INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PIRIPIRI
CURSO TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

FRANCISNILTO DOS SANTOS NASCIMENTO

CLINCARE: Desenvolvimento de um Assistente Conversacional Inteligente para Apoio ao Acompanhamento Domiciliar de Idosos em Vulnerabilidade Social por Agentes Comunitários de Saúde

PIRIPIRI
2026

FRANCISNILTO DOS SANTOS NASCIMENTO

CLINCARE: Desenvolvimento de um Assistente Conversacional Inteligente para Apoio ao Acompanhamento Domiciliar de Idosos em Vulnerabilidade Social por Agentes Comunitários de Saúde

Trabalho de Conclusão de Curso (relatório técnico de *software*) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Piripiri .

Orientador: Prof. Dr. Ricardo Moura Sekeff Budaruiche.

PIRIPIRI

2026

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

NASCIMENTO, Francisnilto dos Santos

N244c Clincare : desenvolvimento de um assistente conversacional inteligente para apoio ao acompanhamento domiciliar de idosos em vulnerabilidade social por agentes comunitários de saúde / Francisnilto dos Santos NASCIMENTO. - 2026.
79 f.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Piripiri, 2026.
Orientador : Prof Dr. Ricardo Moura Sekeff Budaruiche.

1. Inteligência artificial. 2. Agente de saúde. 3. Assistente conversacional. I. Título.

CDD - 004.21

Elaborado por Júlio César Oliveira Rodrigues CRB CRB 3/1125

FRANCISNILTO DOS SANTOS NASCIMENTO

CLINCARE: Desenvolvimento de um Assistente Conversacional Inteligente para Apoio ao Acompanhamento Domiciliar de Idosos em Vulnerabilidade Social por Agentes Comunitários de Saúde

Trabalho de Conclusão de Curso (relatório técnico de *software*) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Piripiri.

Aprovada em: 14/07/2026.

BANCA EXAMINADORA

_____
Prof. Dr. Ricardo Moura Sekeff Budaruiche(Orientador)
Instituto Federal do Piauí (IFPI)

_____
Prof. Me. Jeferson do Nascimento Soares
Instituto Federal do Piauí (IFPI)

_____
Prof. Esp. Marcos Ramon Paulino Resende
Instituto Federal do Piauí (IFPI)

Dedico este trabalho à minha avó paterna
e aos meus amigos e mestres.

AGRADECIMENTOS

Agradeço a todos os professores e servidores do IFPI do Campus Piripiri, pois todos direto e indiretamente contribuíram para o alcance desse resultado.

Agradeço, também, a minha mãe e avó que deram todo o apoio necessário para que eu chegasse até aqui. Agradeço ao meu Deus, que me deu força quando mais precisava. Agradeço à minha psicóloga, que me ajudou no controle de minhas emoções em momentos turbulentos e com o planejamento do cronograma de atividades deste trabalho.

O que prevemos raramente ocorre; o que menos esperamos geralmente acontece.

Benjamin Disraeli

RESUMO

Este trabalho apresenta o desenvolvimento de um relatório técnico de *software* voltado para o fortalecimento da Atenção Primária à Saúde no contexto do Sistema Único de Saúde. O projeto propõe uma plataforma *web* inteligente para auxiliar o trabalho dos Agentes Comunitários de Saúde e o acompanhamento de pacientes idosos em situação de vulnerabilidade social. A solução integra um agente conversacional por voz baseado em modelos de linguagem de grande porte, especificamente o *Llama* 3.2 3B, utilizando a técnica de geração aumentada por recuperação embasada em protocolos oficiais do Sistema Único de Saúde. Entre as funcionalidades desenvolvidas, destacam-se a triagem inteligente com níveis de prioridade no atendimento domiciliar, a classificação automática de intenções, a gestão da carteira de idosos para o agente de saúde e um fluxo de cadastro assistido para mitigar a exclusão digital de idosos. Os resultados demonstram a viabilidade técnica da implementação de inteligência artificial em contextos de recursos computacionais restritos, com tempos de processamento médios de 6,75 segundos, proporcionando um mecanismo sistematizado para a identificação contínua de riscos e a melhoria da comunicação entre o cidadão e o serviço de saúde

Palavras-chave: inteligência artificial; agente de saúde; idosos; assistente conversacional; triagem inteligente; acessibilidade.

ABSTRACT

This work presents the development of a software technical report aimed at strengthening Primary Health Care within the context of the Unified Health System. The project proposes an intelligent web platform to assist Community Health Workers and the monitoring of elderly patients in situations of social vulnerability. The solution integrates a voice-based conversational agent powered by large language models, specifically Llama 3.2 3B, using retrieval-augmented generation grounded in official Unified Health System protocols. Among the developed features, key highlights include intelligent triage with priority levels for home care visits, automatic intent classification, management of the elderly patient roster for the health worker, and an assisted registration workflow to mitigate digital exclusion among the elderly. The results demonstrate the technical feasibility of deploying artificial intelligence in resource-constrained environments, with average processing times of 6.75 seconds, providing a systematized mechanism for continuous risk identification and the improvement of communication between citizens and the health service.

Keywords: artificial intelligence; community health worker; elderly; conversational assistant; intelligent triage; accessibility.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACS	Agentes Comunitários de Saúde
API	Application Programming Interface
APK	Android APplication Package
APS	Atenção Primária à Saúde
BIT	Binary Digit
CBO	Código Brasileiro de Ocupações
CEP	Código de Endereçamento Postal
CNES	Cadastro Nacional de Estabelecimentos de Saúde
CPF	Cadastro de Pessoas Físicas
CPU	Central Processing Unit
CSV	Comma-Separated Values
CSS	Cascading Style Sheets
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network Library
DBML	Database Markup Language
e-SUS PEC	e-SUS do Prontuário Eletrônico do Cidadão
ER	Entidade-Relacionamento
FAB	Floating Action Button
GB	Gigabytes
GPU	Graphics Processing Unit
HCT	Hue, Chroma and Tone
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
IP	Internet Protocol
JSON	JavaScript Object Notation
MD3	Material Design 3
MVP	Minimum Viable Product
ORM	Object-Relational Mapping
PDF	Portable Document Format
POO	Programação Orientada a Objetos
RAG	Retrieval-Augmented Generation
REST	Representational State Transfer
RF	Requisito Funcional
RNF	Requisito Não Funcional
SDK	Software Development Kit
SIMET	Sistema de Informações do Ministério da Saúde
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSR	Server-Side Rendering
SUS	Sistema Único de Saúde
TTS	Text-to-Speech
TTL	Time to Live

UBS	Unidade Básica de Saúde
UID	User Identifier
UML	Unified Modeling Language
VRAM	Video Random Access Memory
XML	eXtensible Markup Language

LISTA DE SÍMBOLOS

≥ Maior ou igual a

% Porcentagem

SUMÁRIO

1 INTRODUÇÃO

O principal nível de organização do cuidado no Sistema Único de Saúde (SUS) está relacionado à garantia do direito constitucional à saúde por meio da Atenção Primária à Saúde (APS). Nesse contexto, o SUS é responsável pela coordenação dos processos de gestão, vigilância, preservação e acompanhamento contínuo da população, tanto no âmbito das Unidades Básicas de Saúde (UBS) quanto em domicílio (Lima *et al.*, 2021). Todavia, as mudanças demográficas, como o envelhecimento populacional e o aumento das doenças crônicas, associadas às condições de vulnerabilidade social, econômica e ambiental, vêm acarretando impasses gerenciais para o atendimento das crescentes demandas assistenciais em saúde (Giacomin, 2021).

Os Agentes Comunitários de Saúde (ACS) desempenham papel fundamental na APS, sendo responsáveis pelo acompanhamento contínuo da população adscrita e pela realização de visitas domiciliares. De acordo com Genaro *et al.*, (2025), observa-se que a atuação dos ACS junto às famílias brasileiras em situação de vulnerabilidade social enfrenta limitações estruturais. Entre os principais desafios, destacam-se a escassez de tecnologias de comunicação entre cidadãos de baixa renda e os ACS, bem como as dificuldades logísticas relacionadas ao acompanhamento dessas populações (Genaro *et al.*, 2025). Além disso, a ausência de mecanismos sistematizados para a identificação contínua de indivíduos em situação de risco compromete a efetividade do acompanhamento domiciliar (Torres; Wermelinger; Ferreira, 2025).

Paralelamente, observa-se um crescente impulso ao desenvolvimento de soluções tecnológicas baseadas em Inteligência Artificial (IA) voltadas ao setor da saúde, com aplicações na estratificação de riscos, no apoio à decisão clínica e no monitoramento remoto de idosos com restrições de mobilidade (Alves *et al.*, 2025; Lin; Wai; Chang Chien, 2026). Nesse contexto, estudos evidenciam o potencial da IA na APS, especialmente quando desenvolvida para atender às demandas dos cenários reais de atuação dos profissionais de saúde e às características socioeconômicas e culturais das populações assistidas (Lopes Júnior *et al.*, 2025; Resende *et al.*, 2025).

Dessa forma, a incorporação da IA nessas aplicações de saúde pode contribuir para o aumento da precisão dos processos de estratificação de risco, para

o fortalecimento da tomada de decisão clínica e para uma organização mais eficiente da demanda assistencial dos ACS (Khan; Sherani, 2024). Contudo, a efetividade dessas soluções não depende apenas do desempenho dos modelos de IA, mas também de sua capacidade de atender às necessidades dos usuários e de se integrar aos fluxos reais de trabalho dos serviços de saúde (Williams *et al.*, 2021).

Além disso, o desenvolvimento de soluções tecnológicas para a saúde exige que aspectos como representatividade e acessibilidade sejam considerados desde as etapas de concepção da aplicação. Embora os avanços em IA tenham ampliado as possibilidades de inovação na área da saúde, Rajan (2025) e Hackers (2024) demonstram que grande parte dessas soluções permanecem concentradas em países desenvolvidos, sendo minimamente adaptadas às realidades socioeconômicas e culturais dos países em desenvolvimento. Como consequência, populações em situação de vulnerabilidade frequentemente permanecem excluídas dessas iniciativas ou dispõem de ferramentas com baixa aderência aos fluxos reais de trabalho da APS.

Em concordância com Rajan (2025) e Hackers (2024), Shams-Ghahfarokhi (2025) ainda pontua que a inadequação tecnológica às necessidades de acessibilidade de usuários idosos com baixa alfabetização e restrições de mobilidade configura um obstáculo significativo à inclusão digital. Fica claro, pois, que a representatividade está relacionada ao reconhecimento das características sociais e das necessidades desse público. Por outro lado, a acessibilidade refere-se à implementação efetiva de recursos capazes de garantir autonomia na utilização das tecnologias.

O desenvolvimento de novas soluções deve contemplar simultaneamente essas duas perspectivas, uma vez que a ausência de acessibilidade reduz a representatividade a uma ação meramente simbólica. Consequentemente, a representatividade, quando dissociada da acessibilidade, torna-se incapaz de promover uma inclusão efetiva e de minimizar as estruturas de exclusão social que afetam essas populações vulneráveis. Em outras palavras, sem a viabilidade técnica da acessibilidade, a representatividade converte-se em uma estratégia de visibilidade sem emancipação, preservando as estruturas de exclusão sob uma aparência superficial de diversidade (Burci, 2017; Sousa; Guimarães, 2025).

Uma das alternativas mais promissoras para ampliar o acesso aos serviços de saúde, especialmente entre populações periféricas, consiste na implementação de

interfaces conversacionais baseadas em modelos de *Machine Learning*. Conforme a literatura, assistentes conversacionais inteligentes proporcionam interações mais intuitivas, reduzindo barreiras de acessibilidade decorrentes de baixos níveis de alfabetismo, baixa visão ou cegueira. Dessa forma, essas tecnologias apresentam potencial para promover maior autonomia no autocuidado, além de viabilizar o monitoramento contínuo de idosos em ambiente domiciliar por familiares e profissionais de saúde (Abadir, 2025).

Além do mais, as tecnologias de IA, quando integradas ao trabalho dos ACS, apresentam elevado potencial para fortalecer o modelo de atuação territorializada da APS. Por meio dessas ferramentas inteligentes, são viabilizados mecanismos de apoio à coleta estruturada de dados, à identificação ágil de riscos e ao acompanhamento longitudinal de idosos em situação de vulnerabilidade socioeconômica e ambiental (Rodrigues *et al.*, 2022).

Paralelamente, destaca-se a relevância dos documentos de orientação à APS e demais manuais técnicos oficiais disponibilizados pelo Ministério da Saúde para subsidiar as ações desenvolvidas na APS. Esses documentos reúnem orientações baseadas em evidências científicas voltadas à promoção da saúde, prevenção de agravos, acompanhamento de condições clínicas e apoio à tomada de decisão pelos profissionais da APS.

Considerando sua importância para a prática assistencial, esses materiais foram empregados neste trabalho como base de conhecimento do agente inteligente, sendo previamente indexados por meio da técnica de *Retrieval-Augmented Generation* (*RAG*). Dessa forma, as respostas produzidas pela IA passam a ser fundamentadas nas recomendações contidas nesses documentos oficiais, contribuindo para reduzir respostas sem embasamento e aumentar a confiabilidade das orientações fornecidas pelo sistema.

Diante das limitações observadas no modelo tradicional de acompanhamento domiciliar de pessoas idosas analfabetas e com restrições de mobilidade até uma UBS, evidencia-se a necessidade de soluções tecnológicas mais democráticas e inclusivas. Nesse contexto, propõe-se o desenvolvimento de um *Minimum Viable Product* (*MVP*) para a *web* e escalável para dispositivos móveis, voltado ao fortalecimento das ações da APS, sem substituir o papel desempenhado pelos ACS.

Por conseguinte, este *MVP* tem como finalidade conectar idosos analfabetos que enfrentam restrições locomotivas para deslocamento até uma UBS aos seus

respectivos ACS, integrando dashboards exclusivos para ambos os perfis de usuários e funcionalidades essenciais ao acompanhamento assistencial. Para isso, a solução incorpora recursos como um assistente inteligente por voz e identificação do nível de risco em saúde voltada à triagem inteligente.

Além disso, a proposta fundamenta-se no conceito de *Human-in-the-Loop* (Humano no Ciclo), segundo o qual a tecnologia atua como mecanismo de suporte às decisões dos profissionais de saúde, preservando o protagonismo da análise e da tomada de decisão humana no processo assistencial (Rodrigues *et al.*, 2022). Desse modo, os recursos de IA são empregados para ampliar a capacidade de monitoramento, análise e priorização de atendimentos, promovendo uma integração equilibrada entre inovação tecnológica, eficiência operacional e cuidado humanizado (Rodrigues *et al.*, 2022).

A relevância deste trabalho fundamenta-se em três dimensões complementares: social, institucional e científica. Sob a perspectiva social, a proposta busca reduzir barreiras de acesso à informação e às orientações básicas em saúde para populações vulneráveis, ampliando sua inclusão nos serviços de APS (Abisha *et al.*, 2024; Catapan *et al.*, 2024; Rodrigues *et al.*, 2022).

No âmbito institucional, a solução visa apoiar o trabalho dos ACS por meio da otimização de processos de acompanhamento e gestão da APS, contribuindo para a eficiência das atividades assistenciais realizadas em domicílio (Lamas *et al.*, 2025; Santos *et al.*, 2025; Rodrigues *et al.*, 2022). Sob a perspectiva científica, pretende-se contribuir para a geração de evidências sobre a aplicação de IA em contextos reais da saúde, especialmente em cenários marcados por restrições socioeconômicas e limitações de acesso aos serviços de cuidado (Blondino *et al.*, 2024; Martins, 2024; Rodrigues *et al.*, 2022).

Diante desse contexto, o presente trabalho busca responder à seguinte questão de pesquisa: De que maneira um sistema inteligente integrado a uma IA generativa com arquitetura conversacional pode contribuir para o acompanhamento contínuo e a priorização do atendimento domiciliar de idosos com baixa alfabetização e restrições de mobilidade no contexto da Atenção Primária à Saúde?

1.1 OBJETIVOS

Esta seção detalha o propósito fundamental desta pesquisa, estabelecendo a

meta central a ser alcançada pelo desenvolvimento do software proposto, bem como os passos técnicos e operacionais necessários para a concretização desse projeto.

1.1.1 GERAL

Desenvolver um sistema inteligente *web* responsivo, integrado a uma IA generativa com arquitetura conversacional e funcionalidades de triagem inteligente, capaz de auxiliar na priorização do atendimento domiciliar, contribuindo para o fortalecimento da assistência remota e do monitoramento preventivo de idosos com restrições de mobilidade no âmbito da APS e das UBS.

1.1.2 ESPECÍFICOS

Para viabilizar a execução do objetivo geral, foram traçadas as seguintes metas específicas, que delineiam as etapas de modelagem, desenvolvimento e validação do sistema:

- Desenvolver um sistema *web* que realize o cadastro, a autenticação e o gerenciamento de usuários, contemplando os perfis de idosos e ACS, incluindo mecanismos de recuperação de senha, verificação de identidade e controle seguro de acesso.

- Integrar ao sistema um agente de IA generativa com arquitetura conversacional capaz de interagir com o usuário idoso, promovendo acessibilidade e facilitando a comunicação com indivíduos com baixa alfabetização ou limitações funcionais.

- Implementar a funcionalidade para o registro estruturado das informações de saúde obtidas durante as interações entre o idoso e o agente inteligente.

- Implementar uma funcionalidade para disponibilizar orientações informativas sobre cuidados básicos de saúde ao idoso, respeitando o caráter assistivo da tecnologia, sem substituir a atuação do ACS.

- Implementar a funcionalidade para o processamento das interações realizadas, possibilitando a geração automática de relatórios, seu armazenamento em banco de dados e o acompanhamento contínuo das informações registradas.

- Implementar a funcionalidade de classificação automática dos relatórios clínicos gerados por níveis de prioridade, disponibilizando essas informações aos ACS para apoiar a organização e a priorização do atendimento domiciliar.

2 METODOLOGIA

Esta seção detalha o delineamento metodológico adotado para o desenvolvimento da pesquisa, descrevendo os procedimentos, a abordagem científica e os critérios que fundamentam a construção da solução proposta. O objetivo é apresentar, de forma estruturada, a trajetória técnica percorrida desde a concepção do *MVP* até a validação de suas funcionalidades, assegurando o rigor necessário para a verificação da aplicabilidade da ferramenta no contexto da APS.

2.1 CARACTERIZAÇÃO DA PESQUISA

Este trabalho caracteriza-se como uma pesquisa aplicada, por estar voltado ao desenvolvimento de uma solução tecnológica destinada à resolução de um problema identificado no contexto da APS. O estudo possui natureza tecnológica, uma vez que seu principal resultado consiste na concepção, implementação e validação de um *MVP* denominado *ClinCare*, desenvolvido para apoiar o acompanhamento domiciliar de idosos em situação de vulnerabilidade social por meio de um assistente conversacional inteligente. A proposta busca contribuir para o fortalecimento das atividades desempenhadas pelos ACS, sem substituir sua atuação, utilizando recursos de IA como mecanismo de apoio à coleta de informações, ao monitoramento contínuo e à priorização assistencial

Sob o ponto de vista metodológico, a pesquisa adota abordagem predominantemente qualitativa, uma vez que o foco está na concepção da arquitetura da solução, na definição das funcionalidades e na avaliação de sua capacidade de atender às necessidades do domínio estudado. O desenvolvimento concentrou-se na construção de um protótipo funcional que reúne as principais funcionalidades previstas para a plataforma, permitindo demonstrar sua viabilidade técnica e sua aderência ao problema investigado.

2.2 PROCESSO DE DESENVOLVIMENTO

O desenvolvimento do MVP foi conduzido de forma incremental, permitindo a implementação gradual das funcionalidades e sua validação contínua ao longo do projeto. Inicialmente, foram definidos os requisitos e produzidos os artefatos de

modelagem, incluindo diagramas de casos de uso, classes, arquitetura do sistema e modelo entidade-relacionamento, que serviram como base para a implementação da solução.

Na etapa seguinte, foi implementada uma arquitetura em camadas, organizada em componentes de apresentação, controle, serviços, persistência, domínio e extensão, favorecendo a separação de responsabilidades, a redução do acoplamento entre módulos e a manutenção evolutiva da aplicação. Essa organização também facilitou a integração entre os diferentes serviços responsáveis pela autenticação dos usuários, persistência dos dados, processamento de linguagem natural, recuperação semântica dos documentos de saúde e comunicação com serviços externos.

Com a infraestrutura arquitetural estabelecida, foram implementadas as funcionalidades destinadas aos dois perfis de usuários da plataforma. Para os idosos, desenvolveu-se uma interface conversacional baseada em IA generativa, capaz de realizar interações em linguagem natural, registrar informações de saúde e fornecer orientações embasadas em documentos informativos oficiais do SUS sobre saúde. Para os ACS, foram implementadas funcionalidades relacionadas ao acompanhamento clínico, geração automática de relatórios, classificação de prioridades assistenciais e gerenciamento dos idosos vinculados à plataforma.

Paralelamente, foram realizadas integrações com serviços externos responsáveis pela autenticação dos usuários, consulta às bases do Cadastro Nacional de Estabelecimentos de Saúde (CNES), recuperação automática de endereços e processamento local do modelo de linguagem utilizado pelo assistente conversacional. Essa abordagem possibilitou a construção de um *MVP* funcional, capaz de reproduzir os principais fluxos operacionais previstos para a solução proposta.

2.3 VALIDAÇÃO DA SOLUÇÃO

A validação concentrou-se na verificação funcional do *MVP* desenvolvido, buscando confirmar o correto funcionamento das principais funcionalidades previstas nos requisitos da aplicação. Durante essa etapa, foram realizados testes unitários dos fluxos relacionados ao cadastro e autenticação dos usuários, interação com o assistente conversacional.

Os testes unitários foram empregados para verificar, de forma isolada, o comportamento dos serviços que compõem a camada de negócio da aplicação, utilizando o *framework Pytest* em conjunto com as bibliotecas *pytest-mock* e *unittest.mock*. Essa abordagem permitiu substituir dependências externas por objetos simulados (*mocks*), possibilitando a validação das regras de negócio independentemente da disponibilidade de serviços como banco de dados, mecanismos de autenticação, modelos de IA e demais componentes externos. Por meio dessa metodologia, foi possível avaliar tanto os fluxos de execução bem-sucedidos quanto os mecanismos de tratamento de exceções e de recuperação de falhas implementados na arquitetura do sistema.

As suítes de teste abrangeram os principais módulos da aplicação, contemplando os serviços de autenticação, cadastro de usuários, chat conversacional, gerenciamento do vínculo entre idosos e ACS, processamento de IA, mecanismo de *RAG*, integração com o serviço do CNES, síntese de voz e mecanismos de autenticação e autorização das rotas protegidas. Essa estratégia permitiu validar o comportamento dos componentes considerados críticos para o funcionamento da plataforma, assegurando a consistência das regras de negócio e a robustez das integrações implementadas.

Em complemento, validaram-se a geração automática de relatórios clínicos, a classificação dos níveis de prioridade, o gerenciamento dos grupos segmentados dos idosos por condições crônicas e integração com os serviços externos empregados pela plataforma. Esses testes permitiram identificar inconsistências durante o desenvolvimento e promover ajustes sucessivos até a obtenção de uma versão funcional da solução. A existência de testes unitários para os principais serviços da aplicação também foi considerada como requisito de qualidade da arquitetura proposta.

Além da correta execução dessas funcionalidades, também foram analisados aspectos relacionados ao desempenho da arquitetura de IA, incluindo os tempos de processamento das respostas, o comportamento do mecanismo de recuperação de *RAG* e a viabilidade da execução local do modelo de linguagem empregado pelo sistema. Os resultados obtidos por meio dessas estratégias de validação foram posteriormente analisados e discutidos, abrangendo os indicadores de cobertura das suítes de teste, o desempenho dos módulos implementados e as evidências da viabilidade técnica da solução proposta.

Adicionalmente, foram produzidos registros audiovisuais demonstrando o funcionamento do *MVP*, permitindo evidenciar, em ambiente controlado, os principais fluxos de utilização da plataforma e o comportamento das funcionalidades desenvolvidas. Por tratar-se de um *MVP*, esta pesquisa não contemplou a implantação da plataforma em ambiente de produção nem sua disponibilização para utilização em larga escala. A validação teve como objetivo demonstrar a viabilidade técnica da arquitetura proposta e verificar sua capacidade de atender aos requisitos definidos para o contexto da APS. Este trabalho constitui uma etapa inicial para futuras avaliações em cenários reais de utilização e para estudos envolvendo usuários finais e profissionais da saúde.

3 TECNOLOGIAS ENVOLVIDAS

Para o desenvolvimento deste trabalho, propôs-se a construção de um *MVP* inteligente, acessível por meio da *web* e de um *Android Application Package (APK)* em versão *debug*, utilizando um *web view* empacotado. Considerando a viabilidade técnica da solução, optou-se por uma arquitetura em camadas, conteinerizada e preparada para execução com aceleração por *GPU* por meio do *NVIDIA Container Toolkit*. A solução integra um *backend* composto por banco de dados, autenticação, IA generativa, serviços de áudio e voz, exportação dos documentos e infraestrutura baseada em *Docker*, conteinerizada e orquestrada por meio do *Docker Compose*. Por meio dessa integração, viabilizou uma infraestrutura de desenvolvimento que contribui para a flexibilidade, a escalabilidade e a viabilidade técnica, social e econômica da solução.

A principal linguagem de programação utilizada neste trabalho foi o *Python* 3.10.11. Sua escolha fundamentou-se não apenas em seu ecossistema consolidado para aplicações de IA e processamento de dados, mas em sua viabilidade para o desenvolvimento de aplicações *web*. Como *framework* principal, foi utilizado o *Flask*, responsável pela construção do *backend* baseado em arquitetura *Representational State Transfer (REST)*, bem como pela implementação das rotas, sessões e interfaces do sistema. Para a renderização das páginas do *frontend*, utilizou-se o mecanismo de *templates* da biblioteca nativa *Jinja2*.

Para a modelagem da base de dados, foi adotado o sistema gerenciador de banco de dados *Postgre Structured Query Language* (*SQL*) 13, implementado de forma compatível com o módulo *SQLAlchemy*. Em conjunto, essas tecnologias viabilizam a construção de uma camada de abstração eficiente para o modelo relacional por meio do padrão *Object-Relational Mapping (ORM)*. Além disso, essa integração estabelece uma ponte entre o paradigma da Programação Orientada a Objetos (POO) e o modelo relacional, reduzindo a necessidade da escrita manual de comandos *SQL*.

Para a autenticação dos usuários, o sistema utiliza o *Firebase Authentication*, com suporte do *Firebase Admin Software Development Kit* (*SDK*) para operações privilegiadas executadas no *backend* e da biblioteca *Pyrebase4* para autenticação no lado do cliente. Além disso, foi integrada a autenticação social por meio do *Google OAuth 2.0*, utilizando a biblioteca *Authlib*. Essa arquitetura de autenticação

amplia a usabilidade da aplicação, permitindo o acesso tanto por meio de e-mail e senha quanto por uma conta *Google*.

Na camada de IA generativa, priorizou-se a independência em relação a *Application Programming Interface (APIs)* comerciais, adotando o *Ollama* como servidor de inferência *REST* para execução local dos modelos de linguagem. Como modelo de IA, o *Llama 3.2:3B* foi selecionado por apresentar um equilíbrio adequado entre desempenho computacional, baixa latência e reduzido consumo de recursos de *hardware*, dispensando a utilização de unidades de processamento gráfico de alto desempenho.

A escolha desse modelo justifica-se pelo equilíbrio entre a redução do número de parâmetros e as técnicas de *distillation* empregadas. A versão utilizada possui aproximadamente três bilhões de parâmetros, permitindo níveis de quantização em 8, 4 e 2 *Binary Digits* (*BITs)*, o que viabiliza sua execução em *Video Random Access Memory (VRAM)* de aproximadamente 4,5 *GigaBytes* (*GB*), 3 *GB* e 2 *GB*, respectivamente. Em contrapartida, modelos com 7 ou 8 bilhões de parâmetros podem exigir entre 14 *GB* e 16 *GB* de *VRAM* em precisão nativa de 16 *BITs*, ou entre 8 *GB* e 10 *GB* quando executados com quantização de 8 *BIT*, apenas para o carregamento inicial na *Graphics Processing Unit (GPU)*.

Como tecnologia complementar, fez o uso do *ChromaDB* como banco de dados vetorial persistente para a recuperação de informações e a contextualização das respostas produzidas pelo modelo *Llama 3.2:3B*, empregando a abordagem *RAG*. Para a geração dos *embeddings* semânticos, adotou-se a biblioteca *SentenceTransformers*, utilizando o modelo *all-MiniLM-L6-v2*. Já a extração do conteúdo dos documentos de saúde de apoio utilizados pela IA foi realizada por meio da biblioteca *PyPDF*.

Optou-se pelo *all-MiniLM-L6-v2* por seu equilíbrio entre velocidade de inferência e tamanho reduzido, compatível com os recursos computacionais restritos do projeto. Embora não seja a versão explicitamente multilíngue, o modelo foi treinado sobre um *corpus* amplo que inclui pares de sentenças em múltiplos idiomas via *knowledge distillation*, apresentando desempenho aceitável para similaridade semântica em português no contexto de teste realizado. Reconhece-se, contudo, como limitação e trabalho futuro, a substituição por uma variante nativamente multilíngue, como o *paraphrase-multilingual-MiniLM-L12-v2* para validação comparativa de acurácia.

No processamento vetorial, os cálculos de similaridade semântica ficaram a cargo da biblioteca *NumPy*. Para a execução dos modelos de aprendizado de máquina, empregou-se o *PyTorch* 2.2.1, com suporte às tecnologias *CUDA* 12.1 e *cuDNN* 8, em ambiente equipado com *GPU NVIDIA*. Essas tecnologias foram selecionadas em razão do elevado desempenho computacional proporcionado durante a geração e a consulta dos *embeddings*, garantindo maior eficiência na recuperação semântica dos documentos de saúde do SUS de referência utilizados como base para a geração das respostas da IA e fortalecendo o caráter técnico e informativo das informações disponibilizadas aos usuários.

Para atender às funcionalidades de interação por voz, foram utilizadas tecnologias voltadas tanto à síntese quanto ao reconhecimento de fala. Para isso, adotou-se o *Microsoft Edge Text-to-Speech* (*TTS*) para a conversão de texto em áudio e a biblioteca *SpeechRecognition* para a transcrição dos comandos de voz, a fim de capturar e reproduzir conteúdos sonoros com baixa latência. Esses recursos tornam a interface mais inclusiva, aproximando a solução da proposta de atender usuários com limitações de leitura, escrita ou mobilidade, reforçando a acessibilidade como princípio central do desenvolvimento.

No contexto da geração e exportação dos documentos resultantes da triagem inteligente realizada na interface do ACS, foram empregadas bibliotecas especializadas em conversão e renderização de conteúdo. Para isso, utilizaram-se os módulos *Markdown*, *WeasyPrint* e *Flask-WeasyPrint*. O *Markdown* foi empregado para converter documentos escritos em sua linguagem de marcação para o formato *HyperText Markup Language (HTML)*, simplificando a elaboração dos documentos sem a necessidade da construção manual de estruturas *HTML* complexas.

A biblioteca *WeasyPrint* é responsável pela conversão dos documentos *HTML* e *Cascading Style Sheets (CSS)* em arquivos *Portable Document Format* (*PDF)*. Seu mecanismo de renderização interpreta as regras de formatação definidas nas folhas de estilo, permitindo a geração de documentos vetoriais com elevada qualidade visual diretamente pela aplicação.

A extensão *Flask-WeasyPrint* é responsável por integrar o *WeasyPrint* ao *framework Flask*. Essa integração possibilita converter diretamente o conteúdo retornado pelas rotas da aplicação em arquivos *PDF*, permitindo o reaproveitamento dos *templates* desenvolvidos com o *Flask* e reduzindo significativamente a

quantidade de código necessária para a geração dos documentos formatados destinados ao usuário final.

No desenvolvimento da interface do sistema, o *frontend* foi desenvolvido utilizando as tecnologias *HTML5*, *CSS3* e *JavaScript*, seguindo as diretrizes do *Material Design 3 (MD3)*. Além disso, a *Fetch API* foi utilizada para a comunicação assíncrona com o *backend* e o *HTML Media Element API* foi empregado para a reprodução dos conteúdos de áudio processados pelo servidor. Embora este trabalho não tenha sido desenvolvido com tecnologias nativas para aplicações móveis, essa composição tecnológica possibilita a construção de interfaces responsivas, adaptáveis a diferentes resoluções e dimensões de tela.

A infraestrutura de implantação foi organizada utilizando *Docker* e *Docker Compose*, permitindo a conteinerização da aplicação e a orquestração dos serviços que compõem o sistema, incluindo o banco de dados e servidor *Flask*. Além disso, utilizou-se o *NVIDIA Container Toolkit*, responsável por disponibilizar acesso à *GPU* dentro dos contêineres, possibilitando melhor desempenho durante a execução dos modelos de IA. Complementarmente, o gerenciamento das configurações da aplicação foi realizado por meio de variáveis de ambiente administradas pela biblioteca *python-dotenv*, contribuindo para maior organização, segurança e reprodutibilidade do ambiente de execução.

Também foi integrado um mecanismo de gerenciamento de *cache* utilizando o *Redis*, configurado com tempo de vida (*Time To Live – TTL*) de 24 horas para armazenar temporariamente as consultas realizadas ao *Simple Object Access Protocol* (*SOAP*) *web service* do Ministério da Saúde. Essa estratégia reduz chamadas repetidas ao serviço externo, melhora a latência da aplicação e é gerenciada por meio da implementação do padrão de projeto *Singleton* durante todo o ciclo de vida da aplicação.

Para o consumo deste *web service SOAP*, adotou-se a biblioteca *Zeep*, cliente *Python* responsável pela desserialização das respostas em formato *eXtensible Markup Language (XML)* retornadas pelo serviço *SOAP*-CNES. Por meio desse *web service*, é disponibilizado o acesso estruturado à relação de UBS por código do Instituto Brasileiro de Geografia e Estatística (IBGE).

Como integração adicional com serviços externos, utilizou-se a *API* pública *ViaCEP* para o preenchimento automático de informações de endereçamento durante o cadastro de usuários. Por meio da biblioteca *Requests*, são obtidos dados

como Código de Endereçamento Postal (CEP), município e unidade federativa, reduzindo o preenchimento manual dessas informações. Diante do exposto, a Tabela 1 sintetiza o conjunto de tecnologias, bibliotecas e serviços integrados ao projeto, bem como suas atribuições operacionais.

Tabela 1 – Principais tecnologias utilizadas

Tecnologia	**Função**
Python 3.10.11	Linguagem principal
Flask ≥ 3.0	*Framework web REST* e renderização de páginas
SQLAlchemy	*ORM* para acesso ao banco relacional
PostgreSQL 13	Banco de dados relacional principal
Redis ≥ 5.0	Cache em memória (*TTL* 24h) para consultas CNES/*SOAP*
Firebase Authentication	Autenticação de usuários
Google OAuth 2.0	*Login* social via conta *Google*
Ollama	Servidor local de inferência para modelo de linguagem generativa
Llama 3.2:3B	Modelo de linguagem principal
ChromaDB	Banco vetorial para o *pipeline RAG*
SentenceTransformers (all-MiniLM-L6-v2)	Geração de *embeddings* semânticos
Microsoft Edge TTS	Síntese de voz
SpeechRecognition	Transcrição de comandos de voz
Zeep	Cliente *SOAP* para consulta ao CNES
ViaCEP (*Requests*)	Preenchimento automático de endereços por CEP
Docker + Docker Compose	Conteinerização e orquestração dos serviços

Fonte: elaborada pelo autor (2026).

4 MODELAGEM DO PROJETO

Esta seção apresenta a modelagem do sistema desenvolvido, contemplando os principais artefatos produzidos durante as fases de análise e *design*. Seu propósito é formalizar as decisões de projeto por meio de representações estruturadas, proporcionando uma visão clara, organizada e documentada da solução antes e durante sua implementação (Sommerville, 2019).

Para esse fim, são apresentados o levantamento dos Requisitos Funcionais (RF) e não funcionais (RNF), os Diagramas de Casos de Uso, de Classes, de Arquitetura do Sistema, o Diagrama Entidade-Relacionamento (ER) do banco de dados e as interfaces desenvolvidas. Em conjunto, esses artefatos fornecem uma base consistente para a compreensão do escopo do sistema, das responsabilidades atribuídas a cada componente e das decisões técnicas que nortearam seu desenvolvimento

4.1 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos constitui a etapa inicial da modelagem do sistema, sendo responsável por identificar e documentar as necessidades que a solução deverá atender. Neste trabalho, adotaram-se sessões de *brainstorming* e entrevistas semi-estruturadas com os principais atores envolvidos no domínio da aplicação, representados na Figura 1. Essas técnicas possibilitaram compreender os processos relacionados ao contexto da APS e identificar os RF e RNF da plataforma, conforme recomendado pela literatura (Osborn, 1953; Sommerville, 2019).

A partir dessas técnicas, foram identificados os objetivos da aplicação, os perfis de usuários e as interações esperadas entre os atores e o sistema. Os requisitos foram organizados em duas categorias: RF e RNF, responsáveis por descrever os serviços e comportamentos esperados da aplicação e estabelecer restrições relacionadas à qualidade, desempenho, segurança e arquitetura do sistema.

.

4.1.1 REQUISITOS FUNCIONAIS

Esta subseção detalha os RFs estabelecidos para o *MVP ClinCare*, descrevendo os serviços, comportamentos e funções que o sistema deve oferecer aos seus usuários. A definição destes requisitos visa delimitar o escopo da solução, contemplando as interações necessárias entre os perfis de idosos e ACS, bem como a integração com os mecanismos de IA e serviços externos que sustentam a operação da plataforma.

RF001: O sistema deve permitir o cadastro de novos usuários nos perfis de idoso e ACS.

RF002: O sistema deve autenticar usuários por meio do *Firebase Authentication* e associar cada usuário autenticado ao respectivo registro armazenado no banco de dados *PostgreSQL*.

RF003: O sistema deve validar o endereço de *e-mail* informado durante o cadastro utilizando os mecanismos disponibilizados pelo *Firebase Authentication*.

RF004: O sistema deve permitir a autenticação por *e-mail* e senha, validando as credenciais simultaneamente no *Firebase Authentication* e no banco de dados *PostgreSQL*.

RF005: O sistema deve permitir autenticação utilizando uma Conta *Google* por meio do *OAuth 2.0*.

RF006: O sistema deve permitir a recuperação da senha mediante envio de um *e-mail* de redefinição pelo *Firebase Authentication*.

RF007: O sistema deve permitir que usuários do perfil idoso atualizem seu endereço de *e-mail* mediante confirmação do novo endereço por meio de um *link* de verificação.

RF008: O sistema deve permitir que o usuário exclua permanentemente sua conta.

RF009: O sistema deve disponibilizar *dashboards* distintos para usuários dos perfis idoso e ACS.

RF010: O sistema deve gerar automaticamente um código alfanumérico único composto por seis caracteres após a conclusão do cadastro da conta, destinado ao vínculo do idoso ao *dashboard* do ACS.

RF011: O sistema deve integrar a *API ViaCEP* para preencher automaticamente o CEP, o estado e o município durante o cadastro dos usuários.

RF012: O sistema deve permitir que idosos autenticados interajam com um agente inteligente de saúde por meio de um *chat* conversacional baseado em voz.

RF013: O agente inteligente deve classificar automaticamente cada interação de mensagem realizada pelo idoso nas categorias *GREETING*, *HEALTH_QUERY*, *EMERGENCY* ou *OTHER*.

RF014: O agente inteligente deve utilizar documentos de orientação à saúde do SUS, por meio da abordagem *RAG*, para recuperar informações de contexto e produzir respostas compatíveis com a classificação da intenção de mensagem do idoso identificada.

RF015: O sistema deve disponibilizar a janela de contexto do idoso estruturada no formato *JavaScript Object Notation (JSON)* para utilização pelo agente de IA.

RF016: O sistema deve atualizar incrementalmente a janela de contexto do idoso conforme o histórico de conversas registrado em cada sessão do *chat*.

RF017: O sistema deve permitir que o ACS vincule idosos ao seu *dashboard* utilizando um código único de identificação.

RF018: O sistema deve permitir que o ACS solicite a geração ou atualização do relatório diário de acompanhamento de um idoso.

RF019: O agente de IA deve gerar relatórios diários de acompanhamento do idoso utilizando as informações armazenadas na janela de contexto.

RF020: O sistema deve permitir que o ACS consulte o histórico completo dos relatórios diários de cada idoso vinculado.

RF021: O sistema deve permitir que o ACS realize o *download* de qualquer relatório diário em *PDF*, preservando um *layout* padronizado e adequado para impressão.

RF022: O sistema deve executar automaticamente a triagem inteligente dos idosos vinculados ao ACS, classificando-os nas prioridades ALTA, MÉDIA ou BAIXA, acompanhadas da respectiva justificativa clínica.

RF023: O sistema deve permitir que o ACS realize o cadastro assistido de idosos, gerando credenciais temporárias e um *link* para recuperação de acesso.

RF024: O sistema deve permitir que o ACS compartilhe um *link* temporário para recuperação de acesso destinado aos idosos cadastrados em modo assistido.

RF025: O sistema deve permitir que o ACS crie grupos de idosos, informando nome e descrição para cada grupo.

RF026: O sistema deve permitir que o ACS adicione e remova idosos dos grupos sob sua responsabilidade.

RF027: O sistema deve garantir que cada ACS gerencie exclusivamente os grupos e idosos vinculados ao seu próprio cadastro.

RF028: O sistema deve integrar-se à *API SOAP* do CNES para consultar as Unidades UBS disponíveis em um município a partir do respectivo código do IBGE.

RF029: O sistema deve indexar os documentos utilizando o modelo de *embeddings all-MiniLM-L6-v2*, empregando aceleração por *GPU (CUDA)* quando disponível.

4.1.2 REQUISITOS NÃO FUNCIONAIS

Esta subseção apresenta os RNFs definidos para o *MVP ClinCare*, estabelecendo os atributos de qualidade e as restrições técnicas que orientam o comportamento e a estrutura da aplicação. Enquanto os RFs delimitam as capacidades operacionais do sistema, os RNFs asseguram que a solução opere com o desempenho, a segurança, a usabilidade e a robustez necessários para sua viabilidade técnica. A definição destes critérios é fundamental para garantir que a arquitetura proposta atenda às expectativas de qualidade exigidas para um ambiente de suporte à saúde, balizando decisões de projeto que abrangem desde a latência na resposta da IA até a integridade no gerenciamento de dados dos usuários.

RNF001: O sistema deve classificar a intenção de cada mensagem do idosos em tempo inferior a 500 milissegundos, utilizando o *cache* de *embeddings* previamente calculados e mantidos em memória.

RNF002: O sistema deve gerar respostas utilizando o modelo *Llama 3.2:3B* em tempo máximo de sete segundos para cada solicitação.

RNF003: O sistema deve realizar o pré aquecimento do modelo de linguagem durante a inicialização do servidor, reduzindo a latência da primeira requisição.

RNF004: O sistema deve armazenar, no *Redis*, as informações obtidas da *API SOAP* do CNES utilizando *TTL* de 24 horas, reduzindo consultas repetidas ao serviço externo.

RNF005: O sistema deve permitir que a reprodução do áudio seja iniciada antes da conclusão completa da geração da resposta textual.

RNF006: O sistema deve validar a sessão do usuário, a cada requisição protegida, tanto no *Firebase Authentication* quanto no banco de dados *PostgreSQL*, garantindo a integridade da autenticação.

RNF007: O sistema deve utilizar *tokens* de sessão assinados por meio de uma chave secreta (*SECRET_KEY*) armazenada exclusivamente em variáveis de ambiente.

RNF008: O sistema deve garantir o isolamento dos dados entre os ACS, impedindo o acesso a informações, idosos e grupos pertencentes a outros agentes.

RNF009: O sistema não deve armazenar senhas de usuários no banco de dados *PostgreSQL*, delegando integralmente seu gerenciamento ao *Firebase Authentication*.

RNF010: O sistema deve exigir a confirmação do novo endereço de *e-mail* por meio de um *link* de verificação antes de efetivar sua atualização.

RNF011: O sistema deve armazenar as credenciais do *Firebase Admin SDK* em arquivo no formato *JSON*, protegido e excluído do controle de versão por meio do arquivo *.gitignore*.

RNF012: O sistema deve gerar *links* temporários de recuperação de acesso contendo *token* assinado e prazo de expiração previamente definido.

RNF013: O agente inteligente deve produzir respostas curtas, objetivas e escritas em linguagem simplificada, adequada ao perfil cognitivo do público-alvo.

RNF014: O agente inteligente deve sintetizar todas as respostas em áudio utilizando voz em português brasileiro.

RNF015: O sistema deve apresentar mensagens de erro em linguagem humanizada, sem expor informações técnicas, códigos internos ou detalhes da infraestrutura de autenticação.

RNF016: O sistema deve operar em modo degradado quando o mecanismo de recuperação semântica baseado em *RAG* estiver indisponível, retornando uma mensagem padrão sem interromper o funcionamento da aplicação.

RNF017: O sistema deve executar, em segundo plano, a geração de relatórios e a atualização da janela de contexto, evitando bloqueios nas respostas *HyperText Transfer Protocol (HTTP)*.

RNF018: O sistema deve executar automaticamente operações de *rollback* no banco de dados *PostgreSQL* sempre que ocorrer falha durante operações de escrita.

RNF019: O sistema deve garantir que exista apenas um relatório diário para cada idoso em uma mesma data.

RNF020: O sistema deve atualizar a janela de contexto do idoso de forma incremental, processando apenas as mensagens ainda não incorporadas ao contexto.

RNF021: O sistema deve adotar arquitetura em camadas, mantendo separação entre rotas, serviços, repositórios e modelos, sem acesso direto aos dados pelas rotas da aplicação.

RNF022: O sistema deve registrar eventos utilizando mecanismo estruturado de *logging*, configurável por níveis (*DEBUG*, *INFO* e *ERROR*), identificando o módulo responsável por cada ocorrência.

RNF023: O sistema deve possuir testes unitários capazes de validar os principais serviços responsáveis pelas regras de negócio.

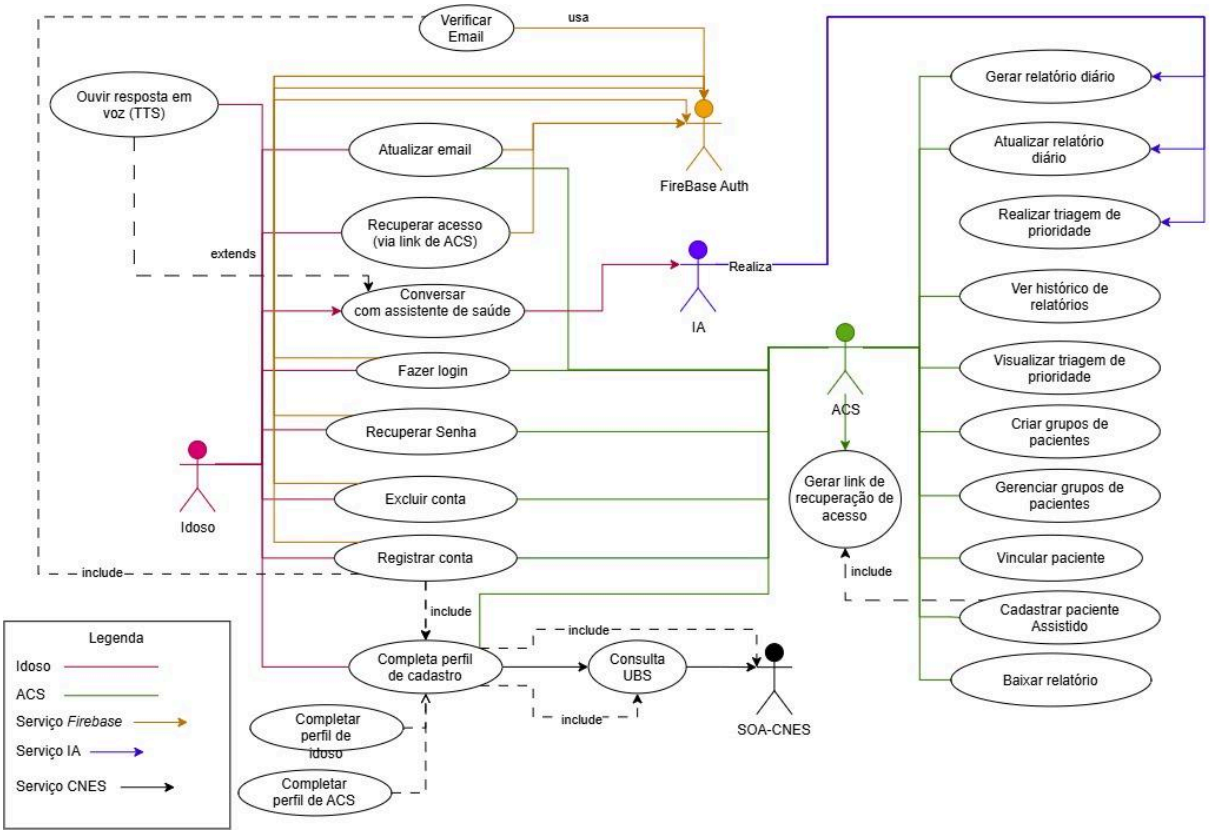
RNF024: O sistema deve ser executável em contêineres *Docker*, utilizando o *Docker Compose* para orquestrar os serviços *Flask*, *PostgreSQL*, *Redis* e *Ollama*.

4.2 CASOS DE USO

Esta subseção apresenta o Diagrama de Casos de Uso, ilustrado na Figura 1, com o objetivo de identificar os atores envolvidos, representar as interações estabelecidas entre eles e evidenciar as funcionalidades disponibilizadas pelo *MVP*.

A modelagem foi elaborada utilizando a notação da *Unified Modeling Language* (*UML*), empregando os estereótipos *include* e *extend* para representar dependências funcionais entre os casos de uso, além da utilização de atores secundários para representar serviços externos integrados à plataforma.

Figura 1 - Diagrama de casos de uso



Fonte: elaborada pelo autor (2026).

O Diagrama de Casos de Uso identifica dois atores primários: o idoso e o ACS. Ambos representam os usuários finais da plataforma, desempenhando papéis distintos conforme suas responsabilidades no contexto da APS. O idoso corresponde ao indivíduo assistido pelo sistema de saúde comunitária e possui acesso às funcionalidades de cadastro, autenticação, recuperação de senha, gerenciamento da própria conta e interação com o assistente inteligente de saúde. O perfil do ACS possui acesso a um conjunto de funcionalidades exclusivas destinadas ao monitoramento clínico, incluindo a vinculação de idosos no *dashboard*, o acompanhamento por meio de relatórios automatizados, a triagem inteligente e a criação e o gerenciamento de grupos segmentados de idosos conforme suas condições clínicas.

O diagrama também é composto por três atores secundários, responsáveis por representar serviços externos integrados ao sistema: *Firebase Authentication*, *SOAP*-CNES e o modelo de linguagem *Llama 3.2:3B*. O *Firebase Authentication* fornece toda a infraestrutura necessária para autenticação dos usuários, contemplando funcionalidades como registro de contas, autenticação, recuperação de senha, atualização de endereço de e-mail e exclusão de contas. O serviço *SOAP*-CNES disponibiliza os dados do CNES por meio de um *web service SOAP*, auxiliando o processo de cadastro complementar realizado exclusivamente pelo ACS. O modelo de linguagem *Llama 3.2:3B* executa internamente os casos de uso relacionados ao processamento de linguagem natural empregados pelo assistente inteligente da plataforma.

Ao lado do ator idoso, observam-se casos de uso voltados ao autocuidado e à interação com o agente inteligente de saúde. O fluxo de autenticação é compartilhado entre os dois perfis de usuário e contempla os casos de uso Registrar Conta, *Login*, Recuperação de Senha e Exclusão de Conta. O caso de uso Registrar Conta apresenta uma dependência obrigatória do tipo *include* em relação ao caso de uso Verificar E-mail, refletindo o fluxo em que a ativação da conta permanece condicionada à confirmação do endereço de e-mail por meio do *Firebase Authentication*.

Após a autenticação, o idoso passa a utilizar o assistente conversacional baseado no modelo de linguagem *Llama 3.2:3B*, que representa o principal caso de uso da plataforma sob sua perspectiva. Internamente, esse caso de uso incorpora mecanismos inteligentes responsáveis pela classificação da intenção da mensagem, pela recuperação dos documentos de orientação à APS utilizando a abordagem *RAG* e pela geração da resposta utilizando o modelo de linguagem. Dessa forma, evidencia-se que toda interação conversacional percorre esse fluxo de processamento de IA.

As funcionalidades de síntese de voz e de atualização da janela de contexto clínico são representadas por relacionamentos do tipo *extend*, uma vez que correspondem a comportamentos condicionais. A síntese de voz é executada apenas quando o idoso utiliza a interface de comunicação por voz, enquanto a atualização da janela de contexto ocorre de forma assíncrona após o encerramento da sessão de *chat*.

Em relação ao ator ACS, observa-se um conjunto significativamente maior de casos de uso, refletindo sua função de gerenciamento e acompanhamento dos idosos vinculados à plataforma. Esse perfil possibilita a execução de operações relacionadas ao monitoramento clínico coletivo e ao acompanhamento individual dos idosos. Os casos de uso Gerar Relatório Diário e Atualizar Relatório dependem do caso de uso responsável pela geração automatizada de relatórios clínicos utilizando o modelo de IA. Para essa finalidade, o modelo sumariza o histórico de conversas do idoso armazenado na janela de contexto, estruturando essas informações em formato *JSON* para produzir relatórios clínicos organizados e consistentes.

A funcionalidade de triagem inteligente contempla tanto a análise do quadro clínico quanto a geração da respectiva justificativa clínica pelo modelo de linguagem. Com base nessa avaliação, o sistema classifica automaticamente os idosos em níveis de prioridade alta, média ou baixa. Essas informações subsidiam o planejamento das visitas domiciliares e das ações de acompanhamento realizadas pelo ACS.

O caso de uso Cadastrar Idoso em Modo Assistido foi concebido para atender usuários idosos que apresentam dificuldades relacionadas ao letramento digital. Por meio dessa funcionalidade, o próprio ACS realiza o cadastro inicial da conta dele e gera um *link* temporário destinado ao responsável pelo idoso para a definição das credenciais de acesso. Essa abordagem é necessária para situações em que o usuário não possui endereço de e-mail ou apresenta dificuldades para utilizar serviços de correio eletrônico, como *Gmail*, *Hotmail* ou *Outlook*. Assim, o cuidador responsável pode concluir a configuração das credenciais de acesso sem que o ACS tenha conhecimento dessas informações, preservando a confidencialidade dos dados de autenticação.

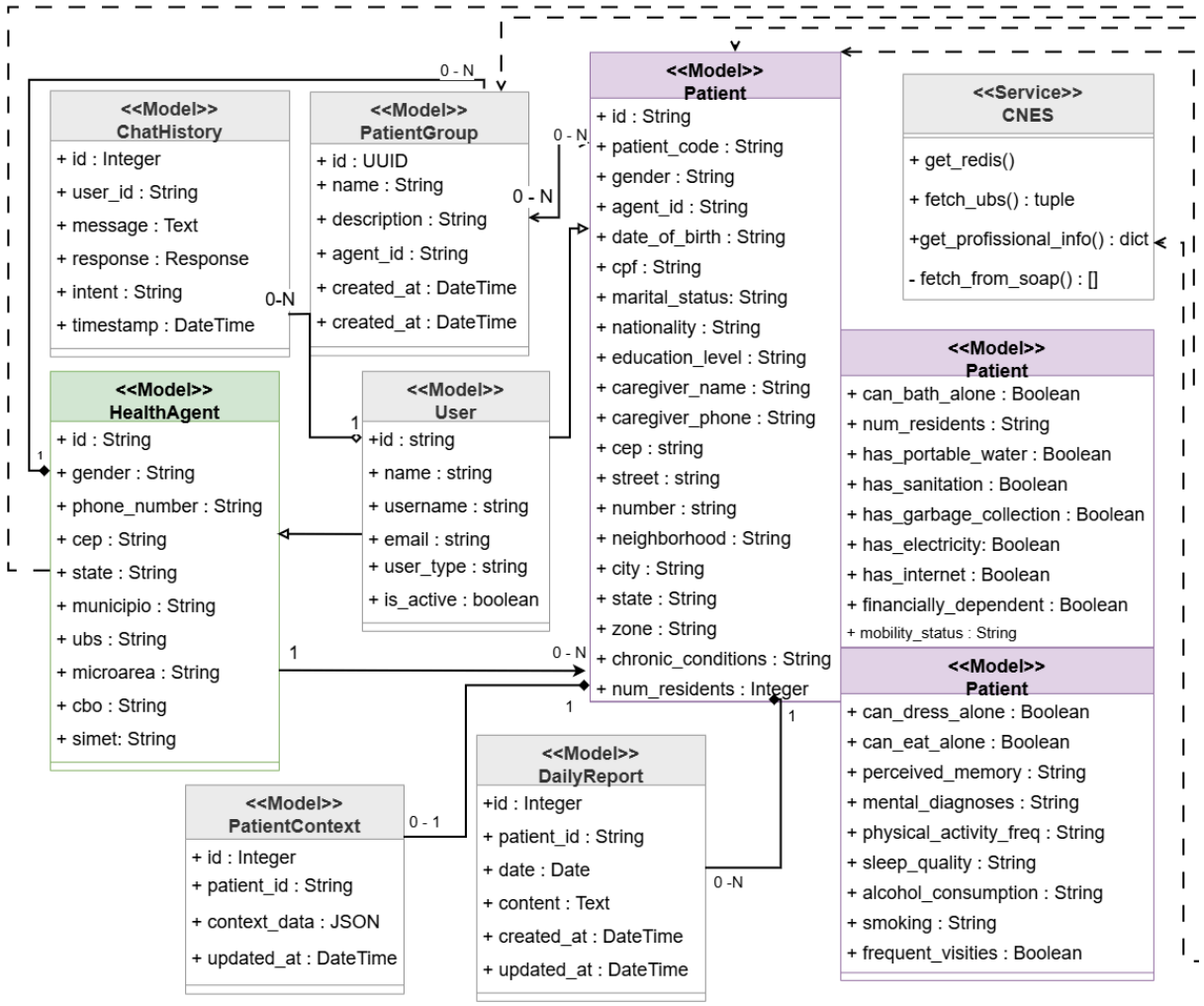
Os atores secundários *SOAP*-CNES e *Firebase Authentication*, posicionados na região central e na extremidade superior esquerda do diagrama, evidenciam sua função como serviços de suporte às funcionalidades da plataforma. O *Firebase Authentication* participa dos fluxos relacionados à autenticação, verificação de e-mail, recuperação de senha e exclusão de contas. Já o serviço *SOAP*-CNES é consultado durante o cadastro do ACS para fornecer a relação de UBS disponíveis no município informado, bem como os respectivos dados de classificação ocupacional.

4.3 DIAGRAMA DE CLASSES

Nesta subseção é apresentado o Diagrama de Classes, elaborado para orientar o desenvolvimento segundo os paradigmas da POO. Sua modelagem fundamenta-se em princípios como encapsulamento, abstração, composição, testabilidade isolada e redução do acoplamento por herança. Além disso, incorpora padrões arquiteturais como o *Singleton*, implementado por meio de extensões do *Flask*, classes de repositório e decoradores, garantindo instância única durante todo o ciclo de vida da aplicação, separação das responsabilidades em camadas e reutilização de comportamentos sem a necessidade de herança por meio de funções de alta ordem.

O Diagrama de Classes da plataforma reflete uma arquitetura em camadas bem definida, estruturada em quatro níveis principais de abstração: modelos de domínio, repositórios, serviços e extensões de infraestrutura. Essa organização constitui uma aplicação direta do Princípio da Responsabilidade Única (*Single Responsibility Principle – SRP*), no qual cada classe desempenha uma responsabilidade específica e claramente delimitada dentro da arquitetura do sistema (Martin, 2017). A primeira camada, representada pela Figura 2 corresponde aos modelos de domínio (*Model*), sendo composta pelas entidades persistidas no banco de dados relacional adotado pelo sistema e mapeadas por meio do *ORM*.

Figura 2 - Diagrama de classes dos modelos de domínio



Fonte: elaborada pelo autor (2026).

A classe *User* ocupa posição central nessa camada, pois representa a identidade básica de todos os usuários da plataforma. A partir dela, por meio de associações, derivam as classes *HealthAgent* e *Patient*, responsáveis por especializar os perfis de usuário conforme os dois atores primários do sistema.

A entidade *HealthAgent* armazena as informações profissionais do ACS, como o Código Brasileiro de Ocupações (CBO), UBS vinculada e a microárea de atuação. Por sua vez, a entidade *Patient* concentra as informações clínicas e demográficas do idoso, incluindo condições crônicas, localidade e dados de contato do cuidador domiciliar. Essa organização evita a redundância de informações na tabela *Users*, representada na Figura 6, ao manter os atributos específicos de cada perfil em tabelas especializadas, preservando a normalização do banco de dados.

Associadas à entidade *Patient*, existem três classes responsáveis por representar o ciclo de acompanhamento clínico realizado pela plataforma. A primeira

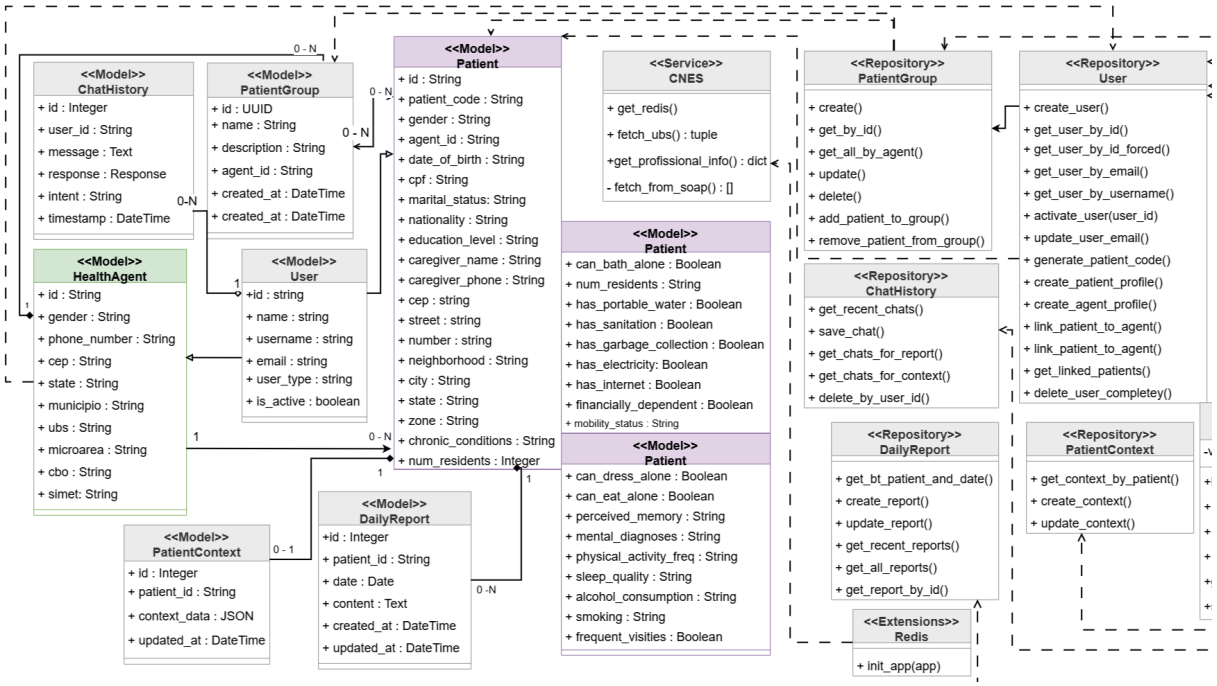
delas é a classe *ChatHistory*, responsável por registrar cada par de mensagem e resposta das interações entre o idoso e o assistente inteligente, armazenando também a intenção identificada pelo *pipeline* de Processamento de Linguagem Natural (*PLN*).

A segunda classe é *DailyReport*, responsável por persistir os relatórios clínicos diários gerados automaticamente pelo modelo de linguagem, oferecendo suporte à atualização incremental dessas informações ao longo do dia. A terceira classe é *PatientContext*, responsável por manter uma representação estruturada, no formato *JSON*, do estado clínico atual do idoso. Essa estrutura armazena informações como sintomas recorrentes, medicações de uso contínuo relatadas e ações recomendadas, sendo posteriormente injetada como contexto nos *prompts* enviados ao modelo de linguagem.

Ainda entre as classes relacionadas aos atores primários, destaca-se a classe *PatientGroup*, associada à entidade *HealthAgent*. Essa classe permite organizar os idosos vinculados ao ACS em grupos definidos segundo critérios clínicos, facilitando o gerenciamento das atividades de acompanhamento realizadas pelo profissional de saúde.

A camada de repositórios (*Repository*), representada pela Figura 3 é responsável por abstrair completamente o acesso ao banco de dados, expondo uma interface composta por métodos de alto nível para cada entidade de domínio. A classe *UserRepository* concentra as operações de criação, consulta, ativação, atualização de endereço de e-mail e exclusão de usuários. Além disso, encapsula a lógica de vinculação entre as entidades *Patient* e *HealthAgent*, bem como a geração do código único de identificação do idoso.

Figura 3 - Diagrama de classes dos modelos de repositórios

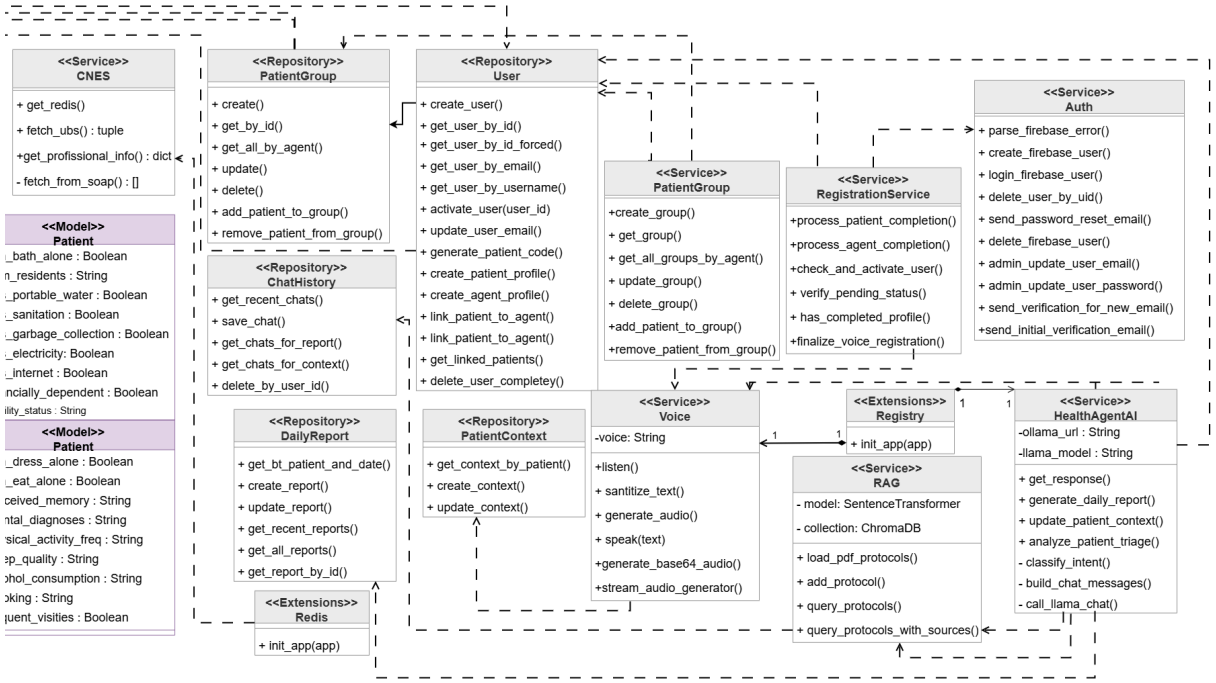


Fonte: elaborada pelo autor (2026).

As demais classes pertencentes à camada de repositórios seguem o mesmo padrão arquitetural, garantindo que a camada de serviços permaneça desacoplada do *ORM*. Essa decisão de projeto favorece a testabilidade da aplicação, permitindo que os repositórios sejam substituídos por *mocks* em memória durante a execução das suítes de testes.

A camada de serviços (*Service*), representada pela Figura 4, concentra toda a lógica de negócio da aplicação e representa o nível arquitetural que mais se aproxima do princípio de responsabilidades isoladas (Martin, 2017). Nessa camada, a classe *HealthAgentAI* constitui o principal serviço da plataforma, reunindo atributos de configuração do modelo *Llama* e métodos responsáveis por todo o ciclo de processamento da IA empregado neste *MVP*.

Figura 4 - Diagrama de classes dos modelos de serviços e extensões



Fonte: elaborada pelo autor (2026).

O *HealthAgentAI* realiza a classificação da intenção da mensagem do idoso, a construção do *prompt* contendo o contexto clínico dele, a geração da resposta pelo modelo de linguagem, a sumarização em relatórios clínicos diários e a execução da triagem inteligente. A organização desse *pipeline* é materializada pela classe *RAGService*, responsável pela integração com o banco vetorial *ChromaDB* por meio do modelo *SentenceTransformer*. Essa arquitetura possibilita a codificação semântica das consultas utilizadas na recuperação dos documentos de orientação à APS do SUS, previamente armazenados no banco vetorial e empregados como contexto para fundamentar as respostas geradas pelo modelo de linguagem.

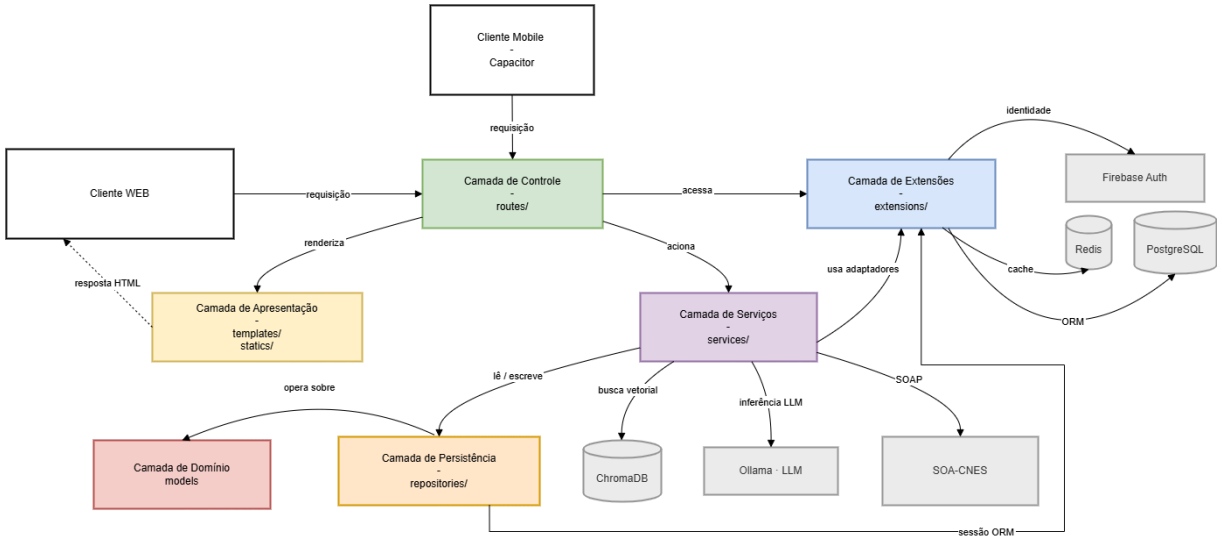
A classe *AuthService* encapsula toda a comunicação com o *Firebase Authentication*, isolando as chamadas às respectivas *APIs* de autenticação do restante da aplicação. O *RegistrationService* é responsável pelo gerenciamento do ciclo de vida das etapas de cadastro dos usuários. O *PatientGroupService* implementa as regras de negócio relacionadas ao gerenciamento de grupos de idoso, validando a propriedade dos recursos pertencentes ao ACS. A classe *VoiceService* encapsula as integrações com os mecanismos de síntese e reconhecimento de voz. Já o CNES*Service* é responsável pela comunicação com o *web service SOAP* do CNES, realizando a consulta dos dados necessários ao processo de cadastro dos profissionais de saúde.

A camada de extensões (*Extension*) é representada pelas classes *ServiceRegistry* e *RedisExtension*, cuja finalidade é inicializar e disponibilizar os serviços da aplicação como *singletons* para o *framework Flask*. A classe *ServiceRegistry* registra os serviços *HealthAgentAI* e *VoiceService* como extensões da aplicação, tornando-os acessíveis globalmente durante todo o ciclo de vida da execução. A classe *RedisExtension* disponibiliza o CNES*Service* utilizando o *Redis* como camada de *cache*, configurada com um *TTL* de 24 horas para as consultas de UBS realizadas por código Sistema de Informações do Ministério da Saúde (SIMET*)*. Essa estratégia reduz chamadas repetidas ao *web service SOAP* do *Service-Oriented Architecture*-CNES (*SOA*-CNES), contribuindo para a diminuição da latência das consultas e para a otimização do desempenho da aplicação.

4.4 ARQUITETURA DO SISTEMA

Para este projeto, adotou-se uma arquitetura em camadas (*Layered Architecture*), conforme ilustrado na Figura 5. Esse padrão arquitetural proporciona uma separação mais clara das responsabilidades, facilita a manutenção evolutiva do sistema e reduz o acoplamento entre os componentes. Nesse modelo, cada camada possui um conjunto bem definido de responsabilidades e se comunica exclusivamente com os níveis que respeitam a hierarquia de dependências, estabelecendo um fluxo que parte da camada de apresentação em direção ao domínio.

Figura 5 – Diagrama de arquitetura do sistema



Fonte: elaborada pelo autor (2026).

A arquitetura integra a Camada de Apresentação, responsável pelas interfaces disponibilizadas diretamente ao usuário. Essa camada utiliza *templates* renderizados no lado do servidor (*Server-Side Rendering - SSR*) e arquivos estáticos, caracterizando-se como um nível totalmente passivo da aplicação. Ela é composta por um conjunto de arquivos responsáveis por receber os dados estruturados provenientes das camadas inferiores e apresentá-los conforme as regras de negócio definidas. Dessa forma, toda a lógica de negócio permanece restrita às camadas inferiores da arquitetura.

A Camada de Controle representa o ponto de entrada de todas as requisições encaminhadas ao servidor, independentemente de sua origem. Esse nível arquitetural é organizado em módulos funcionais que agrupam os controladores por domínio, sendo responsável por interpretar as requisições recebidas. Essa camada ainda realiza as verificações das condições de autorização da sessão, aciona os serviços apropriados e, após o processamento, determina o tipo de resposta que será encaminhada à interface. A camada de controle não executa operações de persistência nem gerencia regras de negócio complexas. Sua responsabilidade limita-se à mediação entre as requisições recebidas e a lógica de negócio implementada na camada de serviços.

A Camada de Serviços concentra a inteligência operacional do sistema. Nela residem as regras de negócio, os fluxos de orquestração e as integrações com serviços externos especializados. Esse nível arquitetural concentra a implementação das funcionalidades relacionadas ao PLN, ao cadastro e à autenticação dos usuários junto ao provedor externo de identidade, à consulta ao *SOA*-CNES e ao gerenciamento de grupos segmentados de idosos. Essa organização permite que cada serviço constitua um módulo coeso e de baixo acoplamento, operando sobre abstrações disponibilizadas pelas camadas de persistência e de extensões, sem depender diretamente da camada de controle responsável por sua invocação.

A Camada de Persistência implementa o padrão *Repository*, responsável por abstrair o mecanismo de acesso aos dados por meio de uma interface orientada ao domínio. Esse padrão executa as operações sobre as entidades do sistema sem que as camadas superiores possuam qualquer conhecimento acerca da tecnologia de banco de dados utilizada. Nessa camada, para cada agregado do domínio são

definidos repositórios específicos, garantindo que a lógica de transação e o tratamento dos erros de persistência permaneçam isolados das demais camadas da aplicação.

A Camada de Domínio define os modelos centrais de dados do sistema por meio do *ORM*, representando os conceitos fundamentais do domínio de negócio e constituindo o núcleo estável da aplicação. Destaca-se que essa camada possui caráter exclusivamente declarativo, definindo apenas as estruturas de dados e seus relacionamentos. Em razão dessa característica, não estabelece dependências em relação aos demais níveis arquitetônicos da aplicação.

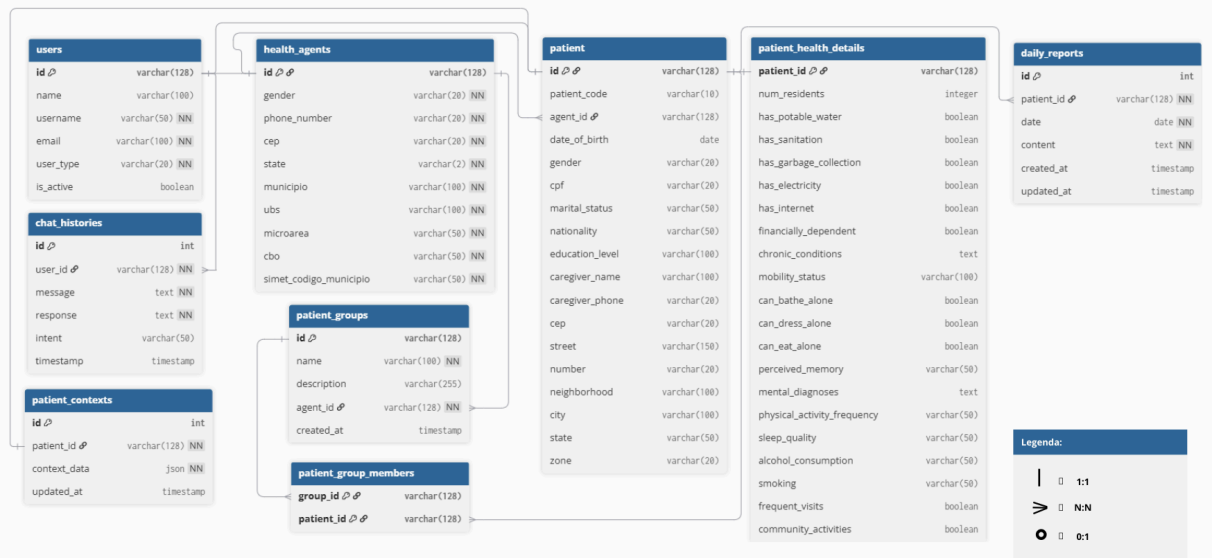
A Camada de Extensões atua como o nível arquitetural responsável pela adaptação à infraestrutura, funcionando como um componente transversal de suporte às camadas de controle, serviços e persistência. Esse fragmento arquitetural reúne os adaptadores encarregados de inicializar e disponibilizar, de forma controlada, as conexões com os serviços externos utilizados pela aplicação. Nessa camada, todos esses componentes são inicializados durante a criação da aplicação, no contexto da *Application Factory*, garantindo que cada extensão seja configurada apenas uma única vez e compartilhada durante todo o ciclo de execução do sistema.

4.5 DIAGRAMA ENTIDADE-RELACIONAMENTO

Nesta subseção é apresentado o Diagrama ER, ilustrado na Figura 6, que constitui um elemento fundamental para o planejamento e o desenvolvimento do *software*, proporcionando a estruturação e a compreensão lógica do banco de dados relacional deste *MVP*. Esse tipo de diagrama evidencia as entidades persistidas, seus atributos, seus relacionamentos e as cardinalidades estabelecidas entre elas.

Para a elaboração da modelagem deste diagrama, foi utilizada a plataforma *web DBDiagram*, que possibilitou a construção do modelo relacional por meio da linguagem *Database Markup Language* (*DBML*). A modelagem utiliza o *PostgreSQL* como sistema gerenciador de banco de dados e o *SQLAlchemy* como camada de *ORM*, de modo que cada entidade representada no diagrama corresponde diretamente a um modelo *Python* definido na aplicação *Flask*.

Figura 6 – Diagrama de entidade-relacionamento



Fonte: elaborada pelo autor (2026).

O modelo relacional da plataforma é composto por oito tabelas organizadas em torno de 3 entidades centrais, *users*, *Patient* e *health_Agents*. A tabela responsável pela identidade do sistema é *users*, que armazena os dados comuns a todos os usuários da aplicação. Essa tabela possui atributos como nome, endereço de e-mail, tipo de perfil e estado de ativação da conta do usuário.

A partir da tabela *users*, é realizada a especialização dos usuários por meio do atributo *user_ype*, originando as tabelas *health_Agents* e *Patient*. Ambas referenciam a tabela *users* por meio de uma chave primária compartilhada do tipo *VARCHAR*, com capacidade para 128 caracteres, correspondente ao identificador único gerado pelo *Firebase Authentication*. Para essa estratégia de modelagem, adotou-se o conceito de herança de tabelas por chave compartilhada, com o objetivo de evitar a duplicação de atributos comuns e garantir a consistência referencial entre os diferentes perfis de usuários.

A tabela *Patient* apresenta a maior cardinalidade do modelo, concentrando dados de identificação, demografia e condições clínicas do usuário. Nessa tabela, o campo *agent_Id* estabelece diretamente o vínculo entre o idoso e o ACS, representando uma associação muitos-para-um. Isso significa que múltiplos idosos podem estar vinculados a um único ACS, enquanto cada idoso pode pertencer a apenas um agente por vez. Além disso, o atributo Cadastro de Pessoas Físicas (CPF) possui restrição de unicidade, garantindo que cada valor seja registrado apenas uma única vez no sistema.

As tabelas *chat_histories*, *daily_reports* e *patient_contexts* dependem diretamente da entidade *Patient*, representando o histórico clínico acumulado ao longo das interações realizadas durante uma sessão ativa de *chat*. A tabela *daily_reports* é responsável por persistir os relatórios clínicos diários gerados automaticamente pelo agente de IA, oferecendo suporte às atualizações incrementais realizadas ao término de cada sessão conversacional por meio dos campos *created_at* e *updated_at*.

Para estruturar os dados obtidos durante a conversa entre o idoso e o modelo de linguagem, foi modelada a tabela *patient_contexts*, que organiza o estado clínico atual do idoso em formato *JSON*, mantendo um único registro por usuário. Essa tabela estabelece uma relação um-para-um com as tabelas *Patient* e *daily_reports* ao encerramento de cada sessão de *chat* com o modelo de linguagem natural.

A tabela *Patient_group* é responsável pela manipulação dos dados segmentados dos grupos de idosos criados pelo ACS. A tabela *Patient_group_members* implementa o relacionamento muitos-para-muitos entre grupos e idosos por meio de uma tabela associativa composta por chaves estrangeiras para ambas as entidades. Essa estrutura permite que um idoso pertença a múltiplos grupos e vice-versa, sem redundância de dados.

Do ponto de vista da integridade dos dados, o modelo estabelece que os atributos obrigatórios de cada entidade sejam definidos pela restrição *NOT NULL*, garantindo o preenchimento das informações essenciais. Em contrapartida, atributos opcionais, como dados complementares de contato e informações clínicas específicas, permanecem configurados para aceitar valores nulos, permitindo maior flexibilidade durante o cadastro dos usuários e o acompanhamento evolutivo dos idosos.

Além disso, foram implementadas restrições de unicidade (*UNIQUE*) para atributos críticos, como CPF, endereço de e-mail e código de identificação do idoso, assegurando a consistência das informações armazenadas. As relações entre as tabelas são mantidas por meio de chaves estrangeiras (*FOREIGN KEY*), garantindo a integridade referencial e impedindo a existência de registros órfãos no banco de dados.

4.6 INTERFACES

Esta subseção descreve as decisões de *design* de interface adotadas no desenvolvimento deste *MVP*. Essas decisões fundamentam a construção das telas a partir de uma metodologia estruturada em um conjunto de diretrizes voltadas à promoção da acessibilidade, da consistência e da facilidade de compreensão pelos diferentes perfis de usuários. Nessa perspectiva, o *frontend* deste *software* é desenvolvido com base nos princípios do sistema de *design* selecionado, que define a paleta de cores, a hierarquia tipográfica e as soluções de *layout* empregadas para atender a interfaces responsivas e acessíveis para a *web*.

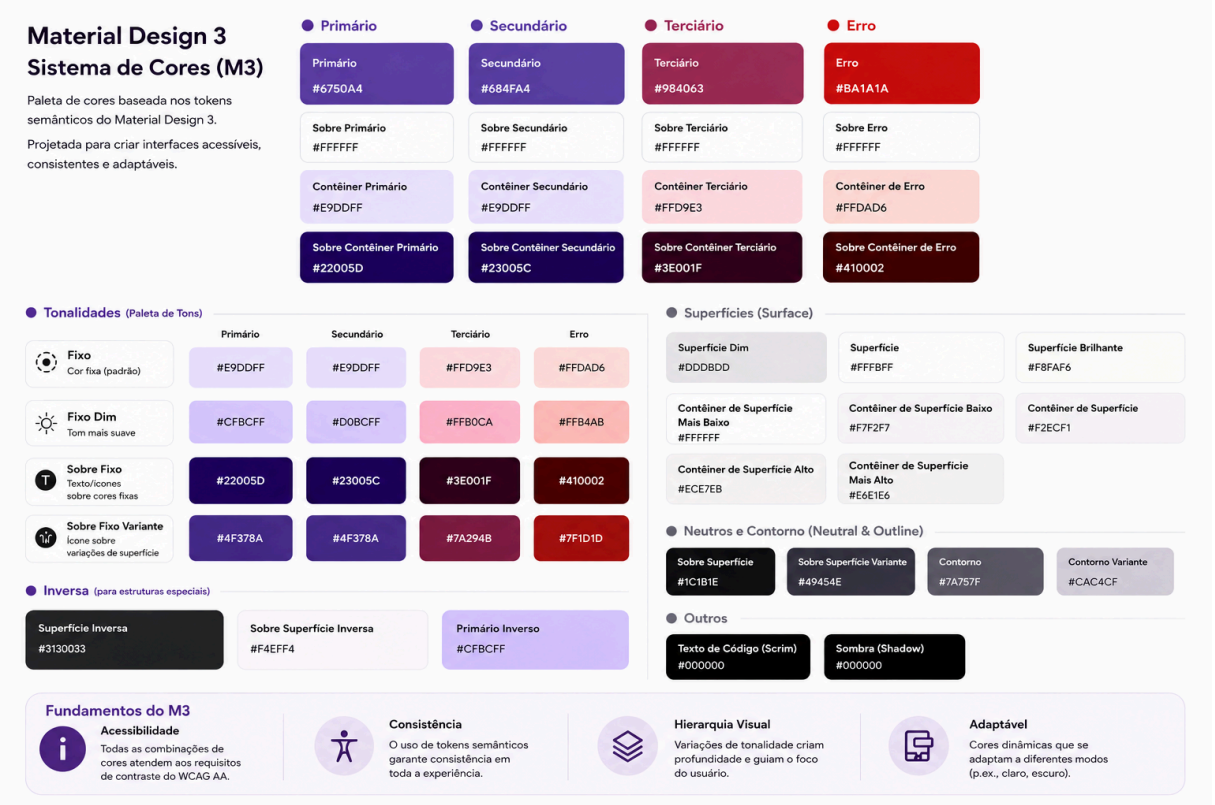
Para a elaboração das telas deste projeto, utilizou-se o *MD3* como documentação de referência para o *design*. Essa documentação disponibiliza especificações para sistemas publicados pelo *Google*, estruturadas em três pilares: personalização, adaptação e acessibilidade como requisitos do projeto de *layout*. Essas especificações fornecem uma linguagem visual coerente por meio de um conjunto de regras para a definição de cores, tipografia, formas dos componentes e comportamentos de estado. Essa documentação permite que as decisões de interface sejam sustentadas por um contrato de *design* documentado e replicável ao longo de todas as telas da aplicação (Google, 2026).

O *MD3* disponibiliza uma arquitetura de cores constituída por um sistema dinâmico de paletas tonais derivadas de uma cor semente gerada pelo algoritmo *Hue, Chroma and Tone* (*HCT*), responsável por produzir tonalidades distribuídas uniformemente em um espectro de 0 a 100 tons e atribuir cores padronizadas aos componentes da interface. Essa atribuição é realizada por meio do mapeamento dos papéis semânticos de cor, representados pelos grupos primário, secundário, terciário, superfície e erro, juntamente com seus respectivos papéis de contraste identificados pelo prefixo “sobre-”. Essa diferenciação cromática garante a legibilidade dos conteúdos sobrepostos e destina-se a comunicar visualmente ações primárias de confirmação, componentes de suporte, notificações de validação negativa e indicadores de estados diferenciados.

A adoção do *MD3* não se restringiu à referência conceitual, mas foi implementada por meio de um conjunto de classes *CSS* com variáveis semânticas que mapeiam os *tokens* de cor utilizando *CSS Custom Properties*, além da tipografia especificada pela documentação oficial (Google, 2026). Essa abordagem permitiu que as telas da plataforma compartilhassem uma identidade visual unificada, apresentando aparência e comportamento consistentes entre os módulos de

autenticação e os *dashboards* do idoso e do ACS. Conforme apresentado na Figura 7, observa-se o padrão de papéis semânticos de cor adotado para as telas de autenticação, cadastro e *dashboard* do idoso.

Figura 7 – Paleta de cores das telas de autenticação, cadastro e *dashboard* do idoso



Fonte: gerada pelo *Figma* e adaptada pelo autor (2026).

As ações de maior relevância da interface são representadas pelo papel primário, caracterizado pela cor roxa de matiz violeta-azulado, derivada da paleta de referência do *MD3* para um esquema claro. Os componentes de suporte, como botões tonais e campos destinados à apresentação dos requisitos de senha, utilizam o papel secundário, representado por uma tonalidade violeta acinzentada. O papel terciário é representado por um tom rosa-vinhoso dessaturado, complementando a paleta com um terceiro elemento cromático destinado a estados diferenciados, sem conotação de erro. Para o tratamento de falhas e de mensagens de validação negativa dos formulários, faz o uso do papel erro, em conformidade com sua função semântica amplamente consolidada. O o papel superfície define a tonalidade base do cartão de autenticação, enquanto as variantes de superfície são destinadas aos

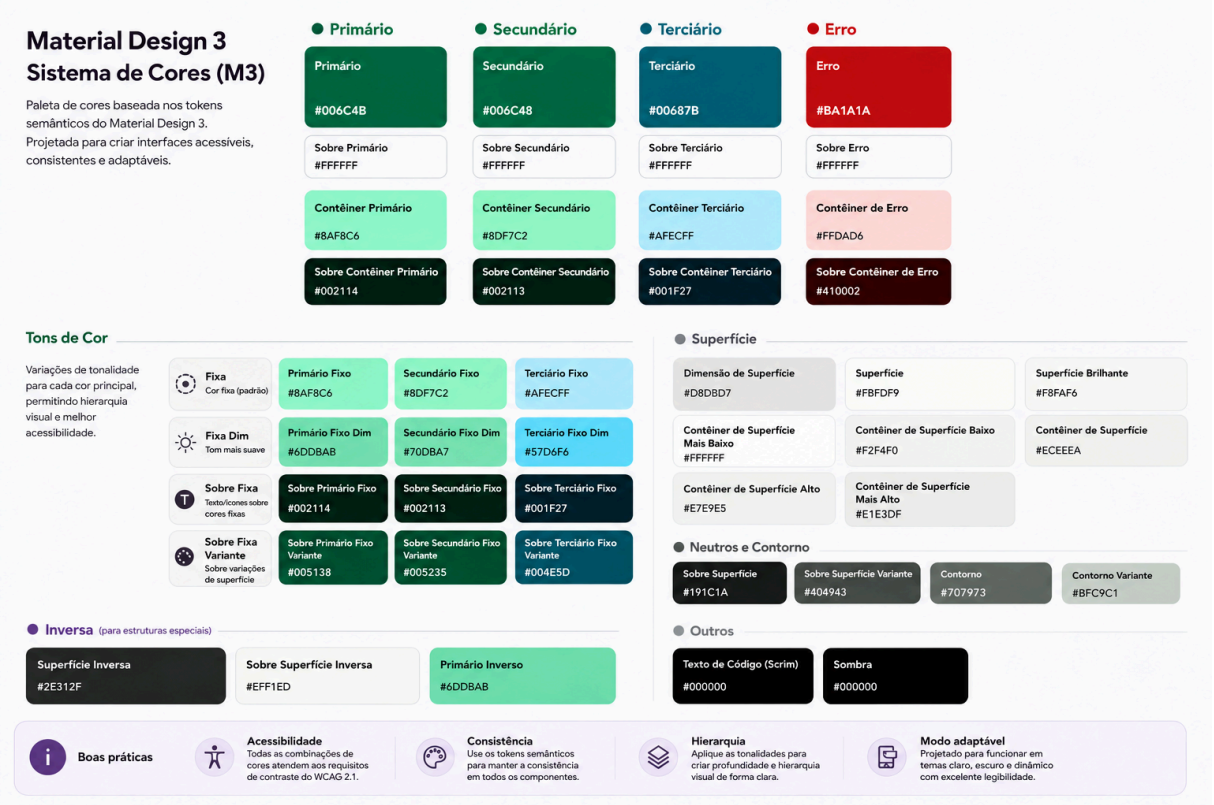
separadores, às bordas suaves e aos fundos dos campos que apresentam os requisitos de senha.

O *dashboard* do idoso adota praticamente os mesmos princípios presentes na paleta de cores apresentada na Figura 7, preservando a continuidade visual da identidade baseada em tons de roxo entre as telas de *login*, cadastro e área principal da aplicação. Essa continuidade foi adotada de forma intencional, uma vez que o usuário percorre um fluxo único desde a autenticação até o acesso ao *dashboard*, sem que ocorra ruptura cromática entre os módulos. Assim como os módulos de autenticação e cadastro, o *dashboard* utiliza os grupos semânticos primário, secundário, terciário e superfície. Como diferencial, o papel erro é empregado para sinalizar funcionalidades de risco ao usuário, como a ação “Deletar Conta”.

O papel secundário apresenta uma diferenciação tonal sutil e proposital em relação ao primário, sendo destinado aos componentes de menor hierarquia visual, como botões tonais, *chips* e itens de navegação. O papel terciário, representado por um tom rosa-vinho quente, é empregado nas animações do assistente virtual durante o *modal* de gravação, conferindo dinamismo cromático à interação por voz e distinguindo-a dos controles estáticos da interface. Já o papel superfície apresenta uma tonalidade levemente mais quente em relação ao módulo de autenticação, adequando-se ao contexto de uso pessoal e ao conceito de bem-estar proposto para essa área da aplicação.

As interfaces do ACS adotam uma paleta cromática independente, cuja cor semente é o verde-esmeralda, processado diretamente pelo algoritmo tonal do *Material Theme Builder*. A adoção dessa distinção entre as paletas para este *MVP* fundamenta-se na necessidade de sinalizar ao ACS que ele se encontra em um ambiente de trabalho distinto, dotado de responsabilidades e ferramentas diferentes daquelas disponibilizadas ao idoso. Além disso, essa escolha é justificada pela forte associação do verde-esmeralda ao imaginário visual relacionado à saúde, ao monitoramento clínico e aos sistemas de gestão hospitalar, reforçando o caráter profissional do módulo. A Figura 8 apresenta a declaração dos grupos semânticos de cores primárias, secundárias, terciárias, superfícies e erros empregados na interface do ACS.

Figura 8 – Paleta de cores das telas de *dashboard* do ACS



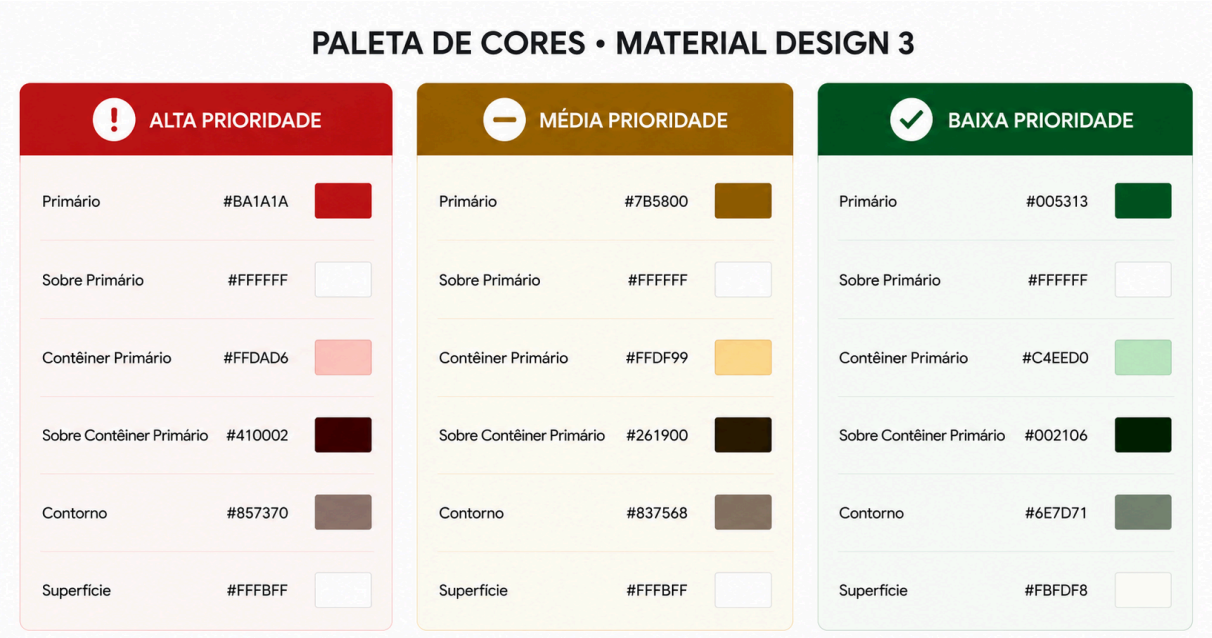
Fonte: gerada pelo *Figma* e adaptada pelo autor (2026).

Nesta paleta, o papel primário é aplicado aos ícones de destaque da barra lateral de navegação, aos títulos das seções do *dashboard* e aos botões de ação principal, como os destinados à geração de relatórios e ao acionamento da triagem de idosos. O contêiner primário é utilizado no item ativo da barra lateral de navegação e no ícone do perfil do ACS. O papel secundário mantém intencionalmente uma relação cromática muito próxima ao primário, atuando como suporte visual para botões tonais e controles secundários sem competir pela atenção do usuário. O papel terciário, representado por um matiz azul-*teal* clínico, é empregado para produzir contraste complementar em recursos de interação e distintivos específicos que exigem destaque fora do fluxo principal de ações. O grupo superfície apresenta uma tonalidade de branco levemente esverdeada, conferindo coerência cromática ao esquema visual sem introduzir contraste excessivo entre o plano de fundo e os cartões sobrepostos.

Além dos papéis canônicos de cor definidos pelo *MD3*, este trabalho propõe uma extensão semântica destinada aos indicadores de triagem clínica gerados pelo *pipeline* de IA. Na Figura 9, os níveis de prioridade de atendimento são representados por meio de emblemas coloridos. Essas cores foram derivadas

analogamente ao sistema tonal do *MD3*, no qual cada cor de fundo corresponde a um tom de contêiner e a respectiva cor do texto corresponde ao tom sobreposto de alta legibilidade.

Figura 9 – Paleta de cores para os indicadores de triagem clínica



Fonte: gerada pelo *Figma* e adaptada pelo autor (2026).

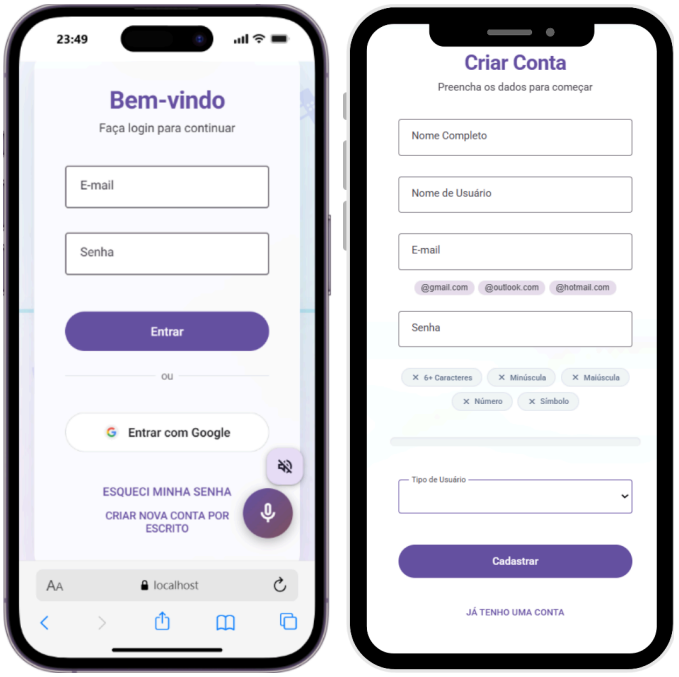
O indicador de prioridade alta é mapeado ao território semântico do papel erro, utilizando um tom rosado de contêiner combinado a texto em vermelho escuro, reforçando visualmente o caráter de urgência clínica. O indicador de prioridade média emprega um tom âmbar claro associado a texto em ocre-escuro, comunicando um estado de atenção sem transmitir a urgência representada pelo vermelho.

Já o indicador de prioridade baixa utiliza um tom verde-menta suave com texto em verde-floresta, sinalizando estabilidade clínica e ausência de necessidade imediata de intervenção. Essas composições colorimétricas dos emblemas não representam uma ruptura estética em relação aos tokens do *MD3*; ao contrário, constituem uma extensão semântica coerente com os princípios de mapeamento funcional estabelecidos pela especificação.

Como é mostrado na Figura 10, a interface de *login* é composta por um cartão central de autenticação, destacado sobre um plano de fundo animado com elementos visuais relacionados à área da saúde em movimentação contínua. Esses

elementos contextualizam o domínio da aplicação e conferem dinamismo à interface sem comprometer a legibilidade do formulário. Em relação aos componentes visuais, o *card* de autenticação disponibiliza acesso por meio de e-mail e senha, autenticação utilizando o *Google*, redefinição de senha, criação de uma nova conta e um botão de navegação acessível por voz.

Figura 10 – Telas de *login* e cadastro



Fonte: elaborada pelo autor (2026).

O processo de cadastro é estruturado em dois *layouts* distintos, em razão da existência de dois perfis de usuários que demandam conjuntos específicos de informações. Inicialmente, o formulário coleta os dados básicos necessários, como nome completo, nome de usuário, endereço de e-mail, senha e tipo de usuário. Após a seleção do perfil correspondente, o usuário é direcionado para uma etapa complementar, na qual é apresentado um formulário específico para o perfil selecionado, sendo um destinado ao preenchimento dos dados do idoso e outro voltado ao ACS.

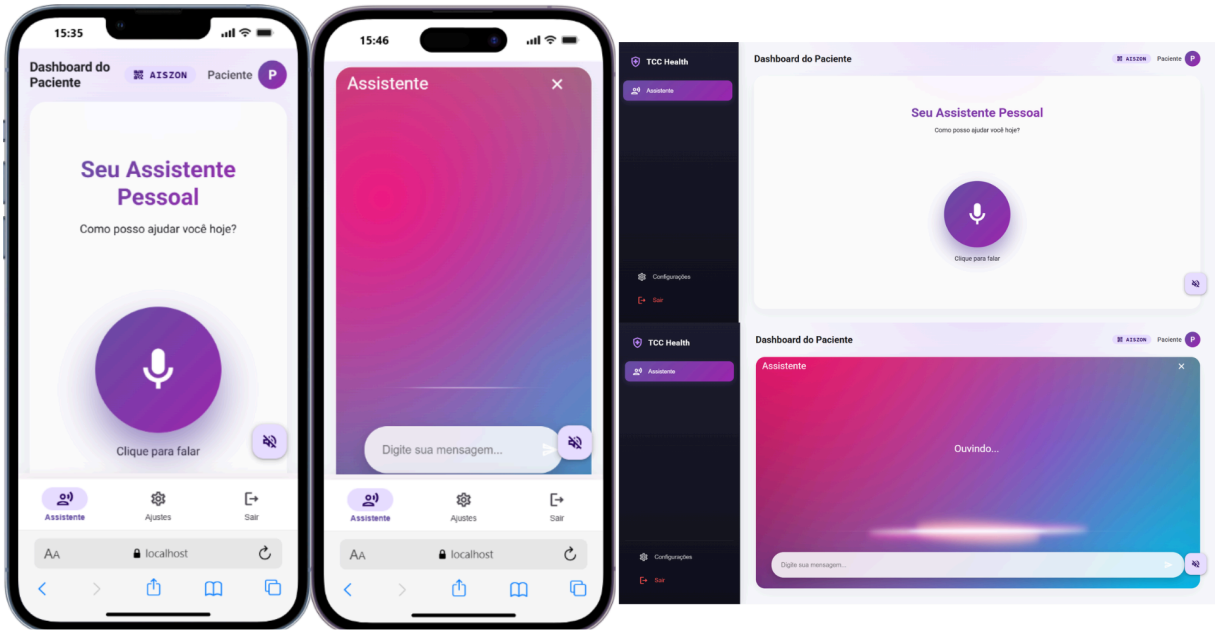
A tela de pré-cadastro disponibiliza recursos voltados à agilidade de uso, como botões de atalho para os domínios de e-mail mais frequentes que, quando acionados, completam automaticamente o sufixo do endereço informado pelo usuário. Outro componente de destaque é o medidor de força da senha. Esse recurso avalia a credencial em tempo real com base em cinco critérios de

segurança, por meio de animações implementadas em *JavaScript*, apresentando-os como *chips* indicadores, cada um acompanhado de um ícone de estado e da respectiva cor correspondente ao nível de robustez da senha.

O campo de seleção “Tipo de Usuário” determina o perfil do cadastro, oferecendo duas opções: Paciente e ACS. Essa distinção é fundamental para o sistema, pois condiciona os fluxos posteriores ao cadastro e os níveis de acesso às funcionalidades da plataforma. O *link* “Já tenho uma conta” direciona o usuário novamente à tela de *login*, concluindo o ciclo de navegação entre os dois fluxos de autenticação. Esse conjunto de fluxos entre as telas de autenticação e cadastro demonstra aderência aos princípios do *design* centrado no usuário, apresentando hierarquia visual clara, *feedback* contextual e elementos visuais que reforçam a identidade temática do sistema voltado à saúde comunitária.

As interfaces do *dashboard* do idoso, ilustradas na Figura 11, apresentam a área de interação com o agente de IA sob duas perspectivas complementares: a versão responsiva para dispositivos móveis e para *desktops*. Essa composição evidencia a estratégia de *design* responsivo adotada durante o desenvolvimento das telas, na qual o mesmo conjunto de componentes e funcionalidades adapta-se às dimensões e às capacidades de interação de cada dispositivo.

Figura 11 – Telas *web* do *dashboard* do idoso responsivas para dispositivos *mobile-first* e *desktop*



Fonte: elaborada pelo autor (2026).

Para dispositivos de mesa, o *dashboard* do idoso é organizado por meio de uma barra lateral fixa posicionada à esquerda, com fundo escuro, concentrando os itens de navegação, a logomarca da aplicação, as opções de configuração e o encerramento da sessão. Além disso, o *dashboard* incorpora a área destinada ao agente de IA, composta por um plano de fundo claro e por um *card* central responsável por renderizar o conteúdo principal da interação.

Em dispositivos com largura inferior a 992 *pixels*, a barra lateral é completamente ocultada e substituída por uma barra de navegação inferior fixa, posicionada junto ao rodapé da tela. Os itens de navegação passam a ser organizados horizontalmente, apresentando ícones posicionados acima dos respectivos rótulos textuais. Essa decisão de *design* segue o padrão de *Bottom Navigation* do *MD3*, no qual o item ativo é destacado por uma pílula preenchida com a cor do contêiner primário. Essa alternância de padrões de navegação garante que a interface permaneça igualmente utilizável em telas reduzidas, sem exigir redimensionamento forçado ou ampliação da página, atendendo às diretrizes de acessibilidade.

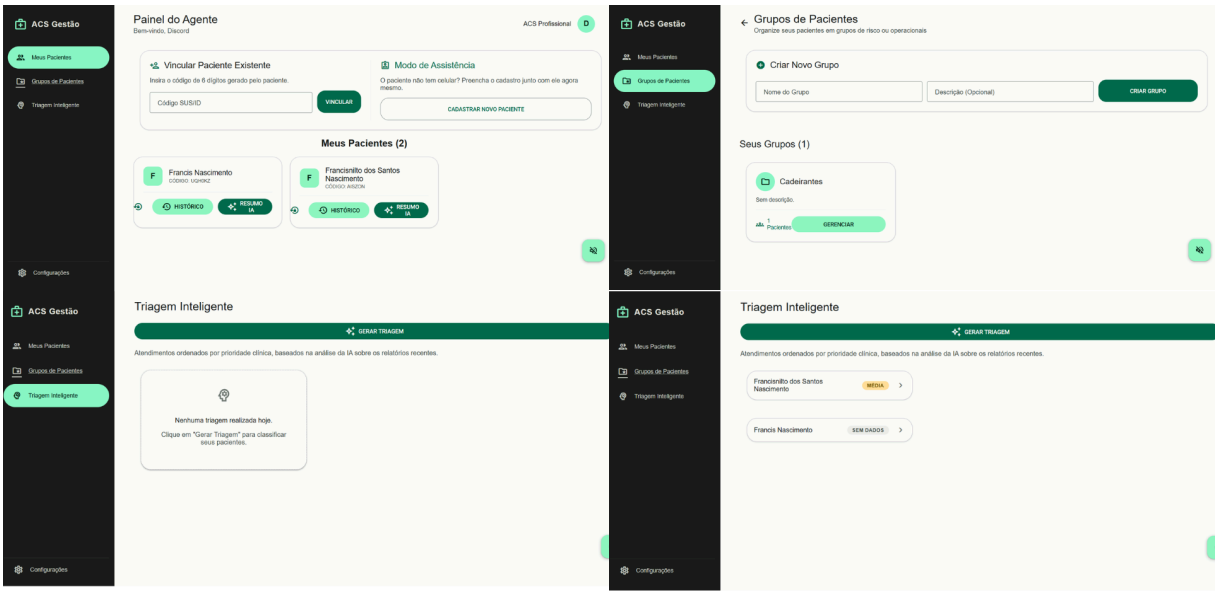
O assistente inteligente por voz está posicionado na área central do *dashboard* e utiliza um botão do tipo *Floating Action Button* (*FAB*) como elemento principal para ativação do agente de IA. Esse botão é preenchido pela cor primária da aplicação e apresenta um ícone de microfone centralizado. Para reforçar visualmente seu estado de disponibilidade, o componente utiliza uma animação contínua de pulso suave acompanhada de uma sombra expansiva ao seu redor, transmitindo ao usuário a percepção de que o sistema permanece ativo e preparado para receber interações.

Ao acionar o botão do microfone, como demonstrado na segunda e na quarta capturas da Figura 9, um *modal* de interação é exibido sobre a interface principal, ocupando integralmente a área destinada ao diálogo com o agente de IA. Nesse estado, o plano de fundo do *modal* passa a utilizar uma composição formada por múltiplos gradientes radiais animados, combinando seis cores (rosa, ciano, violeta, laranja, azul e verde) sobre uma base escura com animação contínua de 25 segundos. A combinação dessas cores em uma interface dinâmica promove um deslocamento contínuo dos gradientes, produzindo a percepção de um plano de fundo vivo e fluido.

Abaixo da área de transcrição textual da IA, um componente animado em ondas de áudio representa visualmente o estado de escuta ativa, tornando-se opaco e dinâmico quando o microfone está em funcionamento e permanecendo discreto durante o estado ocioso. Como complemento, foi integrada uma barra de entrada de texto com efeito de vidro fosco, acompanhada de um botão circular de envio, compondo o *modal* e possibilitando ao usuário interagir com o agente tanto por voz quanto por digitação, de acordo com sua preferência ou necessidade.

A interface do *dashboard* do ACS, juntamente com as principais telas de suas funcionalidades, é apresentada nas Figuras 12 e 13, que reúnem oito *screenshots*. O painel do ACS disponibiliza funcionalidades voltadas ao gerenciamento de idosos, à criação de grupos segmentados por condições crônicas, à geração de relatórios e à triagem inteligente de idosos para o auxílio na identificação de atendimento domiciliar prioritário.

Figura 12 – Interface do *dashboard* do ACS



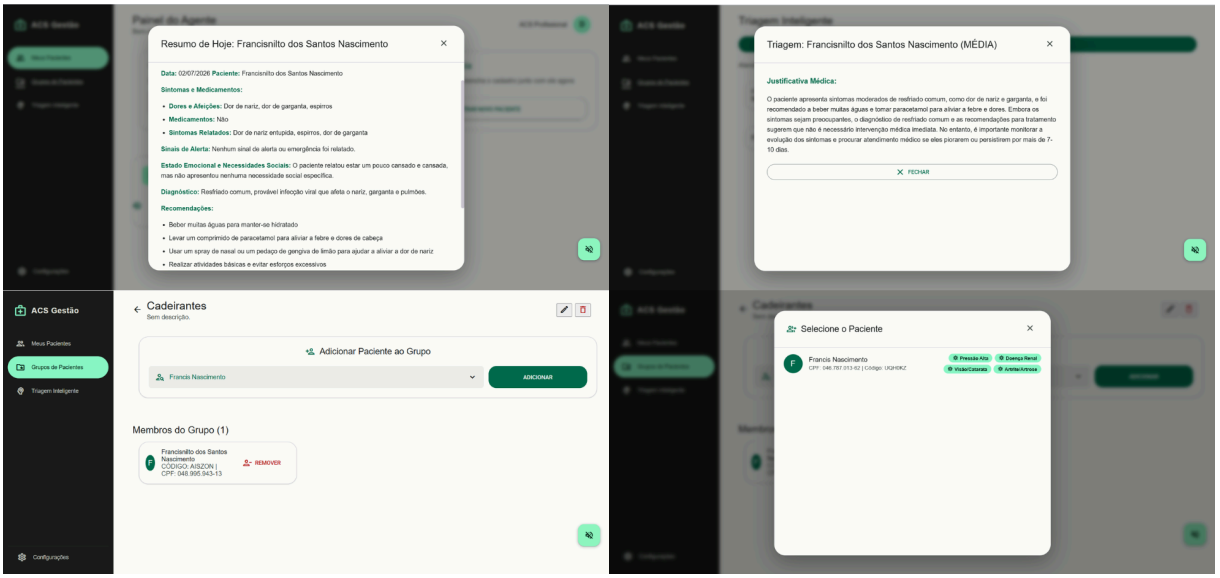
Fonte: elaborada pelo autor (2026).

A barra lateral fixa apresenta um fundo escuro sólido, sobre o qual os elementos textuais são exibidos com contraste adequado para garantir sua legibilidade. Os itens de navegação ativos são destacados por meio do contêiner primário, seguindo o padrão do componente *Navigation Drawer* do *MD3*, enquanto os itens inativos apresentam um estado sutil de *hover*. A área de atuação do ACS é composta por um conjunto de funcionalidades que, ao longo do uso diário da

plataforma, contribuem para a reorganização dinâmica do *layout*, incorporando novos elementos e componentes previstos pelo sistema de *design* adotado.

Como complemento às telas apresentadas, a Figura 13 ilustra os *modais* que materializam decisões específicas de *design*, orientadas pela necessidade de apresentar informações clínicas de elevada densidade de forma clara, reduzindo simultaneamente o número de interações necessárias para a execução das operações recorrentes do ACS.

Figura 13 – Interfaces das funcionalidades principais presentes no *dashboard* do ACS



Fonte: elaborada pelo autor (2026).

Os conteúdos apresentados nos *modais* clínicos são formatados em *Markdown*, tecnologia que contribui para uma estrutura tipográfica rica, composta por títulos em diferentes níveis hierárquicos, destacados com a cor primária do tema, listas estruturadas e palavras em negrito. Esses *modais* oferecem acesso imediato às informações clínicas, reduzindo a necessidade de múltiplos cliques até a localização do conteúdo desejado. Essa abordagem de *design* contribui para a redução da carga cognitiva do ACS e favorece uma curadoria mais consciente na organização das informações clínicas e no gerenciamento dos grupos de idosos.

5 SOFTWARE

O código-fonte integral deste *MVP* está disponível publicamente por meio de um repositório *Git* hospedado na plataforma *GitHub*. Nesse repositório encontram-se todos os arquivos de configuração, módulos da aplicação *Flask*, arquivos de *template*, folhas de estilo, *scripts* de teste e os arquivos *Docker* necessários para a reprodução completa do ambiente de desenvolvimento e execução. O acesso ao repositório pode ser realizado por meio do *link* a seguir: https://github.com/jyunior245/TCC-Workspace.git.

5.1 IMPLANTAÇÃO

Para o desenvolvimento do sistema, optou-se por uma estratégia que orquestra toda a arquitetura de *software* por meio de contêineres *Docker*. Por meio dessa metodologia, foi possível garantir a portabilidade e a reprodutibilidade do ambiente de execução, independentemente do sistema operacional hospedeiro e dos desenvolvedores. A seguir são descritos os procedimentos necessários para a instalação das dependências e a inicialização do servidor *Flask* executado dentro de um contêiner *Docker*.

O *backend* da aplicação foi desenvolvido na linguagem *Python*, juntamente com as bibliotecas necessárias para sua execução, listadas no arquivo *requirements.txt*, localizado no diretório raiz do projeto. Embora seja possível realizar a instalação manual das dependências em um ambiente virtual *Python*, a implantação da solução foi concebida para ocorrer por meio de contêineres *Docker*, garantindo maior reprodutibilidade, isolamento do ambiente e simplificação do processo de configuração.

Recomenda-se que a inicialização da aplicação seja realizada utilizando os arquivos de configuração do *Docker* disponibilizados no projeto, seguindo os procedimentos descritos na documentação oficial do *Docker*[1]. Essa abordagem assegura que todas as dependências e os serviços necessários sejam configurados automaticamente, reduzindo inconsistências entre diferentes ambientes de desenvolvimento e produção.

[1] Documentação oficial do Docker: Docker Compose, primeiros passos. Disponível em: https://docs.docker.com/compose/gettingstarted.

Para o empacotamento da aplicação, foi utilizado o *framework Capacitor*, desenvolvido pela *Ionic*, responsável por converter aplicações *web* em aplicativos nativos para a plataforma *Android* por meio de uma camada de *web view*. A escolha dessa tecnologia justifica-se pela possibilidade de reutilizar integralmente a interface desenvolvida para o ambiente de navegadores, eliminando a necessidade de implementar uma versão nativa da aplicação destinada às versões *debug*.

Essa escolha tecnológica também se justifica pela maior facilidade na distribuição de atualizações, pois modificações realizadas na aplicação *web* tornam-se imediatamente disponíveis aos dispositivos móveis. Essa flexibilização da disponibilidade dos novos recursos minimiza a necessidade de recompilar ou redistribuir um novo arquivo *APK*, exceto quando houver alterações específicas na camada nativa.

Para compreender de forma detalhada o fluxo de execução do *Capacitor* neste projeto, recomenda-se a consulta à documentação oficial do *Capacitor*[2], utilizada como base para a integração entre a aplicação *web* e as plataformas nativas. A documentação apresenta os comandos da interface de linha de comando, o processo de sincronização do projeto e a geração do *APK* em versão *debug*. Por meio dessa consulta, é possível auxiliar na compreensão do funcionamento interno e da implantação do *MVP*.

É importante destacar que o *APK* gerado opera em modo *web view*, ou seja, a aplicação *Android* funciona como um navegador embutido que carrega, em tempo real, as interfaces do servidor *Flask*. Nessa perspectiva, o *APK* não encapsula a lógica da aplicação, mas consome remotamente os recursos disponibilizados pelo servidor. Assim, o servidor *Flask* deve permanecer em execução e acessível pela rede local ou externa, utilizando o endereço *Internet Protocol* (*IP*) da máquina hospedeira configurado no arquivo *capacitor.config.json*, para garantir o funcionamento correto do *APK*.

Contudo, a atual implementação, que utiliza um ambiente *web view* dependente da disponibilidade de um servidor *Flask* na rede local, apresenta limitações estruturais para o público-alvo do projeto. Dada a proposta de acessibilidade a idosos em situação de vulnerabilidade social com restrições de mobilidade domiciliar, a dependência de infraestrutura de rede robusta para comunicação com o backend torna-se uma barreira operacional.

---

[2] Documentação oficial do Capacitor. Disponível em: https://capacitorjs.com/docs/.

Para garantir a viabilidade técnica e a escalabilidade do *MVP* em cenários reais de acompanhamento domiciliar, recomenda-se a migração da arquitetura para um desenvolvimento orientado a dispositivos móveis, utilizando *frameworks* nativos ou tecnologias robustas dedicadas ao desenvolvimento *mobile*, como *flutter* ou *React Native*. Essa transição permitiria otimizar a experiência do usuário, melhorar o desempenho da aplicação e reduzir a dependência de conectividade local constante, alinhando a implementação técnica à proposta de valor central do projeto.

5.2 TESTES

Esta seção descreve a estratégia de testes adotada no desenvolvimento da aplicação, apresentando os tipos de teste utilizados, as ferramentas empregadas e as evidências dos casos de teste executados. Optou-se pela aplicação de testes unitários com o propósito de verificar o comportamento isolado de cada função da camada de serviços do sistema, independentemente de infraestruturas externas, como bancos de dados, serviços de autenticação ou modelos de IA (Fowler, 2018).

Para a execução dos testes, foi utilizado o *framework Pytest*, destinado ao desenvolvimento orientado a testes no ecossistema *Python*. Essa ferramenta foi empregada em conjunto com o *plugin pytest-mock*, que disponibiliza o recurso *mocker* para auxiliar na substituição de dependências externas por objetos simulados, conhecidos como *mocks*. Além disso, a biblioteca padrão *unittest.mock* foi utilizada na criação de objetos *MagicMock*, que consistem em uma subclasse da classe *Mock*, responsável por implementar automaticamente métodos especiais da linguagem, facilitando a simulação do comportamento de objetos durante a execução dos testes.

A Tabela 2 apresenta a cobertura de testes das funcionalidades implementadas na camada de serviço de autenticação, responsável pelo gerenciamento das operações de identidade da aplicação por meio do *Firebase*. Considerando o caráter crítico desse componente para a segurança e o controle de acesso ao sistema, a suíte de testes foi estruturada para validar tanto os fluxos de execução bem-sucedidos quanto os cenários de exceção, contemplando operações de autenticação, gerenciamento de credenciais, recuperação de acesso e tratamento de falhas provenientes de serviços externos. Essa cobertura contribui para assegurar a confiabilidade, a estabilidade e a previsibilidade do serviço,

garantindo que os erros sejam tratados de forma controlada e que as funcionalidades atendem ao comportamento especificado, mesmo diante de condições adversas.

Tabela 2 – Casos de teste para o serviço de autenticação

Caso de Teste	**Resultado Esperado**	**Status**
Tratamento de erro do *Firebase Admin SDK*	Erro	OK
Tratamento de senha fraca	Erro	OK
Tratamento de e-mail já cadastrado	Erro	OK
Tratamento de e-mail inválido	Erro	OK
Tratamento de senha incorreta	Erro	OK
Tratamento de usuário inexistente	Erro	OK
Tratamento de código de erro desconhecido	Erro	OK
Tratamento de exceção genérica do *Python*	Erro	OK
Criação de usuário com e-mail já cadastrado	Erro	OK
Criação de usuário via *Admin SDK*	Sucesso	OK
Autenticação com credenciais válidas	Sucesso	OK
Autenticação com credenciais inválidas	Erro	OK
Exclusão de conta por *idToken*	Sucesso	OK
Atualização de e-mail via *Admin SDK*	Sucesso	OK
Envio de e-mail para recuperação de senha	Sucesso	OK
Envio de e-mail de recuperação para usuário inexistente	Erro	OK
Atualização de senha via *Admin SDK*	Sucesso	OK
Atualização de senha sem *Admin SDK*	Erro	OK
Atualização de senha com falha no	Erro	OK

Firebase		
Exclusão de conta por Identificador de Usuário (*User Identifier - UID*) via *Admin SDK*	Sucesso	OK
Exclusão de conta por *UID* sem *Admin SDK*	Erro	OK
Exclusão de conta por *UID* com falha no *Firebase*	Erro	OK
Criação de usuário via *Admin SDK* com falha no *Firebase*	Erro	OK
Atualização de e-mail sem *Admin SDK*	Erro	OK
Atualização de e-mail com falha no *Firebase*	Erro	OK
Verificação de e-mail para novo endereço sem *Admin SDK*	Erro	OK
Verificação de e-mail para novo endereço sem *API Key*	Erro	OK
Verificação de e-mail para novo endereço com falha na primeira requisição *POST*	Erro	OK
Verificação de e-mail para novo endereço com falha na segunda requisição *POST*	Erro	OK
Envio inicial de e-mail de verificação (duas requisições *REST*)	Sucesso	OK
Envio inicial de e-mail de verificação sem *Admin SDK*	Erro	OK
Envio inicial de e-mail de verificação sem *API Key*	Erro	OK
Envio inicial de e-mail de verificação com falha na primeira requisição *POST*	Erro	OK
Envio inicial de e-mail de verificação com falha na segunda requisição *POST*	Erro	OK
Exclusão por *idToken* com resiliência e sem propagação da exceção	Erro	OK

Fonte: elaborada pelo autor (2026).

A Tabela 3 apresenta a cobertura de testes do serviço de cadastro, responsável por orquestrar o fluxo de registro, a ativação de contas e a validação das informações cadastrais dos usuários. A suíte de testes foi estruturada para verificar as principais regras de negócio associadas ao processo de cadastro, incluindo desde a validação do estado da conta no *Firebase Authentication* até o mapeamento dos dados entre a camada de apresentação e o domínio da aplicação. Os resultados obtidos por essa cobertura são fundamentais para assegurar a integridade dos dados, a consistência das regras de ativação de usuários e a confiabilidade do processo de cadastro. Assim, essa cobertura contribui para reduzir a ocorrência de inconsistências cadastrais e garantir o correto funcionamento desse serviço essencial ao sistema.

Tabela 3 – Casos de teste para o serviço de cadastro

Caso de Teste	**Resultado Esperado**	**Status**
Ativação de usuário com (@tcchealth.com)	Sucesso	OK
Ativação de usuário com e-mail verificado	Sucesso	OK
Bloqueio de usuário com e-mail não verificado	Erro	OK
Validação de perfil inexistente no banco de dados	Erro	OK
Cadastro de idoso com perfil completo	Sucesso	OK
Cadastro de ACS com perfil incompleto	Erro	OK
Verificação de pendência com e-mail verificado institucional	Sucesso	OK
Verificação de pendência com e-mail ainda não verificado	Erro	OK
Conclusão do perfil de idoso (mapeamento do formulário)	Sucesso	OK
Conclusão do perfil de ACS (mapeamento do formulário)	Sucesso	OK

Fonte: elaborada pelo autor (2026).

A Tabela 4 apresenta a cobertura de testes do serviço responsável pelo vínculo de idosos ao *dashboard* do ACS, funcionalidade essencial para o gerenciamento da carteira de atendimentos sob sua responsabilidade. A suíte de testes foi estruturada para validar os principais cenários de sucesso e falha relacionados ao processo de vinculação, contemplando a verificação da existência do idoso, das regras de exclusividade do vínculo e da persistência das informações no banco de dados. Essa cobertura de testes é fundamental para assegurar a integridade dos dados, a correta aplicação das regras de negócio e a consistência das relações entre ambos os usuários, contribuindo para a confiabilidade e a segurança das informações gerenciadas pelo sistema.

Tabela 4 – Casos de teste para o vínculo de idosos

Caso de Teste	**Resultado Esperado**	**Status**
Vínculo de idoso com código válido e sem vínculo prévio	Sucesso	OK
Vínculo de idoso com código inválido	Erro	OK
Vínculo de idoso já associado a outro ACS	Erro	OK
Vínculo de idoso já associado ao próprio ACS	Erro	OK

Fonte: elaborada pelo autor (2026).

A Tabela 5 apresenta a cobertura de testes do serviço de segmentação de idosos por grupos de condições crônicas, funcionalidade destinada a apoiar a organização e o acompanhamento clínico realizados pelo ACS. A suíte de testes foi estruturada para validar as principais operações do serviço, incluindo a criação, a consulta, a atualização e a exclusão de grupos, bem como a associação e a desassociação de idosos, contemplando tanto os fluxos de sucesso quanto os cenários de falha e a validação das regras de negócio. Essa cobertura de testes é essencial para garantir a consistência das informações, a integridade das operações de gerenciamento dos grupos e a confiabilidade do serviço, assegurando que a

segmentação dos idosos ocorra em conformidade com os requisitos funcionais definidos para a aplicação.

Tabela 5 – Casos de teste para a segmentação de idoso por grupos

Caso de Teste	Resultado Esperado	Status
Criação de grupo com nome vazio ou em branco	Erro	OK
Criação de grupo com dados válidos	Sucesso	OK
Listagem de grupos cadastrados	Sucesso	OK
Remoção de idoso do grupo	Sucesso	OK
Inclusão de idoso no grupo	Sucesso	OK
Atualização do nome do grupo	Sucesso	OK
Atualização da descrição do grupo sem alteração do nome	Sucesso	OK
Exclusão de grupo existente	Sucesso	OK

Fonte: elaborada pelo autor (2026).

A Tabela 6 apresenta a cobertura de testes do serviço de IA, responsável pelo processamento das interações entre idosos e o agente conversacional, pela classificação da intenção das mensagens, pela geração de relatórios clínicos e pela manutenção do contexto persistido das conversas. Em razão de concentrar as funcionalidades mais críticas do sistema, a suíte de testes foi planejada para validar seus principais fluxos funcionais e os mecanismos de tratamento de exceções, contemplando tanto os cenários de operação normal quanto as condições de falha. Essa cobertura é fundamental para assegurar a confiabilidade das respostas produzidas, a segurança no processamento das informações clínicas, a integridade dos dados persistidos e a robustez operacional da aplicação, mesmo diante de falhas de comunicação com serviços externos.

Tabela 6 – Casos de teste para o serviço de IA

Caso de Teste	Resultado Esperado	Status
Classificação de mensagem como saudação utilizando *fast path*	Sucesso	OK

Classificação de mensagem como emergência utilizando *fallback*	Sucesso	OK
Classificação de mensagem como dúvida em saúde utilizando *fallback*	Sucesso	OK
Classificação de mensagem como outros assuntos utilizando *fallback*	Sucesso	OK
Classificação de mensagem por *embeddings* com *RAG* ativo	Sucesso	OK
Triagem clínica com resposta *JSON* válida	Sucesso	OK
Triagem clínica com resposta *JSON* malformada	Erro	OK
Triagem clínica com resposta *JSON* válida e chaves ausentes	Erro	OK
Triagem clínica com falha total da *API* (exceção)	Erro	OK
Geração de relatório clínico diário sem interações	Sucesso	OK
Geração de relatório clínico diário com sucesso	Sucesso	OK
Atualização de relatório clínico diário sem novas interações	Sucesso	OK
Atualização de relatório clínico diário com novas interações	Sucesso	OK
Persistência do relatório clínico diário com falha após a geração	Erro	OK
Construção do contexto clínico sem histórico de *chats*	Sucesso	OK
Construção do contexto clínico com histórico de *chats*	Sucesso	OK
Atualização de contexto clínico existente	Sucesso	OK
Construção do contexto clínico com resposta *JSON* inválida e *fallback* seguro	Erro	OK
Fluxo principal (*get_response*) com	Sucesso	OK

pipeline completo		
Fluxo principal (*get_response*) para saudação sem acionamento do *RAG*	Sucesso	OK
Fluxo principal (*get_response*) para emergência com acionamento do *RAG*	Sucesso	OK
Fluxo principal (*get_response*) sem *user_id* (sem persistência)	Sucesso	OK

Fonte: elaborada pelo autor (2026).

A Tabela 7 apresenta a cobertura de testes do serviço de *RAG*, responsável pela indexação dos documentos de saúde em um banco vetorial e pela recuperação de informações contextuais utilizadas para enriquecer as respostas produzidas pelo agente de IA. A suíte de testes foi estruturada para validar os principais fluxos funcionais do serviço, contemplando as operações de indexação, recuperação dos documentos e os mecanismos de degradação controlada em situações de indisponibilidade ou falhas da infraestrutura vetorial. Essa cobertura de testes é essencial para assegurar a confiabilidade da recuperação de informações, a integridade dos dados indexados e a robustez do processo de enriquecimento contextual. Por meio dessa abordagem, confirmou-se que eventuais falhas na camada de recuperação de conhecimento não comprometem a estabilidade nem o funcionamento do agente inteligente.

Tabela 7 – Casos de teste para o serviço de *RAG*

Caso de Teste	**Resultado Esperado**	**Status**
Indexação de documento no banco vetorial (*ChromaDB*)	Sucesso	OK
Consulta simples ao banco vetorial	Sucesso	OK
Consulta com recuperação de referências bibliográficas	Sucesso	OK
Consulta em modo offline (*offline_mode*)	Erro	OK
Consulta com exceção no ChromaDB (*fallback*)	Erro	OK

Fonte: elaborada pelo autor (2026).

Os testes dos serviços de CNES, responsáveis pela consulta de UBS e pela busca do CBO do usuário da saúde, apresentados na Tabela 8, foram estruturados para validar os principais fluxos funcionais desses serviços, abrangendo a estratégia de resiliência empregada nas consultas ao CNES e ao arquivo *Comma-Separated Values* (*CSV*) local. Essa estratégia baseia-se em múltiplas camadas para a recuperação de dados e das operações de consulta ao CBO, bem como nos mecanismos de tratamento de falhas associados à indisponibilidade de recursos externos e locais. Dessa forma, o planejamento desses testes é fundamental para assegurar a confiabilidade das consultas, a continuidade operacional diante de falhas de infraestrutura e a consistência das informações disponibilizadas aos usuários, contribuindo para a robustez e a previsibilidade do serviço.

Tabela 8 – Casos de teste para os serviços de CNES e CBO

Caso de Teste	**Resultado Esperado**	**Status**
Consulta de UBS utilizando *cache* (*Redis*)	Sucesso	OK
Consulta de UBS por meio da *API SOAP* do CNES	Sucesso	OK
Consulta de UBS com *fallback* para arquivo *CSV* (*API* indisponível)	Erro	OK
Consulta local de CBO com código válido	Sucesso	OK
Consulta local de CBO com código inexistente	Erro	OK
Consulta local de CBO com arquivo *CSV* inexistente	Erro	OK

Fonte: elaborada pelo autor (2026).

A Tabela 9 apresenta a cobertura da suíte de testes do serviço de voz, responsável por validar os principais fluxos funcionais do serviço, contemplando o pré-processamento das respostas textuais, a geração de áudio sintetizado, o tratamento de conteúdos inválidos e os mecanismos de tratamento de exceções durante a comunicação com o mecanismo de síntese de voz. Essa cobertura de

testes é fundamental para assegurar a confiabilidade do processo de conversão entre texto e fala, a integridade do conteúdo disponibilizado ao usuário e a robustez operacional do serviço, garantindo a continuidade da funcionalidade mesmo diante de falhas em dependências externas.

Tabela 9 – Casos de teste para o serviço de voz

Caso de Teste	**Resultado Esperado**	**Status**
Sanitização de texto com prefixos e referências	Sucesso	OK
Geração de áudio com texto vazio ou nulo	Erro	OK
Geração de áudio pelo fluxo principal (*Base64* retornado)	Sucesso	OK
Geração de áudio com falha no *Edge-TTS* (exceção)	Erro	OK

Fonte: elaborada pelo autor (2026).

A Tabela 10 apresenta a cobertura de testes dos decoradores de autorização, camada responsável por implementar o mecanismo transversal de controle de acesso às rotas da aplicação. A suíte de testes foi estruturada para validar os principais fluxos de autenticação e autorização, contemplando a verificação da consistência das sessões entre o *Firebase* e o *PostgreSQL*, o controle de acesso baseado em perfis de usuário e os mecanismos de tratamento de inconsistências entre as bases de dados. Essa cobertura de testes é fundamental para assegurar a segurança das rotas protegidas, a integridade do gerenciamento de sessões e a correta aplicação das políticas de autorização, contribuindo para a confiabilidade do sistema e para a prevenção de acessos indevidos.

Tabela 10 – Casos de teste para os decoradores de autorização

Caso de Teste	**Resultado Esperado**	**Status**
login_required: acesso sem sessão ativa (401)	Erro	OK

login_required: acesso com usuário válido (autorizado)	Sucesso	OK
login_required: usuário removido do Firebase (401)	Erro	OK
login_required: usuário removido do banco de dados local (401)	Erro	OK
agent_required: acesso de idoso negado (403)	Erro	OK
patient_required: acesso de ACS negado (403)	Erro	OK
agent_required: acesso de ACS inativo negado (403)	Erro	OK
agent_required: acesso de ACS ativo autorizado	Sucesso	OK
patient_required: acesso de idoso inativo negado (403)	Erro	OK
patient_required: acesso de idoso ativo autorizado	Sucesso	OK

Fonte: elaborada pelo autor (2026).

6 RESULTADOS E DISCUSSÕES

Para o desenvolvimento deste *MVP*, foi realizado o levantamento de 29 RFs e 26 RNFs, totalizando 55 requisitos técnicos. Os requisitos de autenticação e gerenciamento de usuários constituem a camada responsável pelo fluxo fundacional do sistema. Esse fluxo compreende a implementação e a rastreabilidade de funcionalidades como *login* convencional e *Google OAuth 2.0*, cadastro, recuperação de senha, atualização de e-mail com validação pelo *Firebase* e exclusão de contas de usuários. Dessa forma, a implementação desses requisitos estabeleceu uma infraestrutura de identidade robusta, essencial para garantir que as funcionalidades de monitoramento clínico e conversacional operem sobre um contexto seguro e auditável.

Além disso, por meio dos requisitos de gestão de perfil e do ambiente de acompanhamento, foi possível viabilizar a construção de um ecossistema de gestão da carteira de idosos para o ACS, atendendo diretamente à proposta de fortalecer a ação territorializada da APS. Diante desse contexto, foram contempladas funcionalidades como a geração de código único para identificação e vinculação de idoso ao ACS, a criação e o gerenciamento de grupos por condições crônicas e a triagem clínica inteligente por meio de relatórios diários.

O núcleo inteligente da plataforma destinada ao idoso foi fundamentado nos requisitos funcionais relacionados ao agente conversacional por voz, à classificação automática da intenção das mensagens e ao uso de *RAG* por meio dos documentos informativos de saúde do SUS. Além disso, foi contemplada a manutenção, em tempo real, da janela de contexto clínico do idoso, bem como a geração e atualização de relatórios diários e a triagem inteligente com níveis de prioridade alta, média ou baixa.

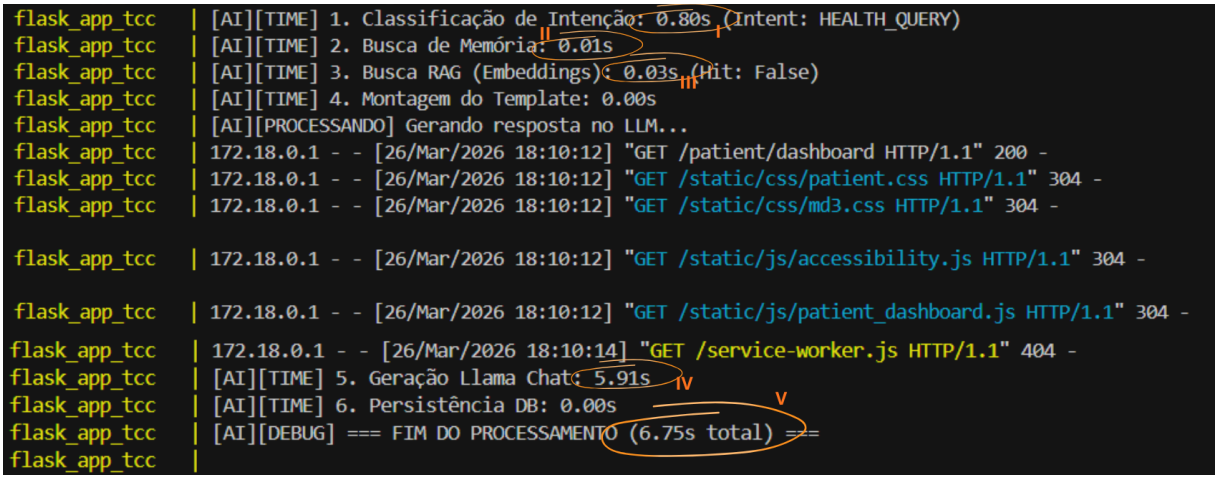
Durante o processo de desenvolvimento, a implementação desses requisitos evidenciou maior grau de complexidade técnica do *MVP*, em virtude das limitações de processamento e de *hardware* do dispositivo utilizado para a codificação. Todavia, o desenvolvimento bem-sucedido demonstrou a viabilidade de integrar modelos de linguagem de grande porte a sistemas de informação em saúde voltados a contextos com recursos computacionais restritos.

Os requisitos não funcionais que materializam o fluxo de cadastro assistido, por meio do qual o próprio ACS realiza presencialmente o cadastro do idoso,

viabilizaram a geração de credenciais temporárias e o compartilhamento de um *link* de recuperação de acesso com o cuidador responsável. Assim, esse fluxo responde diretamente a uma das barreiras estruturais identificadas na Introdução, relacionada à exclusão digital de indivíduos sem domínio de endereços de e-mail ou sem capacidade de realizar cadastro autônomo em plataformas digitais.

No contexto dos requisitos não funcionais, os critérios de desempenho estabeleceram metas mensuráveis para o comportamento do sistema sob condições operacionais controladas. Como demonstrado na Figura 14, a classificação da intenção das mensagens alcançou tempo de retorno inferior a 500 milissegundos (I), resultado obtido por meio do mecanismo de *cache* de *embeddings*.

Figura 14 – Tempo de processamento e retorno das respostas pelo *Llama* 3.2:3B via *terminal*



Fonte: elaborada pelo autor (2026).

Diante desse resultado, o mecanismo realiza cálculos de similaridade entre o vetor da mensagem de entrada e os vetores das categorias de intenção pré-calculados em memória. Por meio desse *cache*, elimina-se a necessidade de recalcular esses vetores a cada nova requisição, reduzindo o tempo de processamento que anteriormente variava entre 1 segundo e 1.25 segundos.

A geração de respostas pelo modelo *Llama* 3.2:3B atingiu tempo de processamento de 5.91 segundos (IV). Já as buscas de contexto na memória (II) e no *RAG* resultaram em tempo inferior a 1 segundo (III), produzindo um tempo médio total de processamento de 6.75 segundos (V). O *pipeline* de *RAG*, composto pelo banco vetorial *ChromaDB*, pelo modelo de *embeddings* (*SentenceTransformer all-MiniLM-L6-v2*) e pela biblioteca *PyPDF* para extração de conteúdos documentais,

demonstrou funcionamento adequado para o conjunto de arquivos do Ministério da Saúde indexados. Por meio desses documentos, foi possível auxiliar o modelo de IA na formulação de respostas baseadas em fontes oficiais para o contexto da APS, priorizando a consulta ao *RAG* para a construção de respostas coerentes aos contextos da saúde e da atenção primária.

Os resultados exibidos na Figura 14 foram obtidos via testes de desempenho executados em um ambiente local controlado de simulação, com o servidor *Flask* e o modelo *Llama* 3.2:3B servido pelo *Ollama*. O ambiente de testes foi configurado em uma máquina com *GPU NVIDIA* de 4 *GB* de VRAM compatível com *Compute Unified Device Architecture* (*CUDA*) 12.1 e *CUDA Deep Neural Network Library* (*cuDNN)* 8 para reproduzir condições operacionais próximas às de um ambiente real de implantação.

Para iniciar o servidor e observar os logs diretamente no terminal, utilizou-se o comando “docker-compose up”, que inicializa a aplicação em modo DEBUG para expor cada etapa do pipeline de processamento. Os cenários contemplaram o fluxo completo da mensagem de saúde, abrangendo classificação, busca de memória, recuperação RAG, montagem do prompt, geração do LLM e persistência no banco.

Para cada cenário, as métricas de tempo foram capturadas nos logs, permitindo a observação isolada do tempo de cada etapa do pipeline bem como do tempo total de processamento exigido na requisição.Esse procedimento possibilitou a identificação de gargalos específicos e a validação empírica de que os limiares de desempenho estabelecidos nos requisitos não funcionais foram atendidos.

Acerca da decisão técnica de utilizar o modelo *Llama* 3.2:3B, servido localmente por meio do *Ollama*, como núcleo da arquitetura de IA deste projeto, a escolha fundamentou-se em critérios técnicos e socioeconômicos deliberadamente articulados ao desempenho e à viabilidade técnica diante das limitações de *hardware*. Sob a perspectiva técnica, o modelo selecionado, com 3 bilhões de parâmetros, permitiu sua operação em ambientes com apenas 3 *GB* de *VRAM* quando quantizado em 4 *BIT*.

O modelo com 3 bilhões de parâmetros representa uma janela de viabilidade significativamente mais ampla do que modelos de 7B ou 8B, que podem exigir entre 8 *GB* e 14 *GB* de *VRAM* em suas configurações de alta fidelidade. Essa escolha está alinhada ao princípio de democratização tecnológica que orienta o desenvolvimento deste trabalho, uma vez que a imposição de requisitos que

demandam *hardware* de alto desempenho contraria diretamente o objetivo de produzir uma solução acessível ao contexto da saúde pública brasileira.

Para eliminar a elevada latência da primeira requisição ao modelo de linguagem natural, adotou-se o mecanismo de pré-aquecimento do *Llama* 3.2:3B durante a inicialização do servidor. Essa estratégia proporcionou uma redução significativa da latência decorrente do carregamento inicial dos pesos do modelo na memória. Ainda, possibilitou que, uma vez em operação, o servidor respondesse de forma consistente dentro da janela de latência estabelecida nos requisitos não funcionais.

Em ambiente equipado com *GPU NVIDIA* compatível com o *CUDA 12.1* e *cuDNN 8*, o modelo operou dentro do intervalo esperado de até seis segundos para a geração de respostas completas. Em contrapartida, quando executado exclusivamente em Unidade Central de Processamento (*Central Processing Unit – CPU*), a latência superou esse limite de forma considerável, configurando uma limitação de *hardware* já identificada anteriormente.

O desenvolvimento do agente conversacional por voz constitui a principal contribuição de acessibilidade do *MVP*, uma vez que a composição tecnológica adotada viabilizou uma interface de interação que não exige capacidade de leitura ou escrita do idoso para sua operação. O módulo de síntese de voz integrado resultou em uma experiência de fala e audição natural no ambiente de teste controlado, por meio do mecanismo de *streaming* por blocos (*chunks*) de áudio.

Por meio dessa estratégia de desenvolvimento, obteve-se o início da reprodução audível antes da conclusão da geração completa do áudio pelo *Edge-TTS*. Assim, esse mecanismo contribuiu para reduzir a percepção da latência durante a interação conversacional, aproximando a experiência do idoso ao fluxo de uma conversa humana natural.

O fluxo de cadastro assistido pelo ACS representa uma solução original e contextualmente relevante para o problema da exclusão digital de idosos analfabetos, que acabam sendo prejudicados por não conseguirem realizar o cadastro de forma autônoma. Essa funcionalidade possibilita que o próprio ACS realize presencialmente o cadastro da conta do idoso, gere as credenciais temporárias e um *link* de recuperação de acesso assinado com prazo de expiração definido.

Posteriormente, esse *link* pode ser disponibilizado ao cuidador domiciliar responsável pelo idoso, que poderá estabelecer as credenciais definitivas de acesso sem que o ACS tenha contato com as informações confidenciais da senha. Assim, essa abordagem preserva a integridade dos dados sensíveis e transfere a responsabilidade pela definição da senha ao núcleo familiar, respeitando a autonomia e a privacidade do indivíduo assistido.

Os resultados das funcionalidades desenvolvidas para a autenticação configuram uma infraestrutura inicial de acesso resiliente e adequada ao perfil diversificado dos usuários da plataforma, desde que acompanhada de fluxos complementares de cadastro assistido e autenticação acessível (Abadir, 2025; Rajan, 2025). Todavia, é importante ressaltar a necessidade de integrar o *login* por leitura biométrica, a fim de contemplar 100% dos usuários, inclusive os idosos analfabetos, uma vez que a autenticação por conta *Google* requer uma ação inicial de um responsável alfabetizado para a criação da conta de e-mail.

O desenvolvimento deste *software* contemplou o ACS com o maior número de funcionalidades em seu *dashboard*, refletindo sua posição central no modelo de atendimento domiciliar da APS. A funcionalidade de triagem inteligente constituiu o principal mecanismo de apoio à tomada de decisão clínica disponibilizado ao ACS. Por meio dessa funcionalidade, o sistema aciona o modelo *Llama* 3.2:3B para analisar o histórico textual das interações de cada idoso vinculado à conta do ACS e produzir uma classificação de risco em três níveis: alta, média ou baixa prioridade, acompanhada de uma justificativa clínica estruturada.

A classificação de prioridade é gerada a partir do conteúdo da janela de contexto clínico do idoso, atualizada de forma incremental ao término de cada sessão de *chat*, preservando a memória longitudinal das interações mais recentes. Essa classificação resulta de uma triagem clínica que fornece ao ACS uma base informacional para organizar a agenda de visitas domiciliares, priorizando os idosos com maior risco clínico identificado pelo modelo.

Como outro resultado do desenvolvimento deste *MVP*, destaca-se a geração automatizada de relatórios diários, baseada em um mecanismo por meio do qual o ACS pode solicitar, a qualquer momento, a geração ou a atualização do relatório de saúde de um determinado idoso vinculado ao seu *dashboard*. Esses relatórios são produzidos pelo modelo de IA a partir da sumarização da janela de contexto clínico

em formato *JSON* e exportados em formato *PDF*, com *layout* objetivo e profissional, por meio da biblioteca *WeasyPrint*.

Aliás, a execução da geração de relatórios em *threads* de segundo plano garante que a experiência do ACS não seja interrompida durante o processamento, respondendo imediatamente com uma mensagem de confirmação enquanto o relatório é produzido de forma assíncrona. Por meio do mecanismo de *threads*, ainda garantiu-se a integridade dos registros clínicos, resultando em apenas um relatório diário persistido por idoso para cada data, preservando a consistência do histórico documental.

No contexto dos testes unitários, as coberturas das suítes, apresentadas entre as Tabelas 2 e 10, demonstraram robustez lógica parcial do sistema construído, pois a execução de 105 casos de teste resultou em 100% de aprovação, com cobertura geral de 53,9% no diretório principal do projeto. O relatório consolidado foi gerado por meio do comando "*pytest --cov=. --cov-report=term-missing*", executado a partir do diretório raiz do projeto, acionando a suíte completa de testes unitários para exibir o total de linhas executáveis, cobertas e não alcançadas em cada arquivo. Embora esse percentual seja parcial para o *MVP* como um todo, ele reflete uma clara e deliberada priorização arquitetural da camada de serviços, onde efetivamente reside a complexidade algorítmica e clínica do sistema, conforme a Tabela 11.

Tabela 11 – Cobertura de testes agrupados por camada arquitetural

Camada Arquitetural	Linhas Executáveis	Linhas Cobertas	Cobertura (%)
Modelos (*Models*)	136	127	93,3%
Utilitários (*Utils*)	78	63	80,7%
Serviços (*Services*)	898	669	74,5%
Extensões (*Extensions*)	166	73	43,9%
Repositórios (*Repositories*)	245	98	40,0%
Rotas (*Routes*)	757	200	26,4%
TOTAL GERAL	**2.280**	**1.230**	**53,9%**

Fonte: elaborada pelo autor (2026).

A camada de Modelos apresentou a maior cobertura dentre todas as camadas, atingindo 93,3%, demonstrando que praticamente todas as estruturas responsáveis pela representação das entidades de domínio foram exercidas durante os testes. Esse cenário valida uma redução significativa do risco de inconsistências relacionadas à manipulação de dados, às validações e às conversões entre os objetos da aplicação.

Em seguida, a camada de Utilitários alcançou 80,7% de cobertura. Esse resultado evidencia a importância dessa camada para a segurança transversal da aplicação, visto que ela abriga predominantemente os decoradores responsáveis pelo controle de acesso e pela validação das sessões. Por meio dessa validação extensiva desses artefatos, assegurou-se que usuários não autenticados ou com sessões corrompidas sejam bloqueados antes mesmo de alcançarem a lógica de negócio, preservando a integridade do ecossistema.

Além disso, a camada de Serviços, que centraliza a complexidade algorítmica e as regras de negócio clínicas, incluindo as integrações com o modelo de IA, o *pipeline RAG* e os processos de triagem, obteve uma cobertura robusta de 74,5%. Esse índice confirma a eficácia da estratégia de isolamento por *mocks*, permitindo testar fluxos críticos, tratamentos de exceção e heurísticas de contingência sem depender da disponibilidade de serviços externos, conferindo estabilidade ao núcleo inteligente do sistema.

A camada de Extensões e Repositórios apresentaram índices mais conservadores, correspondendo a 43,9% e 40,0%, respectivamente. Com base nesses dados, a cobertura parcial das extensões justifica-se por se tratar de rotinas de inicialização de instâncias e credenciais, cujo comportamento depende majoritariamente de variáveis de ambiente. Já nos repositórios, esse índice reflete o limite conceitual da estratégia de testes adotada, uma vez que a validação completa do acesso aos dados exigiria a construção de suítes de testes de integração com instâncias reais do banco de dados.

A camada de Rotas registrou a menor taxa de cobertura do sistema, com 26,4%. Essa diferença constitui uma consequência natural do padrão arquitetural adotado, uma vez que a lógica de negócio foi extraída dos controladores e delegada integralmente à camada de serviços. As rotas assumem um papel exclusivamente relacionado ao recebimento das requisições e ao retorno de códigos *HTTP*. Consequentemente, a ausência de testes específicos para a camada de roteamento

representa um risco clínico reduzido, visto que as regras responsáveis pelas respostas da *API* já foram amplamente validadas na camada subjacente.

Para a obtenção dos índices detalhados por módulo apresentados na Tabela 12, executou-se o comando "*pytest Flask/tests/ --cov=Flask/app --cov-report=term-missing"*, que restringe a execução e gera o relatório delimitado a essas camadas. A utilização da *flag verbose* (-v) permite visualizar individualmente o resultado exato de cada caso de teste executado no processo, facilitando a rápida identificação e correção de eventuais falhas. A tabela logo abaixo detalha o desempenho individual de cada módulo avaliado, evidenciando as áreas que possuem maior robustez estrutural e identificando os serviços que ainda demandam ampliação de cenários.

Tabela 12 – Módulos cobertos e validados pelas suítes de teste dos módulos

Módulo / Serviço	**Linhas Executáveis**	**Linhas Cobertas**	**Cobertura (%)**
ai_service.py	291	229	79%
auth_service.py	136	133	98%
cnes*_service.py*	157	66	42%
decorators.py	78	63	81%
patient_group_service.py	44	35	80%
rag_service.py	102	94	92%
registration_service.py	62	61	98%
voice_service.py	106	51	48%

Fonte: elaborada pelo autor (2026).

Os dados evidenciam o excelente desempenho dos serviços relacionados à segurança e à gestão de usuários. Os módulos de autenticação e cadastro apresentaram os maiores índices de cobertura do conjunto, atingindo 98% cada. Esses resultados demonstram que os fluxos críticos de entrada de usuários no sistema estão amplamente protegidos contra regressões e falhas inesperadas. Outro destaque positivo é o serviço de *RAG*, responsável pela lógica de busca de informações em uma base vetorial, que alcançou expressiva cobertura de 92%.

Esse resultado sugere que os algoritmos de recuperação de contexto e integração de conhecimento encontram-se amplamente validados pelas suítes de teste.

O serviço de IA configura-se como o componente mais extenso e complexo desta análise, alcançando índice de cobertura de 79%. Considerando a natureza desse módulo, esse percentual indica que os principais comportamentos da IA encontram-se adequadamente validados, permanecendo apenas fluxos específicos de exceção ou instabilidades de rede como candidatos à ampliação da cobertura em trabalhos futuros.

O módulo de decoradores e o serviço responsável pela criação de grupos segmentados de idosos mantiveram níveis consistentes de validação, registrando coberturas de 81% e 80%, respectivamente. Esses resultados asseguram que as lógicas transversais, como a verificação de permissões e de *tokens* de autenticação, operem adequadamente, enquanto o serviço de grupos demonstra estabilidade nas operações de criação, leitura, atualização e exclusão.

Os serviços de voz e de CNES representam os principais pontos de atenção da camada, apresentando as menores coberturas entre os módulos avaliados, ambas inferiores a 50%. O serviço de processamento e síntese de voz alcançou 48% de cobertura, resultado que pode ser atribuído à dificuldade inerente de simular e testar rotinas envolvendo arquivos de áudio em ambientes automatizados. Já o serviço de CNES registrou o menor índice da análise, com apenas 42% de cobertura. Esse resultado indica que parte significativa das regras de processamento, integração e consulta de dados referentes ao CNES ainda permanece sem cobertura de testes. Esses dados evidenciam a necessidade de ampliar e refinar as suítes de teste nas próximas etapas de evolução do sistema.

Visando materializar as validações teóricas e arquiteturais obtidas, os registros da aplicação em pleno funcionamento foram documentados em vídeo. Essas demonstrações ilustram, de forma prática, a usabilidade do *software*, a resposta das interfaces assistivas e a fluidez do produto final desenvolvido. O acesso a esse material de validação prática está disponibilizado em um repositório em nuvem (*Google Drive*), podendo ser acessado por meio do *link* disponibilizado a seguir:
https://drive.google.com/drive/folders/1nnIA5LMZHDn5slYI9Bb0i_g5AiyqsPFa?usp=sharing.

7 CONSIDERAÇÕES FINAIS

O presente trabalho propôs o desenvolvimento de um *MVP* voltado à APS, com o objetivo de integrar IA generativa, tecnologias assistivas e ferramentas de gestão clínica para apoiar tanto os idosos quanto os ACS. Ao término do desenvolvimento e das baterias de validação, conclui-se que o sistema atingiu satisfatoriamente os objetivos propostos, demonstrando viabilidade técnica para o contexto ao qual se destina.

A implementação bem-sucedida de 55 requisitos atestou a capacidade da arquitetura de orquestrar múltiplas tecnologias complexas. A adoção de um modelo de linguagem de grande porte, otimizado para execução local em equipamentos com *hardware* de menor capacidade, em conjunto com técnicas de *RAG* fundamentadas em documentos de saúde do Ministério da Saúde, demonstrou ser uma estratégia eficaz. Esse arranjo garantiu respostas contextualizadas e de baixa latência, sem comprometer a segurança dos dados ou depender de *APIs* comerciais onerosas.

Sob a ótica da inclusão digital, o *MVP* minimizou a barreira da exclusão tecnológica frequentemente enfrentada pela população idosa. A interface conversacional por voz, que elimina a necessidade de leitura e escrita, aliada ao fluxo de cadastro assistido conduzido pelo ACS, materializou o conceito de acessibilidade defendido nesta pesquisa. Do ponto de vista do profissional de saúde, a disponibilização de funcionalidades que contemplam o *dashboard* do ACS revelou-se uma ferramenta valiosa para organizar a agenda de visitas domiciliares, reduzindo a sobrecarga cognitiva e otimizando o cuidado territorializado. Além disso, a elevada taxa de cobertura e aprovação (100%) nas suítes de testes unitários evidenciou a maturidade arquitetural e a confiabilidade do código desenvolvido.

Apesar dos êxitos alcançados, o *software* desenvolvido apresenta limitações inerentes ao escopo de um *MVP*, bem como restrições tecnológicas e de infraestrutura. Do ponto de vista de *hardware*, a dependência de uma *GPU* com suporte a *CUDA* e aproximadamente 3 *GB* de *VRAM*, mesmo com quantização em 4 *BIT*, configura uma barreira real para sua adoção nas UBS. Essas unidades frequentemente operam com computadores de entrada, equipados apenas com *CPU*. Dessa forma, as execuções do modelo realizadas exclusivamente em *CPU* apresentaram latências substanciais, o que pode comprometer a fluidez da interação com o agente conversacional.

Em relação ao modelo de IA utilizado, o *Llama* 3.2:3B, em razão do seu tamanho reduzido, possui limitações intrínsecas relacionadas ao raciocínio lógico profundo em quadros clínicos muito específicos ou ambíguos. Além disso, o banco vetorial utilizado pelo *RAG* depende atualmente da alimentação manual de documentos, não dispondo de *pipelines* de atualização automatizada, o que pode ocasionar defasagem documental ao longo do tempo.

No âmbito funcional e da usabilidade, a ausência de um mecanismo de detecção passiva de ativação obriga o idoso a interagir fisicamente com a tela para iniciar a conversa, impondo restrições adicionais a usuários com severas limitações motoras ou visuais. Além disso, a classificação de triagem gerada pela IA não dispara notificações proativas ao ACS em situações emergenciais, dependendo do acesso ativo do profissional ao painel para a identificação dessas ocorrências.

Para as próximas iterações evolutivas deste projeto, delineiam-se caminhos claros de aprimoramento voltados à mitigação das limitações identificadas e à ampliação do impacto da plataforma na saúde pública. No âmbito da usabilidade e da segurança clínica, recomenda-se prioritariamente a implementação de um sistema de notificações proativas, capaz de alertar instantaneamente o ACS sempre que a IA detectar e classificar uma interação do idoso como situação de emergência. Sugere-se, ainda, a criação de um histórico evolutivo da triagem inteligente, permitindo que o profissional acompanhe tendências de agravamento ou de melhora do idoso ao longo do tempo.

Sob a perspectiva tecnológica, a integração com o e-SUS do Prontuário Eletrônico do Cidadão (e-SUS PEC) representa um passo fundamental para consolidar a ferramenta no ecossistema oficial do SUS, evitando a redundância de registros. Em paralelo, a evolução do *frontend* para um aplicativo móvel nativo *offline-first* proporciona maior resiliência ao sistema em áreas com baixa conectividade, permitindo o armazenamento do contexto clínico em *cache* local. Da mesma forma, a incorporação de um mecanismo de ativação por comando de voz livre deve ser priorizada para elevar o nível de acessibilidade da plataforma.

Por fim, recomenda-se, como principal direcionamento para trabalhos futuros, a realização de testes de campo empíricos com usuários reais. A inserção prática do sistema no *locus* de pesquisa, diretamente junto a idosos, cuidadores familiares, ACS e gestores de UBS, permitirá a coleta de métricas qualitativas e quantitativas acerca do uso da plataforma em contexto real. Esse contato com o ambiente

produtivo da APS fornecerá evidências científicas adicionais para confirmar a eficácia, a aceitabilidade e o impacto social do *MVP* na democratização do acesso à saúde no contexto da APS.

REFERÊNCIAS

ABADIR, M. *et al.* Navigating the future of artificial intelligence technologies for improving the care of older adults. **Innovation in Aging**, v. 9, supl. 1, p. S24-S32, 2025. DOI: 10.1093/geroni/igaf092.

ABISHA, D. *et al.* Revolutionizing Rural Healthcare in India: AI-Powered Chatbots for Affordable Symptom Analysis and Medical Guidance. In: **INTERNATIONAL CONFERENCE ON INVENTIVE COMPUTATION TECHNOLOGIES (ICICT), 7., 2024, Coimbatore, India. Proceedings...** Piscataway: IEEE, 2024. p. 181-187. DOI: 10.1109/ICICT60155.2024.10544758.

ALVES, C. V. R. *et al.* Monitoramento da Fragilidade Geriátrica por meio de Heatmaps e Dashboards Inteligentes: uma solução web para instituições de longa permanência no Sul do Brasil. In: **ESCOLA REGIONAL DE APRENDIZADO DE MÁQUINA E INTELIGÊNCIA ARTIFICIAL DA REGIÃO SUL (ERAMIA-RS), 2025. Anais...** Porto Alegre: Sociedade Brasileira de Computação, 2025. p. 292-295.

BLONDINO, C. T. *et al.* The use and potential impact of digital health tools at the community level: results from a multi-country survey of community health workers. **BMC Public Health**, v. 24, n. 1, 650, 2024. DOI: 10.1186/s12889-024-18062-3.

BURCI, T. V. L.; SANTOS, A. R. dos; COSTA, M. L. F. Inclusão com igualdade ou com equidade: primeiras reflexões. **Colloquium Humanarum**, v. 14, n. especial, p. 444-450, 2017. DOI: 10.5747/ch.2017.v14.nesp.000976.

CATAPAN, S. de C. *et al.* Telecare in the Brazilian Unified Health System: where we are and where we are heading. **Ciência & Saúde Coletiva**, v. 29, n. 7, e03302024, 2024. DOI: 10.1590/1413-81232024297.03302024.

FOWLER, M. **Refactoring: Improving the Design of Existing Code**. 2. ed. Boston: Addison-Wesley Professional, 2018. ISBN 9780134757599.

GENARO, L. E. *et al.* Perceptions of Community Health Agents Regarding Home Care Provided to the Elderly. **Journal of Health Sciences**, v. 27, n. 2, p. 51-55, 2025. DOI: 10.17921/2447-8938.2025v27n2p51-55.

GIACOMIN, K. C. Envelhecimento populacional e os desafios para as políticas públicas. In: BERZINS, M. V.; BORGES, M. C. (org.). **Políticas públicas para um país que envelhece**. São Paulo: Martinari, 2012. p. 17-44.

GOOGLE. **Get started with Modern UI design – Material Design 3**. 2026. Disponível em: https://m3.material.io/get-started.

GOOGLE. **Material Web**. 2026. Disponível em: https://material-web.dev.

KHAN, M.; SHERANI, A. M. K. Understanding AI-Driven Cardiovascular Risk Prediction in Underserved Populations: Addressing Social Determinants of Health through Data Analytics. **Global Journal of Universal Studies**, v. 1, n. 2, p. 117-135, 2024. DOI: 10.58990/galas.2025.3.2.117.

LAMAS, C. de A. *et al.* Telehealth initiative to enhance primary care access in Brazil (UBS+Digital Project): multicenter prospective study. **Journal of Medical Internet Research**, v. 27, e68434, 2025. DOI: 10.2196/68434.

LOPES JÚNIOR, J. E. G. *et al.* Entre algoritmos e territórios: uma revisão de escopo sobre saúde digital e inteligência artificial na atenção primária. **Revista Panamericana de Salud Pública**, v. 49, e126, 2025. DOI: 10.26633/RPSP.2025.126.

LIMA, J. G. *et al.* O processo de trabalho dos agentes comunitários de saúde: contribuições para o cuidado em territórios rurais remotos na Amazônia, Brasil. **Cadernos de Saúde Pública**, v. 37, n. 8, e00247820, 2021. DOI: 10.1590/0102-311X00247820.

LIN, K.-C.; WAI, R.-J.; CHANG CHIEN, H.-Y. A Lightweight Temporal Convolutional Network for Contactless SPPB-Aligned Functional Fall-Risk Stratification in Older Adults Using Monocular RGB Video. **Sensors**, v. 26, n. 12, 3894, 2026. DOI: 10.3390/s26123894.

MARTIN, R. C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. Boston: Prentice Hall, 2017. ISBN 9780134494166.

MARTINS, W. de J. *et al.* Plataforma de Inteligência Cooperativa com a Atenção Primária à Saúde (Picaps): soluções tecnocientíficas em saúde digital no enfrentamento da COVID-19 e outras crises. **Ciência & Saúde Coletiva**, v. 29, n. 10, e01622023, 2024. DOI: 10.1590/1413-812320242910.01622023.

OSBORN, A. F. **Applied Imagination: Principles and Procedures of Creative Thinking**. New York: Charles Scribner's Sons, 1953.

RAJAN, M. *et al.* Exploring the Utility of Digital Voice Assistants for Primary Care Patients, Including Those With Physical and Visual Disabilities: Cross-Sectional Study. **JMIR mHealth and uHealth**, v. 13, e66185, 2025. DOI: 10.2196/66185.

RAKERS, M. M. *et al.* Availability of evidence for predictive machine learning algorithms in primary care: a systematic review. **JAMA Network Open**, v. 7, n. 9, e2432990, 2024. DOI: 10.1001/jamanetworkopen.2024.32990.

RESENDE, S. M. *et al.* Tecnologias digitais na Atenção Primária à Saúde: uma revisão de escopo. **Saúde em Debate**, v. 49, n. 145, e9913P, 2025. DOI: 10.1590/2358-289820251459913P.

RODRIGUES, S. M. *et al.* Digital health-enabled community-centered care: scalable model to empower future community health workers using human-in-the-loop artificial intelligence. **JMIR Formative Research**, v. 6, n. 4, e29535, 2022. DOI: 10.2196/29535.

SANTOS, R. C. dos *et al.* Agentes Comunitárias de Saúde e o uso da saúde digital: transformações de um trabalho vivo. **Saúde em Debate**, v. 49, n. 146, e9891P, 2025. DOI: 10.1590/2358-289820251469891P.

SOMMERVILLE, I. **Engenharia de Software**. 10. ed. São Paulo: Pearson Universidades, 2019. ISBN 9788543024974.

SHAMS-GHAHFAROKHI, Z. Challenges in health and technological literacy of older adults: a qualitative study in Isfahan. **BMC Geriatrics**, v. 25, n. 1, 247, 2025. DOI: 10.1186/s12877-025-05893-x.

SOUSA, R. V.; GUIMARÃES, M. J. S. Design inclusivo em embalagens: acessibilidade para pessoas com deficiência visual. **Blucher Design Proceedings**, v. 13, n. 1, p. 1792-1804, 2025. DOI: 10.5151/cidi2025-1081205.

TORRES, D. R.; WERMELINGER, E. D.; FERREIRA, A. P. Aplicação da Inteligência Artificial na Atenção Primária à Saúde: revisão de escopo e avaliação crítica. **Saúde em Debate**, v. 49, n. 145, e10070, 2025. DOI: 10.1590/2358-2898202514510070P.

WILLIAMS, D. *et al.* Deep learning and its application for healthcare delivery in low and middle income countries. **Frontiers in Artificial Intelligence**, v. 4, 553987, 2021. DOI: 10.3389/frai.2021.553987.

APÊNDICE A – INSTRUMENTO DE COLETA DE DADOS DO IDOSO NA ENTREVISTA SEMI-ESTRUTURADA

QUESTIONÁRIO

1. Como você entra em contato com o seu agente de saúde? Você tem facilidade para entrar em contato com ele? Se não, quais dificuldades você enfrenta ao tentar esse contato?
2. Com que frequência o seu agente de saúde visita a sua casa?
3. Durante as visitas, ele leva algum médico ou outro profissional de saúde com ele? Com que frequência isso acontece?
4. Quando o agente de saúde vai à sua casa, que tipos de perguntas ele faz sobre você e sua saúde?
5. Nas visitas de acompanhamento, se você relata algum problema de saúde, o que o agente faz para tentar resolvê-lo?
6. Quando você realiza algum exame, o agente de saúde leva os resultados até você ou você precisa ir buscá-los na UPA ou no local onde o exame foi feito?
7. Após a emissão dos resultados, como você faz para que o médico os analise e passe as orientações ou prescrições necessárias?
8. De que forma você recebe o retorno dos exames analisados pelo médico? O agente de saúde participa desse processo?
9. Em relação ao tempo necessário para obter o retorno desses exames, como você avalia esse prazo?
10. O que poderia ser melhorado nas visitas e nos atendimentos domiciliares realizados pelos agentes e demais profissionais de saúde?

APÊNDICE B – INSTRUMENTO DE COLETA DE DADOS DO ACS NA ENTREVISTA SEMI-ESTRUTURADA

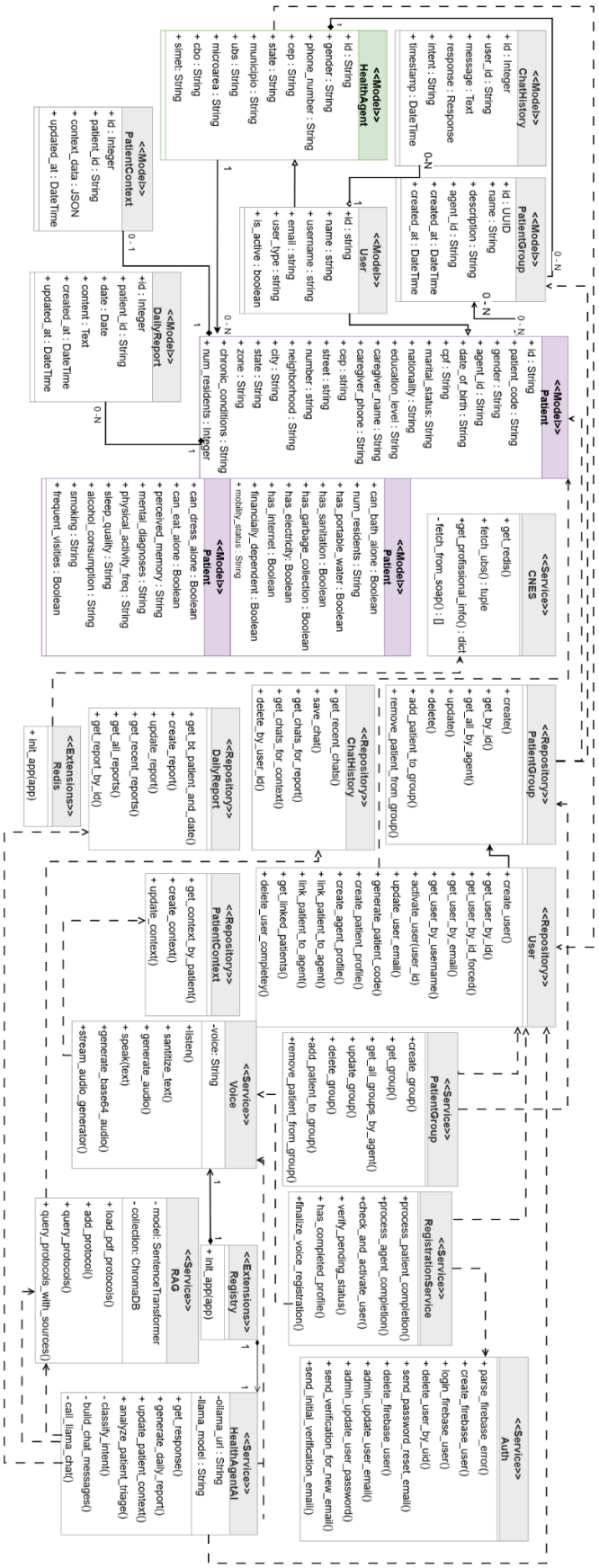
QUESTIONÁRIO

1. Como você identifica os seus pacientes prioritários — ou seja, aqueles que precisam de visitas e atendimentos domiciliares com maior frequência?
2. De que forma os moradores entram em contato com você para solicitar uma visita ou atendimento domiciliar de saúde?
3. A Secretaria Municipal de Saúde estabelece critérios para definir quem tem prioridade no recebimento de visitas e atendimentos domiciliares?
4. Durante as visitas, você segue algum protocolo de comunicação, escuta ativa/ouvidoria ou atendimento? Quais protocolos existentes você pode detalhar?
5. Que tipos de informações sobre o morador você coleta durante a visita domiciliar? Como essas informações auxiliam na prestação dos serviços

básicos de saúde?

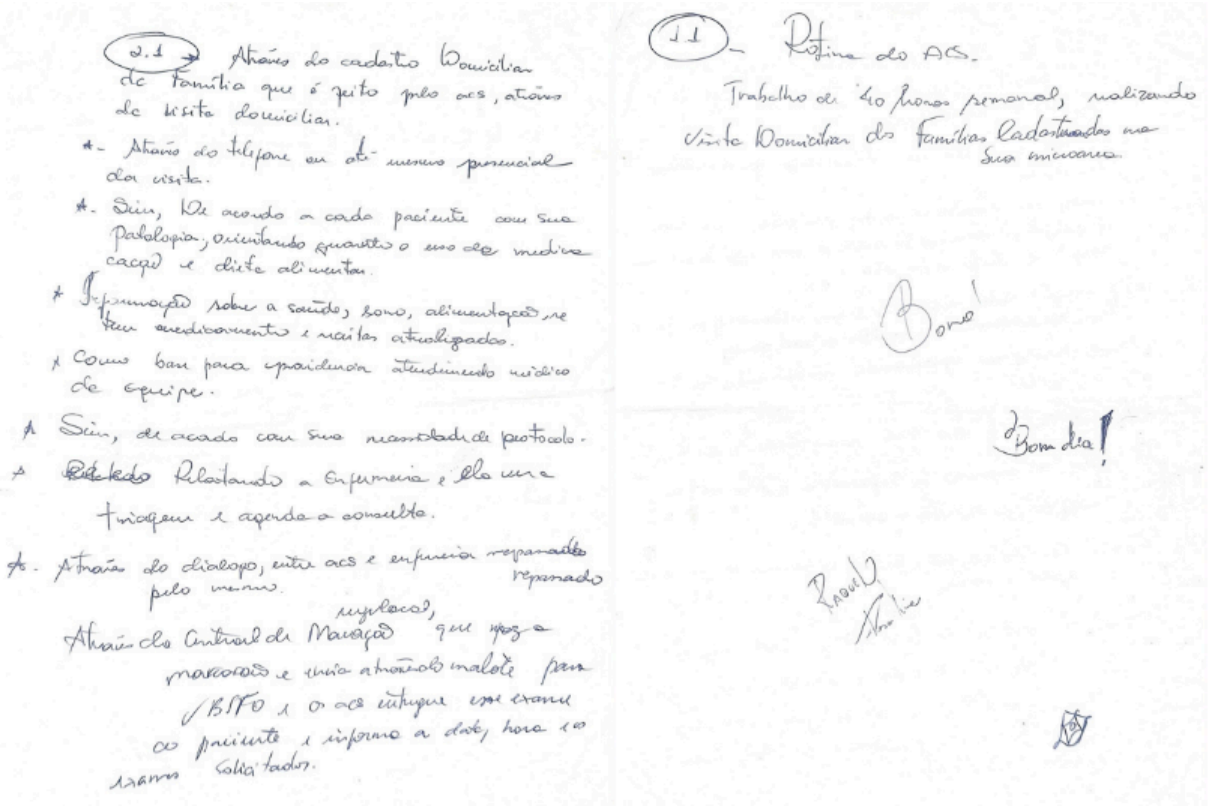
6. O protocolo de atendimento varia de paciente para paciente ou permanece sempre o mesmo?
7. As informações coletadas nas visitas passam por algum processo de triagem para a solicitação de exames laboratoriais ou encaminhamentos médicos na rede pública?
8. Quem são os responsáveis pelo processo de triagem (caso ele exista) e como essa etapa é realizada?
9. De que forma você é notificado quando o resultado de um exame ou o retorno de uma consulta (realizados na UPA ou nas Unidades Básicas de Saúde) fica disponível?

APÊNDICE C – DIAGRAMA DE CLASSES COMPLETO



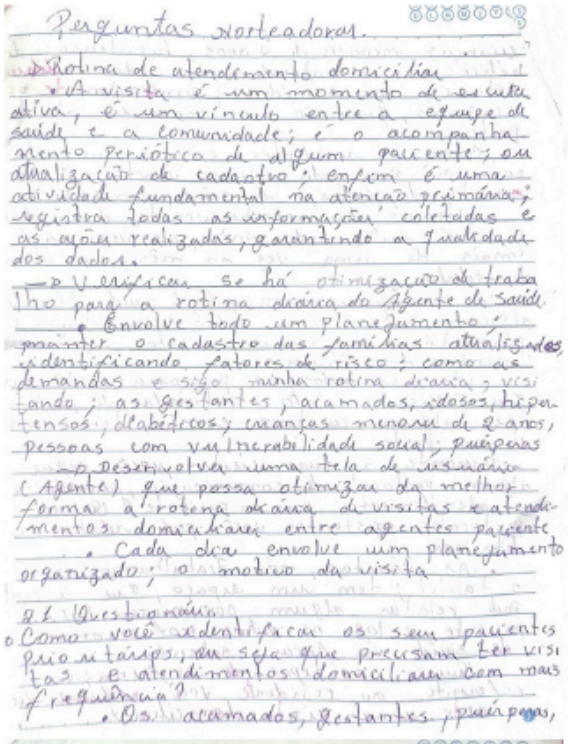
Fonte: elaborada pelo autor (2026).

APÊNDICE D – FOLHAS DE RESPOSTAS DO PRIMEIRO ACS



Fonte: elaborada pelo autor (2026).

APÊNDICE E – FOLHAS DE RESPOSTAS DO SEGUNDO ACS



Perguntas norteadoras.

Rotina de atendimento domiciliar

- A visita é um momento de escuta ativa, é um vínculo entre a equipe de saúde e a comunidade; é o acompanhamento periódico de algum paciente; ou atualização de cadastro; enfim é uma atividade fundamental na atenção primária; registra todas as informações coletadas e as ações realizadas, garantindo a qualidade dos dados.

- Verificar se há otimização de trabalho para a rotina diária do Agente de saúde

- Envolve todo um planejamento; manter o cadastro das famílias atualizados, identificando fatores de risco; como as demandas e siga minha rotina diária; visitando; as gestantes, acamados, idosos, hipertensos, diabéticos; crianças menores de 2 anos, pessoas com vulnerabilidade social; puérperas

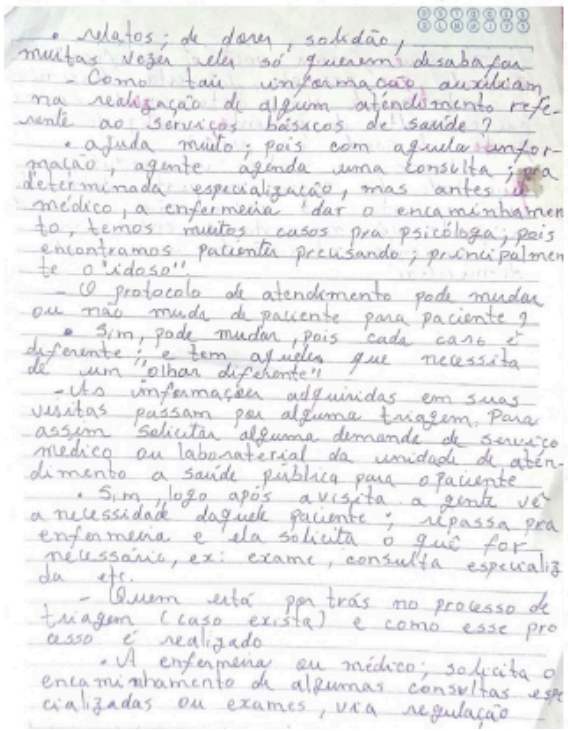
- Desenvolver uma tela de usuário (Agente) que possa otimizar da melhor forma a rotina diária de visitas e atendimentos domiciliares entre agentes paciente

- Cada dia envolve um planejamento organizado; o motivo da visita

2 Questionário

- Como você identificar os seus pacientes prioritários, ou seja que precisam ter visitas e atendimentos domiciliares com mais frequência?

- Os acamados, gestantes, puérperas,



- relatos; de dores, solidão, muitas vezes eles só querem desabafar

- Como tais informações auxiliam na realização de algum atendimento referente ao serviços básicos de saúde?

- ajuda muito; pois com aquela informação, agente agenda uma consulta; pra determinada especialização, mas antes do médico, a enfermeira dar o encaminhamento, temos muitos casos pra psicóloga, pois encontramos paciente precisando; principalmente o "idoso".

- O protocolo de atendimento pode mudar ou não muda de paciente para paciente?

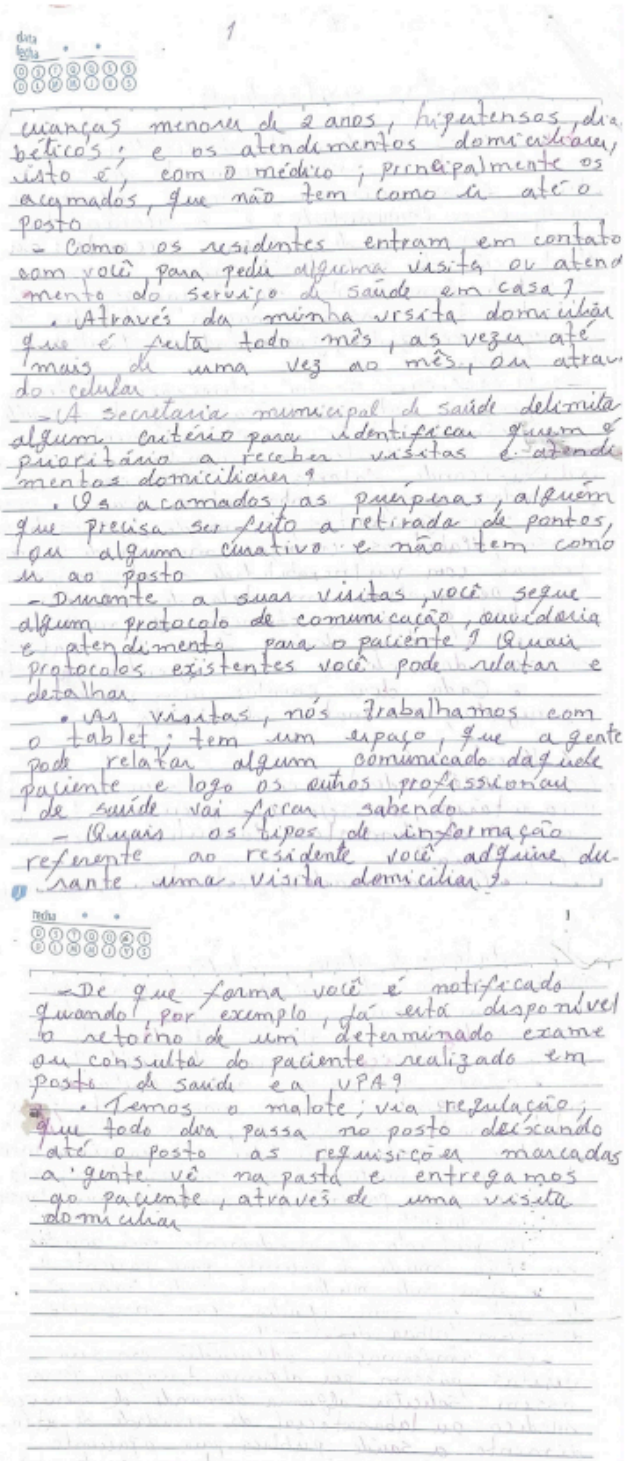
- Sim, pode mudar, pois cada caso é diferente; e tem aqueles que necessita de um "olhar diferente".

- As informações adquiridas em suas visitas passam por alguma triagem. Para assim solicitar alguma demanda de serviço medico ou laboratorial da unidade de atendimento a saúde publica para o paciente

- Sim, logo após a visita a gente vê a necessidade daquele paciente; repassa pra enfermeira e ela solicita o que for necessário, ex: exame, consulta especializada etc.

- Quem está por trás no processo de triagem (caso exista) e como esse processo é realizado

- A enfermeira ou médico; solicita o encaminhamento de algumas consultas especializadas ou exames, via regulação



crianças menores de 2 anos, hipertensos, diabéticos; e os atendimentos domiciliares, isto é, com o médico; principalmente os acamados, que não tem como ir até o posto

- Como os residentes entram em contato com você para pedir alguma visita ou atendimento do serviço de saúde em casa?

- Através da minha visita domiciliar que é feita todo mês, as vezes até mais de uma vez ao mês, ou através celular

- A secretaria municipal de saúde delimita algum critério para identificar quem é prioritário a receber visitas e atendimentos domiciliares?

- Os acamados, as puérperas, alguém que precisa ser feito a retirada de pontos, ou algum curativo e não tem como ir ao posto

- Durante a suas visitas, você segue algum protocolo de comunicação, encaminhamento para o paciente? Quais protocolos existentes você pode relatar e detalhar

- As visitas, nós trabalhamos com o tablet; tem um espaço, que a gente pode relatar algum comunicado daquele paciente e logo os outros profissionais de saúde vai ficar sabendo

- Quais os tipos de informação referente ao residente você adquire durante uma visita domiciliar?

- De que forma você é notificado quando, por exemplo, já está disponível o retorno de um determinado exame ou consulta do paciente realizado em posto de saúde ou a UPA?

- Temos o malote; via regulação; que todo dia passa no posto deixando até o posto as requisições marcadas a gente vê na pasta e entregamos ao paciente, através de uma visita domiciliar

Fonte: elaborada pelo autor (2026).

APÊNDICE F – REGISTRO DAS GRAVAÇÕES EM ÁUDIO DAS ENTREVISTAS COM O ACS E O IDOSO

Fonte: elaborada pelo autor (2026).

ANEXO A – ABERTURA DO PROTOCOLO PARA O REGISTRO DE SOFTWARE

INSTITUTO FEDERAL
Piauí

Processo Eletrônico
23176.000537/2026-19

Data	**Setor de Origem**
04/08/2026 15:58:11	CAPIR - CPA-IFPI - CAMPUS PIRIPIRI
Tipo	**Assunto**
Inovação: Registro de Software	Solicitação de registro de programa de computador.

Interessados
Francisnilto dos Santos Nascimento

Situação
Em trâmite

Trâmites

5 de Agosto de 2026 às 19:41
Recebido por: CONIT-IFPI: Francisco Valdivino Rocha Lima

4 de Agosto de 2026 às 16:29
Enviado por: DENS-IFPI - CAMPUS PIRIPIRI: Alberto Cunha Alves

4 de Agosto de 2026 às 16:28
Recebido por: DENS-IFPI - CAMPUS PIRIPIRI: Alberto Cunha Alves

4 de Agosto de 2026 às 16:00
Enviado por: CPA-IFPI - CAMPUS PIRIPIRI: Saulo Andreane Silva Cardoso

Fonte: elaborada pelo autor (2026).

ANEXO B – DECLARAÇÃO DE ENTREGA DE CÓDIGO-FONTE À COORDENAÇÃO DO CURSO

Ministério da Educação
Instituto Federal de Educação, Ciência e Tecnologia do Piauí
IFPI - CAMPUS PIRIPIRI
Av. Rio dos Matos, Germano, S/N, Germano, Piripiri / PI, CEP 64.260-000
Fone: Site: www.ifpi.edu.br

DECLARAÇÃO 4/2026 - COTANDESSIS/PIR/DENS/DG-PIRIPIR/CAPIR/IFPI

Piripiri, 12 de agosto de 2026.

Declaramos, para os devidos fins, que a Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas (ADS) do Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI) – Campus Piripiri recebeu o **código-fonte do software desenvolvido como produto do Trabalho de Conclusão de Curso (TCC), na modalidade Relatório Técnico de Software**, elaborado pelo discente **Francisnilto dos Santos Nascimento**, matrícula **2024116TADS0012**.

O material foi entregue à Coordenação do Curso para fins de **registro, arquivamento e comprovação da entrega do produto técnico resultante do TCC**, permanecendo sob guarda institucional, conforme os procedimentos acadêmicos aplicáveis.

Por ser verdade, firmamos a presente declaração.

Iallen Gábio de Sousa Santos

Coordenador do Curso Tecnólogo em Análise e Desenvolvimento de Sistemas

Documento assinado eletronicamente por:

- **Iallen Gabio de Sousa Santos, COORDENADOR(A) DE CURSO - FUC0001 - COTANDESSIS/PIR-IFPI - CAMPUS PIRIPIRI**, em 12/08/2026 17:35:35.

Este documento foi emitido pelo SUAP em 12/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpi.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 474748
Código de Autenticação: 5b17b34b8f

Fonte: elaborada pelo autor (2026).