INSTITUTO FEDERAL GOIANO - CAMPUS MORRINHOS

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

NICOLAS SILVA ARAÚJO

BETESAÚDE: DESENVOLVIMENTO DE APLICAÇÃO MÓVEL MULTIPLATAFORMA PARA SUPORTE, EDUCAÇÃO E MONITORAMENTO DO DIABETES MELLITUS

MORRINHOS - GO
2026

NICOLAS SILVA ARAÚJO

BETESAÚDE: DESENVOLVIMENTO DE APLICAÇÃO MÓVEL MULTIPLATAFORMA PARA SUPORTE, EDUCAÇÃO E MONITORAMENTO DO DIABETES MELLITUS

Trabalho de curso apresentado ao Curso Superior de Tecnologia em Sistemas para Internet do Instituto Federal Goiano – Campus Morrinhos, como requisito parcial para obtenção de título em Tecnólogo em Sistemas para Internet.
Área de concentração: SISTEMAS DE INFORMAÇÃO
Orientador: Marcel da Silva Melo

MORRINHOS - GO
2026

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema Integrado de Bibliotecas do IF Goiano - SIBi

A663

Araújo, Nicolas Silva
BETESAÚDE: DESENVOLVIMENTO DE APLICAÇÃO MÓVEL MULTIPLATAFORMA PARA SUPORTE, EDUCAÇÃO E MONITORAMENTO DO DIABETES MELLITUS / Nicolas Silva Araújo. Morrinhos 2026.

64f. il.

Orientador: Prof. Me. Marcel da Silva Melo.
Tcc (Bacharel) - Instituto Federal Goiano, curso de 0421171 - [MO.GRAD] Curso Superior de Tecnologia em Sistemas para Internet - Morrinhos (Campus Morrinhos).
1. Aplicativo. 2. Sistema Web. 3. Diabéticos. 4. Acessibilidade digital. I. Título.

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Nicolas Silva Araújo

Matrícula:
2022204211710068

Título do trabalho:
BETESAÚDE: DESENVOLVIMENTO DE APLICAÇÃO MÓVEL MULTIPLATAFORMA PARA SUPORTE, EDUCAÇÃO E MONITORAMENTO DO DIABETES MELLITUS

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 09 /04 /2026

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Morrinhos — Local
09 /04 /2026 — Data

gov.br Documento assinado digitalmente
NICOLAS SILVA ARAUJO
Data: 08/04/2026 22:43:04-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

gov.br Documento assinado digitalmente
MARCEL DA SILVA MELO
Data: 09/04/2026 13:51:24-0300
Verifique em https://validar.iti.gov.br

Assinatura do(a) orientador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 6/2026 - CCEPTNM-MO/CEPTNM-MO/DE-MO/CMPMHOS/IFGOIANO

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ATA DE APRESENTAÇÃO DE TRABALHO DE CURSO

Aos **23** dias do mês de **março** de **2026**, às **19:30** horas, foi realizada, remotamente via Google Meet, a apresentação pública do trabalho de curso do discente **NICOLAS SILVA ARAÚJO** (2022204211710068) intitulado **BETESAÚDE: Desenvolvimento de aplicação móvel multiplataforma para suporte, educação e monitoramento do Diabetes *Mellitus***, como requisito necessário para a conclusão do curso de Tecnologia em Sistemas para Internet. A Banca Examinadora, constituída pelos professores: **Marcel da Silva Melo** – orientador, **Fernando Barbosa Matos e Antônio Neco de Oliveira**. Após a análise, emitiram o seguinte resultado:

( ) Aprovado

(X) Aprovado com ressalva

- **Observações**: Remoção do cálculo da insulina e criação do histórico detalhado da medição de glicose, informando uso de medicamentos e alimentos consumidos. Botão de emergência.

( ) Reprovado com o seguinte parecer:

Morrinhos, 23 de março de 2026

Por ser verdade firmamos a presente:

(Assinado Eletronicamente)
Marcel da Silva Melo
(Presidente da banca)

(Assinado Eletronicamente)
Fernando Barbosa Matos
(Membro)

(Assinado Eletronicamente)
Antônio Neco de Oliveira
(Membro)

Documento assinado eletronicamente por:

- **Marcel da Silva Melo**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 23/03/2026 20:52:12.
- **Fernando Barbosa Matos**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 23/03/2026 20:59:26.
- **Antonio Neco de Oliveira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 23/03/2026 21:18:26.

Este documento foi emitido pelo SUAP em 23/03/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 803257
Código de Autenticação: ccbc5f09f6

INSTITUTO FEDERAL GOIANO
Campus Morrinhos
Rodovia BR-153, Km 633, Zona Rural, SN, Zona Rural, MORRINHOS / GO, CEP 75650-000
(64) 3413-7900

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por ter me abençoado durante todas as etapas do curso. Em segundo lugar, à minha esposa, que esteve ao meu lado, compreendendo-me, ajudando-me e apoiando-me; e aos meus pais, que sempre estiveram comigo durante essa jornada. Agradeço ainda ao meu orientador pelo suporte e pelos ensinamentos, e a todos os colegas e professores que contribuíram para a minha trajetória acadêmica.

RESUMO

Este trabalho tem como objetivo apresentar o desenvolvimento de um aplicativo móvel multiplataforma, utilizando a linguagem Dart e o *framework* Flutter, com foco no monitoramento da glicemia, educação em saúde para pessoas com diabetes e recursos de acessibilidade digital. A proposta visa oferecer uma ferramenta de apoio que permita o registro e acompanhamento histórico dos níveis glicêmicos, fornecendo também conteúdos educativos relevantes para o autocuidado e a gestão da condição diabética. O sistema foi projetado com atenção à acessibilidade digital, buscando atender usuários com diferentes necessidades, como pessoas com baixa visão ou limitações motoras. A pesquisa está inserida na interseção entre tecnologia e saúde, destacando-se como uma solução inovadora no apoio do paciente por meio da tecnologia móvel. O desenvolvimento considerou princípios de usabilidade, design responsivo, alinhando-se às demandas atuais por soluções digitais inclusivas e personalizadas no contexto da saúde.

Palavras-chave: Aplicativo; Sistema Web; Diabéticos; Acessibilidade digital

ABSTRACT

This work aims to present the development of a multiplatform mobile application, using the Dart language and the Flutter framework, focusing on blood glucose monitoring, health education for people with diabetes, and digital accessibilityt features. The proposal aims to offer a support tool that allows for the recording and historical monitoring of glycemic levels, also providing relevant educational content for self-care and managemen digital accessibilityt of the diabetic condition. The system was designed with attention to digital accessibility, seeking to meet the needs of users with different requirements, such as people with low vision or motor limitations. The research is situated at the intersection of technology and health, standing out as an innovative technological solution in supporting patients through mobility. The current development adheres to principles of usability and responsive design, aligning with current demands for inclusive and personalized digital solutions in the health context.

Keywords: Application; Web System; Diabetics; Digital accessibility

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AOT – *Ahead-of-Time*

API – *Application Programming Interface*

CGM – *Continuous Glucose Monitoring*

COFEN – Conselho Federal de Enfermagem

DM – *Diabetes Mellitus*

FAQ – *Frequently Asked Questions*

IBGE – Instituto Brasileiro de Geografia e Estatística

IDF – *International Diabetes Federation*

IHC – Interação Humano-Computador

mHealth – *Mobile Health*

NIH – *National Institutes of Health*

NLU – *Natural Language Understanding*

NoSQL – *Not Only SQL*

OMS – Organização Mundial da Saúde

PDA – *Personal Digital Assistant*

SAMU – Serviço de Atendimento Móvel de Urgência

SBD – Sociedade Brasileira de Diabetes

SDK – *Software Development Kit*

Sembast – *Simple Embedded Application Store database*

STT – *Speech-to-Text*

TCC – Trabalho de Conclusão de Curso

TTS – *Text-to-Speech*

UFF – Universidade Federal Fluminense

UI – *User Interface*

UX – *User Experience*

W3C – *World Wide Web Consortium*

WCAG – *Web Content Accessibility Guidelines*

SUMÁRIO

1 INTRODUÇÃO

O diabetes *mellitus* representa um dos maiores desafios para a saúde pública no Brasil, impactando uma parcela significativa da população. A dimensão do problema pode ser compreendida ao se considerar o contingente populacional do país que, de acordo com os dados do Censo Demográfico 2022, realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), é de 203.080.756 habitantes *(IBGE, 2023).*

Com base nestes dados demográficos, a Sociedade Brasileira de Diabetes analisa o impacto da doença no cenário nacional. Com uma prevalência de 10,2% de diagnósticos autorreferidos, conforme apurado pelo inquérito Vigitel do Ministério da Saúde e divulgado pela Sociedade Brasileira de Diabetes *(SBD, 2025),* estima-se que aproximadamente 20 milhões de brasileiros convivam com esta condição crônica, evidenciando a urgência no desenvolvimento de ferramentas que auxiliem no monitoramento, acessibilidade digital e suporte aos pacientes.

Para além dos números, o diagnóstico de diabetes impõe uma jornada de adaptação e aprendizado contínuo. A jornada do paciente, especialmente a do recém-diagnosticado, é marcada por barreiras que a literatura busca compreender. Estudos que analisam a adesão ao tratamento evidenciam que as "dificuldades para seguir o tratamento" e a "complexidade" das rotinas impostas, como o monitoramento glicêmico e o monitoramento da insulina, são fatores centrais na experiência desses indivíduos *(SUPLICI et al., 2021).* A estes obstáculos, soma-se o desafio de encontrar informações confiáveis para a autogestão da doença. A ausência de ferramentas centralizadas, gratuitas e acessíveis que integrem educação, monitoramento e suporte primário cria uma lacuna significativa no cuidado, deixando o paciente vulnerável a erros de monitoramento e à incerteza sobre a gravidade de seus sintomas.

Diante desse cenário, o presente Trabalho de Conclusão de Curso (TCC) propõe o desenvolvimento da aplicação *mobile* responsiva Betesaúde, uma opção tecnológica pensada para preencher essa lacuna de acessibilidade digital. O aplicativo tem por escopo oferecer um suporte integrado ao paciente diabético, contemplando funcionalidades essenciais como o apontamento prático da

glicemia integrado ao registro de medicação e alimentação, uma seção educativa robusta e um inovador sistema de pré-consulta para triagem de sintomas, classificando-os para orientar o usuário, além de um sistema de emergência de acesso rápido.

Apesar da crescente oferta de tecnologias móveis voltadas para a saúde, uma análise do cenário atual revela que muitas das soluções disponíveis para o gerenciamento do diabetes apresentam barreiras que limitam sua eficácia. Muitas dessas soluções tecnológicas disponíveis para o gerenciamento do diabetes é limitada, especialmente para o público leigo ou recém-diagnosticado. A literatura científica, por meio de estudos de *benchmarking* de aplicativos, evidencia que entre os principais obstáculos estão a baixa usabilidade e a falta de recursos de acessibilidade digital adequados, que comprometem a experiência do usuário e a efetividade da ferramenta *(LOPES et al., 2024).*

A preocupação com a desinformação digital é validada por um estudo recente da Universidade Federal Fluminense (*UFF*), que ao analisar as fontes *on-line* sobre o tratamento do diabetes *mellitus*, revelou um cenário de "desordem informacional" e materiais de baixa fidedignidade. Tal panorama não apenas confunde o paciente, mas o expõe a riscos significativos, podendo levar à adoção de práticas inadequadas e à descontinuidade de terapias comprovadamente eficazes, o que reforça a necessidade de aplicações que centralizem e ofereçam conteúdo educativo curado e confiável *(UFF, 2025)*.

Aplicativos consolidados no mercado, como o Glic (*GLIC, 2024*), embora ofereçam robustas ferramentas de monitoramento, frequentemente concentram suas funcionalidades mais avançadas em planos de assinatura, criando uma barreira financeira e consequentemente limitações de uso. Outras soluções de alcance global, como o mySugr *(ROCHE DIABETES CARE GMBH, 2024)*, destacam-se pela gamificação e pela interface amigável, mas podem apresentar uma curva de aprendizado acentuada devido à vasta gama de funcionalidades e integrações, o que pode ser intimidador para usuários que buscam simplicidade e orientação inicial. Além disso, a carência de um sistema integrado de triagem de sintomas em grande parte dessas ferramentas força o usuário a buscar canais externos para avaliar a urgência de seu estado de saúde, fragmentando a experiência de cuidado.

Nesse contexto, o diferencial do Betesaúde reside na sua abordagem focada na simplicidade e na integração de funcionalidades essenciais e gratuitas, pensadas para acolher o paciente desde o momento do diagnóstico. Diferentemente de soluções que segmentam seus recursos, a proposta desse trabalho é oferecer, em uma única plataforma, o tripé fundamental do cuidado: monitoramento, educação e orientação.

A principal vantagem competitiva e de inovação do projeto é a inclusão do sistema de pré-consulta, uma ferramenta de triagem de sintomas que oferece ao usuário uma orientação primária imediata, recurso escasso ou inexistente nos principais aplicativos do mercado. Aliado a um *design* responsivo, com acessibilidade digital que garante o acesso em qualquer dispositivo, o Betesaúde busca ser não apenas um diário digital, mas um verdadeiro assistente de saúde para o paciente diabético.

Portanto, este trabalho tem como objetivo geral desenvolver uma aplicação *mobile* responsiva Betesaúde como uma ferramenta eficaz e acessível para o suporte ao gerenciamento do diabetes *mellitus*. Para atingir este fim, foram estabelecidos os seguintes objetivos específicos:

a) implementar um sistema intuitivo para o registro de glicemia;
b) desenvolver uma aplicação mobile com acessibilidade digital;
c) criar e integrar um sistema de triagem de sintomas para orientação do paciente;
d) estruturar uma seção de conteúdo educativo de fácil compreensão;
e) garantir a usabilidade e o acesso da aplicação em múltiplas plataformas.

2 REFERENCIAL TEÓRICO

A fundamentação teórica desse trabalho busca consolidar os conceitos que sustentam o desenvolvimento do ecossistema Betesaúde, transitando pela patologia do diabetes, a ascensão das tecnologias de *Mobile Health* (mHealth) e as ferramentas de engenharia de software empregadas na construção da solução.

2.1 O Diabetes Mellitus e os Desafios do Autocuidado

O Diabetes *Mellitus* (DM) configura-se como uma desordem metabólica crônica de crescente prevalência global, caracterizada fisiopatologicamente pela hiperglicemia decorrente de uma deficiência total na produção de insulina (DM Tipo 1) ou de uma incapacidade do organismo em utilizar eficazmente o hormônio produzido (DM Tipo 2) (*MINISTÉRIO DA SAÚDE*, 2009; *SBD*, 2023).

De acordo com estimativas da Organização Mundial da Saúde (*OMS*, 2023), a doença afeta centenas de milhões de indivíduos globalmente, com projeções que indicam um aumento substancial desse número nas próximas décadas caso não haja intervenções preventivas efetivas. No panorama nacional, o Brasil ocupa posição de destaque entre os países com o maior contingente de adultos diagnosticados com a condição, cenário que sobrecarrega o sistema público e suplementar de saúde, exigindo estratégias inovadoras para o controle populacional da doença (*INTERNATIONAL DIABETES FEDERATION*, 2021). Neste cenário, as necessidades clínicas dos pacientes transcendem o simples controle glicêmico, abrangendo também a prevenção de complicações agudas e crônicas, a limitação de incapacidades e a reabilitação (*OMS*, 2023).

O monitoramento diário dessa condição impõe desafios significativos. Segundo a Biblioteca Nacional de Medicina, vinculada aos Institutos Nacionais de Saúde dos Estados Unidos (*NIH*, 2022), a concepção de autocuidado no contexto do diabetes representa um marco transformador no processo educativo e na aquisição de saberes necessários para a convivência com a patologia. O autocuidado eficaz exige do paciente uma tomada de decisão autônoma, diária e ininterrupta, envolvendo o monitoramento de glicemia, adequação dietética e administração de medicamentos (*SBD*, 2023).

Para mitigar a carga cognitiva, física e emocional imposta por essa rotina rigorosa, a literatura médica tem incentivado fortemente a utilização da tecnologia. A inserção de ferramentas de saúde digital (digital *health*) no cotidiano do paciente tem se provado uma estratégia essencial para otimizar o acompanhamento longitudinal, facilitar o letramento em saúde e proporcionar uma melhor gestão integral do diabetes (*NIH*, 2022; *INTERNATIONAL DIABETES FEDERATION*, 2021).

2.2 Tecnologias Móveis Aplicadas à Saúde (mHealth)

Conforme publicações do Conselho Federal de Enfermagem (COFEN, 2022), o *mHealth* (acrônimo para *Mobile Health* ou Saúde Móvel) é definido como a prática médica e de saúde pública suportada por dispositivos móveis, englobando celulares, tablets, assistentes digitais (PDAs) e aparelhos de monitoramento sem fio. Esse conceito transforma a forma tradicional de prestação de cuidados médicos, permitindo que o monitoramento contínuo e a assistência ocorram "a qualquer hora e em qualquer lugar", transcendendo o espaço físico dos consultórios e hospitais (*COFEN*, 2022).

A utilização de dispositivos móveis para o suporte à saúde tem se mostrado uma estratégia eficaz na mitigação de barreiras de acesso. No entanto, a literatura aponta que a eficácia dessas ferramentas depende diretamente de atributos de usabilidade e acessibilidade digital, sendo a ocorrência da "desordem informacional" em ambientes digitais um risco real para a segurança do paciente (*COFEN*, 2022).

Ainda de acordo com o Conselho Federal de Enfermagem (2022), os principais objetivos e características do *mHealth* incluem a promoção da acessibilidade digital e prevenção, a otimização dos sistemas de atendimento e a versatilidade tecnológica. Em resumo, essa abordagem utiliza a mobilidade tecnológica para tornar a saúde mais conectada, acessível e contínua.

Com essa transformação estrutural, os limites físicos dos consultórios tradicionais são rompidos, permitindo que a prestação de cuidados e o monitoramento do paciente ocorram de forma descentralizada (*COFEN*, 2022).

2.3 Desenvolvimento Multiplataforma com o Framework Flutter

O *Flutter*, lançado pelo Google em 2018, consolidou-se como um *framework* de código aberto para o desenvolvimento de interfaces de usuário (UI). De acordo com sua documentação oficial (*FLUTTER*, 2026), sua principal proposta de valor reside na capacidade de permitir a criação de aplicações para múltiplas plataformas, incluindo: *iOS, Android, Web, Windows, MacOS* e *Linux,* utilizando uma única base de código. Diferentemente do desenvolvimento nativo tradicional, que exige códigos específicos para cada sistema operacional (resultando em maiores custos e tempo de desenvolvimento), o *Flutter* adota uma abordagem multiplataforma que busca unificar o processo sem sacrificar a performance (*FLUTTER*, 2026).

Para atingir um desempenho próximo ao nativo, o framework utiliza a linguagem de programação *Dart*, também desenvolvida pelo Google, que é compilada diretamente para código de máquina. Além disso, a tecnologia não depende dos componentes visuais originais de cada sistema hospedeiro, em vez disso, emprega seu próprio motor gráfico para renderizar a interface (*FLUTTER*, 2026). Essa arquitetura garante que a aparência e o comportamento da aplicação sejam consistentes em qualquer dispositivo, evitando discrepâncias visuais entre sistemas distintos.

A construção da interface nesse ambiente é baseada no conceito de "*widgets*", que são elementos modulares e personalizáveis que variam desde botões simples até *layouts* complexos, proporcionando grande flexibilidade visual (FLUTTER, 2026). A sigla UI, do inglês *User Interface* (Interface do Usuário), representa, em termos gerais, o ponto de interação entre um indivíduo e um sistema computacional, englobando tudo aquilo que o usuário vê, toca ou com o qual interage em um aplicativo (*PREECE; ROGERS; SHARP*, 2013).

No contexto de um framework de UI como o *Flutter* ou o *React Native*, essa definição ganha uma profundidade técnica maior. A interface deixa de ser apenas o "desenho" da tela e passa a ser compreendida como uma camada estruturada de código e componentes. Conforme os preceitos técnicos de desenvolvimento e design de interação, um framework de UI fornece um conjunto de ferramentas e blocos de construção pré-fabricados projetados para transformar regras de negócio (dados e lógica) em elementos visuais de forma eficiente e padronizada (*MICROSOFT*, 2026).

2.4 Persistência de Dados Local e Arquitetura NoSQL

No desenvolvimento de aplicações móveis voltadas à saúde (*mHealth*), particularmente aquelas dedicadas ao monitoramento diário de condições crônicas como o diabetes *mellitus*, a disponibilidade ininterrupta do sistema é um requisito não funcional crítico. Nesse cenário, a adoção de uma arquitetura *offline-first* (prioridade *offline*) emerge como uma estratégia fundamental para garantir que o usuário tenha acesso aos seus dados independentemente da conectividade (*SANTOS*, 2025). Para viabilizar essa arquitetura, a adoção de bancos de dados NoSQL (*Not Only SQL*, uma categoria de bancos não relacionais com esquemas flexíveis de armazenamento) embarcados simplifica o mapeamento de objetos complexos e oferece alto desempenho estrutural (*FOWLER; SADALAGE*, 2013).

Diferente das aplicações *online-first*, que dependem de chamadas constantes a servidores em nuvem (*APIs*) e travam quando há queda de rede, o paradigma *offline-first* oferece alta confiabilidade de uso (*SANTOS*, 2025). Esse modelo garante que a aplicação funcione primariamente a partir de uma base de dados local, assegurando que o paciente consiga registrar dados vitais, como índices glicêmicos, doses de insulina e sintomas, e acessar o histórico médico instantaneamente em qualquer situação (*FOWLER; SADALAGE*, 2013). A conexão com a internet continua sendo importante para a sincronização futura com um servidor, mas a sua ausência temporária não impede o uso contínuo da ferramenta (*SANTOS*, 2025).

Dentro do ecossistema de desenvolvimento multiplataforma em *Flutter*, destaca-se a utilização de soluções NoSQL embarcadas nativas, como o banco de dados *Sembast* (*Simple Embedded Application Store database*). Por ser escrito inteiramente na linguagem da aplicação (*Dart*), o *Sembast* opera de maneira fluida e integrada, oferecendo alto desempenho por meio de APIs totalmente assíncronas. Essa assincronicidade é vital para manter a interface de usuário (UI) responsiva, pois evita o bloqueio da *thread* principal durante as operações de leitura e escrita dos dados de saúde (*TEKARTIK*, 2023).

Além disso, a versatilidade dessa ferramenta permite que o mesmo código de persistência opere sobre diferentes tecnologias de armazenamento base, de acordo com a plataforma em execução. O sistema utiliza arquivos locais em

dispositivos *Android* e *iOS*, e encapsula o padrão *IndexedDB* quando a aplicação é executada em navegadores Web (*TEKARTIK*, 2023).

Dessa forma, a combinação de uma estratégia *offline-first* com a agilidade de um banco de dados NoSQL embarcado proporciona uma infraestrutura resiliente, segura e de alta performance, atributos indispensáveis para aplicações de autocuidado e suporte à decisão clínica (*FOWLER; SADALAGE*, 2013; *SANTOS*, 2025).

2.5 Interfaces Multimodais e Acessibilidade digital Assistiva

A literatura especializada em Interação Humano-Computador (IHC) define a Interface de Usuário (UI) como o espaço, físico ou virtual, onde ocorre a comunicação e a interação direta entre o indivíduo e o sistema computacional.

Em sua essência, a UI engloba todos os elementos perceptíveis na tela que permitem ao usuário inserir comandos, acessar funcionalidades e visualizar as respostas do software (*PREECE; ROGERS; SHARP*, 2013).

A evolução da IHC tem demonstrado que o desenvolvimento de interfaces multimodais, aquelas que combinam diferentes formas de entrada e saída de dados, como visão, tato e audição, é uma abordagem fundamental para a criação de sistemas verdadeiramente inclusivos. Segundo *Preece, Rogers* e *Sharp* (2013), a multimodalidade reduz a sobrecarga cognitiva e física, permitindo que a interação com o sistema ocorra de maneira mais natural e adaptável às limitações momentâneas ou permanentes do indivíduo.

No cenário específico do monitoramento do diabetes *mellitus*, a implementação de recursos de acessibilidade digital assistiva transcende a mera conveniência ergonômica, configurando-se como um recurso crítico de segurança e suporte clínico. Complicações agudas da doença, como os episódios de hipoglicemia, frequentemente desencadeiam uma sintomatologia que afeta diretamente a capacidade de interação com dispositivos móveis. Nesses episódios, os pacientes podem apresentar tremores, confusão mental, incoordenação motora e distúrbios visuais transitórios, como visão turva (SBD, 2023). Adicionalmente, as complicações crônicas, a exemplo da retinopatia diabética, podem comprometer severamente a acuidade visual a longo prazo (SBD, 2023).

Diante dessas barreiras fisiológicas, a dependência exclusiva de interfaces gráficas tradicionais, baseadas em leitura de pequenos textos e toques precisos na tela, torna-se um fator limitante e de risco (*PREECE; ROGERS; SHARP*, 2013). É neste contexto que a integração de tecnologias de Processamento de Linguagem Natural, especificamente a síntese de voz (*Text-to-Speech - TTS*) e o reconhecimento de voz (*Speech-to-Text - STT*), ganha relevância. A presença dessas ferramentas permite que a interface multimodal atue não apenas como um *software* de registro, mas como um assistente de orientação ativa e imediata (*W3C*, 2018).

Ao possibilitar que o paciente formule perguntas verbalmente e receba instruções auditivas claras, especialmente em momentos de crise, nos quais a coordenação motora e a visão estão temporariamente incapacitadas, o sistema promove a autonomia do usuário e mitiga os riscos de agravamento do quadro clínico. Essa abordagem garante que orientações vitais de saúde cheguem ao paciente de forma rápida e segura, alinhando-se diretamente aos princípios fundamentais de acessibilidade digital e design universal (*W3C*, 2018).

3 METODOLOGIA E AMBIENTE DE DESENVOLVIMENTO

O desenvolvimento do Betesaúde foi conduzido por um processo iterativo e incremental, alinhado às práticas Ágeis, em contraponto ao modelo sequencial *Waterfall* (Cascata) (*SOMMERVILLE*, 2019), dada a sua rigidez e dificuldade em acomodar ajustes de requisitos. Neste contexto, foram adotados os princípios das Metodologias Ágeis (*BECK et al.,* 2001), especificamente uma abordagem híbrida que combina os conceitos estruturais do Scrum (*SCHWABER; SUTHERLAND*, 2020) com a gestão de fluxo visual do Kanban (*KANBAN UNIVERSITY*, 2021).

Foi criado um protótipo inicial contendo as funcionalidades de base (registro e histórico), que foi progressivamente refinado e expandido com módulos mais complexos, como o Assistente Virtual e a Seção Educativa. Estas metodologias permitiram a validação contínua dos requisitos e a flexibilidade para aprimorar a arquitetura ao longo do ciclo de vida do projeto.

3.1 Ferramentas e Tecnologias

A seleção da *stack* de desenvolvimento (conjunto de tecnologias utilizadas) foi uma decisão estratégica, pautada nos pilares de *performance*, portabilidade e agilidade. As escolhas, documentadas no arquivo manifesto pubspec.yaml, são detalhadas e justificadas no Quadro 1.

Quadro 1: Ecossistema Tecnológico do Projeto Betesaúde

Categoria	Tecnologia/Ferramenta	Justificativa Técnica da Escolha
Linguagem	Dart (SDK 3.6.1)	Linguagem de programação utilizada no Flutter. Possui natureza moderna, fortemente tipada e otimizada para compilação *Ahead-of-Time* (AOT), o que garante alta performance em produção.

Framework de UI	Flutter	Adotado por sua arquitetura reativa baseada em *widgets* e pela sua capacidade de gerar uma base de código única e compilada nativamente para *web* e *mobile*. Esta escolha foi fundamental para cumprir o requisito de responsividade e acessibilidade digital em múltiplos dispositivos, cerne de um "Sistema para Internet".
Persistência Local	Sembast (com sembast_web)	Selecionado por ser um banco de dados NoSQL embarcado, escrito em Dart puro, o que elimina a necessidade de *bindings* externos. Sua API assíncrona e a utilização da IndexedDB no ambiente *web* garantem uma solução de armazenamento local performática e ideal para uma aplicação *offline-first*.
Interfaces de Voz	flutter_tts, speech_to_text	Pacotes que abstraem as APIs nativas de síntese e reconhecimento de voz, respectivamente. Foram cruciais para a implementação da interface multimodal do Assistente Virtual, um dos principais diferenciais de inovação e acessibilidade digital do projeto.
Utilitários	url_launcher, intl, audioplayers	Componentes de suporte que enriqueceram a funcionalidade: url_launcher para conectar o módulo educativo a fontes externas; intl para a formatação padronizada de datas e horas; e audioplayers para fornecer *feedback* auditivo, aprimorando a experiência do usuário.

Fonte: elaborado pelo autor (2025)

3.2 Arquitetura de Software

A arquitetura do Betesaúde foi concebida sob o princípio fundamental da Separação de Preocupações (*Separation of Concerns*), visando garantir a alta coesão e o baixo acoplamento entre os diferentes componentes do sistema.

3.2.1 Arquitetura em Camadas

O projeto foi estruturado em uma arquitetura de camadas lógicas, onde cada diretório possui uma responsabilidade única:

- Camada de Apresentação (UI): Localizada nos diretórios *screens*/ e *widgets*/, é responsável por toda a renderização da interface e pela

captura de interações do usuário. Esta camada é livre de regras de negócio;

- Camada de Lógica de Negócio: Distribuída entre as próprias telas (*StatefulWidgets*) para lógicas de UI e em funções específicas, como o motor de decisão (avaliarRespostasDetalhada). Esta camada orquestra os dados e aplica as regras que definem o comportamento do aplicativo;
- Camada de Acesso a Dados: Encapsulada no diretório db/, abstrai toda a complexidade da comunicação com o banco de dados *Sembast*. As telas interagem com esta camada através de uma API de serviço (SembastGlicemiaService), desconhecendo os detalhes de implementação da persistência.

3.2.2 Gestão de Estado

A gestão de estado foi abordada de forma pragmática. Para o estado global, como o perfil do usuário, optou-se por uma solução simples e eficaz através de uma classe com membros estáticos (PerfilUsuarioSelecionado), apresentada na Figura 1, evitando a complexidade de pacotes de gerenciamento de estado mais robustos, que seriam excessivos para a escala do projeto.

lib > shared > perfil_usuario.dart > ...

```dart
class PerfilUsuarioSelecionado {
  static String? _perfil;

  // Define o perfil selecionado (ex: 'recem' ou 'experiente')
  static void selecionar(String perfil) {
    _perfil = perfil;
  }

  // Retorna o perfil atualmente selecionado
  static String? get perfil => _perfil;

  // Reseta o perfil (útil para voltar à tela inicial)
  static void resetar() {
    _perfil = null;
  }
}
```

Figura 1 — Código da classe: PerfilUsuarioSelecionado

Fonte: elaborado pelo autor (2025).

Para o estado local das telas, como o controle de formulários ou o passo atual do assistente, utilizou-se o gerenciador de estado nativo do Flutter, o StatefulWidget, que se mostrou perfeitamente adequado e performático.

3.2.3 Estratégia de Compilação Condicional

Um dos desafios técnicos do projeto foi a necessidade de utilizar a biblioteca dart:html para a incorporação de vídeos, que é incompatível com plataformas *mobile*. A solução adotada foi a estratégia de compilação condicional.

Um arquivo "roteador" (iframe_video.dart - Figura 2) utiliza uma diretiva de exportação condicional para expor a implementação web (iframe_video_web.dart - Figura 3) ou uma implementação substituta (iframe_video_stub.dart - Figura 4) com base no ambiente de compilação.

Para a renderização dos vídeos, como visto na Figura 2, o projeto utiliza a técnica de compilação condicional por meio do arquivo iframe_video.dart. Na versão web, ele renderiza o *HtmlElementView* para exibir o *iframe* do YouTube, e em outras plataformas, exibe um texto substituto.

Esta abordagem demonstra uma prática avançada de engenharia de *software*, garantindo a portabilidade e a robustez do código-fonte em um ecossistema multiplataforma.

Figura 2 — Classe iframe_video.dart

Fonte: elaborado pelo autor (2025).

lib > screens > iframe_video_web.dart > ...

```dart
// lib/widgets/iframe_video_web.dart
// ignore: avoid_web_libraries_in_flutter
import 'dart:html';
import 'dart:ui' as ui;

import 'package:flutter/material.dart';

class IframeVideoWeb extends StatelessWidget {
  final String url;
  final String id;

  const IframeVideoWeb({super.key, required this.url, required this.id});

  @override
  Widget build(BuildContext context) {
    // ignore: undefined_prefixed_name
    ui.platformViewRegistry.registerViewFactory(
      id,
      (int viewId) {
        final iframe = IFrameElement()
          ..src = url
          ..style.border = 'none'
          ..allowFullscreen = true
          ..width = '100%'
          ..height = '100%';
        return iframe;
      },
    );

    return HtmlElementView(viewType: id);
  }
}
```

Figura 3 — Classe iframe_video_web.dart

Fonte: elaborado pelo autor (2025).

lib > screens > iframe_video_stub.dart > ...

```dart
// lib/widgets/iframe_video_stub.dart
import 'package:flutter/material.dart';

class IframeVideoWeb extends StatelessWidget {
  final String url;
  final String id;

  const IframeVideoWeb({super.key, required this.url, required this.id});

  @override
  Widget build(BuildContext context) {
    return const Center(
      child: Padding(
        padding: EdgeInsets.all(24.0),
        child: Text(
          "Este recurso de vídeo só está disponível na versão Web do aplicativo.",
          textAlign: TextAlign.center,
          style: TextStyle(fontSize: 16),
        ), // Text
      ), // Padding
    ); // Center
  }
}
```

Figura 4 — Classe iframe_video_stub.dart

Fonte: elaborado pelo autor (2025).

3.3 Implementação dos Módulos Funcionais

Nesta seção é detalhada a implementação dos módulos que constituem a experiência do usuário no Betesaúde.

3.3.1 Módulo de *Onboarding* e Personalização

A jornada do usuário é iniciada pela SelecaoPerfilScreen, que implementa a segmentação. A escolha do perfil é persistida no gerenciador de estado global PerfilUsuarioSelecionado, que servirá como fonte da verdade para outras partes do sistema. Para o usuário recém-diagnosticado, o sistema dispara a *OnboardingScreen*, um guia introdutório implementado com um *PageView* que apresenta informações basilares sobre a condição e a aplicação, cumprindo o objetivo de criar uma experiência de acolhimento e reduzir a carga cognitiva inicial.

3.3.2 Módulo de Monitoramento de Glicemia e Hábitos

Este módulo é o coração da funcionalidade de autocuidado. A classe RegistroGlicemiaScreen apresenta um formulário validado para a entrada de dados.

```dart
class _RegistroGlicemiaScreenState extends State<RegistroGlicemiaScreen> {
  final _formKey = GlobalKey<FormState>();

  final _valorController = TextEditingController();
  final _medicacaoController = TextEditingController();
  final _alimentacaoController = TextEditingController();

  String _refeicao = 'Café da manhã';

  Future<void> _salvarRegistro() async {
    if (_formKey.currentState!.validate()) {
      try {
        final valorTexto = _valorController.text.replaceAll(',', '.');
        final valor = double.parse(valorTexto);

        if (valor < 20 || valor >= 500) {
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
                content: Text(
                    "Valor de glicemia inválido. Insira um valor entre 20 e 500 mg/dL.")),
          );
          return;
        }

        final horario = DateFormat('dd/MM/yyyy HH:mm').format(DateTime.now());

        final novo = Glicemia(
          valor: valor,
          horario: horario,
          refeicao: _refeicao,
          medicacao: _medicacaoController.text,
          alimentacao: _alimentacaoController.text,
        );

        await SembastGlicemiaService.salvarGlicemia(novo,
            resposta: "registro_manual");
```

Figura 5 — Registro da Glicemia e hábitos

Fonte: elaborado pelo autor (2025).

O código apresentado na Figura 5 foca em uma visão holística, permitindo ao usuário registrar não apenas o valor numérico da glicose, mas também *textos descrevendo a Medicação Tomada e a Alimentação Consumida*, salvando tudo em uma instância única no banco Sembast. A orquestração do fluxo de

persistência ocorre na função _salvarRegistro(), acionada pelo botão "Salvar", que primeiramente valida o formulário via _formKey, coleta e formata os dados dos controladores e da data.

3.3.3 Módulo de Orientação: O Assistente Virtual e o Motor de Decisão

O diferencial competitivo do Betesaúde reside em seu módulo de orientação. A classe AssistenteScreen implementa uma interface multimodal que combina reconhecimento e síntese de voz.

```dart
class AssistenteScreen extends StatefulWidget {
  const AssistenteScreen({super.key});

  @override
  State<AssistenteScreen> createState() => _AssistenteScreenState();
}

class _AssistenteScreenState extends State<AssistenteScreen> {
  int passo = 0;
  String? glicemia;
  String? sintomas;
  String? alimentacao;

  final FlutterTts flutterTts = FlutterTts();
  final stt.SpeechToText _speech = stt.SpeechToText();
  final AudioPlayer _audioPlayer = AudioPlayer();
  bool ouvindo = false;

  final perguntas = [
    "Como está sua glicemia?",
    "Você está com sintomas (exemplo: tontura, tremores)?",
    "Você se alimentou nas últimas 2 horas?"
  ];

  final respostasValidas = [
    ["baixo", "normal", "alta"],
    ["sim", "não"],
    ["sim", "não"]
  ];

  final palavrasChave = [
    {
      "baixo": ["baixa", "baixo", "glicose baixa"],
      "normal": ["normal"],
      "alta": ["alta", "alto", "glicose alta"]
    },
```

Figura 6 — Estado e Base de Conhecimento do Assistente

Fonte: elaborado pelo autor (2025).

Esse trecho de código, apresentado na Figura 6, define a configuração inicial e o estado da AssistenteScreen.

1. Variáveis de Estado:
    - O controlador da máquina de estados, este inteiro (iniciado em 0) rastreia qual pergunta está sendo feita.

- Glicemia, sintomas, alimentação: Variáveis de *String*? (com opção do valor nulo) que armazenam as respostas do usuário à medida que ele avança pelos passos.

2. Instâncias de Serviço:
   - *flutterTts, _speech, _audioPlayer*: São as instâncias finais dos pacotes que dão ao assistente suas capacidades multimodais: *flutterTts* para falar (*Text-to-Speech*), _speech para ouvir (*Speech-to-Text*) e *_audioPlayer* para tocar sons de *feedback*.
3. Base de Conhecimento (As Listas):
   - perguntas: Uma lista (*List<String>*) que armazena o texto exato de cada pergunta do fluxo de triagem, em ordem sequencial;
   - respostasValidas: Mapeia cada pergunta para uma lista de respostas canônicas. Isso é usado para validar a entrada;
   - palavrasChave: Esta é a parte mais "inteligente". É uma lista de mapas que funciona como um motor simples de Interpretação de Linguagem Natural (NLU). Ela permite que o sistema entenda diversas formas de fala do usuário (ex: "glicose baixa" ou "baixo") e as normalize para uma única resposta válida (ex: "baixo"), que é então armazenada nas variáveis de estado.

```dart
class _AssistenteScreenState extends State<AssistenteScreen> {
  String avaliar() {
    if (glicemia == "baixo") {
      return sintomas == "sim"
          ? "Atenção: hipoglicemia sintomática. Consuma glicose rápida e procure ajuda se não melhorar."
          : "Glicemia baixa. Tome açúcar simples e monitore em 15 minutos.";
    } else if (glicemia == "normal") {
      return "Sua glicemia está normal. Continue com bons hábitos!";
    } else if (glicemia == "alta") {
      if (sintomas == "sim") {
        return "Glicemia alta com sintomas. Procure orientação médica.";
      } else if (alimentacao == "sim") {
        return "Glicemia alta após refeição. Monitore novamente em 2 horas.";
      } else {
        return "Hiperglicemia em jejum. Reavalie sua dieta ou busque orientação profissional.";
      }
    }
    return "Não foi possível avaliar sua condição.";
  }

  Future<void> falarPergunta() async {
    if (passo < perguntas.length) {
      await flutterTts.setLanguage("pt-BR");
      await flutterTts.setSpeechRate(0.9);
      await flutterTts.speak(perguntas[passo]);
    }
  }
```

Figura 7 — Lógica de Decisão e Vocalização

Fonte: elaborado pelo autor (2025).

Este trecho apresentado na Figura 7 mostra duas das funções mais importantes do assistente: a capacidade de pensar (avaliar) e de falar (falarPergunta).

A função avaliar(), considerada o "cérebro" do assistente virtual, é acionada imediatamente após a coleta de todas as respostas do usuário, utilizando as variáveis de estado (glicemia, sintomas e alimentação) preenchidas durante o fluxo interativo. Por meio de uma árvore de decisão implementada com estruturas if/else aninhadas, o sistema realiza o cruzamento dessas informações para classificar a situação do indivíduo de forma precisa. Tal lógica apresenta grande utilidade preventiva ao diferenciar quadros de hipoglicemia com e sem sintomas, bem como distinções entre hiperglicemia sintomática, pós-refeição ou em jejum.

Ao final do processamento, a função retorna a String exata da recomendação personalizada, que serve como parâmetro de entrada para o direcionamento automático às telas de ParabensScreen ou RecomendacaoScreen.

A função falarPergunta(), identificada como o Atuador de Voz do sistema, consiste em uma implementação assíncrona (Future<void>) da tecnologia Text-to-Speech (TTS). O funcionamento desse componente baseia-se no resgate da pergunta pertinente dentro da lista de diálogos, empregando a variável de estado passo como índice para localizar a instrução correta.

Para a execução da saída sonora, a função realiza a configuração técnica do motor de TTS (flutterTts), definindo parâmetros cruciais como o idioma e a velocidade da fala, culminando na execução do comando flutterTts.speak() para a vocalização efetiva ao usuário. É essa funcionalidade que assegura a interatividade e a acessibilidade digital do assistente virtual, permitindo que a aplicação se comunique de forma clara com o paciente.

Esta função implementa um sistema especialista baseado em regras, utilizando uma árvore de decisão para analisar um conjunto de cinco variáveis de entrada e classificar a condição do usuário.

A lógica é projetada para ser segura, priorizando sempre os sinais de alerta mais graves. Por exemplo, a presença de sintomas sobrepõe-se a outras variáveis para recomendar a busca por orientação médica imediata. A granularidade das regras permite distinguir cenários como hiperglicemia pós-prandial e hiperglicemia de jejum com não adesão à medicação, fornecendo recomendações acionáveis e de maior precisão.

O resultado dessa avaliação direciona o usuário para telas de *feedback* contextual: a ParabensScreen para reforço positivo e a RecomendacaoScreen para alertas, fechando o ciclo de interação com uma resposta clara e apropriada.

O componente central dessa funcionalidade é o seu motor de decisão, consolidado na função avaliar(), conforme demonstrado anteriormente na Figura 7. (Nota: Em iterações preliminares do escopo, a função chegou a ser mapeada como avaliarRespostasDetalhada, sendo posteriormente refatorada e otimizada para o modelo final da aplicação).

```dart
String avaliarRespostasDetalhada({
  required String glicemia,
  required String sintomas,
  required String alimentacao,
  required String sedeOuUrina,
  required String medicacao,
}) {
  if (glicemia == "baixo") {
    if (sintomas == "sim") {
      return "Atenção: possível hipoglicemia sintomática. Tome glicose de ação rápida e procure ajuda se não melhorar.";
    }
    return "Glicemia baixa. Tome algo doce e monitore sua glicemia em 15 minutos.";
  }

  if (glicemia == "normal") {
    return "Sua glicemia está normal. Continue mantendo bons hábitos alimentares e a medicação em dia.";
  }

  if (glicemia == "alto") {
    if (sintomas == "sim" || sedeOuUrina == "sim") {
      return "Glicemia alta com sinais de alerta. Procure orientação médica imediatamente.";
    }
    if (alimentacao == "sim") {
      return "Glicemia alta após refeição. Reavalie sua alimentação e monitore novamente em 2 horas.";
    }
    if (alimentacao == "nao" && medicacao == "nao") {
      return "Glicemia elevada em jejum e sem uso de medicação. Consulte seu médico para ajustar o tratamento.";
    }
    return "Hiperglicemia leve. Mantenha a hidratação, siga a dieta e monitore nas próximas horas.";
  }

  return "Não foi possível avaliar com precisão. Verifique os dados e tente novamente.";
}
```

Figura 8 — Motor de Decisão Avançado

Fonte: elaborado pelo autor (2025).

O código apresentado na Figura 8 é o "motor de decisão" (ou "cérebro") do Assistente Virtual, em sua versão mais completa. É uma função pura que recebe cinco parâmetros (String) sobre o estado do usuário e retorna uma String de recomendação.

A lógica de triagem implementa uma árvore de decisão estruturada através de condições if/else aninhadas, simulando o raciocínio clínico de uma triagem primária. Diferentemente de assistentes mais simplificados, esta versão destaca-se pela robustez ao cruzar cinco variáveis distintas, o que permite orientações preventivas mais precisas.

No tratamento específico da hiperglicemia, o algoritmo prioriza a identificação de sinais de alerta, como sintomas gerais, sede excessiva ou aumento da frequência urinária, para indicar casos de urgência, além de diferenciar elevações glicêmicas pós-prandiais daquelas ocorridas em jejum, identificando cenários de risco específicos como a hiperglicemia em jejum combinada à não administração da medicação.

O resultado desse processamento é uma saída não apenas informativa, mas estritamente acionável, fornecendo orientações claras como a necessidade

de procurar ajuda médica imediatamente ou realizar novo monitoramento em curtos intervalos de tempo.

3.3.4 Módulo Educacional e de Suporte

Para combater a desinformação, a classe EducacaoScreen, apresentada na Figura 9, centraliza conteúdo curado, combinando vídeos embutidos e *links* para artigos de fontes cientificamente validadas. A implementação de um sistema de rastreamento de progresso, persistido localmente via Sembast, introduz um elemento de gamificação que incentiva o engajamento do usuário.

```
class _EducacaoScreenState extends State<EducacaoScreen> {
  Future<void> _initDb() async {
    final dbFactory = databaseFactoryWeb;
    _db = await dbFactory.openDatabase('betesaude_educacao.db');
    final storedData = await _store.find(_db);
    setState(() {
      for (var video in videos) {
        final id = video['id']!;
        progresso[id] =
            storedData.any((snap) => snap.key == id && snap.value == true);
      }
    });
  }

  Future<void> marcarComoAssistido(String id) async {
    await _store.record(id).put(_db, true);
    setState(() => progresso[id] = true);
  }

  void abrirLink(String url) async {
    final uri = Uri.parse(url);
    if (await canLaunchUrl(uri)) {
      await launchUrl(uri, mode: LaunchMode.externalApplication);
    }
  }

  Future<void> falarTexto(String texto) async {
    await flutterTts.setLanguage("pt-BR");
    await flutterTts.setPitch(1.0);
    await flutterTts.speak(texto);
  }

  void irParaHome() {
    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (_) => const HomeScreen()),
      (route) => false,
```

Figura 9 — Classe EducacaoScreen

Fonte: elaborado pelo autor (2025).

A classe EducacaoScreen é uma tela com estado (StatefulWidget) que gerencia o progresso de aprendizado do usuário. O fluxo técnico segue os seguintes passos:

1. Carregamento (initState/_initDb): Ao iniciar, a tela consulta o banco de dados Sembast para carregar o progresso salvo, verificando quais vídeos já foram marcados como assistidos.
2. Atualização de Estado (marcarComoAssistido): Quando o usuário toca no botão de "*check*", a função marcarComoAssistido é disparada. Ela atualiza o estado local (progresso[id] = true), salva essa informação de forma persistente no Sembast (_store.record(id).put(_db, true)) e reconstrói a UI para refletir a mudança, atualizando a barra de progresso.
3. Navegação Externa: Ao clicar nos links de artigos, a função abrirLink (que utiliza o pacote url_launcher) é chamada para abrir o navegador padrão do usuário.
4. Esta tela é a resposta direta ao problema da desinformação on-line. Ao centralizar conteúdo multimídia de fontes seguras e rastrear o progresso do usuário, o Betesaúde se posiciona como um agente ativo de educação. A integração de TTS e a oferta de múltiplos formatos (vídeo e texto) reforçam o pilar de Acessibilidade digital, garantindo que o conhecimento seja acessível a usuários com diferentes níveis de literacia ou necessidades visuais.

A Classe PerguntasPredefinidasScreen (Figura 10) atua como um FAQ interativo, utilizando a mesma tecnologia de reconhecimento de voz do assistente para fornecer respostas rápidas a dúvidas comuns, ampliando o alcance do suporte educacional. Esta classe define a classe de estado (PerguntasPredefinidasScreenState) que gerencia toda a lógica da tela de FAQ

```dart
class PerguntasPredefinidasScreen extends StatefulWidget {
  const PerguntasPredefinidasScreen({super.key});

  @override
  State<PerguntasPredefinidasScreen> createState() =>
      _PerguntasPredefinidasScreenState();
}

class _PerguntasPredefinidasScreenState
    extends State<PerguntasPredefinidasScreen> {
  final FlutterTts flutterTts = FlutterTts();
  final stt.SpeechToText _speech = stt.SpeechToText();
  final TextEditingController _controller = TextEditingController();

  String? resposta;
  bool _ouvindo = false;
  bool _falando = false;

  final Map<String, String> perguntasERespostas = {
    'o que é diabetes':
        'Diabetes é uma condição em que o corpo não produz ou não utiliza adequadamente a insulina.',
    'sintomas do diabetes':
        'Os sintomas incluem sede excessiva, urinar frequentemente, fadiga e perda de peso.',
    'como prevenir complicações':
        'Manter a glicemia controlada, seguir dieta adequada e praticar exercícios físicos.',
    'como aplicar insulina':
        'A insulina deve ser aplicada com orientação médica, geralmente no abdômen ou braço.',
    'alimentação para diabéticos':
        'É recomendada uma dieta com baixo índice glicêmico, rica em fibras e sem açúcar refinado.',
    'quais são os sintomas do diabetes':
        'Os sintomas incluem sede excessiva, urinar frequentemente, fadiga e perda de peso.',
    'alimentação ideal para quem tem diabetes':
        'Prefira alimentos integrais, vegetais, legumes e evite açúcar refinado.',
  };
```

Figura 10 — Configuração do Estado e Base de Conhecimento

Fonte: elaborado pelo autor (2025).

A inicialização dos controladores é fundamental para estabelecer a interatividade multimodal da aplicação, empregando o FlutterTts para a síntese de voz (*Text-to-Speech*), o stt.SpeechToText para o reconhecimento de fala (*Speech-to-Text*) e o TextEditingController para o gerenciamento do campo de entrada textual. No que tange ao gerenciamento de estado, utilizam-se variáveis booleanas como _ouvindo e _falando para controlar dinamicamente os elementos da interface, enquanto a variável nulável String? resposta armazena o retorno da consulta, permitindo que a interface exiba o resultado apenas quando este estiver disponível.

O núcleo dessa funcionalidade reside na estrutura perguntasERespostas (Map<String, String>), que atua como uma base de conhecimento local. Nela as chaves representam os termos de busca esperados na interação do usuário e os valores contêm as respostas curadas e confiáveis, materializando a proposta do projeto de combater a desinformação através do fornecimento de conteúdo pré-validado.

```dart
void buscarResposta() async {
  final perguntaRaw = _controller.text.trim();
  if (perguntaRaw.isEmpty) return;

  final perguntaNormalizada = _removerAcentos(perguntaRaw.toLowerCase());
  String? respostaEncontrada;

  for (var entrada in baseConhecimento.entries) {
    for (var palavraChave in entrada.key) {
      if (perguntaNormalizada.contains(palavraChave)) {
        respostaEncontrada = entrada.value;
        break;
      }
    }
    if (respostaEncontrada != null) break;
  }

  respostaEncontrada ??=
      "Desculpe, não encontrei uma resposta exata para sua pergunta. Consulte seu médico para orientações precisas.";
```

Figura 11 — Lógica de Busca de Resposta

Fonte: elaborado pelo autor (2025).

A função buscarResposta, apresentada na Figura 11, é o "motor de busca" da tela de FAQ. Ela é chamada quando o usuário pressiona "enviar" ou termina de falar no microfone. A função inicia o processamento com a normalização da entrada do usuário, executada pela instrução final pergunta = _controller.text.toLowerCase().trim().

Este passo é crucial para assegurar a flexibilidade da busca, convertendo o texto para caracteres minúsculos e removendo espaços excedentes, o que torna o sistema imune a inconsistências de capitalização.

Na sequência, a lógica de correspondência itera sobre as chaves da base de conhecimento utilizando o método *contains* em detrimento de uma comparação de igualdade estrita (==); essa abordagem permite que o sistema identifique a intenção do usuário mesmo quando a palavra-chave está inserida em frases mais complexas ou coloquiais. A eficiência do algoritmo é garantida pela interrupção imediata do laço de repetição (*break*) assim que a primeira correspondência é localizada, enquanto o operador de atribuição nula (??=) atua como um mecanismo de segurança (*fallback*), definindo uma mensagem padrão caso nenhuma resposta seja encontrada, assegurando assim que o usuário sempre receba um *feedback* do sistema.

3.3.5 Interface e Experiência do Usuário (UI/UX)

A coesão da experiência do usuário é garantida por um mini "*Design System*" definido em main.dart, com uma paleta de cores consistente (liderada pelo tom Colors.teal) e uma tipografia padronizada. A responsividade da interface é explicitamente tratada na *HomeScreen*, que emprega um *LayoutBuilder* para adaptar o número de colunas de um *GridView*, garantindo uma usabilidade ótima em dispositivos de diferentes dimensões, desde *smartphones* a *desktops*. A preocupação com a acessibilidade digital é um tema transversal, manifestado no uso recorrente da síntese de voz e na previsão de fontes ampliadas na ConfiguracoesScreen.

```
class EscolhaUsuarioScreen extends StatelessWidget {
  Widget build(BuildContext context) {
                const SizedBox(height: 32),
                const Icon(Icons.health_and_safety, size: 60, color: Colors.teal),
                const SizedBox(height: 24),
                const Text(
                  "Você é recém-diagnosticado com diabetes ou já convive com a condição?",
                  textAlign: TextAlign.center,
                  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
                ), // Text
                const SizedBox(height: 32),
                ElevatedButton.icon(
                  icon: const Icon(Icons.fiber_new),
                  label: const Text("Sou recém-diagnosticado"),
                  onPressed: () {
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(
                        builder: (_) =>
                            OnboardingScreen(destinoFinal: '/educacao'),
                      ), // MaterialPageRoute
                    );
                  },
                  style: ElevatedButton.styleFrom(
                    minimumSize: const Size(double.infinity, 50),
                    backgroundColor: isDark ? Colors.teal[400] : Colors.teal,
                    foregroundColor: Colors.white,
                  ),
                ), // ElevatedButton.icon
                const SizedBox(height: 16),
                ElevatedButton.icon(
                  icon: const Icon(Icons.check_circle_outline),
                  label: const Text("Já convivo com diabetes"),
                  onPressed: () {
                    Navigator.pushReplacementNamed(context, '/home');
                  },
```

Figura 12 — Lógica de Navegação da Tela de Segmentação

Fonte: elaborado pelo autor (2025).

O código demonstrado na Figura 12 foi implementado pela classe EscolhaUsuarioScreen (definida no arquivo main.dart) e atua como o ponto de entrada (*home*) do componente *MaterialApp*. A lógica de fluxo é controlada pelos

eventos *onPressed* dos dois botões, orientando a navegação conforme o fluxo a seguir:

Fluxo 1 (Recém-diagnosticado): Ao selecionar a primeira opção, o sistema utiliza o método *Navigator.pushReplacement* para navegar até a *OnboardingScreen*. Esta, por sua vez, é instruída (via parâmetro destinoFinal: '/educacao') a encaminhar o usuário diretamente à EducacaoScreen após a sua conclusão. Este fluxo prioriza o pilar de Educação.

Fluxo 2 (Usuário Experiente): Ao selecionar a segunda opção, o sistema executa o comando *Navigator.pushReplacementNamed(context, '/home'),* direcionando o usuário imediatamente ao painel principal (*HomeScreen*). Este fluxo, por sua vez, prioriza o pilar de Monitoramento.

```
class _HomeScreenState extends State<HomeScreen> with TickerProviderStateMixin {
  Widget build(BuildContext context) {
                    "Assistente Virtual",
                    Icons.smart_toy_outlined,
                    const AssistenteScreen(),
                  ),
                  _buildFeatureCard(
                    "Educação em Saúde",
                    Icons.school_outlined,
                    const EducacaoScreen(),
                  ),
                  _buildFeatureCard(
                    "Registrar Glicemia",
                    Icons.add_chart,
                    RegistroGlicemiaScreen(),
                  ),
                  _buildFeatureCard(
                    "Histórico de Glicemia",
                    Icons.history,
                    const HistoricoGlicemiaScreen(),
                  ),
                  _buildFeatureCard(
                    "Perguntas Frequentes",
                    Icons.question_answer_outlined,
                    const PerguntasPredefinidasScreen(),
                  ),
                ],
              ); // GridView.count
            },
```

Figura 13 — Navegação e Composição do Dashboard

Fonte: elaborado pelo autor (2025).

Conforme demonstrado na Figura 13, o código da HomeScreen funciona como o mapa funcional e o roteador de navegação da aplicação. Ele demonstra como o dashboard centraliza e unifica o acesso a todas as funcionalidades-

chave do Betesaúde de forma organizada e modular. Este *dashboard* dá ao usuário acesso direto e organizado às ferramentas de Monitoramento (Registro/Histórico), Educação (Educação) e Orientação (Assistente Virtual). O *design* responsivo reforça diretamente o objetivo de acessibilidade digital e de criar uma solução multiplataforma robusta.

```dart
class ParabensScreen extends StatefulWidget {
  final String mensagem;
  const ParabensScreen({super.key, required this.mensagem});

  @override
  State<ParabensScreen> createState() => _ParabensScreenState();
}

class _ParabensScreenState extends State<ParabensScreen> {
  final FlutterTts flutterTts = FlutterTts();

  @override
  void initState() {
    super.initState();
    falarMensagem();
  }

  Future<void> falarMensagem() async {
    await flutterTts.setLanguage("pt-BR");
    await flutterTts.setSpeechRate(0.9);
    await flutterTts.speak(widget.mensagem);
  }

  @override
  void dispose() {
    flutterTts.stop();
    super.dispose();
  }
```

Figura 14 — Lógica da classe de parabenização

Fonte: elaborado pelo autor (2025).

O código demonstrado na Figura 14 define a tela de *feedback* positivo. A única função dessa tela é apresentar o resultado de forma amigável.

A classe ParabensScreen é estruturada tecnicamente como um *StatefulWidget* que recebe, por meio de seu construtor, a *String* contendo a mensagem de resultado positivo, a qual é originada na função de decisão avaliarRespostasDetalhada. A lógica de *feedback* auditivo (*Text-to-Speech* - TTS) é orquestrada principalmente no ciclo de vida inicial do componente, especificamente no método *initState*. Assim que a tela é construída, este método aciona automaticamente a função falarMensagem(), que utiliza o pacote flutter_tts para sintetizar e ler em voz alta a mensagem de congratulação, garantindo que o retorno ao usuário seja imediato e acessível.

```dart
class RecomendacaoScreen extends StatefulWidget {
  final String mensagem;
  const RecomendacaoScreen({super.key, required this.mensagem});

  @override
  State<RecomendacaoScreen> createState() => _RecomendacaoScreenState();
}

class _RecomendacaoScreenState extends State<RecomendacaoScreen> {
  final FlutterTts flutterTts = FlutterTts();

  @override
  void initState() {
    super.initState();
    falarMensagem();
  }

  Future<void> falarMensagem() async {
    await flutterTts.setLanguage("pt-BR");
    await flutterTts.setSpeechRate(0.9);
    await flutterTts.speak(widget.mensagem);
  }

  @override
  void dispose() {
    flutterTts.stop();
    super.dispose();
  }
```

Figura 15 — Lógica da classe de recomendação

Fonte: elaborado pelo autor (2025).

O código demonstrado na Figura 15 define a tela de *feedback* de alerta, sendo a contraparte da classe ParabensScreen. Sob uma perspectiva técnica, a estrutura dessa classe espelha a da classe ParabensScreen, sendo implementada como um *StatefulWidget* que recebe via construtor uma *String* contendo a mensagem de alerta gerada pela função de avaliação. O mecanismo de *feedback* auditivo opera de maneira análoga, invocando a função falarMensagem() dentro do método *initState* para vocalizar imediatamente a recomendação.

Esta funcionalidade, contudo, assume uma importância crítica no contexto da acessibilidade digital e segurança do paciente. Em episódios de hipoglicemia sintomática, nos quais o usuário pode sofrer com tonturas ou visão turva e estar impossibilitado de ler o texto na tela, a instrução sonora (como "Consuma glicose de ação rápida") atua como um recurso vital, constituindo um dos argumentos centrais de acessibilidade digital do projeto.

```
class _HistoricoGlicemiaScreenState extends State<HistoricoGlicemiaScreen> {
  Widget build(BuildContext context) {
    ), // AppBar
    body: _registros.isEmpty
        ? const Center(
            child: Text(
              "Nenhum registro encontrado",
              style: TextStyle(fontSize: 16, color: Colors.grey),
            ), // Text
          ) // Center
        : ListView.builder(
            itemCount: _registros.length,
            itemBuilder: (_, i) {
              final g = _registros[i];
              return Card(
                color: Colors.white,
                elevation: 3,
                margin:
                    const EdgeInsets.symmetric(horizontal: 12, vertical: 6),
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12)), // RoundedRectangleBorder
                child: ListTile(
                  leading:
                      const Icon(Icons.monitor_heart, color: Colors.teal),
                  title: Text(
                    '${g.valor.toStringAsFixed(1)} mg/dL',
                    style: const TextStyle(fontWeight: FontWeight.bold),
                  ), // Text
                  subtitle: Text(
                    '${g.horario}\n${g.refeicao}\nMedicação: ${g.medicacao.isNotEmpty ? g.medicacao : "-"}\nAlimentação: ${g.alime
                    style: const TextStyle(height: 1.4),
                  ), // Text
                  isThreeLine: true,
                ), // ListTile
              ); // Card
```

Figura 16 — Exibição do código da tela Histórico de Glicemia

Fonte: elaborado pelo autor (2025).

O código demonstrado na Figura 16 é o controlador de estado da tela de histórico. Sua responsabilidade é buscar os dados no banco de dados e disponibilizá-los para a interface visual.

1. List<Glicemia> _registros = [ ]; - Esta é a variável de estado principal. É uma lista vazia que irá armazenar todos os objetos Glicemia (os registros) depois que eles forem carregados do banco.
2. @override void initState() - Este método é um "gatilho" do Flutter, executado automaticamente uma vez quando a tela é aberta. A única coisa que ele faz é chamar a função carregar(), iniciando o processo de busca de dados.
3. Future<void> carregar() async - Esta é a função mais importante da tela. Ela é marcada como async (assíncrona) porque a leitura do banco de dados (SembastGlicemiaService) é uma operação que não tem resultado imediato (é uma operação de I/O).
    - final dados = await SembastGlicemiaService.listarGlicemias();:

      Esta é a linha de comunicação com a camada de persistência. O *await* pausa a execução dessa função até que o SembastGlicemiaService termine de consultar o banco e retorne a List<Glicemia>. Isso demonstra uma arquitetura limpa, onde a tela (UI) não sabe *como* os dados são salvos, ela apenas *pede* os dados ao serviço.
    - dados.sort((a, b) => b.horario.compareTo(a.horario));:

      Esta é uma linha crucial de lógica de negócio. Ela ordena a lista de registros em ordem cronológica decrescente (do mais novo para o mais antigo), garantindo que o usuário veja as medições mais recentes no topo da tela.
    - *setState(()* { registros = dados; });:

      Esta é a atualização de estado. A função *setState* informa ao Flutter que a variável _registros (que estava vazia) agora contém os dados carregados e ordenados. Isso faz com que o Flutter redesenhe a tela, e o *ListView.builder* (visualizado na análise anterior) passa a exibir os itens.
    - *try/catch*:

O bloco *try/catch* garante que, se houver qualquer falha durante a leitura do banco, o aplicativo não irá "quebrar", mas sim exibir uma mensagem de erro amigável *(SnackBar)* para o usuário.

3.3.6 Módulo de Emergência e Ligações Rápidas

Foi implementado um botão flutuante de ação rápida (*Floating Action Button*) na tela inicial, que ao ser acionado abre um Modal interativo. Este modal utiliza o pacote *url_launcher* do *Flutter* para acionar o discador nativo do dispositivo, permitindo realizar chamadas diretas e rápidas ao SAMU (192). Além disso, o recurso integra-se a uma instância dedicada do banco de dados local *Sembast*, permitindo que o paciente cadastre, edite e disque imediatamente para o número de telefone de um contato familiar, proporcionando uma rede de apoio acessível com apenas um toque em situações de crise de hipoglicemia ou hiperglicemia.

```
class _HomeScreenState extends State<HomeScreen> with TickerProviderStateMixin {
  void _abrirEmergencia() {
                ElevatedButton.icon(
                  onPressed: () => _ligarNumero('192'),
                  icon: const Icon(Icons.local_hospital),
                  label: const Text("LIGAR PARA SAMU (192)"),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.redAccent,
                    foregroundColor: Colors.white,
                    minimumSize: const Size(double.infinity, 50),
                    textStyle: const TextStyle(fontWeight: FontWeight.bold),
                  ),
                ), // ElevatedButton.icon
                const SizedBox(height: 20),
                const Divider(),
                const SizedBox(height: 10),
                const Text("Contato de Familiar Rápido",
                    style: TextStyle(fontWeight: FontWeight.bold)), // Text
                const SizedBox(height: 10),

                if (editando) ...[
                  TextField(
                    controller: nomeCtrl,
                    decoration: const InputDecoration(
                      labelText: "Nome do Familiar",
                      border: OutlineInputBorder(),
                      prefixIcon: Icon(Icons.person),
                    ), // InputDecoration
                  ), // TextField
                  const SizedBox(height: 10),
                  TextField(
                    controller: telCtrl,
                    keyboardType: TextInputType.phone,
                    decoration: const InputDecoration(
                      labelText: "Telefone",
                      border: OutlineInputBorder(),
```

Figura 17 — Lógica do modal de Emergência no Dashboard

Fonte: elaborado pelo autor (2025).

A estrutura lógica que viabiliza essa funcionalidade, integrando o acionamento do discador nativo e a interface do modal de configuração do banco de dados, pode ser observada na Figura 17.

4 DESENVOLVIMENTO E IMPLEMENTAÇÃO DO SISTEMA

Este capítulo documenta o processo de desenvolvimento e a implementação da aplicação mobile responsiva Betesaúde. A construção do sistema foi orientada à resolução dos problemas centrais identificados: a complexidade no monitoramento da glicemia e a carência de orientação confiável. Dessa forma, o texto aprofunda as decisões técnicas, justificando a adoção do Flutter para o desenvolvimento multiplataforma e a escolha do Sembast para a persistência de dados local. O foco da análise mostra sobre a implementação dos módulos de Monitoramento e Educação, e, de forma central, sobre a arquitetura do Assistente Virtual de triagem de sintomas, trazendo acessibilidade digital, o que constitui o principal diferencial inovador dessa proposta.

4.1 Fluxo de dados

Figura 18 — Tela de Segmentação de Perfil

Fonte: elaborado pelo autor (2025).

O fluxo de dados do Betesaúde inicia-se com a tela de segmentação de perfil, ilustrada na Figura 18. Esta é a primeira interface apresentada ao usuário e serve como um divisor estratégico para personalizar toda a sua jornada subsequente. Visualmente, a tela (Figura 18) apresenta a identidade do aplicativo (logo e título "Bem-vindo ao Betesaúde") e propõe uma questão-chave: "Você é recém-diagnosticado com diabetes ou já convive com a condição?". A interface é minimalista, com dois botões de ação claros que correspondem às respostas possíveis:

1. "Sou recém-diagnosticado"
2. "Já convivo com diabetes"

O objetivo dessa tela é direcionar o usuário ao caminho mais adequado, atendendo a uma das "dores" centrais do projeto: a complexidade de aplicativos genéricos para quem acabou de receber o diagnóstico.

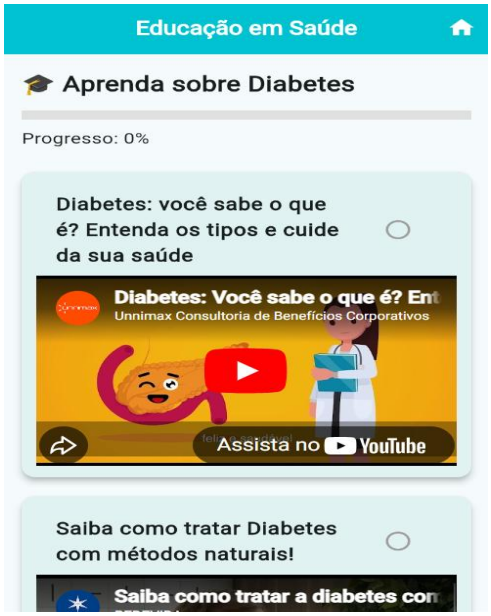

Figura 19 — Aprenda sobre Diabetes (Vídeos)

Fonte: elaborado pelo autor (2025).

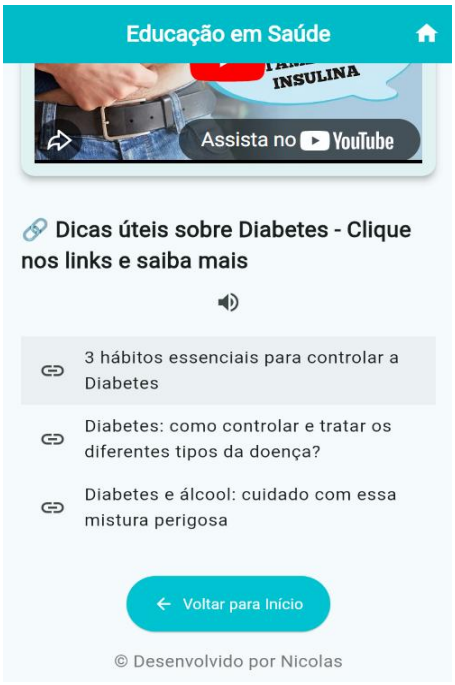

Figura 20 — Dicas úteis (Links)

Fonte: elaborado pelo autor (2025).

Como ilustrado nas Figuras 19 e 20, a tela é estruturada em duas seções principais, precedidas por uma barra de progresso:

1. Aprenda sobre Diabetes (Vídeos): Uma lista de vídeos educativos do YouTube é embutida diretamente na interface através de *iframes*. Cada vídeo possui um título e um botão para o usuário "marcar como assistido".
2. Dicas úteis (Links): Uma lista de links para artigos externos de fontes de autoridade em saúde (como Unimed, Einstein e *SBD*).

A tela inclui funcionalidades de acessibilidade digital, como um ícone de "ouvir" que utiliza *Text-to-Speech* (TTS) para ler os títulos dos artigos em voz alta.

4.1.1 Tela Principal (*Dashboard*)

A classe *HomeScreen* é o painel de controle central do Betesaúde. Conforme o fluxo de segmentação, esta é a tela principal para o usuário que seleciona a opção "Já convivo com diabetes", priorizando o acesso rápido às ferramentas de monitoramento. Essa tela também fica disponível para o recém-

diagnosticado após repasse pela tela da Figura 19. Análise Visual e Funcional (Classe HomeScreen):

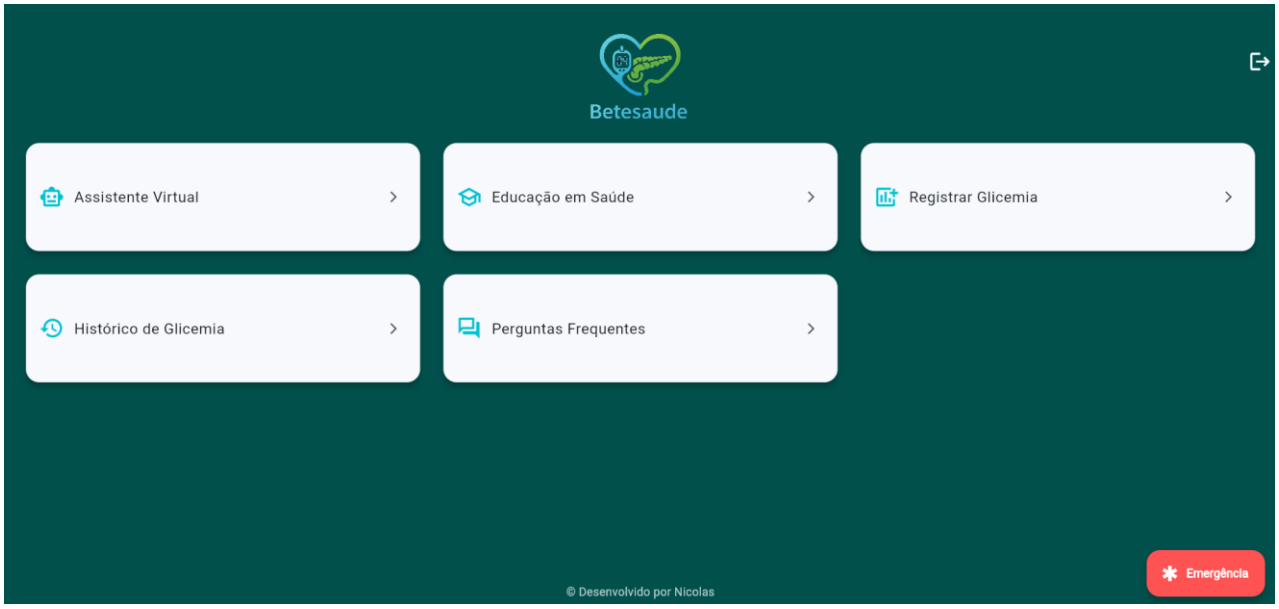

Figura 21 — Tela principal da aplicação Betesaude

Fonte: elaborado pelo autor (2025).

Visualmente, a tela (Figura 21) apresenta um *dashboard* limpo com a logo do Betesaúde e uma grade de botões de navegação. Cada botão representa um módulo principal do sistema:

- Assistente Virtual (Pilar de Orientação)
- Educação em Saúde (Pilar de Educação)
- Registrar Glicemia (Pilar de Monitoramento)
- Histórico de Glicemia (Pilar de Monitoramento)
- Perguntas Frequentes (Pilar de Educação)
- Botão de Emergência: Posicionado de forma flutuante e contínua no canto inferior direito da tela, oferece acesso rápido e direto para realizar ligações emergenciais para o SAMU e para contatos de apoio pré-cadastrados pelo usuário. (Pilar de Comunicação)

Esta tela demonstra a responsividade da aplicação, um requisito central do projeto. Em uma tela mais larga (como um *tablet* ou navegador web), a grade se adapta, exibindo os botões em múltiplas colunas (neste caso, três), em vez da coluna única de um smartphone.

A integração operacional deste módulo de emergência à tela principal é ilustrada nas Figuras 22 e 23. A Figura 22 demonstra o estado inicial do componente modal interativo, que se sobrepõe à interface principal após o acionamento do botão flutuante. O design dessa interface foi fundamentado em princípios de alerta visual, utilizando uma paleta de cores quentes (vermelho) no botão principal para destacar a ação de discagem imediata ao Serviço de Atendimento Móvel de Urgência (SAMU - 192), visando minimizar o tempo de resposta cognitivo do paciente em casos de crises severas.

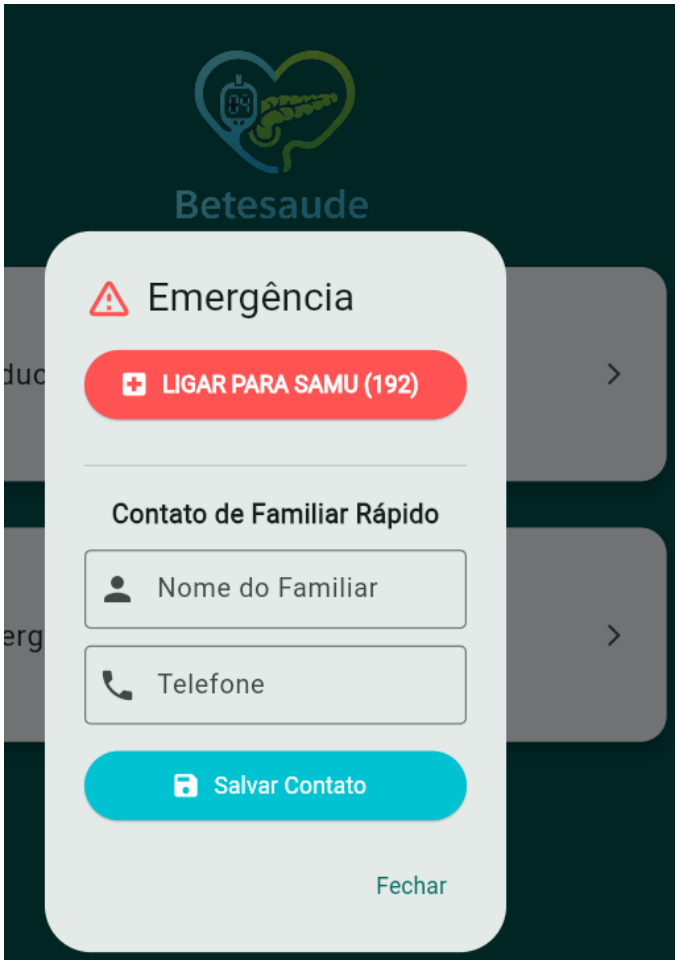

Figura 22 — Tela do botão emergência

Fonte: elaborado pelo autor (2025).

A Figura 23 apresenta a evolução do estado do modal após o usuário realizar o cadastramento de sua rede de apoio familiar. Observa-se que, devido à integração com a arquitetura offline-first do banco de dados Sembast, os campos de entrada de texto (nome e telefone) são ocultados e substituídos

dinamicamente por um botão de ação direta ('Ligar para Mãe', no exemplo). Essa abordagem de Interface do Usuário (UI) evita redundâncias e garante que, em um momento de hipoglicemia sintomática, onde a coordenação motora do usuário pode estar comprometida, a solicitação de socorro ocorra com um único toque na tela.

Figura 23 — Tela de emergência com contato cadastrado

Fonte: elaborado pelo autor (2025).

4.1.2 Fluxo de Orientação: Assistente Virtual

O pilar de Orientação é materializado pelo Assistente Virtual, uma funcionalidade inovadora de triagem de sintomas. O fluxo se inicia na AssistenteScreen (Figura 24), que atua como a interface de coleta de dados. Análise Visual e Funcional (Classe AssistenteScreen):

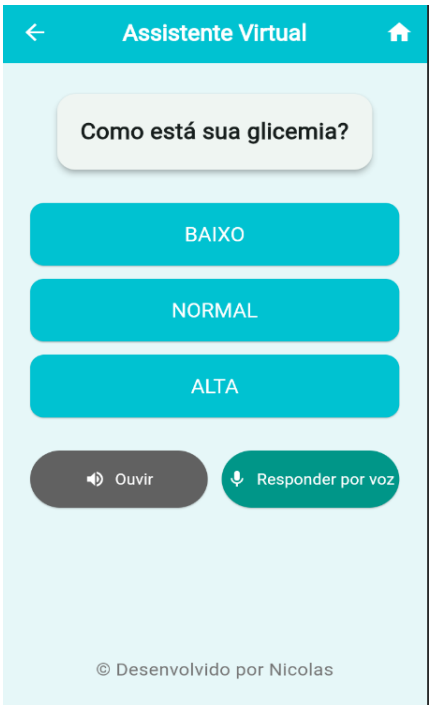

Figura 24 — Interface do Assistente Virtual

Fonte: elaborado pelo autor (2025).

Como apresentado na figura 24, a tela apresenta ao usuário uma interface de diálogo, passo a passo, começando com a pergunta "Como está sua glicemia?". O design foca na acessibilidade digital e na interação multimodal:

- Entrada por Toque: O usuário pode selecionar botões de resposta rápida ("BAIXO", "NORMAL", "ALTA").
- Entrada por Voz: O botão "Responder por voz" (ícone de microfone) ativa o pacote *speech_to_text* para capturar a resposta falada.
- Saída Auditiva: O botão "Ouvir" utiliza o flutter_tts para ler a pergunta em voz alta.

Após cada resposta, o assistente avança para a próxima pergunta (sintomas, alimentação, etc.) até coletar todas as informações necessárias.

4.1.3 Telas de Resultado: RecomendacaoScreen e ParabensScreen

Após o usuário responder a todas as perguntas da triagem, o Assistente Virtual entra na fase de Processamento e Saída. A função avançar chama o "motor de decisão", a função avaliar(), que processa as respostas coletadas. Com base no resultado dessa avaliação, o sistema utiliza uma navegação condicional para exibir uma das duas telas de resultado:

Figura 25 — RecomendacaoScreen

Fonte: elaborado pelo autor (2025).

Se a avaliação identificar um risco (ex: glicemia alta com sintomas), a tela de alerta é exibida (Figura 25). Ela utiliza uma paleta de cores vermelha e um ícone de aviso para comunicar urgência e fornecer uma orientação acionável (ex: "Procure orientação médica.").

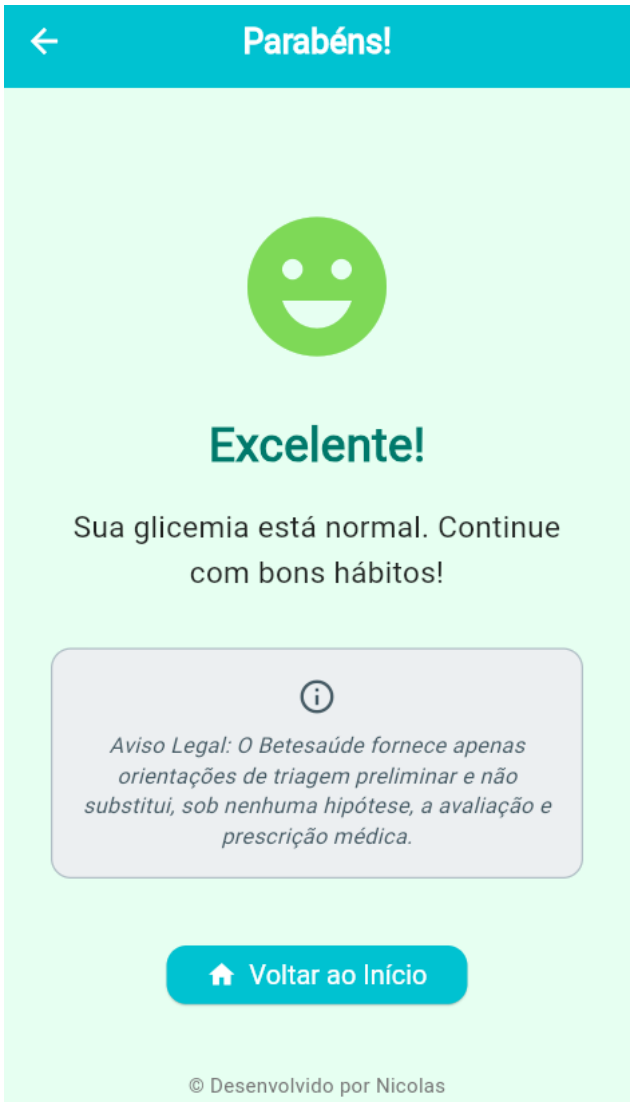

Figura 26 — Tela ParabensScreen
Fonte: elaborado pelo autor (2025).

Se a avaliação for positiva (ex: glicemia normal), a tela de parabéns é exibida (Figura 26). Ela usa cores verdes e um ícone de reforço positivo para incentivar o usuário a manter os bons hábitos.

4.1.4 Tela de Registro de Glicemia

Esta tela é a principal ferramenta de entrada de dados do pilar de Monitoramento, permitindo ao usuário registrar seus níveis de glicose atrelados aos seus hábitos alimentares e medicamentosos do momento. Análise Visual e Funcional (RegistroGlicemiaScreen):

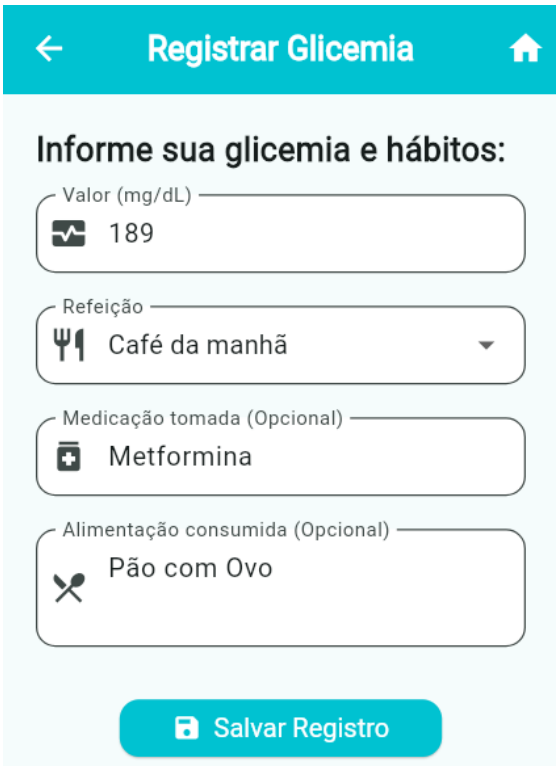

Figura 27 — Tela de registro de Glicemia

Fonte: elaborado pelo autor (2025).

A interface demonstrada na Figura 27 é um formulário simples e objetivo, projetado para agilizar o processo de registro:

1. Campo "Valor (mg/dL)": Um *TextFormField* que aceita a entrada numérica da medição de glicemia.
2. Campo "Refeição": Um *DropdownButtonFormField* que, quando tocado, permite ao usuário classificar a medição (ex: "Café da manhã", "Almoço", "Jantar", "Outro").
3. Botão "Salvar Registro": O *ElevatedButton* que dispara a lógica de processamento e salvamento dos dados.

4.1.5 Tela de Histórico de Glicemia

A tela de Histórico de Glicemia, apresentada na Figura 28, atua como o diário digital do paciente, exibindo todos os registros de glicemia e as interações do assistente que foram salvas no banco de dados local.

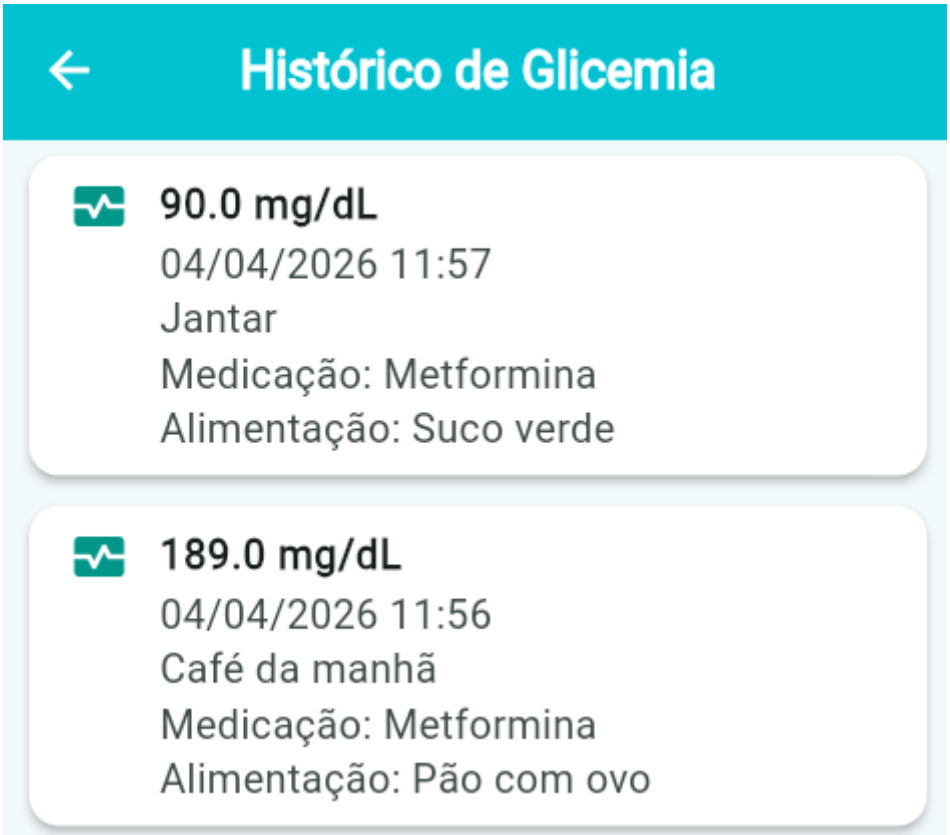

Figura 28 — Exibição do Histórico de Glicemia

Fonte: elaborado pelo autor (2025)..

A interface demonstrada na Figura 28, consiste em uma lista cronológica de registros. Cada item é apresentado em um Card individual para facilitar a leitura. Cada card exibe de forma clara as quatro informações essenciais do modelo Glicemia:

1. **Valor**: O nível de glicemia medido (ex: "520.0 mg/dL").
2. **Horário**: A data e hora exata do registro.
3. **Contexto**: A origem do registro, seja uma refeição (ex: "Café da manhã") ou uma interação (ex: "Assistente Virtual").
4. **Hábitos**: A medicação e alimentação fornecidas pelo usuário.

Como demonstrado na imagem, os registros são ordenados do mais recente para o mais antigo, permitindo ao usuário uma visualização imediata de suas últimas medições.

4.1.6 Tela de Perguntas Frequentes

A tela Perguntas Frequentes, apresentada na Figura 29, funciona como um assistente de FAQ interativo, permitindo ao usuário sanar dúvidas comuns sobre o diabetes através de texto ou voz e recebendo respostas imediatas e curadas.

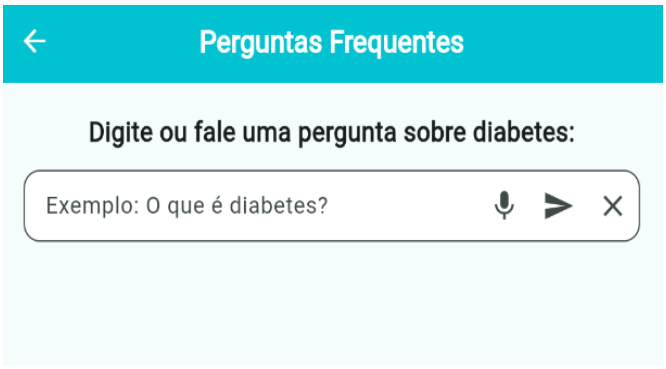

Figura 29 – Tela de Perguntas Frequentes

Fonte: elaborado pelo autor (2025).

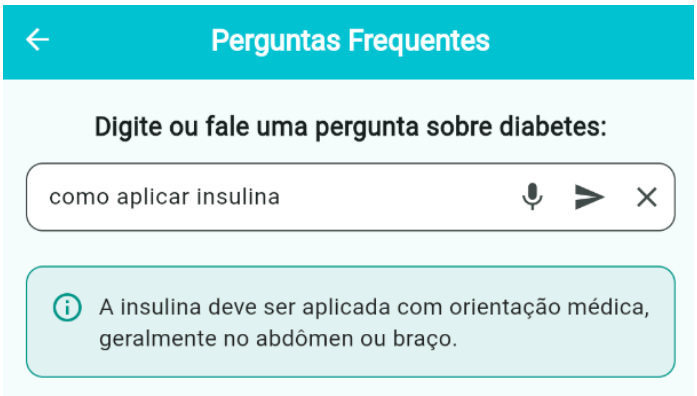

Figura 30 — Tela de Perguntas Frequentes com resposta

Fonte: elaborado pelo autor (2025).

A interface é centrada em um campo de entrada de texto (*TextField*), como visto na Figura 29 (estado inicial) e Figura 30 (estado com resposta). A tela conta com uma entrada Multimodal onde o usuário pode:

1. Digitar a pergunta (ex: "como aplicar insulina").
2. Falar a pergunta, usando o ícone de microfone.

A tela apresenta resposta Dinâmica, como ilustrado na Figura 30, após a pergunta ser enviada, a resposta correspondente aparece dinamicamente na tela dentro de um Card de *feedback*, sem que o usuário precise sair da interface.

5 CONCLUSÃO

O presente trabalho tem como objetivo central projetar e desenvolver o Betesaúde, uma aplicação móvel responsiva focada no suporte, monitoramento e educação de pacientes com diabetes *mellitus*. Diante do cenário crescente de melhorar a desordem informacional e da alta complexidade técnica observada nas ferramentas disponíveis no mercado, a proposta buscou entregar uma solução gratuita, acessível e centrada no usuário, preenchendo uma lacuna significativa na jornada do paciente, especialmente para indivíduos recém-diagnosticados.

A adoção do *framework* Flutter demonstrou-se uma escolha tecnológica altamente acertada. Sua utilização permitiu a consolidação de uma interface fluida, responsiva e consistente para múltiplas plataformas a partir de uma base de código única, otimizando o tempo de desenvolvimento. No âmbito da engenharia de *software*, a integração do banco de dados NoSQL embarcado Sembast garantiu a resiliência do sistema por meio de uma arquitetura *offline-first*. Essa decisão técnica assegurou que o paciente tenha acesso ininterrupto ao seu histórico e possa realizar o registro de sua glicemia atrelado aos seus hábitos, como a medicação tomada e a alimentação consumida, mesmo em cenários sem conexão com a internet.

O principal diferencial e ponto de inovação alcançado por este projeto foi a implementação do Assistente Virtual para triagem de sintomas. A construção de um motor de decisão baseado em regras clínicas, aliado a uma interface multimodal com integração de síntese e reconhecimento de voz (*Text-to-Speech* e *Speech-to-Text*), atendeu plenamente aos requisitos de acessibilidade digital estipulados. Essa funcionalidade comprovou-se viável e indispensável para melhorar a segurança do usuário em momentos de crise, como em episódios de hipoglicemia sintomática, cenários em que a interação tátil ou visual com o dispositivo pode estar severamente comprometida.

Contudo, o desenvolvimento de *software* aplicado à saúde exige validação e aprimoramento contínuos. Reconhece-se que, em sua versão atual, o motor de triagem do assistente possui limitações inerentes a uma árvore de decisão estática, fornecendo uma orientação primária que não substitui, sob

nenhuma hipótese, a avaliação médica profissional. Além disso, o foco exclusivo na persistência local, embora resolva a questão da disponibilidade *offline*, restringe temporariamente o compartilhamento dos dados consolidados com a equipe de saúde ou familiares responsáveis pelo paciente.

Diante disso, como propostas para trabalhos futuros, sugere-se a implementação de um módulo de sincronização de dados em nuvem (*cloud computing*), permitindo a geração de relatórios exportáveis para facilitar as consultas médicas. Recomenda-se também a futura integração do aplicativo com dispositivos vestíveis (*wearables*) e biossensores de monitoramento contínuo de glicose (CGM) via *Bluetooth*. Adicionalmente, a evolução do motor de busca da seção educativa para uma arquitetura baseada em Inteligência Artificial Generativa e a expansão dos recursos de gamificação poderiam maximizar ainda mais o engajamento a longo prazo.

Conclui-se, portanto, que o Betesaúde cumpre seu propósito fundamental de democratizar o acesso à informação segura e de simplificar o autocuidado diário. A execução desse projeto não apenas consolida os conhecimentos técnicos em desenvolvimento mobile e arquitetura de sistemas adquiridos ao longo da formação no curso de Tecnologia em Sistemas para a Internet, mas também entrega um artefato computacional com real potencial de impacto social.

O sistema apresenta-se como um passo promissor para transformar a complexidade imposta pelo tratamento do diabetes em uma jornada mais assistida, autônoma e acolhedora.

REFERÊNCIAS

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE). Censo Demográfico 2022: Primeiros resultados de População e Domicílios. Rio de Janeiro**: IBGE, 2023.**

Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv102011.pdf >. Acesso em 19 set. 2025.

SOCIEDADE BRASILEIRA DE DIABETES (SBD). [Brasil já tem cerca de 20 milhões de pessoas com diabetes, publicado em 2025]. São Paulo: **SBD, 2025.**

Disponível em: <https://diabetes.org.br/brasil-ja-tem-cerca-de-20-milhoes-de-pessoas-com-diabetes/>. Acesso em 19 set. 2025.

GLIC. **Glic**: controle do diabetes. Glic, 2024. Aplicativo de software. Disponível em: <https://apps.apple.com/br/app/glic-diabetes-e-glicemia/id1184941726 e https://play.google.com/store/apps/details?id=br.com.quasar.gliconline&hl=pt_BR>. Acesso em: 3 out. 2025.

ROCHE DIABETES CARE GMBH. **mySugr**: Diário de Diabetes. Google Play Store, Versão 3.98.31, 2024. Aplicativo de software. Disponível em: <https://apps.apple.com/br/app/mysugr-di%C3%A1rio-da-diabetes/id516509211>

e

<https://play.google.com/store/apps/details?id=com.mysugr.android.companion&hl=pt_BR>. Acesso em: 7 out. 2025.

UNIVERSIDADE FEDERAL FLUMINENSE (UFF). **Estudo da UFF revela baixa qualidade e desordem informacional na internet sobre tratamento farmacológico do diabetes mellitus**. UFF, 30 jun. 2025. Disponível em: <https://www.uff.br/30-06-2025/estudo-da-uff-revela-baixa-qualidade-e-desordem-informacional-na-internet-sobre-tratamento-farmacologico-do-diabetes-mellitus/>. Acesso em: 9 out. 2025.

SUPLICI, S. E. R. et al. Adesão ao autocuidado de pessoas com Diabetes Mellitus na Atenção Primária: estudo de método misto. **Escola Anna Nery Revista de Enfermagem**, Florianópolis, v. 25, n. 5, e20210032, 2021. Disponível em: <https://www.scielo.br/j/ean/a/jF5QntVTdRBWTNcVfJ7hpGH/?format=pdf&lang=pt>. Acesso em: 13 out. 2025.

LOPES, R. O. P. et al. Benchmarking de aplicações móveis para a saúde de pessoas com Diabetes Mellitus. **Revista Latino-Americana de Enfermagem**, Ribeirão Preto, v. 32, e4028, 2024. Disponível em: <https://www.scielo.br/j/rlae/a/9vQjtDZ4gxFZSF64rDpwnMf/>. Acesso em: 13 out. 2025.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson Education do Brasil, 2019.

BECK, Kent et al. **Manifesto para Desenvolvimento Ágil de Software**. 2001. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 25 out. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. **O Guia do Scrum: O Guia Definitivo para o Scrum: As Regras do Jogo**. 2020. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-Brazilian.pdf>. Acesso em: 26 out. 2025.

KANBAN UNIVERSITY. **O Guia Oficial do Método Kanban**. 2021. Disponível em: <https://kanban.university/wp-content/uploads/2021/04/The-Official-Kanban-Guide_Portuguese_A4.pdf>. Acesso em: 26 out. 2025.

AHMAD, Farhan; JOSHI, Shiv H. **Práticas de autocuidado e seu papel no monitoramento do diabetes: uma revisão narrativa**. Jul. 2023. Disponível em: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10402910/>. Acesso em: 09 fev. 2026.

CONSELHO FEDERAL DE ENFERMAGEM (COFEN). MHealth: Definição, Interesses, Desafios e Futuro. **Biblioteca Virtual de Enfermagem**, Brasília, DF, 2022. Disponível em: <https://biblioteca.cofen.gov.br/mhealth-definicao-interesses-desafios-futuro/>. Acesso em: 16 fev. 2026.

MICROSOFT. Interface e entrada do usuário - Windows apps. **Microsoft Learn**, 2026. Disponível em: <https://learn.microsoft.com/pt-br/windows/apps/develop/user-interface?tabs=winui-3>. Acesso em: 16 fev. 2026.

FLUTTER. Flutter Docs **Amazon Web Services**, 2026. Disponível em: <https://docs.flutter.dev>. Acesso em: 16 fev. 2026.

PREECE, J.; ROGERS, Y.; SHARP, H. *Design de Interação: além da interação humano-computador*. 3. ed. Porto Alegre: Bookman, 2013. Disponível em: <https://pt.scribd.com/document/364346592/Design-de-Interacao-Alem-Da-Interacao-Humano-Computador>. Acesso em: 16 fev. 2026.

SOCIEDADE BRASILEIRA DE DIABETES (SBD). *Diretrizes da Sociedade Brasileira de Diabetes 2023-2024*. São Paulo: SBD, 2023. Disponível em: <https://diretrizes.diabetes.org.br/>. Acesso em: 16 fev. 2026.

W3C (World Wide Web Consortium). *Diretrizes de Acessibilidade digital para Conteúdo Web (WCAG) 2.1*. 2018. Disponível em: <https://www.w3.org/TR/WCAG21/>. Acesso em: 16 fev. 2026.

FOWLER, M.; SADALAGE, P. J. *NoSQL Essencial: Um guia conciso para o mundo emergente da persistência poliglota. São Paulo: Novatec, 2013.* Disponível em: <https://pt.scribd.com/document/735960683/NoSQL-Essencial>. Acesso em: 16 fev. 2026.

SANTOS, V. C. Online first vs Offline first no Flutter: como usar e quando escolher. Alura, São Paulo, 12 jan. 2025. Disponível em: <https://www.alura.com.br/artigos/online-offline-first>. Acesso em: 16 fev. 2026.

TEKARTIK. *Sembast: Simple Embedded Application Store database*. Documentação Oficial. Pub.dev, 2023. Disponível em: <https://pub.dev/packages/sembast>. Acesso em: 16 fev. 2026.

MINISTÉRIO DA SAÚDE (Brasil). Diabetes. **Biblioteca Virtual em Saúde**, Brasília, DF, 2009. Disponível em: <https://bvsms.saude.gov.br/bvs/dicas/67diabetes.html>. Acesso em: 21 fev. 2026.

INTERNATIONAL DIABETES FEDERATION (IDF). *IDF Diabetes Atlas*. 10. ed. Bruxelas, Bélgica: International Diabetes Federation, 2021. Disponível em: <https://diabetesatlas.org/>. Acesso em: 24 fev. 2026.

NATIONAL INSTITUTES OF HEALTH (NIH). *Diabetes Management: How Lifestyle, Daily Routine Affect Blood Sugar*. National Library of Medicine, Bethesda, MD, 2022. Disponível em: <https://www.medlineplus.gov/diabetes.html>. Acesso em: 24 fev. 2026.

ORGANIZAÇÃO MUNDIAL DA SAÚDE (OMS). *Diabetes*. Genebra: World Health Organization, 2023. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/diabetes>. Acesso em: 24 fev. 2026.