de testes de integração, permitindo identificar com precisão eventuais falhas de comunicação entre os componentes da aplicação.

3.4.3 Testes Funcionais e de Interface (UI)

Além dos testes unitários e de integração, foram aplicados testes funcionais e de interface com o objetivo de validar o comportamento do sistema do ponto de vista do usuário final. Esses testes asseguram que as funcionalidades descritas nos requisitos estejam sendo executadas corretamente quando acionadas por meio da interface gráfica, e que a interação com os componentes visuais esteja ocorrendo conforme o esperado.

Os testes funcionais verificam se o sistema atende aos seus propósitos e realiza as tarefas previstas — como o envio de formulários, navegação entre telas, e a geração de sugestões alimentares com base nos dados inseridos. Já os testes de interface (também chamados de testes de UI) focam na resposta visual e comportamental dos elementos da aplicação, como botões, campos de entrada, mensagens de erro e feedbacks visuais, garantindo que estejam responsivos e alinhados com as diretrizes de usabilidade.

Para essa finalidade, foi empregada ferramentas como a React Native Testing Library, que é uma extensão da popular Testing Library adaptada para aplicações desenvolvidas em React Native. Essa biblioteca ofereceu um conjunto de utilitários para simular interações do usuário e verificar a presença, comportamento e acessibilidade dos componentes da interface. Segundo (Domingues Daniel e Moreira, 2020), a adoção de bibliotecas que promovem testes baseados na perspectiva do usuário contribui significativamente para o aumento da confiabilidade da aplicação e melhora da experiência de uso.

A escolha por realizar testes funcionais e de interface também se justifica pela natureza do público-alvo da aplicação — cuidadores de pessoas com Transtorno do Espectro Autista (TEA) —, o que exige interfaces claras, acessíveis e com feedbacks visuais bem definidos. A

validação contínua dessas interações é, portanto, indispensável para assegurar que o sistema seja compreensível e eficaz em sua proposta.

3.5 Tecnologias

3.5.1 Gestão do projeto

- **GitHub:** O GitHub é uma plataforma baseada em computação em nuvem voltada ao desenvolvimento colaborativo de software. Ele oferece uma série de funcionalidades que auxiliam no gerenciamento de projetos, entre as quais se destacam: o controle de versão distribuído, por meio da integração com o sistema Git; a criação e organização de issues; a geração e distribuição de releases (versões estáveis do sistema); e o uso de quadros no estilo Kanban, que permitem o acompanhamento visual do progresso das atividades.

Conforme apresentado na subseção 3.2.4.0.2 — Política de Branches — o repositório do projeto, hospedado no GitHub, foi estruturado com seis branches distintas. Cada uma delas foi definida para atender a demandas específicas ao longo do ciclo de desenvolvimento, garantindo organização, rastreabilidade e melhor gerenciamento das entregas, essa organização é mostrada na Figura 3.4.

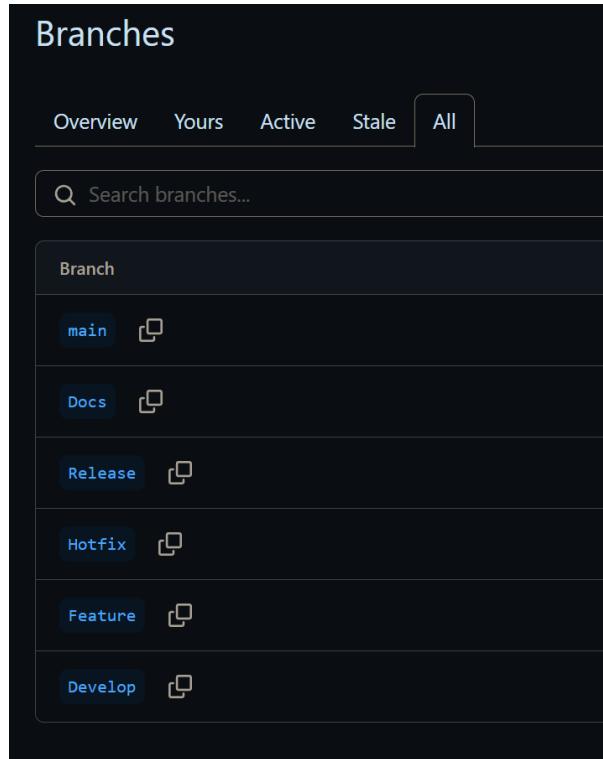


Figura 3.4 – Organização das branches - GitHub

Fonte: Autor, 2025.

3.5.2 Ferramentas de comunicação

- **Microsoft Teams:** O Microsoft Teams é uma plataforma de comunicação corporativa que integra funcionalidades de mensagens instantâneas, reuniões virtuais, armazenamento de arquivos e integração com ferramentas do pacote Microsoft 365, como Word, Excel e PowerPoint. No contexto deste projeto, o Microsoft Teams foi escolhido como principal ferramenta de comunicação entre os membros da equipe de desenvolvimento, o Product Owner (PO) e o orientador. Tal escolha se justifica pela disponibilidade gratuita da plataforma aos estudantes da Universidade de Brasília (UnB), além da centralização de mensagens, agendamento de reuniões, compartilhamento de documentos e histórico de interações, o que contribui significativamente para a organização e acompanhamento das etapas do projeto.

3.5.3 Persistência de dados

A estratégia de persistência de dados foi definida para assegurar a consistência, integridade e escalabilidade do sistema. A modelagem considerou a sensibilidade das informações registradas, como os dados alimentares de pessoas com Transtorno do Espectro Autista (TEA). Essa estruturação prévia permitiu uma implementação coerente dos mecanismos

de armazenamento e acesso, fundamentada nas melhores práticas de engenharia de dados ([Silberschatz; Korth; Sudarshan, 2011](#)).

3.5.4 PostgreSQL

O armazenamento dos dados foi implementado utilizando o PostgreSQL. Este sistema gerenciador de banco de dados relacional (SGBD) foi escolhido por sua robustez, conformidade com o padrão SQL e suporte a transações ACID, características essenciais para garantir a confiabilidade das informações ([The PostgreSQL Global Development Group, 2024](#)).

A escolha de um banco relacional se mostrou adequada para a natureza estruturada dos dados do projeto, como os perfis sensoriais e as sugestões geradas pelo algoritmo. Isso permitiu a definição de restrições de integridade rigorosas e a execução de consultas eficientes. A comunicação com o banco é mediada pela API RESTful em Node.js, que centraliza e valida todas as operações de leitura e escrita, garantindo que o acesso às informações do usuário ocorra de forma segura.

3.5.5 Ferramentas de Apoio à Escrita

Para apoio na revisão textual, estruturação de seções e aprimoramento da clareza da redação, foram utilizadas ferramenta de inteligência artificial, como o ChatGPT e o Gemini. Sua aplicação restringiu-se a sugestões de melhoria linguística e organização de texto, sem interferir no conteúdo técnico ou nos resultados do projeto.

4 Solução Implementada

Este capítulo detalha a solução implementada: um aplicativo para sugestão de trocas alimentares para pessoas com Transtorno do Espectro Autista (TEA), desenvolvido para apoiar cuidadores e pessoas com TEA independentes no manejo da seletividade alimentar.

4.1 Sobre o Aplicativo

O aplicativo foi desenvolvido para usuários com Transtorno do Espectro Autista (TEA) independentes e para cuidadores de pessoas com TEA.

A solução implementada gera sugestões de trocas alimentares personalizadas, utilizando as preferências e sensibilidades sensoriais registradas pelo usuário como dados de entrada para o algoritmo. O objetivo funcional do aplicativo é sugerir alime

4.2 *Backlog* do produto

Conforme introduzido no Capítulo 3, o Backlog do Produto é um artefato central em metodologias ágeis, funcionando como uma lista priorizada de todas as funcionalidades e requisitos do projeto. Para o desenvolvimento da solução proposta neste trabalho, foi criado um backlog específico que serviu como guia para a equipe de desenvolvimento.

Este backlog foi populado com itens na forma de Histórias de Usuário, detalhando as necessidades dos usuários finais. A priorização desses itens foi fundamental para definir o escopo da versão atual do projeto, garantindo que as funcionalidades de maior valor e impacto fossem desenvolvidas primeiro. Portanto, o sistema apresentado neste capítulo é o resultado tangível dos itens que foram selecionados, planejados e executados a partir do topo do nosso backlog.

As Histórias de Usuário foram utilizadas para descrever todas as funcionalidades do sistema sob a perspectiva de quem o utiliza. Adotando o formato padrão de mercado — "Como um [ator], eu quero [realizar uma ação] para que [possa obter um benefício]" —, conseguimos manter o foco na entrega de valor real. Cada história representava um requisito de negócio ou uma necessidade do usuário que o software deveria satisfazer. Em contrapartida, as Tarefas Técnicas foram criadas para registrar trabalhos essenciais que não entregam uma nova funcionalidade necessariamente visível ao usuário final, mas são cruciais para a saúde, qualidade e viabilidade do projeto.

Backlog do Aplicativo (Priorização MoSCoW)

O backlog foi estruturado para atender aos dois perfis de usuário finais: o Cuidador (que gerencia assistidos) e o Usuário Independente (pessoa com TEA que gerencia a própria alimentação).

- **US01 (Must Have, 3 pts):** Como usuário (cuidador ou independente), quero criar uma conta para acessar as funcionalidades do aplicativo.
- **US02 (Must Have, 2 pts):** Como usuário, quero realizar login para acessar meus dados de forma segura.
- **US03 (Must Have, 3 pts):** Como cuidador, quero cadastrar múltiplos assistidos para gerenciar as preferências alimentares de cada um individualmente.
- **US04 (Must Have, 3 pts):** Como usuário independente, quero configurar meu próprio perfil com dados pessoais e nível de suporte.
- **US05 (Must Have, 3 pts):** Como usuário, quero registrar os "Alimentos Seguros"(preferências e aceitação atual) para alimentar a base de dados do algoritmo.
- **US06 (Must Have, 5 pts):** Eu, como sistema, devo processar os "Alimentos Seguros" e identificar um grupo alimentar alvo para sugerir trocas.
- **US07 (Must Have, 5 pts):** Eu, como sistema, devo gerar uma lista de sugestões de trocas alimentares baseada na similaridade sensorial com os alimentos já aceitos.
- **US08 (Must Have, 3 pts):** Como usuário, quero visualizar a lista de sugestões de troca ordenadas pela compatibilidade sensorial.
- **US09 (Should Have, 2 pts):** Como usuário, quero registrar o feedback (aceitou/recusou) sobre uma sugestão para que o sistema aprenda e melhore as próximas recomendações.
- **US10 (Must Have, 3 pts):** Como usuário, quero visualizar um relatório com o histórico das trocas alimentares realizadas (aceitas e rejeitadas) para acompanhar a evolução da dieta.
- **US11 (Should Have, 2 pts):** Como usuário, quero alterar minha senha de acesso para garantir a segurança da conta.

Épicos do Produto

Os requisitos foram agrupados em épicos que representam os grandes módulos funcionais da solução implementada.

- **Épico 1: Gestão de Identidade e Acesso**
 - (Must Have) **US01:** Criação de conta (Cuidador e Independente).
 - (Must Have) **US02:** Autenticação (Login).
 - (Should Have) **US11:** Alteração de senha.
 - (Must Have) **TK01:** Implementar autenticação JWT e controle de sessão.

- **Épico 2: Gerenciamento de Perfis e Assistidos**
 - (Must Have) **US03**: Cadastro de múltiplos assistidos (perfil Cuidador).
 - (Must Have) **US04**: Configuração de auto-perfil (perfil Independente).
 - (Must Have) **TK02**: Modelagem do banco de dados para suportar relacionamento 1:N (Cuidador-Assistidos).
- **Épico 3: Mapeamento do Repertório Alimentar e Sensorial**
 - (Must Have) **US05**: Registro de Alimentos Seguros e preferências sensoriais.
 - (Must Have) **TK03**: Popular banco de dados com a base de alimentos e seus atributos sensoriais detalhados (textura, sabor, cor) para viabilizar o algoritmo.
- **Épico 4: Sistema de Recomendação e Análise de Evolução**
 - (Must Have) **US06**: Identificação automática do grupo-meta para intervenção.
 - (Must Have) **US07**: Algoritmo de recomendação por similaridade sensorial.
 - (Must Have) **US08**: Visualização das sugestões de troca.
 - (Must Have) **US09**: Registro de feedback (Aceitação/Recusa) e atualização automática dos Alimentos Seguros.
 - (Should Have) **US10**: Relatório de histórico das trocas realizadas.
 - (Must Have) **TK04**: Implementação da lógica de pontuação ponderada (Similaridade Sensorial vs. Melhora Nutricional) no serviço de sugestões.

4.3 Identidade Visual

A identidade visual do aplicativo utiliza uma paleta de cores em tons pastéis e a fonte Baloo 2. A escolha da fonte buscou priorizar a legibilidade por meio de formas arredondadas, enquanto as cores de baixa saturação foram selecionadas para criar uma interface de usuário clara, promovendo assim a facilidade de uso.

4.3.1 Logotipo

O logotipo é composto por duas formas orgânicas, cujo movimento simboliza o conceito da troca alimentar. O espaço negativo criado pelo encaixe das formas remete à silhueta de uma peça de quebra-cabeça, um símbolo associado ao TEA.

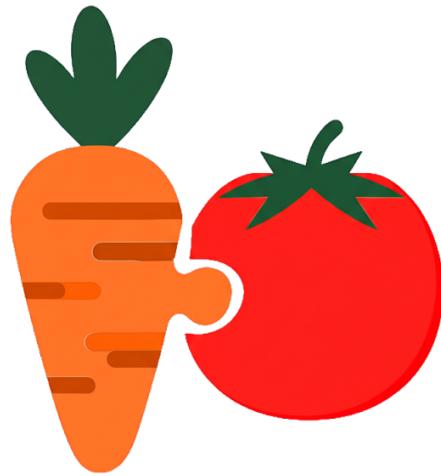


Figura 4.1 – Logo do Aplicativo

Fonte: Autor, 2025.



Figura 4.2 – Paleta de Cores

Fonte: Autor, 2025.

4.3.2 Interface do Aplicativo

A interface do aplicativo foi projetada para guiar o usuário de forma intuitiva através dos principais fluxos de uso: autenticação, gestão de assistidos e o processo de troca alimentar.

O primeiro ponto de contato é o fluxo de acesso. Caso o usuário ainda não possua conta, ele utiliza a tela de **Cadastro** (Figura A.2), onde fornece seus dados básicos. Para usuários recorrentes, o acesso é feito via tela de **Login** (Figura A.1), garantindo a segurança das informações pessoais.

Após a autenticação, o usuário é direcionado para a **Home do Cuidador** (Figura A.3), que atua como o painel central de navegação. Nesta área, é possível acessar as configurações de conta através da tela de **Perfil** (Figura A.4) e, principalmente, gerenciar os indivíduos sob seus cuidados na tela de **Cadastro de Assistido** (Figura A.5). É nesta etapa que se definem os dados críticos como nível de suporte e grau de seletividade.

O núcleo funcional do sistema — a sugestão de trocas — inicia-se com a coleta de dados. O cuidador preenche o **Questionário Alimentar** (Figura A.6), registrando o repertório atual do assistido. Com base nesses dados, o sistema processa as informações e apresenta as opções na tela de **Sugestão de Troca** (Figura A.8), exibindo alternativas sensorialmente compatíveis.

Para fechar o ciclo de aprendizado do algoritmo, o aplicativo solicita o retorno do usuário na tela de **Feedback** (Figura A.9), onde é registrado se a sugestão foi aceita ou recusada, refinando as futuras recomendações apresentadas na listagem de **Refeições Disponíveis** (Figura A.7).

As imagens detalhadas de todas as telas mencionadas encontram-se disponíveis para consulta no Apêndice A.

4.3.3 Interface do Aplicativo

A interface do aplicativo foi desenvolvida com base nos requisitos levantados e na identidade visual definida. Antes da implementação final, foi elaborado um protótipo de média fidelidade para a validação inicial dos fluxos de navegação, o qual pode ser visualizado no Apêndice B.

Além disso, o protótipo interativo de média fidelidade, que serviu de base para o desenvolvimento do *front-end*, está disponível para acesso público através do seguinte endereço: https://www.figma.com/design/ZPEHj7ke48KWDV5uJfHKMw/Prototipo_App_TCC_TEA?node-id=0-1&t=bTELW0yh2FSIr6Hp-1.

4.4 Estrutura do Código da Aplicação

A estrutura do código-fonte da aplicação reflete a arquitetura em 3 camadas (3-Tier). Este padrão de *design* divide a aplicação em três unidades lógicas separadas para garantir a separação de responsabilidades: a Camada de Apresentação (a interface do usuário), a Camada de Negócio (o servidor com a lógica) e a Camada de Dados (o banco de dados).

O projeto foi organizado em um formato *monorepo*, onde o código-fonte do cliente (front-end) e do servidor (back-end) coexistem no mesmo repositório, mas estão desacoplados em diretórios distintos (/app-tea e /back-end) para implementar fisicamente essa separação.

O código-fonte completo da solução implementada está disponível no repositório do projeto e está hospedado na plataforma GitHub, no seguinte endereço: https://github.com/GabrielMS00/Applicativo_TCC_TEA.

4.4.1 Camada de Apresentação (Front-end)

O front-end está contido no diretório /app-tea e consiste em um aplicativo móvel desenvolvido em React Native com o framework Expo. Esta camada é a interface com a qual o usuário interage. Sua estrutura interna é organizada da seguinte forma:

- **Roteamento (app/):** O sistema de navegação utiliza o expo-router, que adota uma abordagem de roteamento baseada em arquivos. Os diretórios dentro de app/ definem as rotas da aplicação, como as rotas de autenticação (auth) e as rotas principais pós-login (tabs).
- **Componentes (components/):** Contém os componentes de UI reutilizáveis, como botões, inputs e cards. A estilização desses componentes é feita com a biblioteca NativeWind.
- **Serviços de API (api/):** Abstrai a comunicação com o back-end. Um cliente apiClient.ts (baseado em Axios) é configurado, e serviços específicos (ex: auth.ts, assistidos.ts) exportam funções para consumir os endpoints.
- **Gerenciamento de Estado (context/):** Utiliza a Context API do React para gerenciamento de estado global, como o contexto de autenticação (AuthContext.tsx), que provê os dados do usuário para a aplicação.

4.4.2 Camada de Negócio e Dados (Back-end)

O back-end está contido no diretório /back-end e consiste na API RESTful desenvolvida em Node.js com o framework Express.js. Esta implementação agrupa tanto a Camada de Negócio quanto a Camada de Dados:

- **Rotas (src/api/routes/):** Define os endpoints da API. O arquivo index.js centraliza os diferentes arquivos de rota (ex: sugestaoRoutes.js, authRoutes.js), que mapeiam os verbos HTTP (GET, POST, etc.) para os controllers correspondentes.
- **Controladores (src/api/controllers/):** Responsáveis por receber as requisições (Request) e formular as respostas (Response). Eles validam os dados de entrada e orquestram a execução da lógica de negócio, acionando os services.

- **Serviços (src/services/) - Camada de Negócio:** Contém a lógica de negócio principal do sistema. Por exemplo, `sugestaoService.js` implementa o algoritmo da Regra de Negócio (comparação sensorial, pontuação), enquanto `processamentoQuestionarioService.js` lida com o processamento das respostas dos questionários.
- **Acesso a Dados (src/api/models/ e config/db.js) - Camada de Dados:** A camada de dados é implementada por `models` que executam consultas SQL diretas no banco de dados PostgreSQL, cuja conexão é gerenciada pelo arquivo `db.js`.
- **Schema do Banco (migrations/):** A estrutura (schema) do banco de dados é gerenciada de forma versionada através de arquivos de migração (ex: `...create-table-cuidadores.js`), utilizando a ferramenta `node-pg-migrate`.

4.4.3 Ambiente e Orquestração

Finalmente, a plataforma Docker é utilizada para padronizar e gerenciar o ambiente de execução. O `Dockerfile` define a imagem de contêiner para o back-end Node.js. O arquivo `docker-compose.yml`, na raiz do projeto, orquestra a inicialização conjunta da Camada de Negócio (serviço `api`) e da Camada de Dados (serviço `postgres`), garantindo um ambiente de desenvolvimento reproduzível e isolado.

4.5 Organização dos Dados

A funcionalidade do aplicativo é suportada por um modelo de dados relacional. Para isso, a arquitetura do banco de dados foi projetada para gerenciar as interações entre usuários, assistidos, avaliações e a lógica de recomendação. O modelo está estruturado nos domínios principais a seguir.

4.5.1 Estrutura de Usuários e assistidos

Neste domínio, é feita a distinção entre o usuário do sistema e o assistido que recebe a intervenção.

- **Entidade User:** Armazena as credenciais de acesso e dados de identificação dos usuários. Os perfis são categorizados em `cuidador` (que gerencia assistidos) ou `assistido` (usuário independente que gerencia o próprio perfil).
- **Entidade Patient:** Representa o sujeito central da intervenção, contendo suas informações demográficas e clínicas. A entidade estabelece o relacionamento de dependência (1:N) com a entidade `User`, onde um cuidador pode ser responsável por um ou mais assistidos, através de uma chave estrangeira (`id_user_caregiver`).

4.5.2 Armazenamento de Dados de Avaliação (Questionários)

Para documentar o estado inicial e a evolução do assistido, o modelo permite o registro de avaliações contínuas.

- **Entidade Questionnaire:** Funciona como um registro mestre para cada instância de um questionário aplicado, armazenando metadados como o tipo de instrumento e a data da aplicação.
- **Entidade Questionnaire_Response:** Projetada com uma estrutura flexível de par chave-valor (pergunta, resposta), esta entidade armazena cada resposta individual, permitindo que o sistema acomode diversos instrumentos sem alterações na estrutura do banco.

4.5.3 Base de Conhecimento Alimentar e Perfil Sensorial

Este domínio estrutura a informação sobre os alimentos, sendo o pilar para o algoritmo de recomendação.

- **Entidade Food:** Constitui o catálogo central de alimentos, contendo suas propriedades intrínsecas, como grupo alimentar e classificação nutricional (e.g., caseiro, processado).
- **Entidade Food_Sensory_Profile:** Vinculada à entidade Food, armazena os atributos extrínsecos de cada alimento (textura, cor, etc.). A separação entre Food e Food_Sensory_Profile é fundamental para a execução do algoritmo de similaridade.
- **Entidade Safe_Food:** Representa o subconjunto de alimentos validados como seguros para um assistido. Esta entidade funciona como o principal insumo para o motor de sugestões, ao definir os "Alimentos Ponte".

4.5.4 Modelo de Sugestão, Opções e Feedback

Este domínio gerencia o ciclo de vida de uma recomendação, desde sua geração até a resposta do usuário.

- **Entidade Exchange_Suggestion:** É uma entidade transacional que registra cada execução do algoritmo, armazenando o contexto da sugestão (assistido, data, grupo-meta e o Safe_Food de referência).
- **Entidade Exchange_Option:** Filha de Exchange_Suggestion, armazena cada uma das opções de troca geradas, vinculando um alimento sugerido a uma refeição e registrando as pontuações calculadas.
- **Entidade Feedback:** Esta entidade fecha o ciclo de aprendizado do sistema, pois armazena a interação do usuário final (cuidador ou assistido) com uma Exchange_Option, registrando o status de ACEITA ou REJEITADA.

4.5.5 Modelo Relacional e DER

Abaixo segue o modelo relacional implementado, que descreve textualmente a estrutura do banco de dados.

- **cuidadores**

- id (PK)
- tipo_usuario
- nome
- email (Unique)
- senha_hash
- cpf (Unique)
- data_nascimento
- data_cadastro
- palavra_seguranca

- **assistidos**

- id (PK)
- nome
- data_nascimento
- nivel_suporte
- grau_seletividade
- cuidador_id (FK → cuidadores.id)

- **alimentos**

- id (PK)
- nome (Unique)
- grupo_alimentar

- **perfis_sensoriais**

- id (PK)
- forma_de_preparo
- textura
- sabor
- cor_predominante
- temperatura_servico
- alimento_id (FK → alimentos.id)
- (Unique: alimento_id, forma_de_preparo)

- **refeicoes**

- id (PK)
- nome (Unique)

- **perfil_refeicao (N:M)**
 - id (PK)
 - perfil_sensorial_id (FK → perfis_sensoriais.id)
 - refeicao_id (FK → refeicoes.id)
 - (Unique: perfil_sensorial_id, refeicao_id)
- **alimentos_seguros (N:M)**
 - id (PK)
 - data_adicao
 - assistido_id (FK → assistidos.id)
 - alimento_id (FK → alimentos.id)
 - (Unique: assistido_id, alimento_id)
- **trocas_alimentares**
 - id (PK)
 - refeicao
 - data_sugestao
 - assistido_id (FK → assistidos.id)
- **detalhes_troca**
 - id (PK)
 - troca_alimentar_id (FK → trocas_alimentares.id)
 - alimento_novo_id (FK → alimentos.id)
 - status
 - perfil_sensorial_id (FK → perfis_sensoriais.id)
 - motivo_sugestao
- **modelos_questionarios**
 - id (PK)
 - nome (Unique)
- **modelos_perguntas**
 - id (PK)
 - texto_pergunta (Unique)
 - modelo_questionario_id (FK → modelos_questionarios.id)
- **modelos_opcoes_respostas**
 - id (PK)
 - texto_opcao
 - modelo_pergunta_id (FK → modelos_perguntas.id)
- **questionarios_respondidos**
 - id (PK)

- data_resposta
- assistido_id (FK → assistidos.id)
- cuidador_id (FK → cuidadores.id)
- modelo_questionario_id (FK → modelos_questionarios.id)

- **respostas**

- id (PK)
- questionario_respondido_id (FK → questionarios_respondidos.id)
- modelo_pergunta_id (FK → modelos_perguntas.id)
- modelo_opcao_resposta_id (FK → modelos_opcoes_respostas.id)

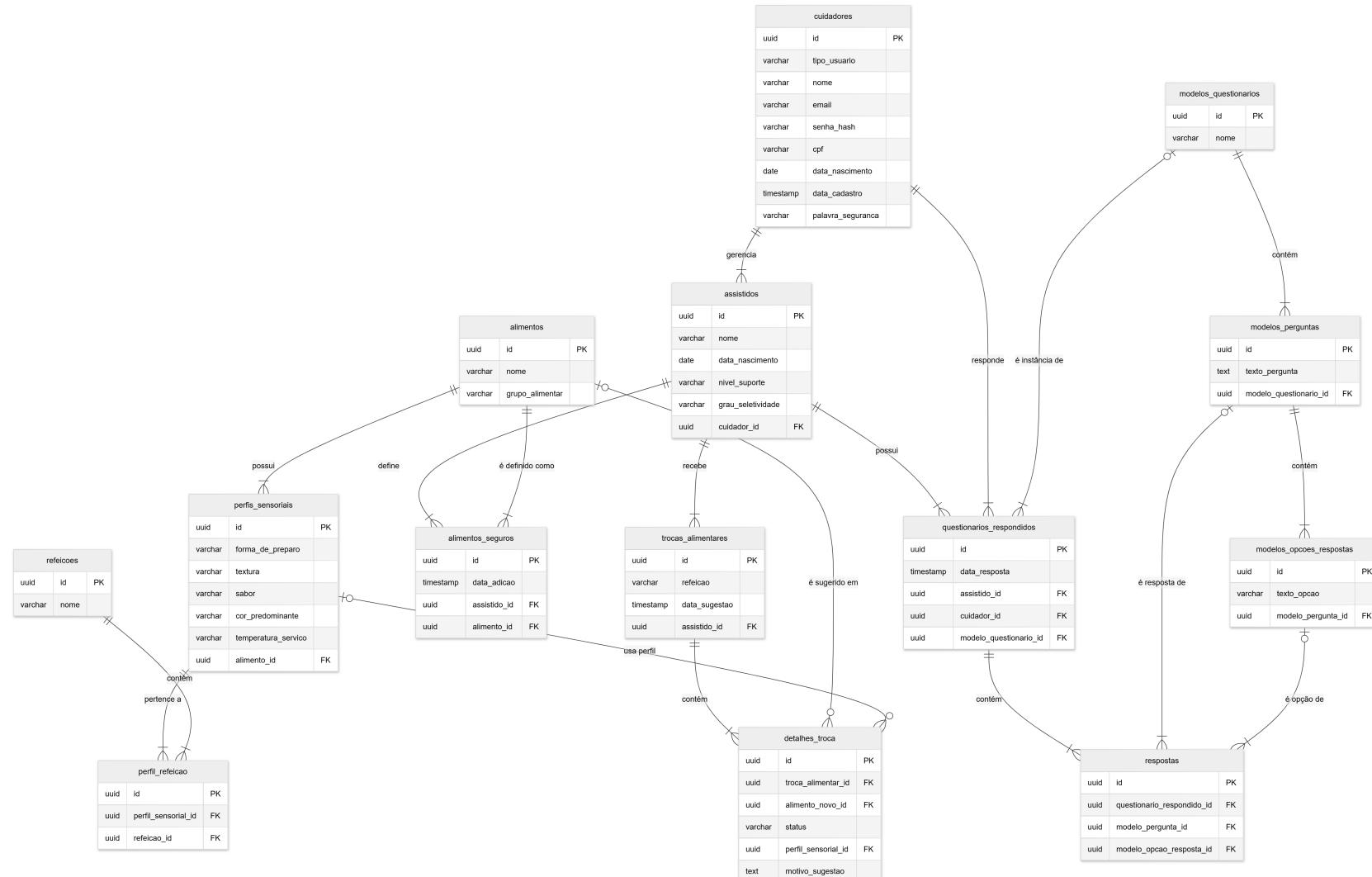


Figura 4.3 – Diagrama Entidade-Relacionamento Banco de Dados

Fonte: Autor, 2025.

Esta modelagem de dados, representada textualmente pelo Modelo Relacional e visualmente pelo Diagrama Entidade-Relacionamento (Figura 4.3), provê a estrutura necessária para que a aplicação gerencie com consistência os dados de seus usuários e execute os algoritmos de recomendação.

4.6 Regra de negócio

O núcleo funcional do sistema é a **Regra de Negócio para Sugestão de Trocas Alimentares**, projetada para operacionalizar a estratégia terapêutica de Encadeamento Alimentar (*Food Chaining*). O objetivo do sistema é apoiar seus usuários — pessoas com TEA independentes ou cuidadores — a promoverem a expansão do repertório alimentar, o que é feito por meio de trocas que sejam sensorialmente compatíveis com alimentos já aceitos.

4.6.1 Fundamentos e Modelo de Dados de Suporte

A operacionalização desta regra é dependente do modelo de dados estruturado. Neste modelo, cada **Alimento Genérico** é classificado em um **Grupo Alimentar**, mas a lógica do sistema opera sobre **Preparações Culinárias Específicas** (e.g., “Nugget de frango industrializado”).

Cada preparação possui **Atributos Sensoriais** (textura, formato, etc.) e uma **Classificação Nutricional** (e.g., processado, caseiro). Esta modelagem resulta em um perfil detalhado para cada item, que serve de base para o processamento do algoritmo.

4.6.2 O Fluxo Operacional do Algoritmo de Sugestão

O processo para geração de uma sugestão de troca alimentar é implementado por um algoritmo que executa as seguintes etapas sequenciais:

1. **Identificação do Ponto de Partida:** O sistema analisa os dados do QFA para selecionar um “Alimento Seguro” do perfil do usuário, recuperando seu perfil sensorial e nutricional.
2. **Definição da Meta Nutricional:** O algoritmo identifica os grupos alimentares com consumo ausente ou deficiente (conforme o QFA) e seleciona um deles como o “grupo-metá” para a intervenção.
3. **Busca e Pontuação de Candidatos:** O algoritmo executa uma busca restrita às preparações pertencentes ao “grupo-metá”. Para cada candidato, é calculada uma “Pontuação de Recomendação”, composta por dois critérios: a **Similaridade Sensorial** com o “Alimento Seguro” e a **Melhora Nutricional**.

-
4. **Geração da Lista Ordenada de Opções:** O sistema compila uma lista com os 3 a 4 candidatos que obtiveram as maiores Pontuações de Recomendação e a apresenta de forma ordenada.

4.6.3 Dinamismo e Adaptação: O Ciclo de Feedback

O sistema é projetado para evoluir com o uso, através de um ciclo de feedback contínuo:

Cenário de Aceitação Ao registrar que uma sugestão foi aceita, o sistema promove esta preparação ao status de um novo “Alimento Seguro”. Este processo enriquece o perfil do usuário e amplia a base sensorial para futuras recomendações.

Cenário de Rejeição Caso a primeira sugestão da lista seja rejeitada, o sistema a marca para não ser oferecida novamente em curto prazo e apresenta a próxima opção da lista, mantendo o processo de interação fluido.

4.7 Testes

Com a conclusão do aplicativo, foram realizadas verificações para garantir o funcionamento correto de recursos específicos e da aplicação como um todo. O processo incluiu testes unitários, funcionais e de integração, descritos nas próximas seções.

4.7.1 Testes Unitários

Conforme definido na metodologia, utilizou-se o *Jest* para a realização dos testes unitários. No *front-end*, foram executados 56 testes, enquanto no *back-end* foram realizados 25. As Figuras 4.4 e 4.5 apresentam a cobertura de testes obtida no projeto. Os resultados foram satisfatórios, alcançando aproximadamente 70% de cobertura geral.

4.7.2 Testes de Integração

Além dos testes implementados no código, foram realizados 13 testes de integração utilizando a plataforma *Postman*. Esses testes consistiram no envio de requisições às rotas da API, com o objetivo de verificar o correto funcionamento dos serviços e suas respostas. Todos os 13 testes foram bem sucedidos.

4.7.3 Teste Funcional

Também foi conduzida uma rodada de testes funcionais, na qual a *Product Owner* avaliou o aplicativo em uso real. Nessa etapa, foram verificadas as principais funcionalidades do sistema e validado o atendimento ao escopo estabelecido para a primeira *release* do projeto.

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Line #s
All files	69.59	50.64	67.14	70.38	
src	88.88	100	0	88.88	
app.js	88.88	100	0	88.88	16
src/api/controllers	61.15	54.86	68.18	61.41	
assistidoController.js	65.45	71.42	80	65.45	5,28-29,40-41,47-61,84-85,92,109-110
authController.js	59.09	52.72	80	59.52	29,40,52-54,91-98,108,130,148-149,154-187
cuidadorController.js	93.33	87.5	100	93.33	27
questionarioController.js	50	62.5	40	48.48	8-13,19-30,41,63,67-68
relatorioController.js	91.3	62.5	100	91.3	87-88
sugestaoController.js	42.22	38	33.33	43.18	28,33,40-41,46-86
src/api/middlewares	73.33	62.5	100	73.33	
authMiddleware.js	73.33	62.5	100	73.33	19,24-25,30
src/api/models	62.8	34.61	60.86	63.24	
AlimentoSeguro.js	17.39	0	0	18.18	9-42
Assistido.js	100	100	100	100	
Cuidador.js	85.71	50	83.33	85.18	56-60
Questionario.js	16.66	0	0	17.64	7-44
Resposta.js	75.86	66.66	100	75.86	16,21-22,33,56-58
src/api/routes	100	100	100	100	
assistidoRoutes.js	100	100	100	100	
authRoutes.js	100	100	100	100	
cuidadorRoutes.js	100	100	100	100	
index.js	100	100	100	100	
questionarioRoutes.js	100	100	100	100	
relatorioRoutes.js	100	100	100	100	
sugestaoRoutes.js	100	100	100	100	
src/config	100	100	100	100	
db.js	100	100	100	100	
src/services	63.63	40.57	66.66	65.18	
processamentoQuestionarioService.js	65.85	40	100	67.56	23-25,39-41,54,74-80
sugestaoService.js	62.83	40.67	60	64.28	51,111-130,138-139,241-288
tests/utils	98.24	33.33	100	98.24	
dbHelper.js	98.24	33.33	100	98.24	79

Test Suites: 8 passed, 8 total
 Tests: 25 passed, 25 total
 Snapshots: 0 total
 Time: 3.701 s, estimated 5 s

Figura 4.4 – Cobertura de testes do Back-End

Fonte: Autor, 2025.

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Line #s
All files	71.97	55.61	70.58	72.42	
api	73.91	48.71	80.95	75.28	
apiClient.ts	59.25	42.85	50	60.78	17-26,44-48,67,78-92
assistidos.ts	83.33	100	66.66	83.33	47,65
auth.ts	100	100	100	100	
cuidador.ts	100	100	100	100	
questionario.ts	100	100	100	100	
relatorio.ts	100	100	100	100	
sugestoes.ts	100	100	100	100	
app/(auth)	82.02	62.5	80.64	82.75	
CreateAccount.tsx	85.18	56.6	90.47	86.53	95-98,129-137,152-153
ForgotPassword.tsx	71.42	76.92	50	71.42	17-18,21-22,35,64
index.tsx	85.71	83.33	66.66	85.71	38,43
app/(tabs)	58.33	53.57	42.85	58.33	
Register.tsx	58.33	53.57	42.85	58.33	30-31,44-45,48-49,71-76,82,108-138
app/(tabs)/Account	90.9	50	66.66	90.9	
ChangePassword.tsx	90.9	50	66.66	90.9	16
app/FoodExchange	75	50	80	75	
MealOption.tsx	75	50	80	75	20-22,32
app/Reports	50	45.83	33.33	51.35	
ViewReport.tsx	50	45.83	33.33	51.35	29-52,65-68,79-136
components	93.93	92.3	88.88	93.75	
Button.tsx	100	100	100	100	
FoodCard.tsx	100	100	100	100	
FoodCardCheckable.tsx	100	50	100	100	24
Input.tsx	100	100	100	100	
MainButton.tsx	100	100	100	100	
MealTypeCard.tsx	100	100	100	100	
RadioButton.tsx	100	100	100	100	
SelectInput.tsx	86.66	100	77.77	85.71	51-53
WatchedCard.tsx	100	100	100	100	
utils	41.66	28.57	25	36.36	
formatters.ts	100	100	100	100	
generateReportHtml.ts	6.66	0	0	6.66	6-50

Test Suites: 24 passed, 24 total
 Tests: 56 passed, 56 total
 Snapshots: 0 total

Figura 4.5 – Cobertura de testes do Front-End

Fonte: Autor, 2025.

4.8 Resultados das Métricas (GQM)

Este tópico apresenta os resultados da aferição do processo de desenvolvimento, com base no GQM (Goal-Question-Metric) definido no Capítulo 3 (Tabela 3.1). A tabela a seguir consolida os resultados obtidos na execução do projeto (TCC 2).

Tabela 4.1 – Resultados das Métricas GQM

Métrica	Cálculo Realizado	Valor Obtido	Valor Esperado	Conclusão
Objetivo 1: Gerenciamento do Backlog (Questões 1.1 e 1.2)				
M 1.1.1 e 1.2.1: Tarefas concluídas do backlog	Contagem de tarefas (US + TK) finalizadas	15	15	O backlog planejado foi integralmente cumprido, atendendo às necessidades mapeadas.
Objetivo 2: Gerenciamento do Cronograma (Questão 2.1)				
M 2.1.1: Funcionalidades mínimas do MVP	(Func. Entregues / Func. MVP Planejadas)	100%	100%	O MVP foi entregue dentro do cronograma estipulado.
Objetivo 3: Progresso do Desenvolvimento (Questões 3.1, 3.2 e 3.3)				
M 3.1: Taxa de bugs corrigidos	(Bugs Corrigidos / Total Bugs Encontrados)	94%	> 90%	A qualidade foi mantida com a correção ágil das falhas.
M 3.2.1: Total de releases entregues	Contagem de releases/versões	2	>= 2	Houve entregas incrementais e contínuas do software.
M 3.3.1: Débitos técnicos registrados	Contagem de débitos (To Do/Refactor)	3	< 5	O endividamento técnico foi controlado e mantido abaixo do esperado.

Fonte: Autor, 2025.

4.9 Cronograma de Desenvolvimento

O desenvolvimento da solução seguiu um cronograma estruturado em entregas incrementais, organizado para priorizar as funcionalidades essenciais do sistema. As atividades foram distribuídas ao longo do segundo semestre de 2025, conforme detalhado a seguir.

A fase inicial, na primeira quinzena de setembro, concentrou-se na implementação dos **Épicos 1 e 2**, estabelecendo as bases de gestão de identidade e o cadastro de perfis e assistidos. Em sequência, a segunda quinzena de setembro e a primeira de outubro foram dedicadas ao **Épico 3**, focado no mapeamento do repertório alimentar e sensorial, etapa fundamental para a entrada de dados no sistema.

Durante todo o mês de outubro, o desenvolvimento voltou-se para o **Épico 4**, o núcleo da solução, onde foram implementados o algoritmo de recomendação e os relatórios de evolução. O mês de novembro foi reservado para a garantia da qualidade: a primeira quinzena focou na validação da solução com os requisitos propostos, enquanto o restante do mês foi dedicado à execução de testes, refatoração de código e correções de erros. Por fim,

a primeira quinzena de dezembro destinou-se à finalização da monografia e à preparação para a defesa.

Tabela 4.2 – Cronograma Executado de Desenvolvimento e Entregas (TCC 2).

Épico / Fase	Setembro		Outubro		Novembro		Dez
	1ª Q	2ª Q	1ª Q	2ª Q	1ª Q	2ª Q	1ª Q
1. Gestão de Identidade e Acesso							
2. Gerenciamento de Perfis e Assistidos							
3. Mapeamento do Repertório Alimentar							
4. Sistema de Recomendação e Análise							
5. Validação da Solução							
6. Testes e Refatoração							
7. Finalização TCC 2 e Apresentação							

Fonte: Autor, 2025.

5 Conclusões

5.1 Introdução

Após o desenvolvimento, o aplicativo se propõe a ser uma solução para a limitação da diversidade alimentar em indivíduos com Transtorno do Espectro Autista e seletividade.

Neste capítulo, serão abordados o cumprimento dos objetivos específicos, as fragilidades mapeadas no projeto e as propostas sugeridas para a evolução do software.

5.1.1 Objetivos Específicos

Os objetivos específicos, que foram definidos previamente ao processo de desenvolvimento, foram cumpridos ao longo do processo de planejamento e de construção do software proposto.

Identificamos os conceitos necessários ao desenvolvimento do aplicativo de software através de reuniões com a PO, que nos explicou os aspectos nessesários para que pudéssemos implementar a lógica das trocas alimentares dentro do contexto estabelecido.

Estabelecemos uma ferramenta de software para apoiar os cuidadores e os próprios indivíduos que possuem TEA e seletividade alimentar, através da criação de um aplicativo que atendesse aos requisitos que foram elicitados mediante entrevistas com a PO.

Por fim, o software será disponibilizado para que a PO possa estabelecer um estudo de caso para avaliar adequação do produto de software aos fins no qual ele se dedica.

5.1.2 Processo de desenvolvimento

O processo definido foi, em geral, cumprido, ainda que com ajustes pontuais que se mostraram necessários durante o desenvolvimento. Um exemplo foi a adequação das branches de gestão de configuração, visto que esta primeira versão foi restrita a dois desenvolvedores e não houve implantação (deploy) dos lançamentos.

O método GQM possibilitou uma avaliação objetiva do andamento do projeto. As métricas utilizadas foram efetivas para validar as decisões adotadas e assegurar o cumprimento dos objetivos. Os dados coletados no processo formaram uma base concreta para o aprimoramento contínuo e a otimização da ferramenta.

Os testes unitários do back-end indicaram um nível satisfatório de cobertura e confiabilidade do código. Sua execução foi fundamental para atestar o funcionamento esperado das funcionalidades críticas e para identificar falhas antecipadamente, contribuindo para

a robustez do sistema. Desta forma, o desenvolvimento manteve um padrão de qualidade consistente e alinhado às práticas de engenharia de software.

Por fim, a aplicação dos testes de usabilidade com a PO confirmou que as necessidades dos usuários finais, levantadas no início do processo, foram atendidas. Além disso, os testes forneceram percepções valiosas sobre usabilidade e navegação, que indicam caminhos para futuras melhorias no projeto.

5.2 Fragilidades e Propostas de Evolução

Apesar do resultado satisfatório, foram mapeados aspectos que se enquadram como fragilidades. Estes devem ser explorados com o propósito de reconhecer as limitações da proposta implementada e mapear futuras soluções para tornar o sistema mais robusto.

As fragilidades identificadas foram classificadas em duas categorias, as internas e as externas. As internas dizem respeito a potenciais vulnerabilidades do código, tais como uma API não otimizada, limitações do banco de dados ou falhas de segurança. As externas abordam limitações fora do escopo do código, que, no caso deste TCC, estão associadas à pesquisa e à utilização do software.

5.2.1 Fragilidades Internas

1. **Porcentagem da cobertura de testes:** Apesar de uma boa cobertura de testes, o sistema não possui 100% de cobertura, o que significa que algum erro ainda pode surgir durante o uso do app.
2. **Recuperação de Credenciais Simplificada:** O fluxo de recuperação de conta (para "esqueci minha senha") foi implementado de forma simplificada. Ao invés de um sistema de redefinição baseado em token enviado por e-mail, o projeto utilizará um mecanismo de "palavra de segurança" definida pelo usuário no cadastro. Esta implementação é temporária. A abordagem foi adotada por não haver, no momento, um serviço de envio de e-mails configurado e uma infraestrutura de hospedagem definitiva. Embora funcional, este método é considerado menos robusto que os padrões de mercado e representa uma fragilidade na segurança da autenticação.
3. **Hospedagem do software:** Não há nenhum serviço de hospedagem sendo utilizado para o software. Além disso, não há um instalador do aplicativo e isso limita o uso do software pois ele só pode ser utilizado localmente.
4. **Gerenciamento de Estado no Aplicativo - Fontend:** O aplicativo pode apresentar fragilidades na gestão de estado, especialmente em cenários de uso offline. Se o usuário registrar uma refeição ou aceitação sem conexão, o sistema pode não ter meca-

nismos robustos para sincronizar esses dados de forma segura quando a conexão for reestabelecida, gerando potencial perda de dados ou inconsistências.

5.2.2 Fragilidades Externas

1. **Limitação da lista de alimentos:** O software opera com uma lista pré-definida de alimentos e seus respectivos aspectos sensoriais. No entanto, essa lista é finita. Tal característica representa uma limitação, pois um usuário que recuse exaustivamente as sugestões eventualmente esgotará as opções disponíveis. Nesse cenário, o aplicativo não será mais capaz de gerar novas propostas.
2. **Viés de Registro do Usuário:** A eficácia do sistema depende inteiramente da precisão e da consistência com que o usuário insere os dados de aceitação alimentar. A subjetividade na avaliação do usuário, como a interpretação do que seria aceitar bem um alimento, ou o esquecimento de registrar refeições podem gerar um perfil sensorial impreciso para o indivíduo, ocasionando sugestões de troca inadequadas.
3. **Fatores Comportamentais Não Mapeados:** O software foca primariamente nos aspectos sensoriais dos alimentos. Contudo, a recusa alimentar no TEA é multifatorial e pode envolver o contexto da refeição, fatores emocionais, ou a forma de apresentação do prato. O aplicativo, em sua versão atual, não captura essas variáveis, o que limita a eficácia de suas sugestões.

5.2.3 Proposta de Evolução

Uma série de sugestões foi mapeada com o propósito de documentação para um futuro trabalho de evolução do aplicativo. A lista com as propostas para futuras implementações é apresentada a seguir.

1. **Criação de um novo tipo de conta:** Atualmente, o software possui dois tipos de conta: cuidador e indivíduo com TEA e seletividade alimentar. No entanto, identificou-se a oportunidade para a criação de um terceiro tipo, destinado ao profissional da área da saúde. Este perfil teria acesso aos dados dos indivíduos com TEA, permitindo a realização de acompanhamentos recorrentes.
2. **Utilização de IA:** O uso de Inteligência Artificial pode aprimorar as sugestões de troca alimentar. A tecnologia permitiria uma análise mais refinada dos padrões sensoriais dos alimentos já consumidos por indivíduos com TEA, com o objetivo de gerar propostas mais assertivas.
3. **Módulo de Relatórios de Progresso:** Para materializar os avanços e facilitar a comunicação com profissionais, sugere-se a implementação de um módulo de relatórios. O sistema poderia gerar gráficos simples sobre a evolução da aceitação alimentar do

indivíduo, como o número de novos alimentos aceitos por período ou a frequência de consumo.

4. **Orientações de Preparo e Receitas:** A aceitação de um novo alimento muitas vezes depende da forma como ele é preparado. O aplicativo poderia ser expandido para incluir um módulo de receitas e orientações de preparo focadas em aspectos sensoriais.
5. **Hospedagem e Publicação nas Lojas:** A aplicação encontra-se em ambiente de desenvolvimento local. Para que a solução cumpra seu papel social e alcance efetivamente o público-alvo, é fundamental realizar a publicação do aplicativo nas lojas oficiais (Google Play Store e Apple App Store). Além disso, propõe-se a migração da infraestrutura do sistema para um serviço de hospedagem em nuvem.
6. **Registro do Software:** Visando a segurança jurídica e a proteção da propriedade intelectual do trabalho desenvolvido, propõe-se a realização do registro do programa de computador junto ao Instituto Nacional da Propriedade Industrial (INPI). Essa etapa é essencial para assegurar os direitos autorais sobre o código-fonte e oficializar a autoria da solução tecnológica antes de sua ampla distribuição.

5.2.4 Conclusões finais

O presente trabalho alcançou seu objetivo principal ao desenvolver e validar uma ferramenta digital voltada para o auxílio de cuidadores e pessoas com TEA no manejo da seletividade alimentar no contexto do Transtorno do Espectro Autista. A solução proposta alcançou os objetivos ao oferecer um meio gratuito e sistematizado para o registro e a sugestão de alimentos com base em perfis sensoriais. Os testes e validações realizados demonstraram que a aplicação atende aos requisitos funcionais estabelecidos, oferecendo uma interface coerente com as necessidades do público-alvo e facilitando a tomada de decisão no ambiente domiciliar.

Além da implementação técnica, este estudo permitiu a identificação de desafios e oportunidades no uso de tecnologia assistiva para o contexto alimentar. As limitações mapeadas durante o processo, como a necessidade de expansão da base de dados e a dependência de registros manuais, servem agora como um roteiro claro para a continuidade da pesquisa. A arquitetura desenvolvida fornece a fundação necessária para as futuras evoluções sugeridas, como a integração com profissionais de saúde e o uso de inteligência artificial, consolidando o potencial do software como um recurso de apoio efetivo na rotina das famílias.

O processo de desenvolvimento do aplicativo proporcionou aprendizados importantes, tanto no aspecto técnico quanto no metodológico. A aplicação prática de conceitos de engenharia de software, aliada à gestão ágil e ao uso de métricas (GQM), mostrou-se fundamental para contornar as limitações de recursos e manter o foco na entrega de valor. Além disso, a interação com os desafios reais da implementação, desde a estruturação da arquitetura até a validação da interface, reforçou a importância de uma abordagem centrada no usuário. Essa

experiência consolidou o entendimento de que a eficácia de uma solução tecnológica para a saúde depende não apenas da robustez do código, mas de sua capacidade de se integrar à rotina de quem a utiliza.

Além disso, a elaboração deste trabalho acadêmico fomentou uma compreensão sobre a metodologia de pesquisa científica. A estruturação rigorosa do estudo, desde a definição do problema até a análise dos resultados, proporcionou uma visão crítica indispensável para a formação profissional. Simultaneamente, o aprofundamento nos fundamentos do Transtorno do Espectro Autista foi essencial. O estudo detalhado sobre as nuances da seletividade alimentar e o impacto sensorial na rotina dos indivíduos permitiu que a solução tecnológica fosse desenhada com a necessária empatia e embasamento teórico, garantindo que o software atendesse às especificidades do seu público-alvo.

Referências

- 650 Industries, Inc. *Expo Documentation*. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <<https://docs.expo.dev/>>. Citado na p. 24.
- American Psychiatric Association. **Diagnostic and statistical manual of mental disorders (DSM-5)**. 5. ed. [S.L.]: American Psychiatric Publishing, 2013. Citado nas pp. 19 e 20.
- BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. In: MARCINIAK, J. J. (Ed.). **Encyclopedia of Software Engineering**. [S.l.]: John Wiley & Sons, 1994. v. 1, p. 528–532. Citado na p. 39.
- BECK, K.; BEEDLE, M.; BENNEKUM, A. van; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. **Manifesto for Agile Software Development**. 2001. Acesso em: 22 nov. 2025. Disponível em: <https://agilemanifesto.org/>. Citado na p. 22.
- BICER, A.; ALSAFFAR, A. A. The prevalence of feeding problems and food selectivity in children with autism spectrum disorder in kuwait. **Kuwait Medical Journal**, v. 52, n. 3, 2020. Citado na p. 19.
- Docker Inc. **Docker Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://docs.docker.com/>. Citado na p. 27.
- DOMINGUES DANIEL E MOREIRA, F. **A Adoção da Biblioteca de Testes do React para Testes de Componentes**. 2020. Artigo em blog técnico ou conferência. Entrada representativa baseada na citação. Detalhes específicos da publicação não foram encontrados. Citado na p. 43.
- DRIESSEN, V. **A successful Git branching model**. 2010. Disponível em: <https://nvie.com/posts/a-successful-git-branching-model/>. Acessado em 08 de Julho de 2025. Citado na p. 37.
- GitHub Inc. **GitHub Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://docs.github.com/>. Citado na p. 26.
- GULLER, M.; YAYLACI, F. The relationship between feeding problems, parental stress, and quality of life in children with autism spectrum disorder. **Journal of Pediatric Nursing**, v. 74, 2024. Citado nas pp. 14 e 15.
- KUEDERLE, O. **React Native - A great choice for mobile development**. 2018. Medium. Acessado em 08 de Julho de 2025. Citado na p. 35.

- LAWLOR, M. **NativeWind: Tailwind CSS for React Native**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://www.nativewind.dev/>. Citado na p. 25.
- LEADER, G.; TUOHY, E.; CHEN, J. L.; MANNION, A.; GILROY, S. P. Gastrointestinal symptoms in a large sample of children with autism spectrum disorder. **Journal of Autism and Developmental Disorders**, v. 51, 2020. O documento cita 2020, embora algumas versões da publicação sejam de 2021. Citado na p. 14.
- Meta Platforms Inc. **React Native Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://reactnative.dev/docs/getting-started>. Citado na p. 23.
- Microsoft Corporation. **Microsoft Teams Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://learn.microsoft.com/en-us/microsoftteams/>. Citado na p. 27.
- Microsoft Corporation. **Visual Studio Code Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://code.visualstudio.com/docs>. Citado na p. 26.
- OpenJS Foundation. **Node.js Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://nodejs.org/en/docs/>. Citado nas pp. 24 e 35.
- PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. Porto Alegre: McGraw-Hill, 2016. Citado na p. 35.
- PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 9. ed. [S.l.]: McGraw-Hill, 2021. Citado nas pp. 21, 33 e 39.
- SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game**. [S.l.], 2020. Acesso em: 22 nov. 2025. Disponível em: <https://scrumguides.org/scrum-guide.html>. Citado na p. 22.
- SHARP, W. G.; POST, B. C.; BECRAFT, J. L.; IWINSKI, V. A.; STIGLER, K. A.; KOTERBA, E. A.; MULVIHILL, C. E. A systematic review and meta-analysis of the prevalence of feeding problems in children with autism spectrum disorder. **Journal of Autism and Developmental Disorders**, v. 48, n. 11, 2018. Citado na p. 14.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistemas de Banco de Dados**. 6. ed. Rio de Janeiro: Editora Campus/Elsevier, 2011. Citado na p. 46.
- SOMMERVILLE, I. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addison-Wesley, 2007. Citado na p. 36.
- SOMMERVILLE, I.; SAWYER, P. **Requirements Engineering: A Good Practice Guide**. [S.l.]: John Wiley & Sons, 1997. Citado na p. 32.
- SPINELLIS, D. The rise of visual studio code. **IEEE Software**, v. 38, n. 3, 2021. Citado na p. 36.

TAYLOR, B. A.; DEQUINZIO, J. A.; HAO, Y. A review of behavioral interventions for food selectivity in children with autism spectrum disorder. **Journal of Applied Behavior Analysis**, v. 50, n. 3, 2017. Citado na p. 14.

The PostgreSQL Global Development Group. **PostgreSQL 16.2 Documentation**. [S.l.], 2024. Acesso em: 22 nov. 2025. Disponível em: <https://www.postgresql.org/docs/>. Citado na p. 25.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL: The World's Most Advanced Open Source Relational Database**. [S.l.], 2024. Disponível em: <https://www.postgresql.org>. Acessado em 08 de Julho de 2025. Citado na p. 46.

UESSUGUE, P.; GOMES, M. M. F.; BARRETO, A. d. S.; VIEIRA, C. M.; SOUZA, G. M. d. AVANÇOS CIENTÍFICOS E TECNOLÓGICOS NO TRANSTORNO DO ESPECTRO AUTISTA E SELETIVIDADE ALIMENTAR: O QUE A LITERATURA MOSTRA. **Revista Políticas Públicas & Cidades**, 2025. ISSN 2359-1552. Citado na p. 21.

ULLOA, J.; TERESHKOVA, A. Differential reinforcement to increase food acceptance in children with asd: A systematic review. **Behavior Analysis: Research and Practice**, v. 20, n. 1, 2020. Citado nas pp. 14 e 15.

ULLOA, J.; TERESHKOVA, A.; ARANKI, J. Sensory-based interventions for feeding difficulties in autism: A systematic review. **Review Journal of Autism and Developmental Disorders**, v. 9, 2022. Citado na p. 19.

VAZQUEZ, M.; FITT, C.; RICH, E. Nutritional interventions for individuals with autism spectrum disorder: A systematic review of the literature. **Research in Autism Spectrum Disorders**, v. 67, 2019. Citado na p. 15.

Apêndices

Apêndice A – Telas do Aplicativo

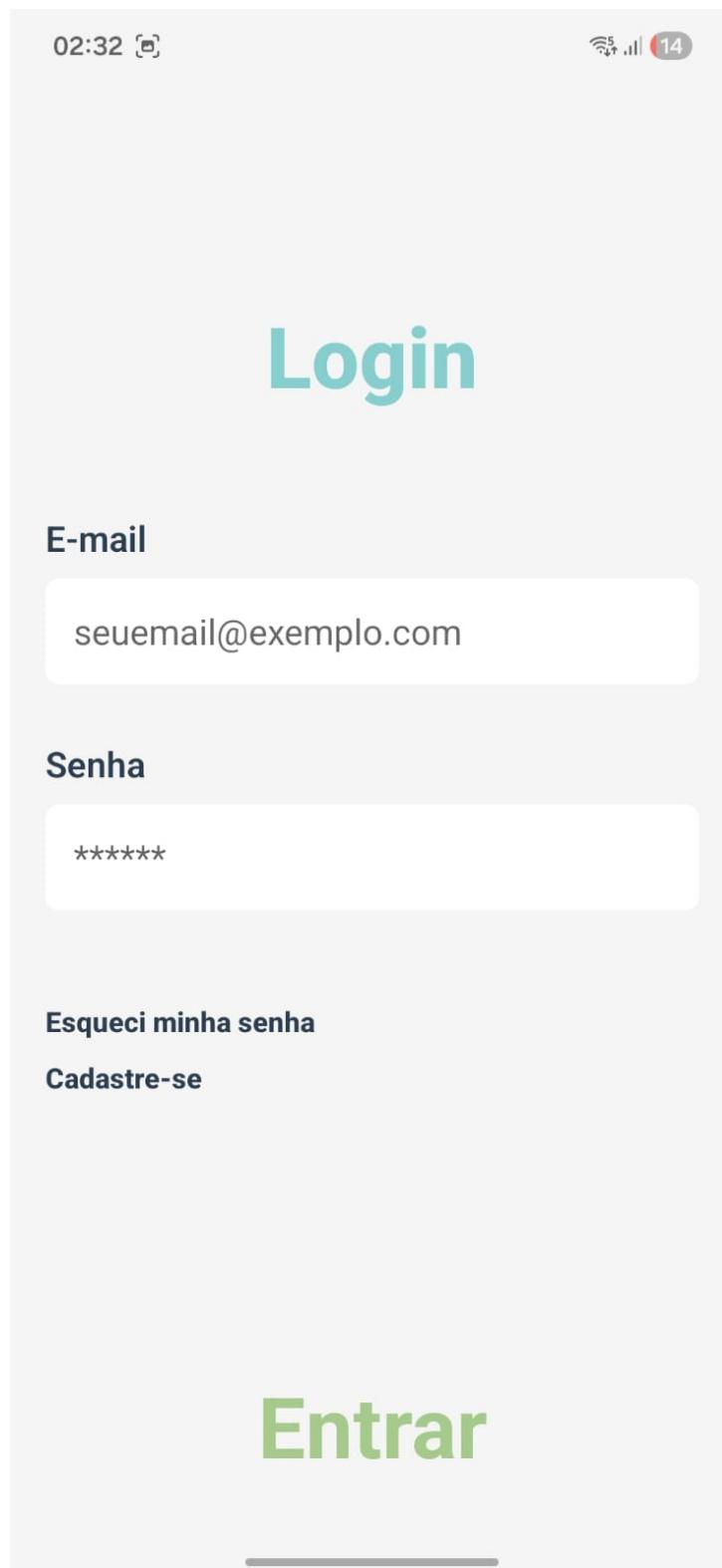


Figura A.1 – Tela de Login

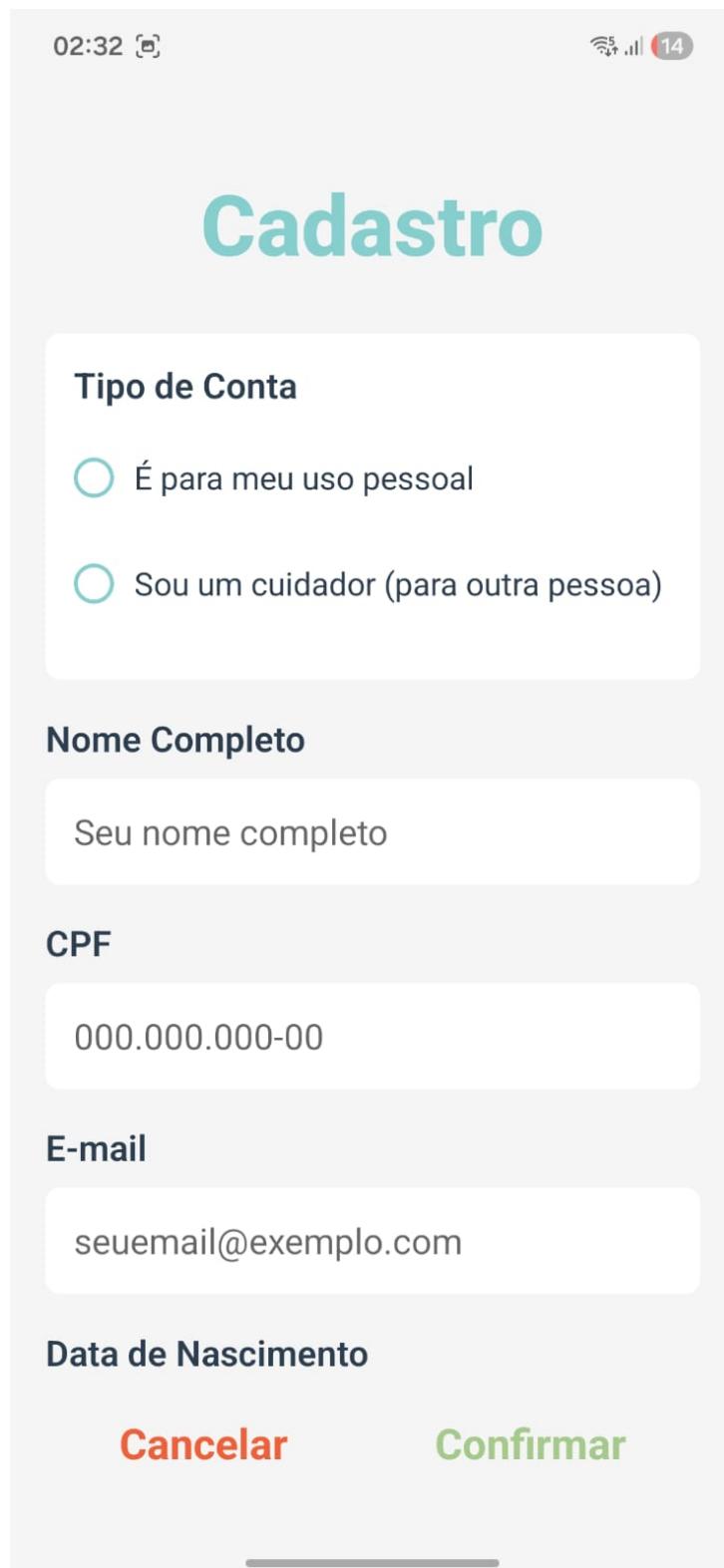


Figura A.2 – Tela de Cadastro de Usuário

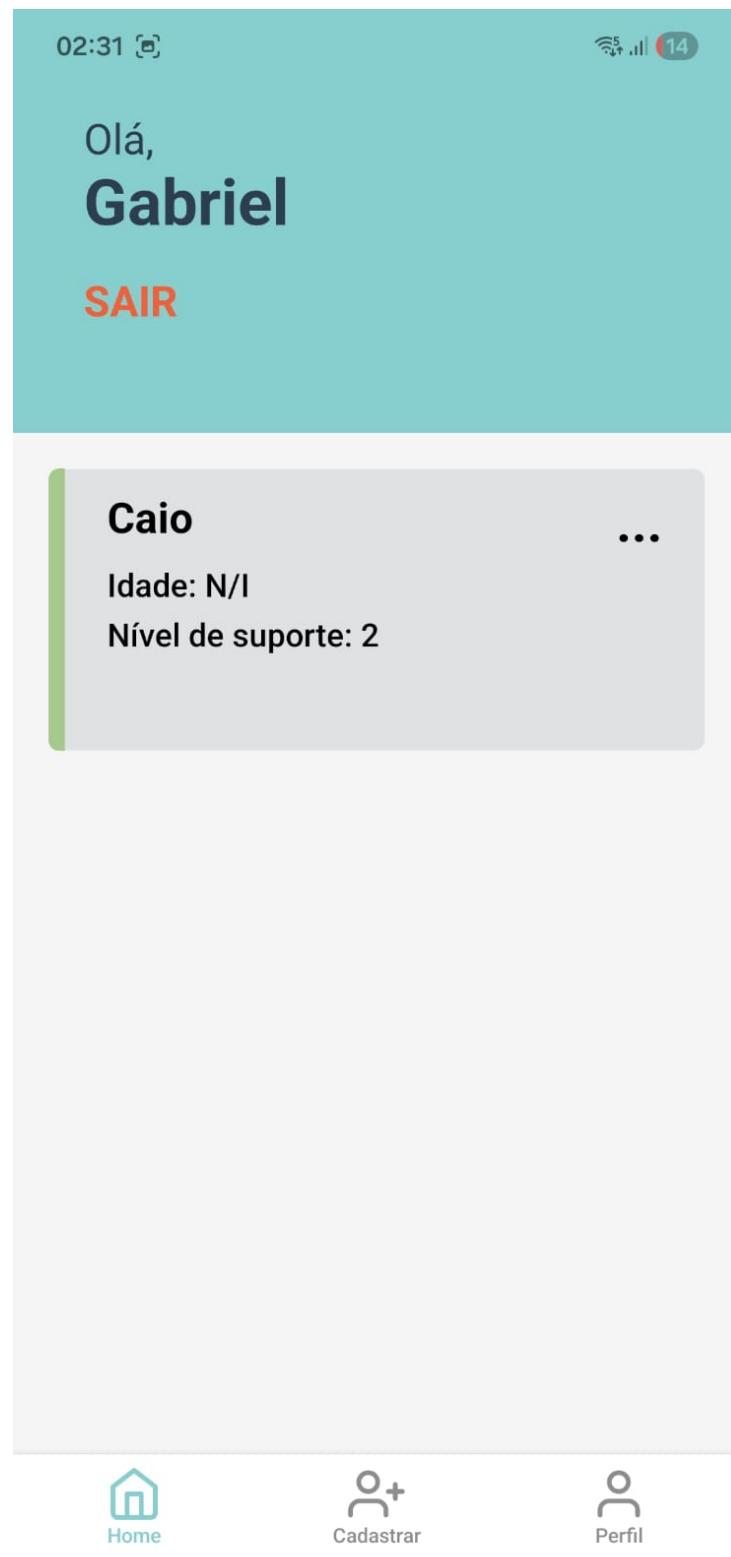


Figura A.3 – Home do Cuidador



Figura A.4 – Perfil do Cuidador

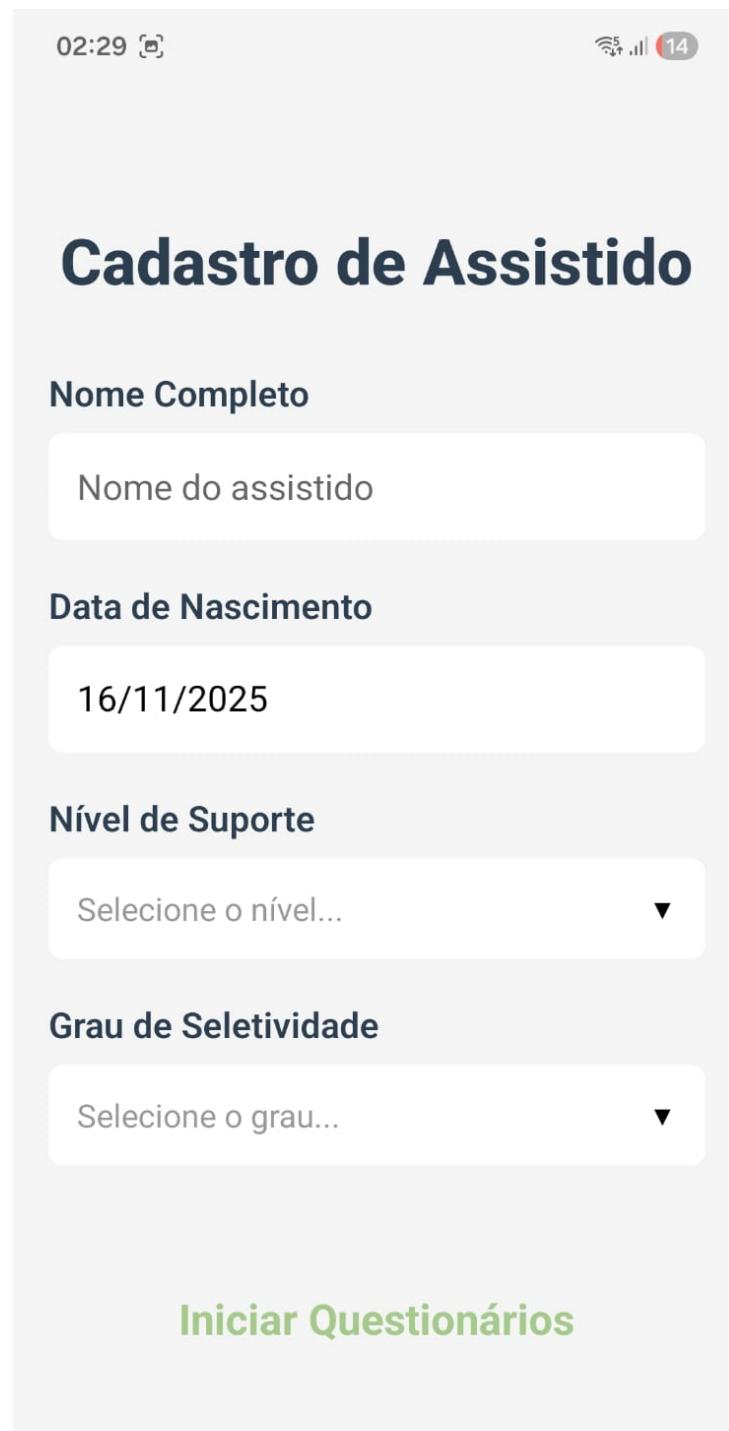


Figura A.5 – Cadastro de Assistido

02:29   14

< Voltar

Frequência Alimentar

Passo 1 de 3

Com que frequência o assistido come banana?

- Nunca
- 1x na semana
- 2-4x na semana
- 5-6x na semana
- 1x por dia ou mais

Com que frequência o assistido come maçã?

- Nunca
- 1x na semana

Figura A.6 – Preenchimento do Questionário



Figura A.7 – Refeições Disponíveis

The image shows a smartphone screen displaying a meal replacement application. The top status bar indicates the time as 02:34, signal strength, and battery level (13%). Below the status bar, a green back arrow and the text '< Voltar' are visible. The main title 'Almoço' is centered at the top of the screen. Three cards are listed vertically, each containing a food item, its suggestion category, and a note about a similar safe food.

- Carne de porco (Grelhada)**
Sugestão
Proteínas
Similar ao seu alimento seguro: Ovo (Cozido)
- Quinoa (Cozida)**
Sugestão
Cereais e Tubérculos
Similar ao seu alimento seguro: Pão (Forma)
- Pimentão (Cru)**
Sugestão
Verduras e Legumes
Similar ao seu alimento seguro: Alface (Crua)

Avaliar Refeição

Figura A.8 – Sugestão de Troca Alimentar



Figura A.9 – Tela de Feedback

Apêndice B – Protótipo de Média Fidelidade

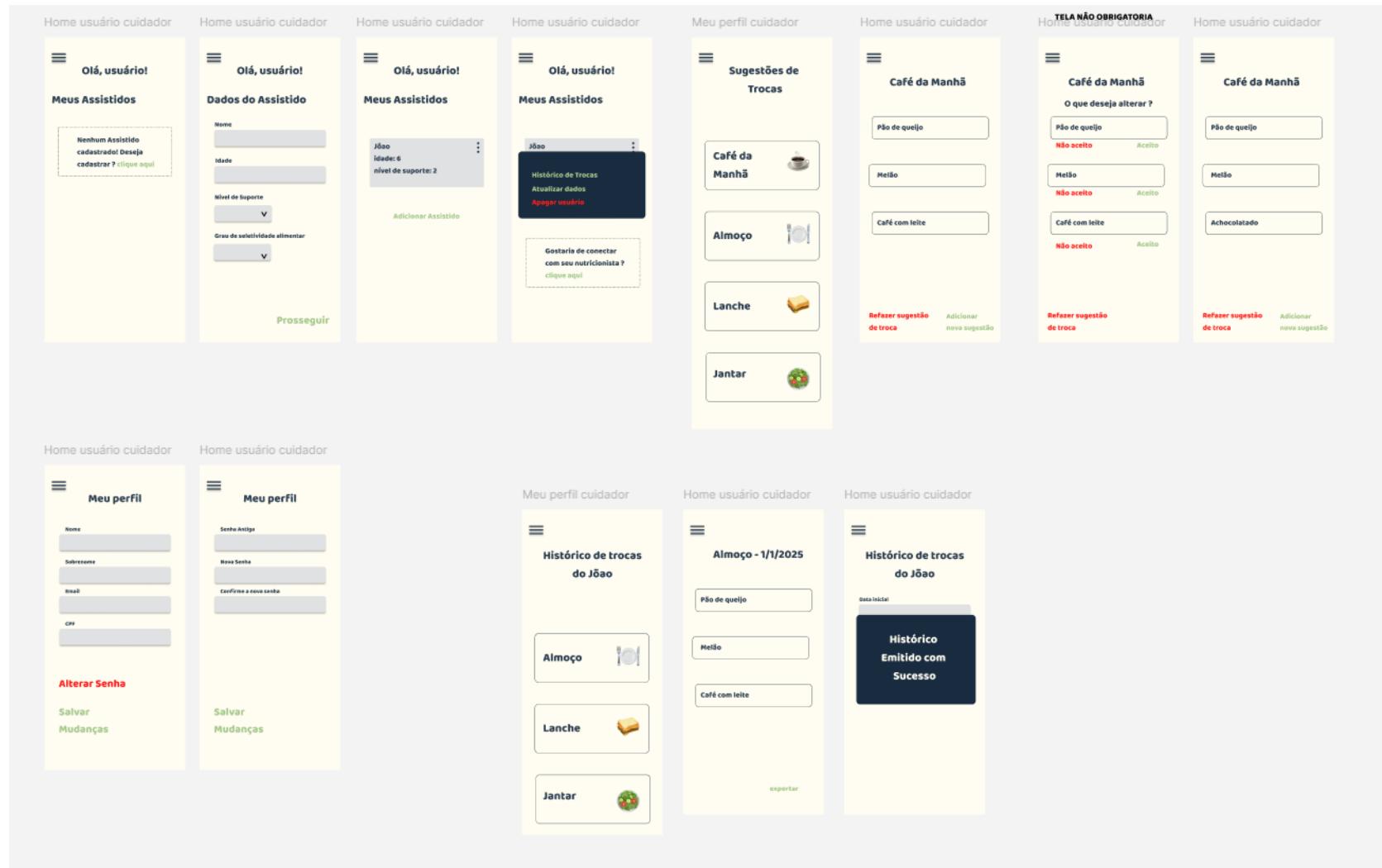


Figura B.1 – Protótipo de Média Fidelidade

