Remo: Um Aplicativo para Regulação Emocional de Pacientes com Doenças Crônicas

Trabalho de Conclusão de Curso

Vitor Hugo Boaventura da Silva

Antonio Carlos dos Santos Souza

Orientador

Instituto Federal da Bahia – IFBA

Curso de Análise e Desenvolvimento de Sistemas

Campus Salvador

Salvador, Bahia, Brasil

Julho 2026

SUMÁRIO

1. Visão Geral

1.1 Declaração do Problema

Segundo ALCIENE (2025), “As doenças crônicas não transmissíveis (DCNT) estão entre os principais problemas de saúde pública do Brasil e do mundo. Segundo a Organização Mundial da Saúde (OMS), as DCNT foram responsáveis por cerca de 70% das mortes ocorridas globalmente em 2019. No Brasil, as DCNT foram responsáveis, em 2019, por 41,8% do total de mortes ocorridas prematuramente, ou seja, entre 30 e 69 anos de idade (Brasil, 2008; Figueiredo et al., 2021).” Dados mais recentes da Organização Mundial da Saúde (2023) apontam que as doenças crônicas não transmissíveis (DCNT) são responsáveis por 74% das mortes no mundo.

É justamente nesse contexto que surge o trabalho intitulado *Prototipagem e Validação de Aplicativo Móvel para Avaliação e Conduta da Regulação Emocional Baseado no Modelo de Calista Roy: Estudo Multimétodos*. Conforme a autora descreve em sua tese, o estudo tem por objetivo o “desenvolvimento de uma tecnologia mobile health para o desenvolvimento do processo de enfermagem com base na regulação emocional de pessoas com Doenças Crônicas Não Transmissíveis”. O desenvolvimento passou por duas etapas distintas: primeiro a coleta de dados com pacientes acometidos por alguma DCNT e, posteriormente, a construção e validação do conteúdo do instrumento de coleta, resultando também na prototipagem e validação de um aplicativo móvel para apoiar a aplicação desse processo de enfermagem.

Ainda em se tratando do estudo realizado por ALCIENE (2025), está fora do escopo do presente trabalho detalhar os pormenores de como cada etapa foi realizada. Cabe aqui apenas mencionar a conclusão da autora, para fins de entendimento das regras de negócio do sistema a ser desenvolvido. Assim sendo, podemos inferir que o bem-estar emocional de pacientes com doenças crônicas está associado a quatro aspectos:

1. Sistema fisiopatológico (adesão terapêutica, choro, letramento, humor);
2. Autocuidado (sono, atividade física, alimentação, mental);
3. Espiritualidade (espírito e natureza humana);
4. Interdependência/comunidade (benefícios sociais).

Nesse sentido, o software almejado pelo estudo tem como proposta orientar os enfermeiros sobre como ajudar seus pacientes a terem um melhor equilíbrio emocional nesses quatro aspectos. O presente Trabalho de Conclusão de Curso tem como objetivo desenvolver o software correspondente ao protótipo idealizado e validado por ALCIENE (2025), possibilitando sua utilização e avaliação em um ambiente real de aplicação. A motivação para desenvolver esse sistema é poder contribuir com a sociedade no sentido de aplicar o conhecimento adquirido no IFBA em algo que seja benéfico para os portadores de DCNT. Já as justificativas do projeto são:

- Aplicação da tecnologia da informação em uma área de conhecimento que é de fundamental importância para a sociedade, que é a área de saúde;
- Oportunidade de validar de maneira prática o resultado levantado por uma tese de doutorado que tem grande potencial de impacto na comunidade acadêmica da UFBA (saúde) e do IFBA (tecnologia da informação);
- Integração de diferentes áreas de conhecimento abrangendo o Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas (CADS) do IFBA e o Programa de Pós-Graduação em Enfermagem e Saúde da Escola de Enfermagem da UFBA.

1.2 Proposta de Solução de Software

Conforme a seção anterior, o software almejado consiste em um aplicativo para dispositivos móveis, seguindo um protótipo desenvolvido como parte do trabalho de Alciene. Os usuários desse sistema serão enfermeiros que estejam acompanhando pacientes acometidos com DCNT.

Dentro do aplicativo, o usuário poderá se registrar e realizar login. Além disso, o sistema permitirá que ele cadastre pacientes. A funcionalidade principal consiste em responder um questionário sobre o estado emocional de cada paciente. Com base no preenchimento deste questionário, o sistema retorna ao usuário um relatório contendo as estratégias de intervenção necessárias para a regulação emocional do paciente em questão. Estas estratégias são arquivos PDF com uma série de orientações que devem ser seguidas pelo enfermeiro como forma de mitigar os pontos de desregulação emocional identificados para este paciente. Estes documentos são de autoria de ALCIENE, 2025, com base em fontes da literatura científica.

As perguntas que compõem o questionário são agrupadas em diferentes categorias, que abrangem os quatro aspectos do bem-estar emocional do paciente, já mencionados na Declaração do Problema. Estes quatro aspectos, por sua vez, possuem subcategorias onde as perguntas estão distribuídas. Todas as perguntas foram redigidas para permitir apenas as respostas “SIM” ou “NÃO”. E cada uma delas possui uma condição que determina qual resposta indica a necessidade de sua respectiva estratégia de intervenção. Uma pergunta possui uma única estratégia de intervenção, mas uma mesma estratégia de intervenção pode estar associada a mais de uma pergunta. Ao todo, são vinte e duas perguntas, quatro categorias, oito subcategorias e quinze estratégias de intervenção. Para facilitar o entendimento de como estes dados se relacionam, foi elaborada a tabela a seguir.

Pergunta	Resposta que Implica Necessidade de Intervenção	Categoria	Subcategoria	Título da Estratégia de Intervenção
Faz uso regular das medicações diárias?	Não	Fisiológico	Medicações	Segurança e Uso Racional das Medicações
Conhece os efeitos adversos da medicação que usa?	Não	Fisiológico	Medicações	Segurança e Uso Racional das Medicações
Marca exames em tempo hábil para consultas?	Não	Fisiológico	Exames	Exames de Acompanhamento
Faz uso de medicação diária para dormir?	Não	Autocuidado	Sono	Disposição do Sono
Conhece sobre a doença e o risco de morte?	Não	Autocuidado	Entendimento Sobre A Patologia	Entendimento Sobre a Patologia
Realiza atividade física aeróbica/anaeróbica?	Não	Autocuidado	Atividade Física	Realização de Exercício Físico
Consome frutas, legumes e vegetais?	Não	Autocuidado	Alimentação	Alimentação
Consome Sal e Açúcar?	Sim	Autocuidado	Alimentação	Alimentação
Consome água?	Não	Autocuidado	Alimentação	Alimentação
Controla quantidade e horários da alimentação?	Não	Autocuidado	Alimentação	Alimentação
Realiza planejamento diário?	Não	Autocuidado	Mental	Planejamento Diário

Faz uso de linguagem positiva?	Não	Autocuidado	Mental	Linguagem Positiva
Costuma ter choros?	Sim	Autocuidado	Mental	Choro com Episódios Recorrentes
Humor oscila facilmente?	Sim	Autocuidado	Mental	Humor
Você costuma fazer planos?	Não	Espiritualidade	Espírito/Natureza Humana	Vontade de Sentido
Você sente que tem uma conexão com o superior?	Não	Espiritualidade	Espírito/Natureza Humana	Conexão com Deus
Você acha que seus relacionamentos são saudáveis?	Não	Espiritualidade	Espírito/Natureza Humana	Papéis e Relacionamentos
Você se sente uma pessoa criativa?	Não	Espiritualidade	Espírito/Natureza Humana	Criatividade
Faz uso de cigarro, álcool e outras drogas?	Sim	Interdependência /Comunidade	Riscos Sociais	Riscos Sociais
Mais de um membro da família possui DCNT?	Sim	Interdependência /Comunidade	Riscos Sociais	Riscos Sociais
Possui rede de apoio familiar?	Não	Interdependência /Comunidade	Riscos Sociais	Riscos Sociais
Há problemas físicos, psíquicos e/ou deficiências na família?	Sim	Interdependência /Comunidade	Riscos Sociais	Riscos Sociais

Tabela 1: Perguntas, condições de retorno, categorias, subcategorias e estratégias de intervenção.
Fonte: Elaborado pelo autor.

Conforme já mencionado, uma vez que todas as perguntas sejam respondidas, o sistema retornará ao usuário um relatório contendo as estratégias de intervenção correspondentes a cada resposta fornecida. Essas informações ficam registradas em um banco de dados para que o enfermeiro possa manter um acompanhamento do estado do paciente ao longo do tempo. Vale ressaltar que os pacientes NÃO serão usuários. Apenas os enfermeiros (e, possivelmente, outros profissionais da saúde futuramente) farão uso da plataforma. Nesse caso, os pacientes portadores de DCNT atuam como fonte de parte dos dados que alimentam o banco, mas serão cadastrados por intermédio dos usuários do sistema. É importante levar em consideração que com esse sistema sendo testado em um ambiente real, como um hospital, existe a possibilidade de um paciente ser atendido por mais de um enfermeiro. Por esse motivo, é importante que o sistema permita que um enfermeiro visualize pacientes cadastrados por outros enfermeiros, para evitar duplicidade de cadastros.

Nesse contexto, o presente trabalho objetiva elencar as principais funcionalidades que precisam estar presentes em uma primeira versão desta aplicação, e, com base nisso, desenvolver um projeto de software que abarque as mais prioritárias. Nesse sentido, o desenvolvimento do aplicativo seguirá o conceito de Produto Viável Mínimo (Minimum Viable Product – MVP), definido como a versão de um produto que contém o conjunto mínimo de funcionalidades necessárias para gerar valor ao usuário e possibilitar a obtenção de feedback validado com o menor esforço de desenvolvimento possível (RIES, 2011). A adoção do MVP permite testar hipóteses fundamentais do projeto em um contexto real de uso, reduzindo riscos técnicos e de escopo, além de orientar decisões futuras de evolução da aplicação com base em evidências empíricas (RIES, 2011).

1.3 Trabalhos Relacionados

Inicialmente, destaca-se o Projeto Pronto Afeto (também do IFBA) desenvolvido por Coutinho (2025), cujo objetivo consiste na construção de uma plataforma digital destinada ao gerenciamento de serviços de cuidado domiciliar, promovendo a interação entre clientes, cuidadores e equipe administrativa. Assim como o sistema REMO, o projeto adota uma arquitetura baseada em aplicação cliente-servidor, composta por aplicações *frontend* responsáveis pela interação com os usuários e uma API REST centralizando as regras de negócio e a persistência dos dados. Embora ambos os trabalhos estejam inseridos no domínio da saúde e utilizem tecnologias semelhantes para o desenvolvimento de aplicações modernas, como será possível verificar na seção de tecnologias adotadas, suas finalidades são distintas. Enquanto o Projeto Pronto Afeto concentra-se na gestão operacional de serviços de cuidado domiciliar, o REMO tem como objetivo apoiar profissionais de enfermagem na avaliação da regulação emocional de pacientes com doenças crônicas, oferecendo estratégias de intervenção fundamentadas no Modelo de Adaptação de Roy (COUTINHO, 2025; MOREIRA, 2024).

Na fundamentação de seu trabalho, Coutinho (2025) recorre a estudos que discutem o emprego de tecnologias digitais no suporte ao cuidado em saúde, referências que também se mostram pertinentes ao desenvolvimento do REMO. Entre esses estudos, Queirós et al. (2018) realizaram uma revisão sistemática sobre tecnologias empregadas no gerenciamento remoto de doenças crônicas, identificando um crescimento significativo da utilização de soluções digitais voltadas ao acompanhamento contínuo de pacientes. Os autores destacam que a efetividade dessas tecnologias depende não apenas da coleta de dados clínicos, mas também da capacidade dos sistemas em oferecer interfaces acessíveis, integração entre diferentes atores envolvidos no cuidado e suporte adequado aos profissionais de saúde.

Em perspectiva semelhante, Sriram, Richardson e McCann (2022) analisaram o emprego de tecnologias assistivas por cuidadores de pessoas com demência em ambiente domiciliar. Os resultados indicam que essas tecnologias podem contribuir para aumentar a segurança dos pacientes, facilitar o acompanhamento das condições de saúde e reduzir a sobrecarga dos cuidadores. Entretanto, os autores ressaltam que aspectos relacionados à usabilidade, simplicidade de utilização e adaptação às necessidades dos usuários exercem influência direta na adoção dessas soluções, reforçando a importância do desenvolvimento de interfaces intuitivas e centradas no usuário.

Os estudos anteriormente apresentados demonstram que aplicações voltadas ao cuidado em saúde vêm sendo empregadas em diferentes contextos, como gerenciamento de serviços assistenciais, monitoramento remoto de pacientes e suporte ao trabalho de cuidadores. Entretanto, o REMO diferencia-se dessas iniciativas por direcionar sua aplicação ao contexto do Processo de Enfermagem, apoiando especificamente a avaliação da regulação emocional de pacientes com doenças crônicas a partir de um instrumento científico validado e de estratégias de intervenção fundamentadas na pesquisa de Silva (2025).

Além dos trabalhos apresentados por Coutinho (2025), a literatura recente evidencia diferentes abordagens para utilização de tecnologias móveis no acompanhamento de pessoas com doenças crônicas. Aplicações de *mobile health* (*mHealth*) frequentemente buscam promover o autocuidado e o monitoramento contínuo das condições clínicas dos pacientes, enquanto sistemas de *Remote Patient Monitoring* (RPM) incorporam a atuação permanente dos profissionais de saúde no acompanhamento das informações coletadas. Segundo Paul et al. (2025), essas tecnologias vêm sendo empregadas como estratégia complementar ao cuidado tradicional, favorecendo a continuidade da assistência e a tomada de decisão baseada em dados.

Uma revisão sistemática realizada por Debon et al. (2019) identificou funcionalidades recorrentes em aplicativos destinados ao acompanhamento de pessoas com doenças crônicas, como registro de informações clínicas, monitoramento de indicadores de saúde, lembretes para tratamentos e incentivo à adoção de hábitos saudáveis. Os autores ressaltam que essas funcionalidades favorecem o acompanhamento contínuo e fortalecem a participação ativa do paciente em seu tratamento.

Sob a perspectiva da arquitetura tecnológica, Butt et al. (2024) destacam que sistemas destinados ao monitoramento remoto em saúde demandam soluções capazes de integrar diferentes componentes computacionais, garantindo interoperabilidade, segurança da informação, disponibilidade e confiabilidade dos dados processados. Complementarmente, Tan et al. (2024) observaram que intervenções baseadas em monitoramento remoto podem produzir impactos positivos na adesão ao tratamento, nos indicadores clínicos e na qualidade de vida de pacientes com doenças crônicas, reforçando o potencial dessas tecnologias como apoio à assistência em saúde.

Embora os trabalhos analisados apresentem contribuições relevantes para o desenvolvimento de aplicações voltadas ao cuidado em saúde, nenhum deles aborda especificamente a avaliação da regulação emocional de pacientes com doenças crônicas fundamentada no Modelo de Adaptação de Roy. Nesse contexto, o REMO propõe uma solução computacional voltada ao apoio do processo de enfermagem, permitindo que profissionais de saúde realizem avaliações estruturadas, obtenham estratégias de intervenção baseadas em evidências científicas e registrem essas informações em uma plataforma desenvolvida especificamente para esse propósito.

2. Requisitos

Com base na proposta levantada na seção 1, o sistema conta, a princípio, com um único tipo de usuário, que é um profissional de saúde. Inicialmente, o aplicativo é pensado para enfermeiros. Dito isso, podem ser estabelecidos os requisitos do sistema.

Porém, antes de adentrar nos requisitos em si, é necessário esclarecer que na Engenharia de Software, os **requisitos de um sistema** descrevem as condições ou capacidades necessárias para atender às necessidades dos usuários e aos objetivos do projeto. Esses requisitos são tradicionalmente classificados em **requisitos funcionais** e **requisitos não funcionais**, distinção amplamente adotada na literatura acadêmica e em normas técnicas (SOMMERVILLE, 2011; IEEE, 1998).

Especificando: os **requisitos funcionais** definem *o que* o sistema deve fazer, delimitando os serviços, comportamentos e funcionalidades oferecidas aos usuários, estando diretamente relacionados às interações entre o sistema e seus atores, como cadastro, registro de dados e comunicação com sistemas externos. Já os **requisitos não funcionais** descrevem *como* o sistema deve se comportar, estabelecendo restrições e atributos de qualidade que influenciam a execução dessas funcionalidades, como desempenho, segurança, usabilidade e confiabilidade. Esses requisitos não se vinculam a funções específicas, mas a características globais do sistema, sendo considerados essenciais para garantir a qualidade do software e a experiência do usuário, conforme descrito na literatura especializada e em normas técnicas (SOMMERVILLE, 2011; IEEE, 1998).

2.1 Requisitos Funcionais

RF01 Como enfermeiro(a), desejo me cadastrar no sistema.
RF02 Como enfermeiro(a), desejo iniciar sessão (realizar login) no sistema.
RF03 Como enfermeiro(a), desejo encerrar sessão (realizar logout) no sistema.
RF04 Como enfermeiro(a), desejo cadastrar pacientes no sistema.
RF05 Como enfermeiro(a), desejo responder questionário de avaliação do estado emocional do paciente cadastrado no sistema.

RF06 Como enfermeiro(a), desejo visualizar o relatório de atendimento do paciente, contendo as estratégias de intervenção recomendadas para cada avaliação realizada.
RF07 Como enfermeiro(a), desejo realizar o download da avaliação do paciente no sistema.
RF08 Como enfermeiro(a), desejo acessar uma relação com todos os pacientes cadastrados no sistema.
RF09 Como enfermeiro(a), desejo acessar os dados de um paciente específico cadastrado no sistema.
RF10 Como enfermeiro(a), desejo editar os dados do paciente cadastrado no sistema.
RF11 Como enfermeiro(a), desejo visualizar o histórico de avaliações de um paciente ao longo do tempo.
RF12 Como enfermeiro(a), desejo registrar uma nova avaliação para um paciente já avaliado anteriormente.
RF13 Como enfermeiro(a), desejo atualizar meus dados cadastrais no sistema.
RF14 Como enfermeiro(a), desejo redefinir minha senha de acesso ao sistema.
RF15 Como enfermeiro(a), desejo excluir um paciente cadastrado do sistema.

2.2 Requisitos Não-Funcionais

RNF01 O sistema deve ser acessado via dispositivo móvel, com sistema operacional *IOS* ou *Android*.
RNF02 O dispositivo móvel deve ter acesso à internet para acessar o aplicativo.
RNF03 O sistema deve garantir autenticação segura dos usuários.
RNF04 O sistema deve garantir confidencialidade e integridade dos dados dos pacientes.
RNF05 O sistema deve apresentar tempo de resposta adequado para operações de cadastro e consulta.
RNF06 O sistema deve possuir interface simples e intuitiva, adequada ao uso por profissionais de saúde.
RNF07 O sistema deve permitir a persistência dos dados em banco de dados relacional.
RNF08 O sistema deve permitir evolução futura para inclusão de novos profissionais de saúde.
RNF09 O sistema deve ser desenvolvido de forma modular, permitindo manutenção e extensões futuras.

3. Design

3.1 Modelo de Banco de Dados

Considerando a necessidade do sistema de armazenar arquivos PDF (estratégias de intervenção e relatórios de atendimento), a persistência de dados do sistema precisa ser composta por dois mecanismos complementares. Para o armazenamento de dados estruturados, como informações de usuários, pacientes, atendimentos, perguntas e respostas, pode ser utilizado um banco de dados relacional. Já para o armazenamento dos arquivos em PDF, pode ser empregada uma solução de armazenamento de objetos (*Blob Storage*), destinada à persistência de dados não estruturados ou semiestruturados. Nesse modelo, os arquivos são armazenados como objetos em contêineres denominados *buckets*, sendo cada objeto identificado por uma chave única e acompanhado de metadados que permitem sua localização e gerenciamento. Em vez de armazenar o conteúdo binário diretamente no banco de dados relacional, mantém-se apenas a referência ao objeto armazenado, reduzindo o volume de dados persistidos no banco e promovendo uma separação entre dados estruturados e arquivos digitais.

3.1.1 Modelo Lógico

O modelo lógico do sistema foi elaborado a partir dos requisitos funcionais definidos na seção 2, utilizando a abordagem relacional para representação dos dados. Nesse modelo, as entidades são organizadas de forma a refletir os principais elementos do domínio da aplicação, seus atributos e os relacionamentos necessários para suportar as funcionalidades de cadastro, avaliação emocional, geração de estratégias de intervenção e emissão de relatórios.

As entidades centrais do modelo são Enfermeiro, Paciente e Atendimento, sendo o atendimento o elemento responsável por associar um profissional de saúde a um paciente em uma determinada data e horário. As entidades Pergunta, Resposta, Categoria e Subcategoria estruturam o questionário de avaliação emocional e permitem a classificação das respostas de acordo com os aspectos do bem-estar considerados pelo sistema.

Além disso, a entidade Estratégia de Intervenção armazena as condutas recomendadas ao profissional de saúde, enquanto a entidade Relatório consolida os resultados de cada atendimento e as estratégias associadas.

A Figura 1 apresenta a representação lógica do núcleo do sistema.

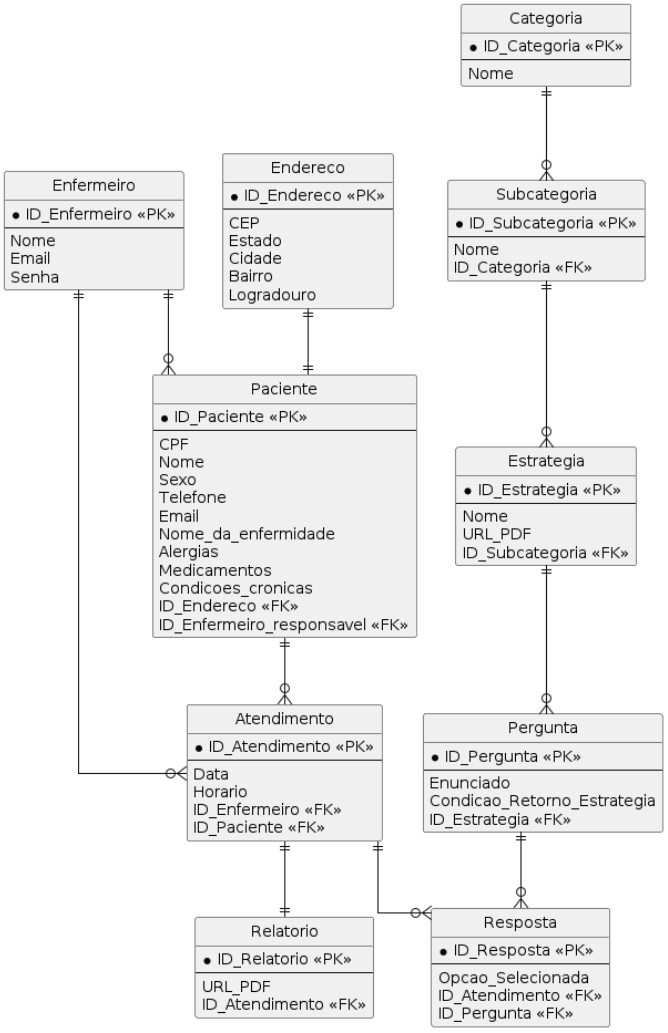

Figura 1: Representação do modelo lógico do cerne do sistema. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

As entidades do banco de dados são:

- **Enfermeiro** – representa os profissionais de enfermagem que utilizam o sistema. Armazena nome, e-mail e senha.
- **Paciente** – representa os indivíduos acompanhados presencialmente pelo enfermeiro, contendo dados pessoais como nome, CPF, sexo, telefone e e-mail.
- **Endereço** – armazena as informações de localização do paciente, incluindo CEP, estado, cidade, bairro e logradouro.
- **Atendimento** – representa cada ocorrência de acompanhamento realizada pelo enfermeiro junto ao paciente, contendo data e horário, funcionando como elemento central de associação entre profissional, paciente e instrumentos de avaliação.
- **Pergunta** – representa os itens do questionário aplicado durante o atendimento, contendo o enunciado e a categoria à qual a pergunta pertence. As perguntas podem ser respondidas negativamente ou afirmativamente, apenas.
- **Resposta** – representa a forma como um paciente respondeu uma determinada pergunta durante um atendimento. As respostas possíveis são *sim* ou *não*.
- **Categoria** – representa diferentes classificações do bem estar do paciente, englobando perguntas e estratégias de intervenção.
- **Subcategoria** – representa subdivisões existentes dentro de cada categoria.
- **Estratégia de Intervenção** – representa as orientações e condutas sugeridas ao profissional de saúde, classificadas por categoria e subcategoria e descritas por meio de um documento PDF explicativo.
- **Relatório** – representa o documento gerado a partir de um atendimento específico, consolidando as informações obtidas sobre o estado emocional do paciente e as estratégias de intervenção associadas.

As entidades se relacionam da seguinte maneira:

- Um **enfermeiro** pode realizar um ou mais **atendimentos**, enquanto cada atendimento está associado a um único enfermeiro.
- Um **paciente** pode estar vinculado a um ou mais **atendimentos**, sendo que cada atendimento refere-se a um único paciente.
- Cada **paciente** possui um **endereço** que armazena suas informações de localização.
- Um **atendimento** está associado a múltiplas **perguntas**, que compõem o questionário aplicado durante a avaliação.
- Um **atendimento** pode resultar na geração de um **relatório**, associado de forma obrigatória a esse atendimento.
- As **estratégias de intervenção** podem estar associadas a um ou mais **relatórios**, e um relatório pode conter múltiplas estratégias, caracterizando um relacionamento muitos-para-muitos.

3.1.2 Modelo Físico

O modelo físico representa a implementação do modelo lógico no sistema gerenciador de banco de dados, especificando como as entidades, atributos e relacionamentos são materializados em tabelas, colunas, chaves e restrições de integridade. Nessa etapa, são definidos aspectos relacionados aos tipos de dados, chaves primárias, chaves estrangeiras e demais características necessárias para a persistência das informações, preservando a estrutura lógica apresentada anteriormente e garantindo a consistência dos dados armazenados. A Figura 2 apresenta a representação física do núcleo do banco de dados da aplicação.

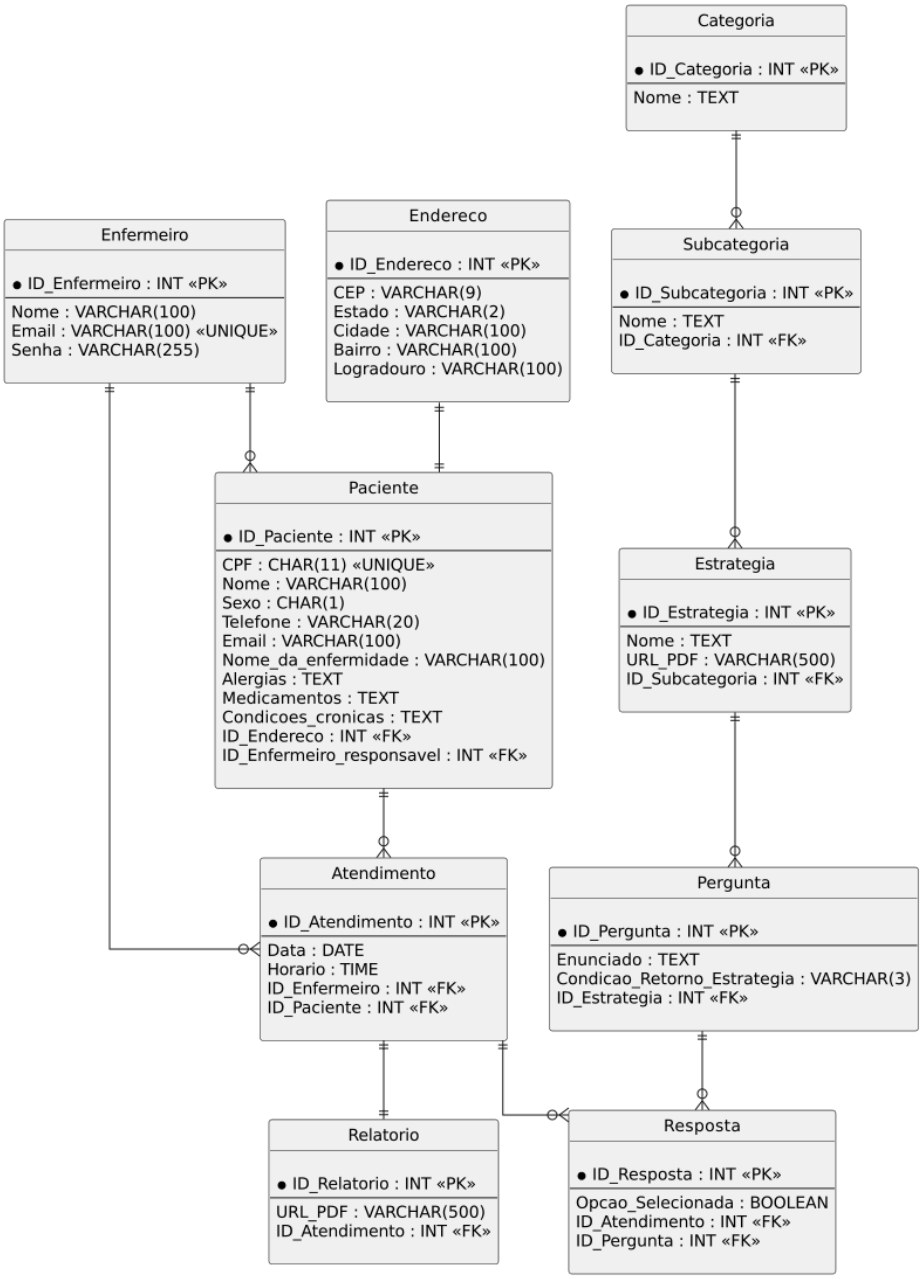

Figura 2: Representação do modelo físico do sistema. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

Vale observar que as entidades Estratégia de Intervenção e Relatório possuem um atributo *URL_PDF*, que nada mais é do que uma referência aos PDF correspondentes, armazenados em *buckets*, conforme descrito na próxima seção.

Após a definição das entidades e de seus relacionamentos, o modelo físico foi estruturado buscando minimizar redundâncias e preservar a consistência das informações armazenadas. Para isso, as entidades foram organizadas de forma independente e relacionadas por meio de chaves estrangeiras, aproximando-se dos princípios da Terceira Forma Normal (3FN). Nessa organização, cada atributo não-chave depende exclusivamente da chave primária da respectiva entidade, reduzindo anomalias de inserção, atualização e remoção e favorecendo a manutenção da integridade dos dados ao longo da evolução da aplicação (ELMASRI; NAVATHE, 2016).

A implementação do modelo também contempla mecanismos de integridade estrutural disponibilizados pelo sistema gerenciador de banco de dados. Cada entidade possui uma chave primária responsável por garantir a identificação única de seus registros, enquanto os relacionamentos entre as tabelas são preservados por meio de chaves estrangeiras. Adicionalmente, atributos de identificação, como CPF e endereço eletrônico, foram definidos com restrição de unicidade, impedindo o cadastramento de registros duplicados, ao passo que informações obrigatórias são protegidas por restrições de obrigatoriedade (NOT NULL), contribuindo para a consistência da base de dados (ELMASRI; NAVATHE, 2016).

Sob a perspectiva das cardinalidades, o modelo foi elaborado para refletir as regras de negócio apresentadas nos requisitos da aplicação. Um paciente pode possuir diversos atendimentos realizados ao longo do tempo, enquanto cada atendimento está associado a um único paciente. Da mesma forma, um enfermeiro pode realizar múltiplos atendimentos, preservando o histórico clínico dos pacientes e permitindo seu acompanhamento longitudinal. Durante cada atendimento são registradas respostas para todas as perguntas do questionário, estabelecendo uma associação entre atendimento, pergunta e resposta. Já as estratégias de intervenção apresentam relacionamento muitos-para-muitos com os relatórios, uma vez que um mesmo documento pode ser recomendado em diferentes atendimentos, e um único relatório pode reunir múltiplas estratégias de intervenção, evitando duplicidade de armazenamento dos documentos e favorecendo sua reutilização.

Embora o volume de dados esperado para o Produto Viável Mínimo seja reduzido, o modelo foi concebido considerando aspectos básicos de desempenho. As chaves primárias são automaticamente indexadas pelo sistema gerenciador de banco de dados, enquanto atributos frequentemente utilizados em operações de busca ou sujeitos a restrições de unicidade, como CPF e endereço eletrônico, podem ser indexados para otimizar consultas e validações. De forma semelhante, as chaves estrangeiras podem ser indexadas quando necessário para melhorar o desempenho de operações envolvendo junções entre tabelas, especialmente em cenários com maior volume de registros (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2025; ORACLE, 2025).

3.1.3 Armazenamento de Objetos

Conforme apresentado na introdução desta seção, a persistência de dados do sistema é composta por um banco de dados relacional para armazenamento das informações estruturadas e por um mecanismo de armazenamento de objetos destinado aos arquivos PDF gerados e utilizados pela aplicação. Nesse modelo, os documentos são armazenados como objetos organizados em *buckets*, sendo identificados por uma chave única utilizada para sua recuperação.

No sistema REMO, esse mecanismo é empregado para armazenar dois tipos de documentos: as estratégias de intervenção, disponibilizadas em formato PDF para consulta durante os atendimentos, e os relatórios gerados ao final de cada avaliação realizada pelo profissional de enfermagem. Em ambos os casos, o banco de dados relacional mantém apenas a referência ao objeto armazenado, enquanto o conteúdo do arquivo permanece persistido no armazenamento de objetos.

A Figura 3 apresenta a organização lógica do armazenamento dos documentos utilizados pela aplicação.

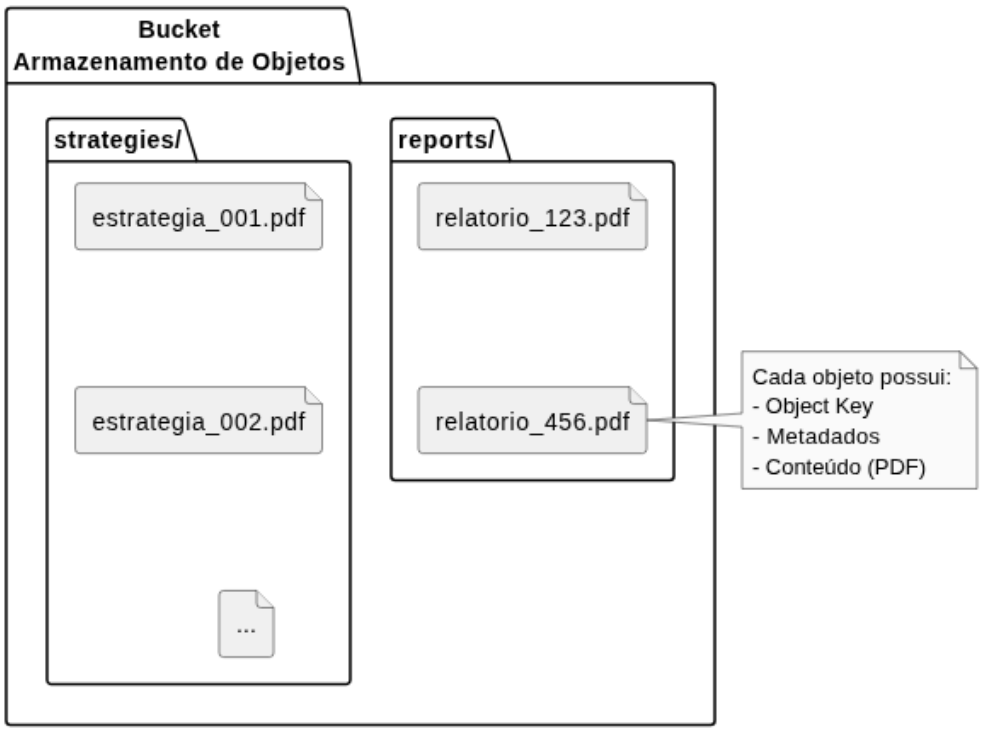

Figura 3: Organização lógica do armazenamento de objetos da aplicação. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

3.2 Projeto UML

A Linguagem de Modelagem Unificada (Unified Modeling Language – UML) constitui um padrão amplamente utilizado para especificação, visualização e documentação de sistemas orientados a objetos. Por meio de diferentes tipos de diagramas, a UML permite representar tanto a estrutura quanto o comportamento da aplicação, contribuindo para a compreensão dos componentes do sistema e das interações existentes entre eles. Nesta seção são apresentados os principais diagramas elaborados para o projeto do REMO, contemplando aspectos estruturais e comportamentais da solução proposta.

3.2.1 Diagrama de Casos de Uso

O diagrama de casos de uso representa as funcionalidades disponibilizadas pelo sistema sob a perspectiva dos atores que interagem com a aplicação. Seu objetivo é identificar os serviços oferecidos pelo sistema e evidenciar as principais interações entre os usuários e as funcionalidades implementadas.

No sistema REMO, o ator principal é o profissional de enfermagem, responsável por realizar o cadastro de pacientes, registrar atendimentos, aplicar o questionário de avaliação emocional, consultar estratégias de intervenção e gerar relatórios contendo os resultados obtidos durante cada atendimento. A Figura 4 apresenta o diagrama de casos de uso elaborado para a aplicação.

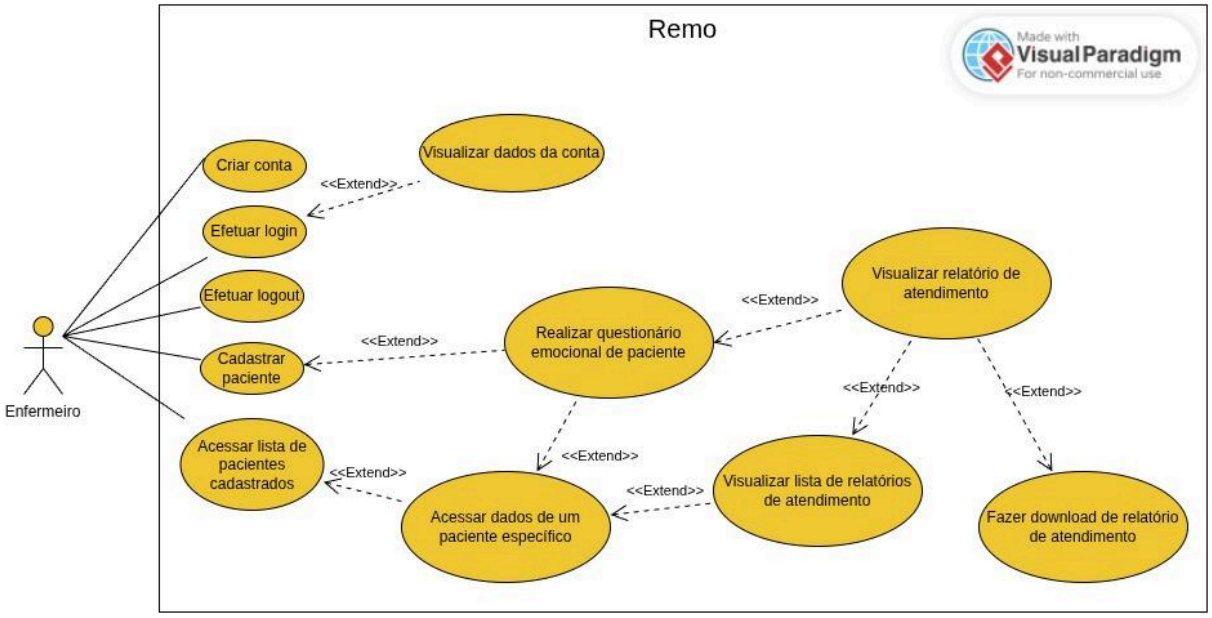

Figura 4: Diagrama de casos de uso do sistema. Fonte: Elaborado pelo autor com a ferramenta Visual Paradigm.

O diagrama evidencia as funcionalidades disponibilizadas ao profissional de enfermagem e demonstra como cada caso de uso contribui para o fluxo de atendimento realizado pela aplicação. Dessa forma, é possível compreender, em um nível funcional, as principais responsabilidades do sistema antes da análise de sua estrutura interna.

3.2.1 Diagrama de Classes

O diagrama de classes representa a estrutura estática da aplicação, descrevendo as principais classes do domínio, seus atributos, operações e relacionamentos. Esse tipo de diagrama possibilita visualizar como os elementos do sistema são organizados e de que forma colaboram para atender aos requisitos funcionais definidos anteriormente.

No REMO, o diagrama de classes foi elaborado com base nas entidades responsáveis pela gestão de pacientes, atendimentos, avaliações emocionais, estratégias de intervenção e geração de relatórios, refletindo a organização lógica da aplicação e servindo como referência para a implementação do sistema. A Figura 5 apresenta o diagrama de classes desenvolvido para o projeto.

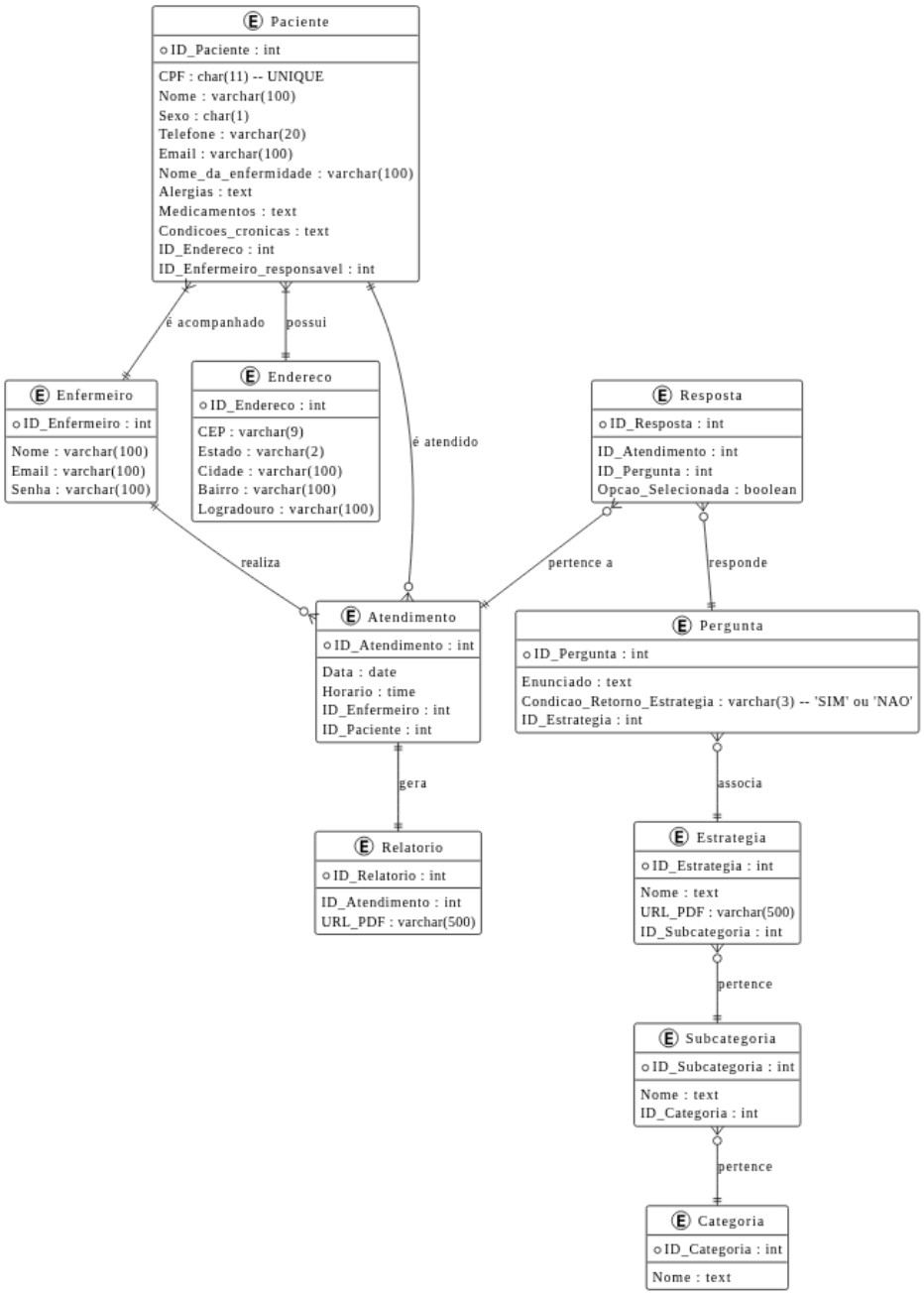

Figura 5: Diagrama de Classe do sistema. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

Observa-se que os relacionamentos entre as classes refletem as regras de negócio estabelecidas para a aplicação, permitindo representar as associações existentes entre profissionais de enfermagem, pacientes, atendimentos, perguntas, respostas, estratégias de intervenção e relatórios. Essa modelagem favorece a organização da implementação e auxilia na compreensão da estrutura do sistema.

3.2.3 Diagrama de Sequência

O diagrama de sequência representa o comportamento dinâmico do sistema, descrevendo a ordem temporal das mensagens trocadas entre os componentes envolvidos na execução de uma determinada funcionalidade. Diferentemente do diagrama de classes, que apresenta a estrutura

estática da aplicação, o diagrama de sequência evidencia como os elementos do sistema colaboram entre si para atingir um objetivo específico.

No contexto do REMO, optou-se por representar o processo de geração do relatório de atendimento por se tratar de uma das funcionalidades centrais da aplicação, envolvendo a recuperação das informações registradas durante a avaliação emocional, a consolidação dos dados obtidos e o armazenamento do documento gerado para posterior consulta pelo profissional de enfermagem. A Figura 6 apresenta o fluxo de interações necessário para a realização desse processo.

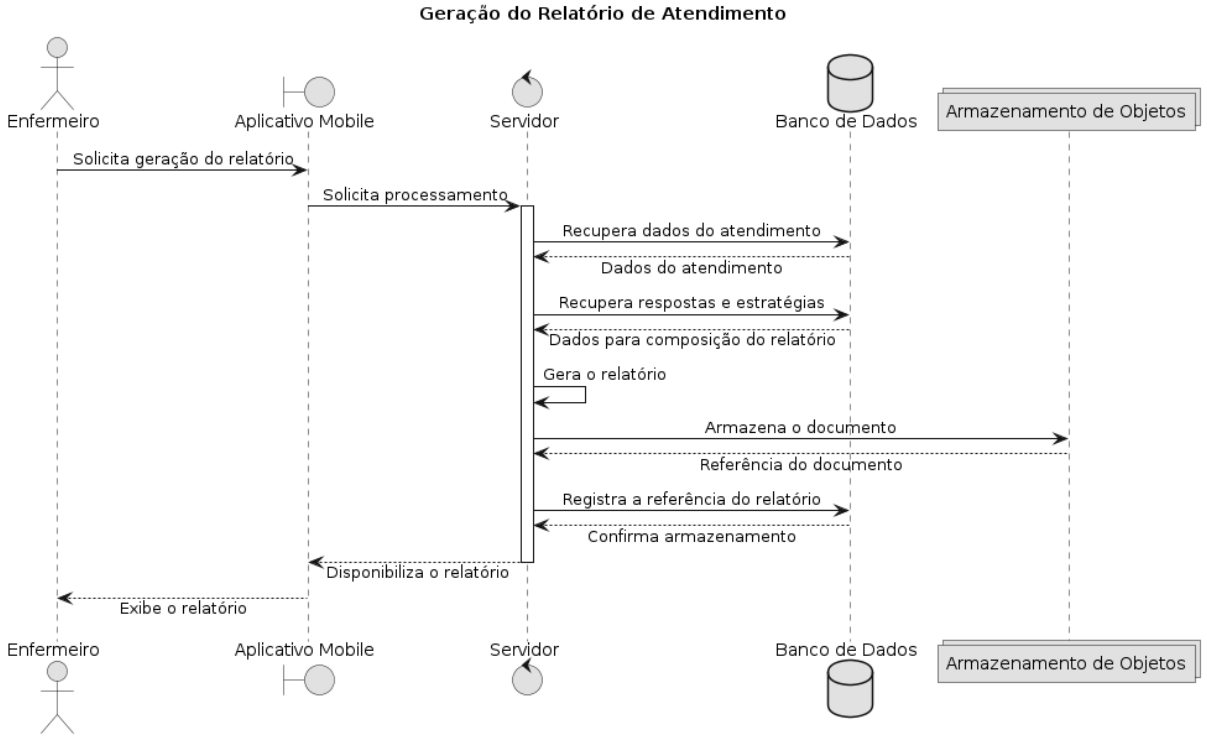

Figura 6: Diagrama de sequência da geração do relatório de atendimento. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

Conforme ilustrado na Figura 6, o processo é iniciado a partir da solicitação do profissional de enfermagem por meio da interface da aplicação. Em seguida, o sistema recupera as informações necessárias ao atendimento, incluindo os dados do paciente, as respostas registradas durante a avaliação e as estratégias de intervenção correspondentes. Após a consolidação dessas informações, o relatório é gerado, armazenado no mecanismo de armazenamento de objetos e sua referência é registrada na base de dados, permitindo que o documento seja posteriormente recuperado e disponibilizado ao usuário.

3.3 Visão Arquitetural

A arquitetura do sistema REMO foi concebida seguindo uma abordagem em camadas, visando promover a separação de responsabilidades, facilitar a manutenção do código e permitir futuras evoluções da aplicação. A solução é composta por quatro elementos principais: aplicação cliente, camada de serviços, banco de dados relacional e mecanismo de armazenamento de objetos.

A aplicação cliente é responsável por fornecer a interface utilizada pelo profissional de enfermagem durante o uso do sistema, permitindo a realização das principais funcionalidades relacionadas ao cadastro de pacientes, execução das avaliações emocionais e consulta das informações geradas ao longo dos atendimentos.

A camada de serviços concentra a lógica de negócio da aplicação, sendo responsável pelo processamento das solicitações recebidas, validação dos dados, gerenciamento das regras de negócio e comunicação com os mecanismos de persistência. A comunicação entre a aplicação cliente e essa camada ocorre por meio de um conjunto de serviços disponibilizados pela aplicação, seguindo o estilo arquitetural REST, o que favorece o desacoplamento entre cliente e servidor e facilita a interoperabilidade entre diferentes plataformas (FIELDING, 2000).

A persistência dos dados é realizada de forma complementar por dois mecanismos distintos. As informações estruturadas da aplicação são armazenadas em um banco de dados relacional, enquanto os documentos digitais produzidos e utilizados pelo sistema são armazenados em um mecanismo de armazenamento de objetos. Essa separação permite que cada tecnologia seja utilizada de acordo com sua finalidade, evitando o armazenamento de arquivos binários no banco de dados relacional e contribuindo para uma arquitetura mais organizada e de fácil manutenção.

A Figura 7 apresenta a visão arquitetural do sistema REMO e os principais fluxos de comunicação entre seus componentes.

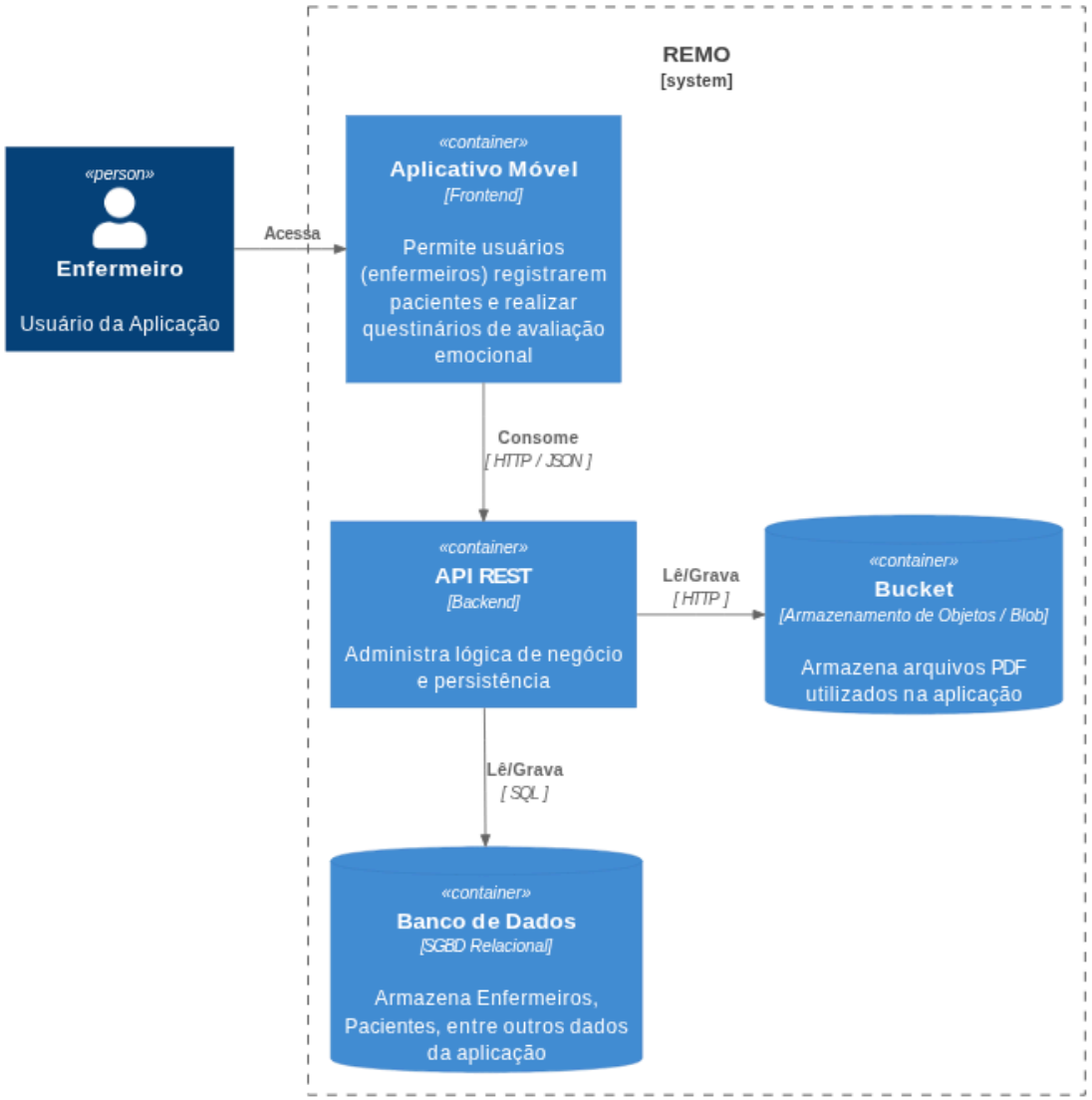

Figura 7: Representação da visão arquitetural do sistema. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

3.3.1 Arquitetura do Backend

O backend do sistema foi organizado segundo uma arquitetura em camadas, na qual cada grupo de componentes possui responsabilidades específicas e bem definidas. Essa organização promove baixo acoplamento entre os módulos da aplicação, facilita a manutenção do código e favorece sua evolução ao longo do desenvolvimento do sistema (FOWLER, 2002; SOMMERVILLE, 2011).

A camada de apresentação é responsável por receber as solicitações provenientes da aplicação cliente e encaminhá-las para o processamento adequado. As regras de negócio são concentradas na camada de aplicação, que coordena os fluxos necessários para execução das funcionalidades do sistema e realiza a comunicação com os mecanismos de persistência.

O domínio da aplicação é representado pelas entidades responsáveis pela modelagem das informações do sistema, bem como pelos objetos utilizados na transferência de dados entre as diferentes camadas da aplicação. Complementarmente, componentes auxiliares são utilizados para realizar conversões entre modelos de domínio e objetos de transferência, tratamento de exceções, autenticação, autorização e demais funcionalidades transversais necessárias ao funcionamento do sistema.

A persistência é realizada por uma camada específica responsável pelo acesso às informações armazenadas. Essa separação evita que as regras de negócio dependam diretamente dos mecanismos de armazenamento, contribuindo para maior modularidade e facilitando futuras alterações na infraestrutura da aplicação.

A Figura 8 apresenta a organização arquitetural adotada para o backend do sistema.

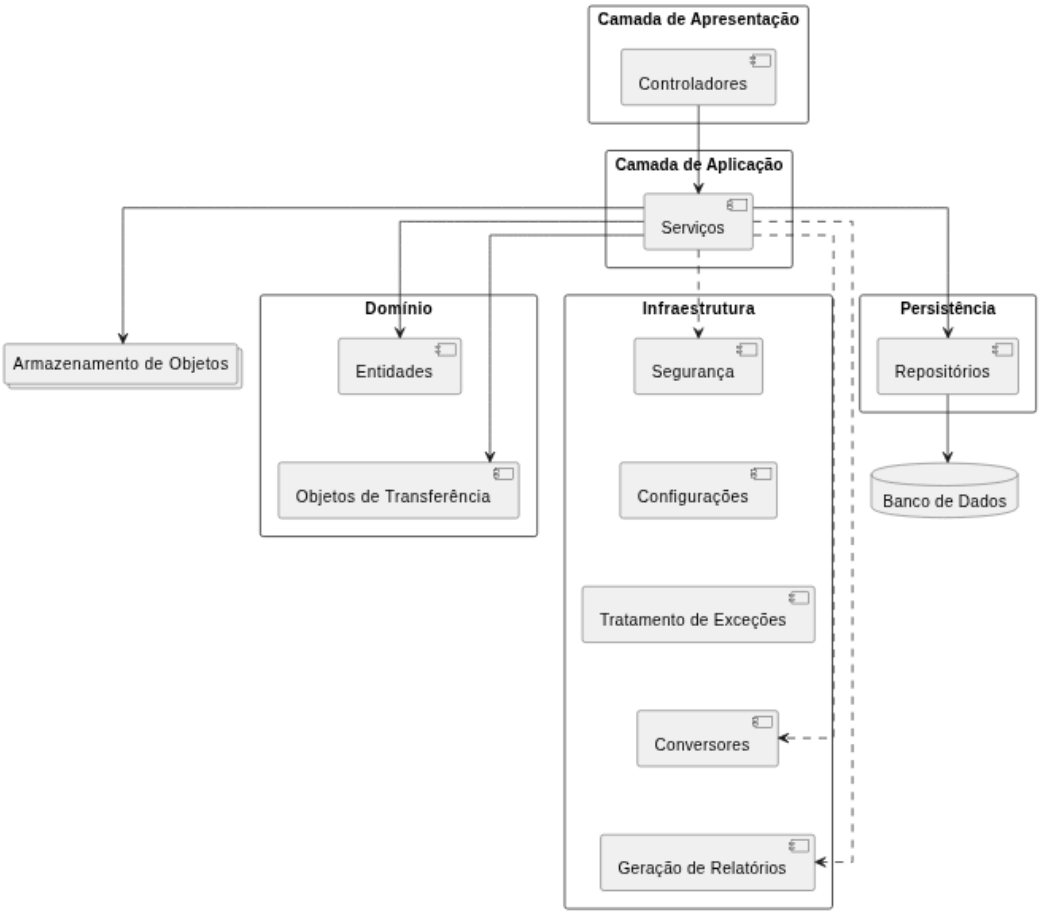

Figura 8: Arquitetura em camadas do backend. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

3.3.2 Arquitetura do Frontend

A arquitetura do frontend foi organizada de forma modular, separando os componentes responsáveis pela navegação, apresentação das informações, gerenciamento do estado da aplicação e comunicação com os serviços disponibilizados pelo backend. Essa organização favorece a reutilização de componentes, reduz o acoplamento entre diferentes partes do sistema e contribui para a manutenção do código ao longo da evolução da aplicação (FOWLER, 2002).

A navegação é responsável pela organização das telas e pelo fluxo de interação do usuário com o sistema. Cada funcionalidade é implementada por meio de telas específicas, que utilizam componentes reutilizáveis para composição da interface e padronização dos elementos visuais da aplicação.

O gerenciamento das informações compartilhadas entre diferentes telas é realizado por componentes responsáveis pela manutenção do estado da aplicação. Dessa forma, evita-se a propagação excessiva de dados entre componentes, permitindo que informações relacionadas ao usuário autenticado, pacientes e questionários permaneçam disponíveis durante a navegação.

A comunicação com o backend é concentrada em uma camada de serviços, responsável por encapsular as operações de acesso aos dados e isolar a interface da aplicação das particularidades da comunicação com a camada de serviços do sistema.

A Figura 9 apresenta a organização arquitetural adotada para o frontend.

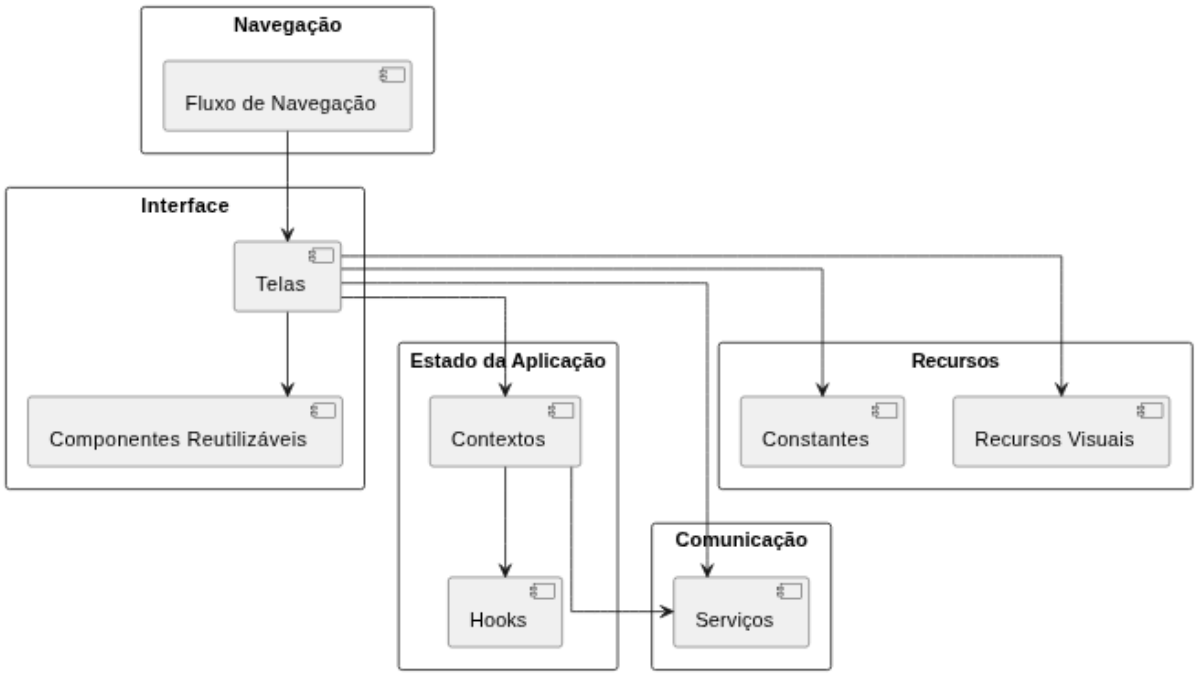

Figura 9: Arquitetura modular do frontend. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

3.4 Tecnologias Adotadas

3.4.1 Frontend

3.4.1.1 React Native

A escolha do React Native mostrou-se adequada ao contexto deste trabalho por permitir o desenvolvimento de aplicações para Android e iOS a partir de uma única base de código, reduzindo

o esforço de implementação e manutenção característico de um Produto Viável Mínimo (MVP). Embora alternativas como Flutter também ofereçam desenvolvimento multiplataforma, o React Native foi selecionado em razão da maturidade de seu ecossistema, da ampla disponibilidade de bibliotecas compatíveis com o Expo, da extensa documentação técnica e de sua ampla adoção na indústria, fatores que contribuem para maior produtividade durante o desenvolvimento (EISENMAN, 2018; REACT NATIVE, 2025; EXPO, 2024). Durante a implementação deste trabalho, a validação do aplicativo foi realizada em dispositivos Android, uma vez que o ambiente de desenvolvimento disponível não contava com dispositivos iOS. Essa decisão restringiu apenas o ambiente de testes, não a arquitetura da aplicação, que permanece apta à futura disponibilização em ambas as plataformas. Adicionalmente, o Android representa a maior parcela do mercado brasileiro de sistemas operacionais móveis, tornando essa plataforma particularmente relevante para a validação inicial do protótipo (STATCOUNTER GLOBAL STATS, 2026).

3.4.1.2 Expo Router

Biblioteca de roteamento baseada em arquivos para aplicações React Native desenvolvidas com Expo. Essa abordagem simplifica a navegação entre telas ao reduzir a necessidade de configurações manuais complexas, promovendo maior organização estrutural do projeto e aumento da produtividade no desenvolvimento (EXPO, 2024). Além disso, sua integração nativa com o ecossistema Expo facilita os processos de teste, build e deploy da aplicação.

3.4.2 Backend

3.4.2.1 Spring Boot

Framework para desenvolvimento de aplicações web em Java que oferece mecanismos de configuração automática e um amplo conjunto de ferramentas integradas. Essas características reduzem a complexidade da configuração inicial e aceleram o desenvolvimento de aplicações robustas e escaláveis. Amplamente adotado em ambientes corporativos e acadêmicos devido à sua maturidade, modularidade e forte integração com o ecossistema Spring (SPRING, 2024).

Embora outras tecnologias, como Jakarta EE e Micronaut, também permitam o desenvolvimento de aplicações corporativas em Java, apresentam características distintas. A especificação Jakarta EE oferece um conjunto padronizado de APIs voltadas ao desenvolvimento de aplicações empresariais, permitindo diferentes implementações compatíveis, enquanto o Micronaut foi concebido com foco em baixo consumo de memória, rápida inicialização e processamento das configurações em tempo de compilação, características especialmente relevantes em ambientes de microsserviços e computação em nuvem (JAKARTA EE, 2025; MICRONAUT FOUNDATION, 2025). Para o contexto deste trabalho, entretanto, o Spring Boot mostrou-se mais adequado por disponibilizar os recursos já mencionados (um ecossistema amplamente consolidado que integra, de forma nativa, recursos de autenticação, persistência, validação e configuração automática, reduzindo significativamente o esforço de desenvolvimento e a quantidade de código de infraestrutura necessária à implementação da aplicação).

3.4.2.2 Arquitetura REST

A adoção do estilo arquitetural REST decorreu das características funcionais do sistema e da necessidade de privilegiar simplicidade, interoperabilidade e facilidade de manutenção. O REMO realiza predominantemente operações de cadastro, consulta, atualização e remoção de dados, organizadas em torno de recursos bem definidos e consumidas por uma única aplicação cliente. Nesse contexto, REST apresenta uma solução suficientemente expressiva, amplamente suportada pelo ecossistema Spring Boot e consolidada no desenvolvimento de aplicações distribuídas (FIELDING, 2000; SPRING, 2024). Alternativas como GraphQL oferecem maior flexibilidade para consultas realizadas pelo cliente, reduzindo problemas de sobrecarga ou subutilização de dados,

enquanto gRPC possibilita comunicação de alta performance baseada em HTTP/2 e serialização binária por Protocol Buffers (GRAPHQL FOUNDATION, 2025; GRPC AUTHORS, 2025). Entretanto, essas abordagens introduzem maior complexidade arquitetural e seus principais benefícios tornam-se mais evidentes em cenários envolvendo múltiplos clientes, grande volume de requisições ou requisitos rigorosos de desempenho, características que não fazem parte do escopo do Produto Viável Mínimo desenvolvido neste trabalho. Dessa forma, a utilização de uma API REST mostrou-se a alternativa mais adequada por conciliar baixo custo de implementação, ampla adoção na indústria e facilidade de evolução da aplicação.

3.4.2.3 Spring Data JPA

O acesso aos dados é realizado por meio do Spring Data JPA, módulo do ecossistema Spring que simplifica a implementação da camada de persistência. A tecnologia permite abstrair operações comuns de banco de dados por meio de interfaces especializadas, reduzindo a quantidade de código necessária para operações de consulta, inserção, atualização e remoção de registros.

Sua utilização contribui para o aumento da produtividade no desenvolvimento e para a padronização da camada de acesso a dados, além de promover maior integração com os demais componentes do framework Spring (SPRING, 2025).

3.4.2.4 Hibernate

O Hibernate é utilizado como framework de mapeamento objeto-relacional (Object Relational Mapping – ORM), responsável por estabelecer a correspondência entre as classes Java da aplicação e as tabelas do banco de dados. Por meio dessa abordagem, torna-se possível manipular dados utilizando objetos da linguagem de programação, reduzindo a necessidade de escrita manual de comandos SQL.

Além de simplificar a persistência dos dados, o Hibernate oferece mecanismos para gerenciamento de relacionamentos entre entidades, controle transacional e otimização de consultas, sendo amplamente adotado em aplicações corporativas desenvolvidas em Java (BAUER; KING; GREGORY, 2015).

3.4.2.5 Spring Security

A segurança da aplicação é implementada utilizando o Spring Security, framework responsável pelo controle de autenticação e autorização dos usuários. Sua integração com o Spring Boot permite a definição centralizada de políticas de acesso, protegendo recursos sensíveis da aplicação contra acessos não autorizados.

A utilização dessa tecnologia contribui para o desenvolvimento de aplicações mais seguras, oferecendo mecanismos consolidados para validação de credenciais, controle de sessões e integração com diferentes estratégias de autenticação (SPILCA, 2021).

3.4.2.6 JSON Web Token (JWT)

A autenticação dos usuários é realizada por meio da tecnologia JSON Web Token (JWT), padrão aberto para transmissão segura de informações entre partes na forma de objetos JSON assinados digitalmente. Após a realização do login, o backend gera um token que passa a acompanhar as requisições subsequentes realizadas pelo usuário autenticado.

Essa abordagem elimina a necessidade de armazenamento de sessões no servidor, favorecendo a escalabilidade da aplicação e reduzindo a complexidade do gerenciamento de autenticação. O padrão JWT é amplamente utilizado em aplicações web e móveis modernas devido à sua simplicidade e eficiência (JONES; BRADLEY; SAKIMURA, 2015).

3.4.2.7 Bean Validation

A validação dos dados recebidos pela API é realizada por meio da especificação Jakarta Bean Validation. Essa tecnologia permite definir regras de validação diretamente nos objetos utilizados pela aplicação, garantindo que informações obrigatórias sejam preenchidas corretamente antes de serem processadas pelo sistema.

A adoção dessa abordagem contribui para a integridade dos dados armazenados, reduzindo inconsistências e melhorando a confiabilidade das operações realizadas pelos usuários (JAKARTA EE, 2023).

3.4.2.8 Apache PDFBox

A geração de relatórios em formato PDF é realizada com o auxílio da biblioteca Apache PDFBox. Essa ferramenta oferece recursos para criação, manipulação, edição e combinação de documentos PDF diretamente em aplicações Java.

No contexto do projeto, sua utilização permite a geração automática de relatórios contendo os resultados das avaliações realizadas e as respectivas estratégias de intervenção recomendadas ao profissional de saúde, contribuindo para a formalização e registro das informações coletadas durante os atendimentos (APACHE SOFTWARE FOUNDATION, 2025).

3.4.2.9 Java 17 e Maven

O backend foi desenvolvido utilizando a linguagem Java na versão 17, considerada uma versão de suporte de longo prazo (Long-Term Support – LTS). Essa versão oferece melhorias de desempenho, segurança e estabilidade em relação a versões anteriores, sendo amplamente adotada em aplicações corporativas modernas (ORACLE, 2025).

O gerenciamento das dependências e do processo de compilação é realizado com o Apache Maven, ferramenta amplamente utilizada no ecossistema Java para automação de builds, gerenciamento de bibliotecas externas e padronização da estrutura dos projetos. Sua adoção facilita a manutenção do ambiente de desenvolvimento e contribui para a reprodutibilidade do projeto em diferentes ambientes (APACHE MAVEN PROJECT, 2025).

3.4.3 Banco de Dados

3.4.3.1 H2 Database

Sistema de gerenciamento de banco de dados relacional em memória, utilizado principalmente durante as fases iniciais de desenvolvimento e testes. Sua principal contribuição na etapa de implementação está na facilidade de configuração, rápida inicialização e integração nativa com o Spring Boot, permitindo a validação da camada de persistência sem a dependência de um banco de dados externo (H2 DATABASE, 2024; SPRING BOOT, 2024). Seu uso é recomendado em contextos de teste para reduzir custos de infraestrutura e acelerar ciclos de desenvolvimento, eliminando a necessidade de servidores de banco de dados externos complexos durante a integração contínua (KLEPPMANN, 2017; CRUDU, 2025).

A utilização do H2 restringe-se ao ambiente de desenvolvimento. Em ambientes de produção, sistemas gerenciadores de banco de dados como PostgreSQL e MySQL disponibilizam mecanismos mais robustos para controle de concorrência, otimização de consultas, recuperação de falhas e suporte à alta disponibilidade, características necessárias para aplicações com múltiplos usuários e grandes volumes de dados (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2025; ORACLE, 2025). Entretanto, para um MVP acadêmico, a inicialização instantânea, integração nativa com

Spring Boot e a ausência de configuração adicional tornam o H2 uma alternativa adequada para validação das funcionalidades implementadas (H2 DATABASE, 2024; SPRING, 2024).

3.4.3.2 MinIO

O armazenamento dos documentos gerados pelo sistema é realizado utilizando o MinIO, uma plataforma de armazenamento de objetos compatível com a API Amazon S3. Outras alternativas, como serviços gerenciados de armazenamento de objetos disponibilizados por provedores de computação em nuvem, a exemplo do Amazon S3, também poderiam ser empregadas para essa finalidade, oferecendo recursos adicionais de escalabilidade, alta disponibilidade e gerenciamento da infraestrutura (AMAZON WEB SERVICES, 2025). Entretanto, para o desenvolvimento do Produto Viável Mínimo, o MinIO mostrou-se mais adequado por disponibilizar compatibilidade com a API S3, execução local sem dependência de serviços em nuvem e facilidade de integração com aplicações Java, permitindo que uma futura migração para soluções gerenciadas possa ser realizada com impacto reduzido sobre a aplicação (MINIO, 2025). Sua utilização também mantém a separação entre o armazenamento de arquivos e o banco de dados relacional, favorecendo a organização da arquitetura da aplicação e a evolução futura da infraestrutura.

3.4.4 Serviços Integrados

Além das tecnologias que compõem diretamente a aplicação, o sistema REMO utiliza serviços externos responsáveis por fornecer funcionalidades complementares ao processo de desenvolvimento e à execução do software. Esses serviços contribuem para a segurança da aplicação e para a obtenção automática de informações utilizadas durante o cadastro dos pacientes.

3.4.4.1 Infisical

Para o gerenciamento de informações sensíveis da aplicação foi utilizado o Infisical. Essa ferramenta permite armazenar credenciais, chaves de acesso e demais variáveis de ambiente de forma centralizada, evitando que esses dados sejam inseridos diretamente no código-fonte ou versionados juntamente com o projeto.

Durante a execução da aplicação, o Infisical disponibiliza essas informações por meio de variáveis de ambiente, permitindo que o sistema acesse credenciais necessárias ao funcionamento de serviços externos sem comprometer a segurança do projeto. Essa abordagem reduz o risco de exposição de informações sensíveis e facilita o gerenciamento de diferentes ambientes de desenvolvimento (INFISICAL, 2025).

3.4.4.2 ViaCEP

Durante o cadastro de pacientes, o sistema utiliza a API pública ViaCEP para realizar a consulta automática de endereços a partir do Código de Endereçamento Postal (CEP). Após a informação do CEP pelo usuário, o serviço retorna dados como logradouro, bairro, município e unidade federativa, reduzindo a necessidade de preenchimento manual e minimizando erros de digitação.

A utilização desse serviço também contribui para aumentar a consistência das informações armazenadas no sistema, uma vez que os dados retornados são obtidos diretamente da base disponibilizada pelo serviço ViaCEP (VIACEP, 2025).

3.4.4.3 API de Localidades do IBGE

Complementando a consulta realizada pelo ViaCEP, o sistema utiliza a API de Localidades do Instituto Brasileiro de Geografia e Estatística (IBGE) para obtenção da relação oficial de estados e

municípios brasileiros. Essas informações são utilizadas na interface da aplicação para disponibilizar listas padronizadas durante o preenchimento dos dados cadastrais dos pacientes.

A utilização da API do IBGE contribui para a padronização dos dados geográficos utilizados pela aplicação, reduzindo inconsistências decorrentes da entrada manual de informações e garantindo alinhamento com a divisão territorial oficial do país (IBGE, 2025).

3.5 Aspectos de Segurança da Informação

O sistema REMO foi concebido considerando requisitos de segurança compatíveis com o contexto de uma aplicação destinada ao registro de informações relacionadas ao acompanhamento de pacientes. Embora o presente trabalho tenha como objetivo o desenvolvimento de um Produto Viável Mínimo (MVP), algumas práticas consolidadas de desenvolvimento seguro foram incorporadas à arquitetura da aplicação visando contribuir para a proteção das informações armazenadas, para o controle de acesso aos recursos do sistema e para a confidencialidade dos dados tratados (SOMMERVILLE, 2011; NIST, 2020).

3.5.1 Proteção dos Dados

A aplicação adota mecanismos destinados à proteção das credenciais de acesso e das informações utilizadas durante sua execução. Conforme apresentado na Seção 3.4.4.1, informações sensíveis, como chaves de acesso e credenciais de serviços externos, são mantidas fora do código-fonte por meio de um gerenciador de segredos, reduzindo o risco de exposição acidental desses dados em repositórios de código ou arquivos de configuração.

As senhas dos usuários não são armazenadas em texto puro. Antes de serem persistidas, passam por um processo de derivação criptográfica utilizando algoritmo apropriado para armazenamento de senhas, prática amplamente recomendada para reduzir os impactos decorrentes de eventual comprometimento da base de dados (PROVOS; MAZIÈRES, 1999; SPILCA, 2021).

Além disso, o sistema mantém separação entre os dados estruturados e os documentos armazenados em formato PDF. Conforme discutido na Seção 3.1.3, o banco de dados relacional armazena apenas as referências aos documentos persistidos no mecanismo de armazenamento de objetos, reduzindo o volume de dados binários mantidos na base relacional e favorecendo uma organização mais adequada da persistência.

3.5.2 Autenticação e Controle de Acesso

O acesso às funcionalidades da aplicação é condicionado à autenticação prévia do profissional de saúde. Conforme descrito na Seção 3.4.2, esse processo é implementado utilizando Spring Security em conjunto com JSON Web Token (JWT), permitindo que cada requisição encaminhada ao backend seja associada a um usuário autenticado. Essa abordagem elimina a necessidade de manutenção de sessões no servidor e contribui para a escalabilidade da aplicação (JONES; BRADLEY; SAKIMURA, 2015; SPILCA, 2021).

Além da autenticação, a aplicação restringe o acesso aos serviços disponibilizados pela API, impedindo que usuários não autenticados executem operações sobre os recursos protegidos. A validação centralizada das credenciais e das permissões de acesso contribui para reduzir a superfície de ataque da aplicação e facilita a implementação de políticas de autorização mais específicas em versões futuras do sistema.

3.5.3 Considerações sobre Privacidade

O REMO manipula informações relacionadas à identificação de pacientes e ao acompanhamento de seu estado de saúde. No contexto da Lei Geral de Proteção de Dados Pessoais (LGPD), essas informações enquadram-se como dados pessoais e dados pessoais sensíveis, exigindo tratamento compatível com os princípios estabelecidos pela legislação brasileira (BRASIL, 2018).

Por se tratar de um MVP desenvolvido para fins acadêmicos, o sistema não contempla mecanismos avançados de governança, auditoria ou gestão do ciclo de vida dos dados. Entretanto, sua arquitetura foi organizada de forma a permitir a incorporação futura de funcionalidades como registro de auditoria, trilhas de rastreabilidade, criptografia de dados em repouso, utilização obrigatória de HTTPS em ambientes de produção e mecanismos mais refinados de controle de acesso, alinhando a evolução da aplicação às boas práticas de segurança da informação e às recomendações para proteção de dados pessoais (NIST, 2020).

4. Implantação

4.1 Projeto de Implantação

A implantação do sistema REMO foi realizada em ambiente local, utilizando um notebook de uso pessoal como plataforma de desenvolvimento e execução da aplicação. A infraestrutura empregada contempla os componentes arquiteturais apresentados na Seção 3.3 e as tecnologias descritas na Seção 3.4, sendo suficiente para execução integrada do Produto Viável Mínimo (MVP).

O ambiente de desenvolvimento é composto por uma aplicação backend desenvolvida em Spring Boot, uma aplicação móvel desenvolvida em React Native com Expo, um banco de dados relacional H2 e um serviço de armazenamento de objetos MinIO para persistência dos documentos PDF utilizados pelo sistema. O gerenciamento das variáveis de ambiente e das credenciais de acesso é realizado por meio do Infisical, evitando o armazenamento de informações sensíveis diretamente no código-fonte.

Durante o desenvolvimento, foram utilizadas as versões gratuitas do MinIO e do Infisical, uma vez que ambas oferecem recursos suficientes para atender às necessidades de um ambiente acadêmico de desenvolvimento e validação funcional da aplicação.

Para disponibilizar o sistema em funcionamento no ambiente local, a sequência de execução adotada consiste em:

1. inicialização da aplicação backend, com carregamento das variáveis de ambiente fornecidas pelo Infisical;
2. inicialização do serviço de armazenamento de objetos (MinIO);
3. conexão do dispositivo Android utilizado para testes, com o modo desenvolvedor habilitado;
4. compilação e execução da aplicação móvel no dispositivo por meio do Expo.

Embora essa infraestrutura tenha sido utilizada durante o desenvolvimento do MVP, sua arquitetura permite a substituição dos componentes de infraestrutura por soluções equivalentes empregadas em ambientes de produção. A utilização do Spring Data JPA desacopla a camada de persistência da implementação específica do sistema gerenciador de banco de dados, permitindo a migração para outros SGBDs compatíveis com JPA com reduzido impacto na lógica da aplicação. De forma semelhante, a adoção da API compatível com Amazon S3 pelo MinIO possibilita sua substituição por serviços de armazenamento de objetos compatíveis, enquanto o uso de variáveis de ambiente para gerenciamento de configurações facilita a adoção de outros gerenciadores de segredos, preservando a implementação da aplicação.

Da mesma forma, os serviços utilizados durante a execução podem ser conteinerizados, permitindo sua inicialização automática e padronizada por meio de ferramentas como Docker e Docker Compose. Essa abordagem reduz diferenças entre ambientes de desenvolvimento, teste e produção, simplifica o processo de implantação, favorece o isolamento entre serviços e contribui para a reprodutibilidade da infraestrutura, características amplamente recomendadas no desenvolvimento moderno de aplicações distribuídas (MERKEL, 2014).

A Figura 10 apresenta o diagrama de implantação correspondente ao ambiente utilizado durante o desenvolvimento do sistema.

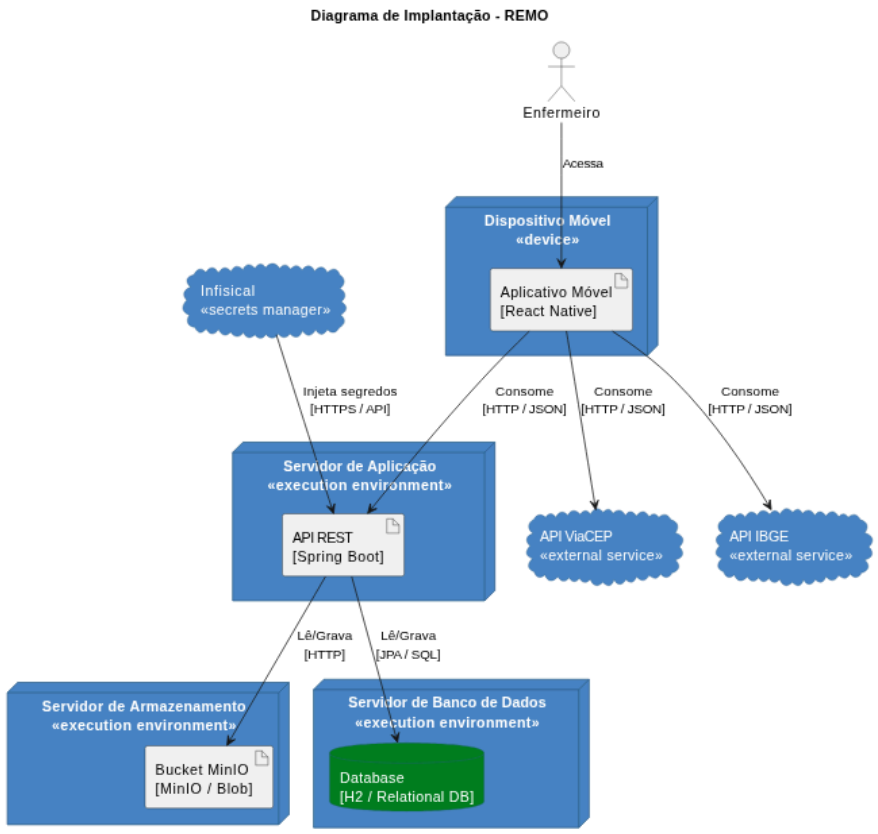

Figura 10: Diagrama de Implantação do REMO. Fonte: Elaborado pelo autor com a ferramenta PlantUML.

5. Trabalhos Futuros

O desenvolvimento do REMO foi conduzido seguindo a abordagem de Produto Viável Mínimo (MVP), priorizando a implementação das funcionalidades essenciais para validação da proposta apresentada neste trabalho. Dessa forma, diversas possibilidades de evolução permanecem abertas para versões futuras da aplicação, tanto sob a perspectiva funcional quanto arquitetural.

No âmbito funcional, uma evolução natural consiste na ampliação dos perfis de usuários suportados pelo sistema. Embora a versão atual tenha sido desenvolvida para utilização por profissionais de enfermagem, a arquitetura adotada permite sua extensão para outros profissionais da saúde, como médicos, psicólogos, fisioterapeutas e assistentes sociais, possibilitando uma abordagem multiprofissional do acompanhamento de pacientes com doenças crônicas.

Outra possibilidade consiste no desenvolvimento de uma interface Web destinada ao gerenciamento administrativo da plataforma. Essa aplicação poderia oferecer funcionalidades complementares às disponíveis no aplicativo móvel, como administração de usuários, gerenciamento das estratégias de intervenção, parametrização dos questionários e acompanhamento estatístico da utilização do sistema.

Também podem ser incorporados mecanismos de visualização gráfica da evolução clínica dos pacientes, permitindo ao profissional de saúde acompanhar indicadores relacionados às avaliações realizadas ao longo do tempo. A utilização de painéis estatísticos (dashboards) pode facilitar a identificação de padrões de evolução, contribuindo para o processo de tomada de decisão baseado em evidências (FEW, 2013).

Do ponto de vista da infraestrutura, a arquitetura apresentada neste trabalho possibilita sua evolução para ambientes de produção com maior demanda de processamento. Conforme discutido nas seções anteriores, componentes atualmente empregados durante o desenvolvimento, como o banco de dados H2 e a implantação local do serviço de armazenamento de objetos, podem ser substituídos por soluções corporativas sem necessidade de alterações significativas nas regras de negócio da aplicação. Da mesma forma, a utilização de contêineres Docker, orquestradores como Kubernetes e serviços de computação em nuvem pode ampliar a disponibilidade, a escalabilidade e a facilidade de implantação da solução em ambientes distribuídos (MERKEL, 2014; BURNS; BEDA; HIGHTOWER, 2022).

Outra evolução relevante consiste na adoção de mecanismos de interoperabilidade com sistemas de informação em saúde por meio do padrão Fast Healthcare Interoperability Resources (FHIR), desenvolvido pela HL7 International. A utilização desse padrão possibilitaria a integração do REMO com prontuários eletrônicos e outras plataformas utilizadas por instituições de saúde, reduzindo a duplicidade de registros e favorecendo o compartilhamento seguro de informações clínicas (HL7 INTERNATIONAL, 2023).

Adicionalmente, futuras versões da aplicação poderão incorporar recursos de inteligência artificial para apoiar a análise das avaliações realizadas e fornecer recomendações complementares ao profissional de saúde. Ressalta-se, entretanto, que tais mecanismos devem atuar como instrumentos de apoio à decisão clínica, preservando a autonomia do profissional responsável pelo atendimento e observando princípios relacionados à transparência, confiabilidade e supervisão humana, conforme recomendações internacionais para utilização de sistemas de inteligência artificial na área da saúde (WORLD HEALTH ORGANIZATION, 2021).

Por fim, podem ser incorporados mecanismos de sincronização offline, permitindo a utilização do sistema em ambientes com conectividade limitada. Nessa abordagem, os dados seriam armazenados temporariamente no dispositivo móvel e sincronizados automaticamente com o servidor quando uma conexão com a Internet estivesse disponível, ampliando as possibilidades de utilização da aplicação em diferentes contextos de atendimento.

6. Manual do Usuário

O aplicativo inicia na tela de login (figura 11). O usuário deve tocar no texto *Cadastre-se*, destacado em verde para ser direcionado à tela de cadastro (figura 12).

Figura 11: Tela de login do REMO. Fonte: Elaborado pelo autor.

Figura 12: Tela de cadastro do REMO. Fonte: Elaborado pelo autor.

Então, o usuário deve preencher seus dados e tocar no botão *Cadastrar*. Feito isso, o usuário será redirecionado para a tela de login, onde deverá inserir suas credenciais e iniciar a sessão tocando no botão *Entrar*.

Depois de efetuar o login, a primeira tela é a *Home* (figura 13).

Figura 13: Tela Home do REMO. Fonte: Elaborado pelo autor.

Nessa tela, o usuário tem as seguintes opções:

- Acessar seu perfil (tocando nas três listras brancas no canto direito superior);
- Cadastrar um novo paciente (tocando em *Novo Cadastro*);
- Visualizar os pacientes cadastrados (tocando em *Meus Pacientes*, *Ver todos* ou no ícone com três bonecos em cinza na base da tela).

A medida que o usuário começar a cadastrar pacientes, os mais recentes serão listados abaixo de *Últimas Coletas*.

A próxima funcionalidade significativa é a de cadastrar um paciente, que é executada na tela de cadastro de paciente (figuras 14 e 15).

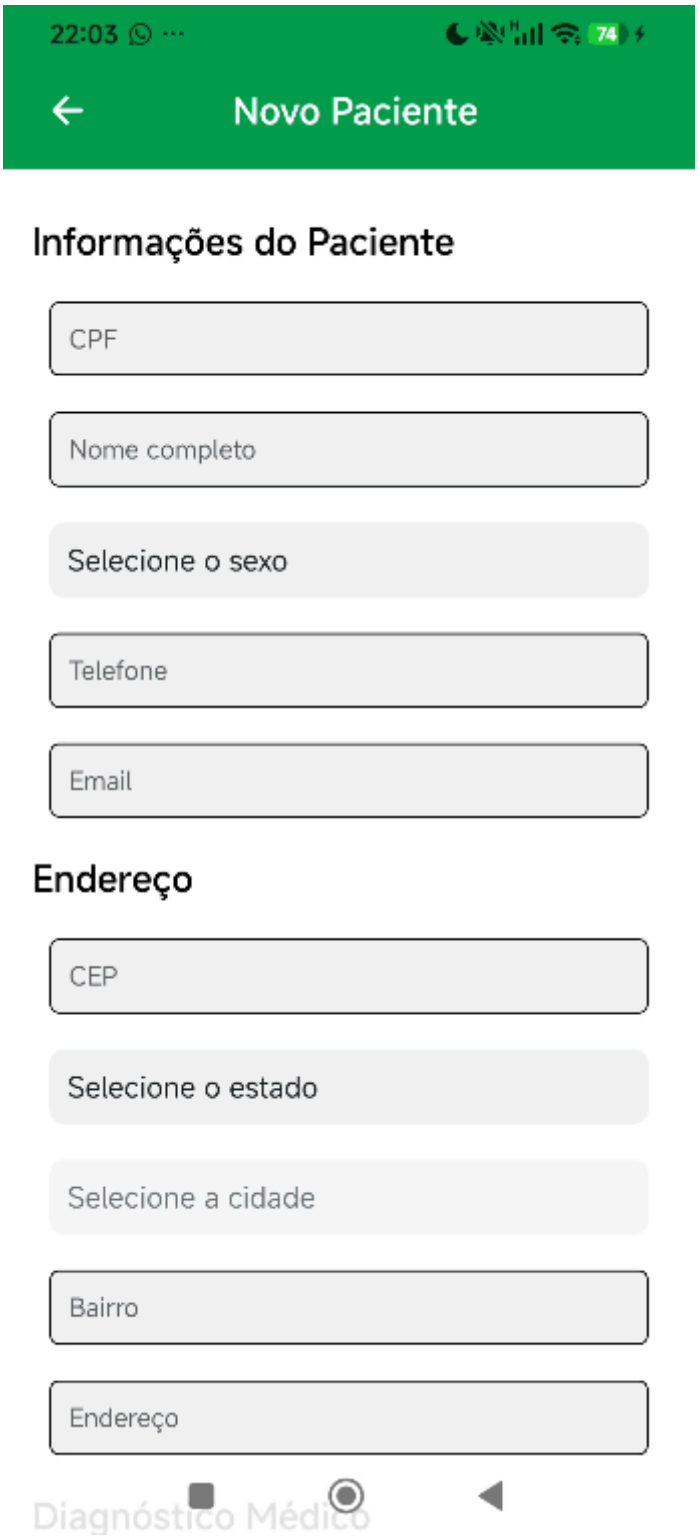

Figura 14: Tela de cadastro de paciente em branco. Fonte: Elaborado pelo autor.

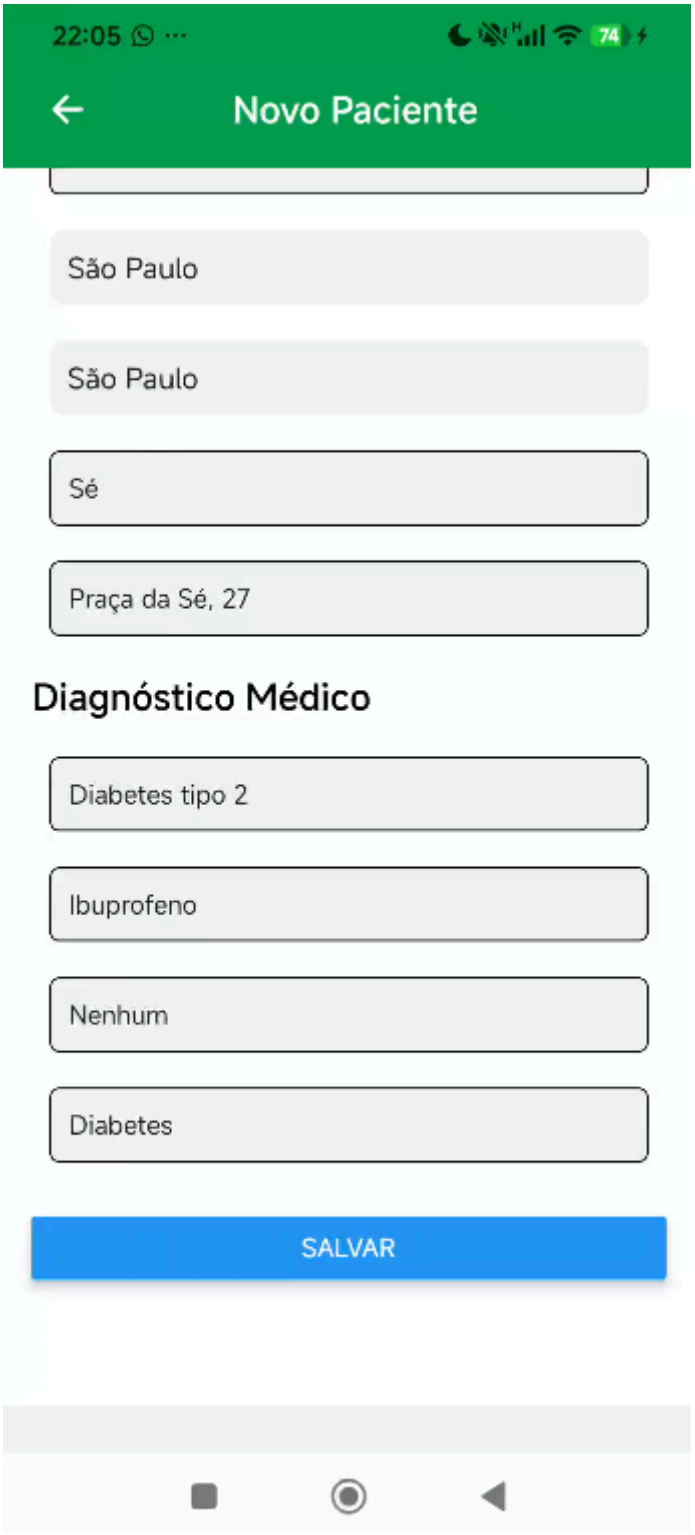

Figura 15: Tela de cadastro de paciente preenchida. Fonte: Elaborado pelo autor.

Nessa tela o usuário deve preencher os dados do paciente e tocar em *Salvar*. Na parte de preenchimento de endereço, basta inserir um CEP válido no início que os campos Estado, Cidade, Bairro e Endereço são preenchidos automaticamente ao tocar fora do campo de CEP. Se a API *ViaCEP* não localizar o CEP informado, é necessário preencher esses campos manualmente.

Uma vez que o usuário salve o novo paciente, ele receberá um aviso informando que o paciente foi registrado e, após tocar em *OK*, será direcionado para o preenchimento do questionário sobre o estado emocional do novo paciente (figuras 16 e 17).

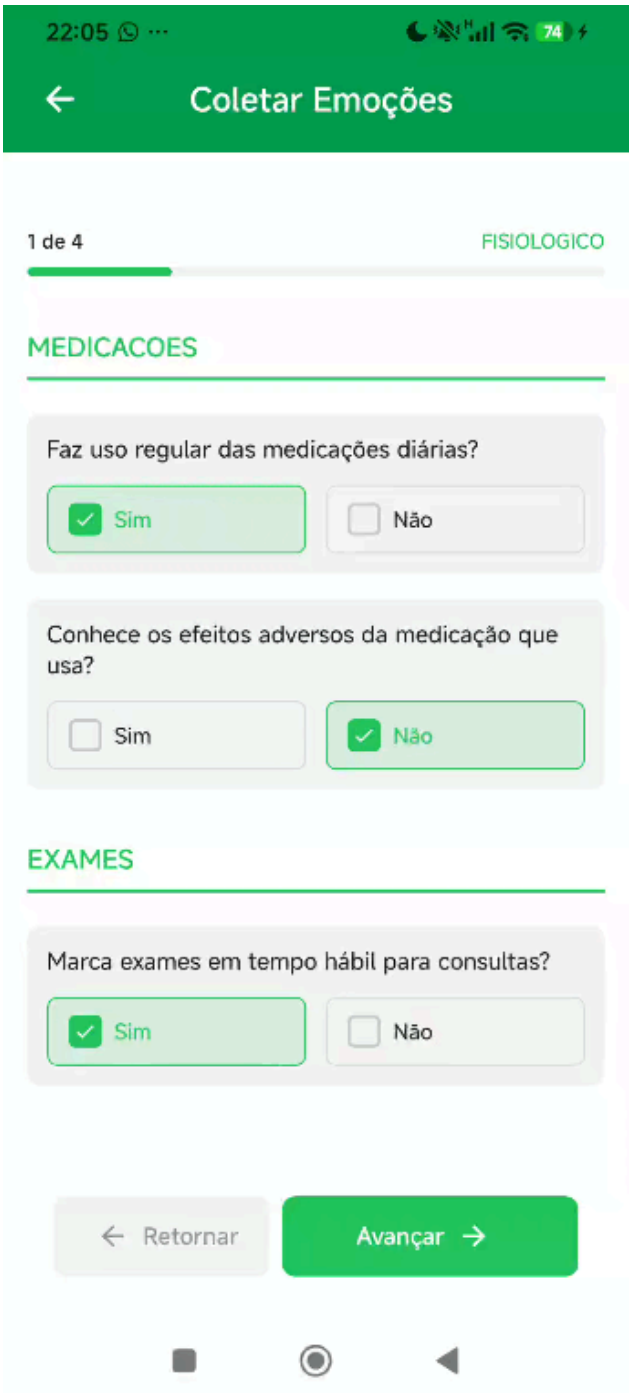

Figura 16: Primeira etapa do questionário emocional. Fonte: Elaborado pelo autor.

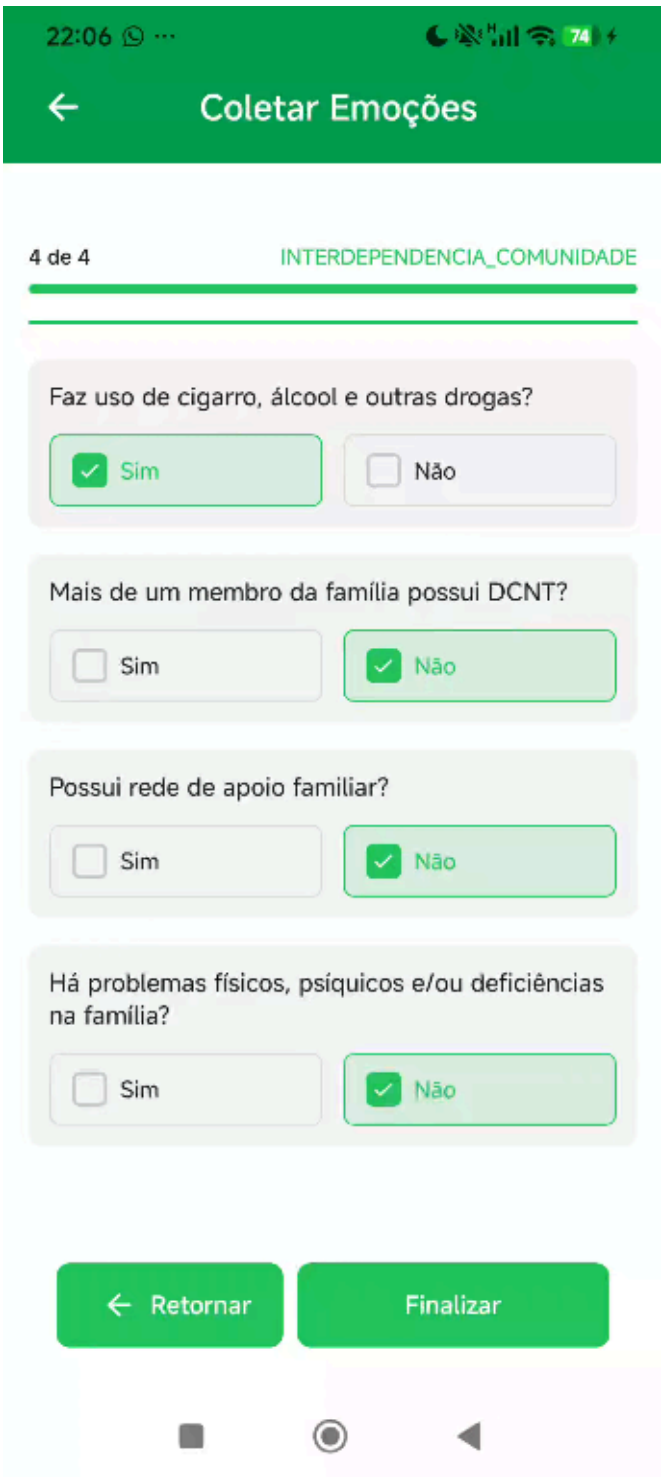

Figura 17: Etapa final do questionário emocional. Fonte: Elaborado pelo autor.

Todas as perguntas devem ser respondidas. Caso contrário, o usuário receberá um aviso informando que há perguntas sem resposta e o relatório não será gerado. Uma vez que todas as perguntas estejam respondidas, o usuário pode tocar em *Finalizar*.

O sistema leva alguns instantes para processar as respostas e gerar o relatório. Então, o usuário pode visualizar o relatório gerado (figura 18).

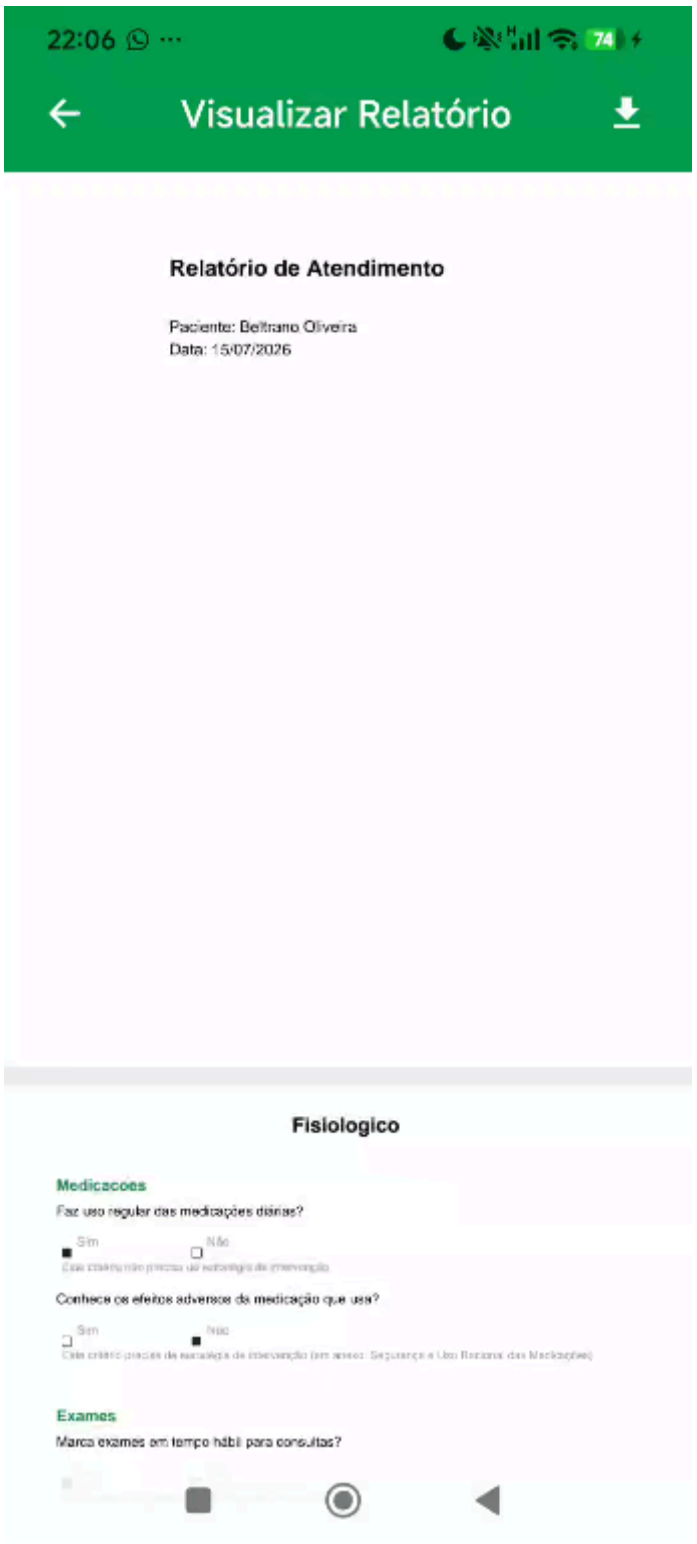

Figura 18: Visualização do relatório. Fonte: Elaborado pelo autor.

Nessa tela, o usuário pode ler todo o conteúdo do relatório e fazer o download do arquivo, tocando no ícone no canto superior direito.

Existem outras telas, já mencionadas, que podem ser acessadas a partir da *Home*.

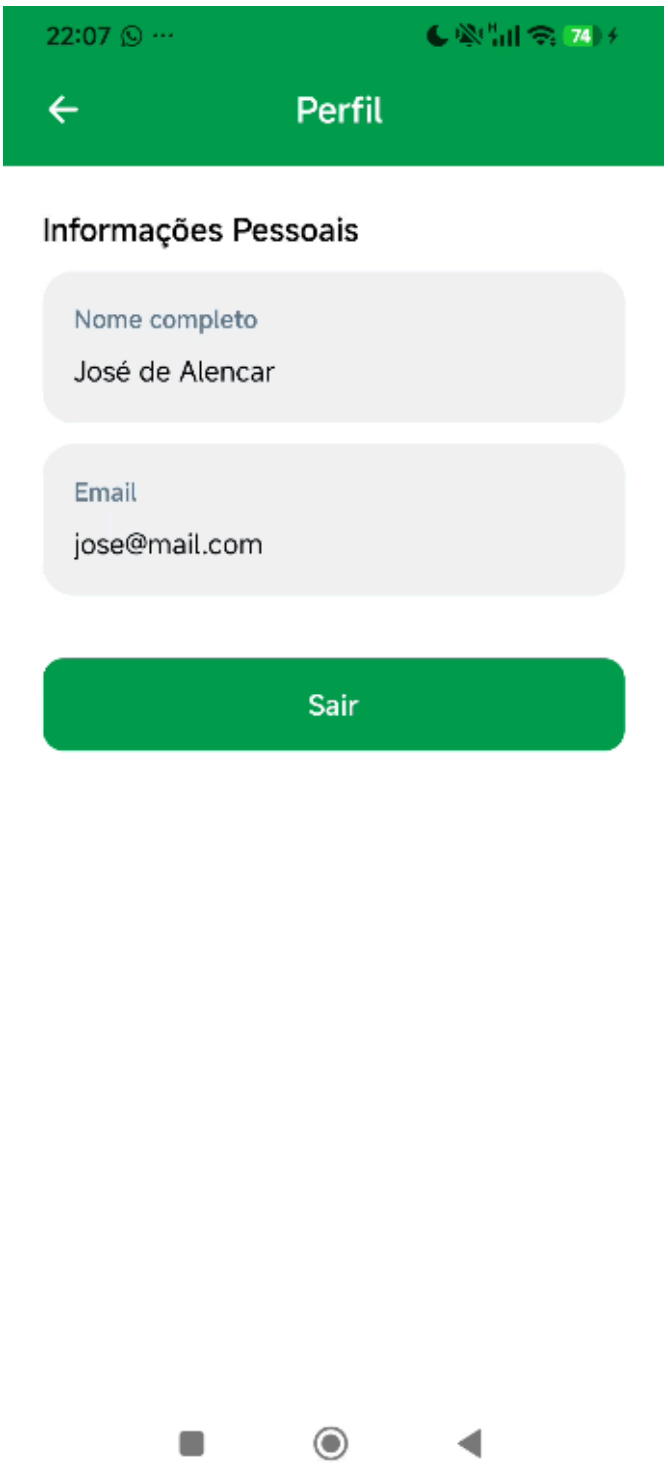

Figura 19: Tela de perfil do usuário. Fonte: Elaborado pelo autor.

Na tela de perfil, o usuário pode visualizar seus dados cadastrais, retornar à tela *Home* (ícone de seta para a esquerda, no canto superior esquerdo) ou efetuar logout (botão *Sair*).

Outra possibilidade, é acessar a tela de listagem dos pacientes cadastrados (figuras 20 e 21).

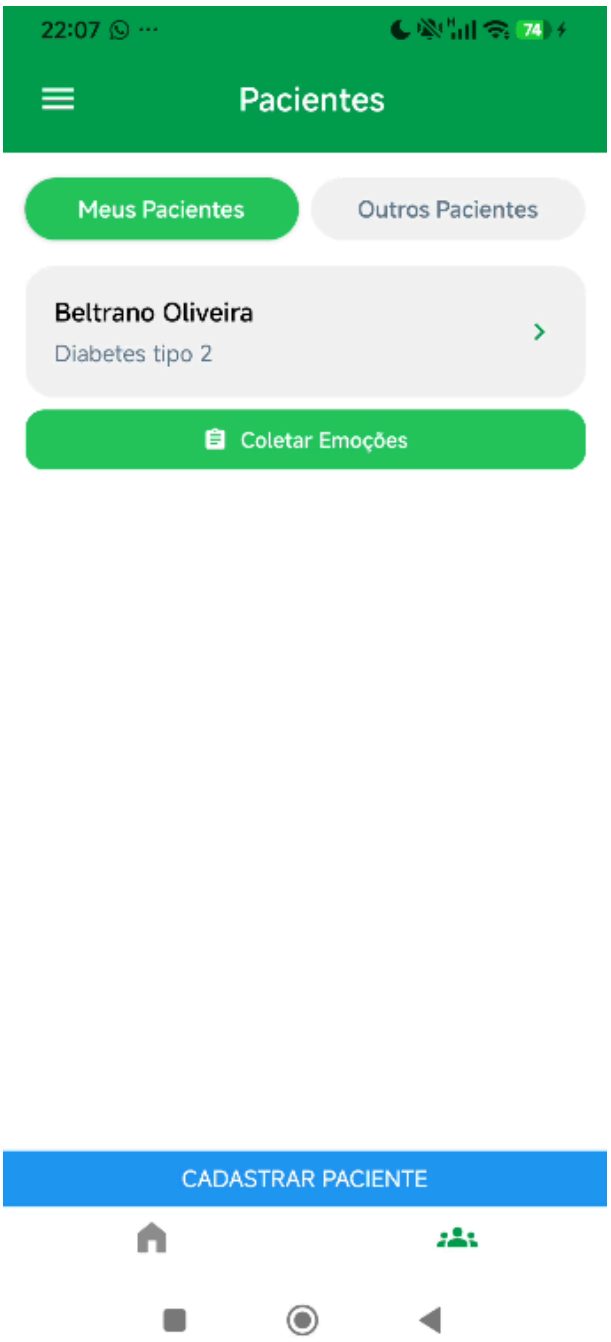

Figura 20: Tela de listagem dos pacientes cadastrados pelo usuário. Fonte: Elaborado pelo autor.

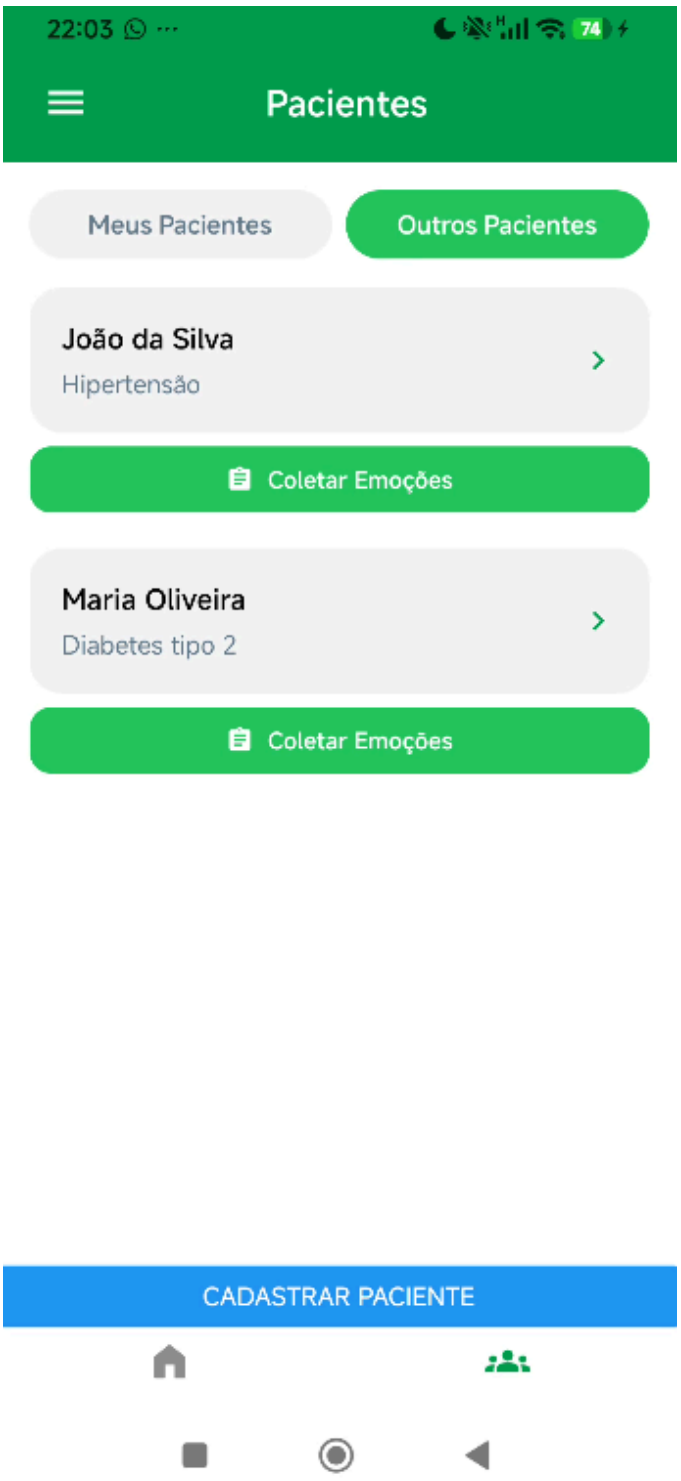

Figura 21: Tela de listagem dos pacientes cadastrados por outros usuários. Fonte: Elaborado pelo autor.

Nessa tela, é possível filtrar a lista de pacientes cadastrados pelo usuário ou por outros usuários tocando em Meus Pacientes ou Outros Pacientes. Além disso, também é possível acessar os detalhes de um determinado paciente, tocando na área acinzentada contendo o nome e a condição crônica deste paciente, ou iniciar o preenchimento de um novo questionário para este paciente tocando em Coletar Emoções, logo abaixo dessa área. Também é possível cadastrar um novo paciente tocando no botão azul na base da tela.

Ademais, há também a tela de detalhamento do paciente (figura 22).

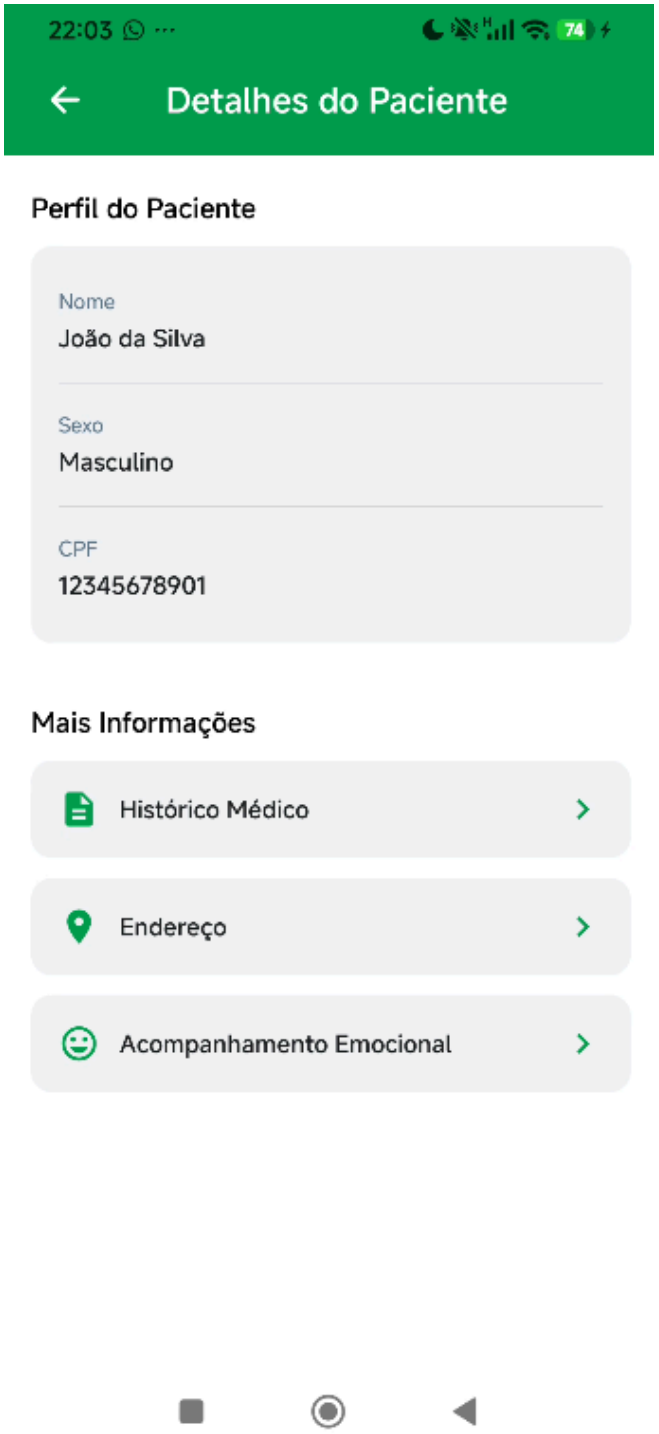

Figura 22: Tela de detalhamento do paciente. Fonte: Elaborado pelo autor.

Nesta tela é possível visualizar os principais dados e também acessar os dados médicos, o endereço e os relatórios gerados para este paciente, tocando em *Histórico Médico*, *Endereço* e *Acompanhamento Emocional*, respectivamente.

Agradecimentos

Em primeiro lugar, quero agradecer a Deus, meu Pai Criador e porto seguro nos momentos de dificuldade.

Agradeço aos meus pais, Carlos Alberto e Maria Augusta, por me trazerem ao mundo, formarem meu caráter e pelo apoio incondicional em todos os momentos.

Às minhas irmãs Nágila e Maria Clara, por sempre torcerem pelo meu sucesso e vibrarem comigo nos momentos de vitória.

Agradeço ao professor Antônio Carlos, que além de aceitar ser meu orientador, ajudou a conseguir o meu primeiro estágio, além de confiar em mim para participar de outros projetos na disciplina Redes de Computadores II. Estendo também esse agradecimento a todos os professores do CADS, que sempre tiveram uma postura humana e empática para com os alunos.

Aos amigos, não apenas colegas, que eu fiz ao longo desta trajetória no IFBA. Sem vocês, não seria a mesma coisa.

Agradeço também a Alciene Pereira da Silva, pela confiança e paciência depositados em mim para o desenvolvimento deste trabalho.

Referências

AMAZON WEB SERVICES. Amazon Simple Storage Service (Amazon S3) Documentation. [S.l.], 2025. Disponível em: https://docs.aws.amazon.com/s3/. Acesso em: 10 jun. 2026.

APACHE MAVEN PROJECT. Maven Documentation. [S.l.], 2025. Disponível em: https://maven.apache.org/guides/. Acesso em: 10 jun. 2026.

APACHE SOFTWARE FOUNDATION. Apache PDFBox Documentation. [S.l.], 2025. Disponível em: https://pdfbox.apache.org/. Acesso em: 10 jun. 2026.

AQUINO, Ana Caroliny Suassuna de; ROLIM, Lucíola Abílio Diniz Melquíades de Medeiros. Uso de aplicativos móveis para o monitoramento de mulheres com diabetes gestacional. Revista Eletrônica Acervo Científico, 2024.

BAUER, Christian; KING, Gavin; GREGORY, Gary. Java Persistence with Hibernate. 2. ed. Shelter Island: Manning Publications, 2015.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Diário Oficial da União: Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 10 jun. 2026.

BURNS, Brendan; BEDA, Joe; HIGHTOWER, Kelsey. Kubernetes: Up and Running. 3. ed. Sebastopol: O'Reilly Media, 2022.

BUTT, S. A. et al. Remote mobile health monitoring frameworks and applications. Engineering Applications of Artificial Intelligence, v. 127, 2024.

CRUDU, Vasile. Effective H2 Database Usage for In-Memory Hibernate Testing. MoldStud, [S. l.], 24 fev. 2025. Disponível em: https://moldstud.com/articles/p-effective-h2-database-usage-for-in-memory-hibernate-testing. Acesso em: 14 jul. 2026.

COUTINHO, Rebeca Vitória de Aguiar. Projeto Pronto Afeto. 2025. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) — Instituto Federal da Bahia, Salvador, 2025.

DEBON, Raquel; COLEONE, Joane Diomara; BELLEI, Ericles Andrei; DE MARCHI, Ana Carolina Bertoletti. Mobile health applications for chronic diseases: a systematic review of features for lifestyle improvement. Diabetes & Metabolic Syndrome: Clinical Research & Reviews, v. 13, n. 4, p. 2507–2512, 2019.

EISENMAN, Bonnie. Learning React Native. 2. ed. Sebastopol: O'Reilly Media, 2018.

ELMASRI, Ramez; NAVATHE, Shamkant B. Fundamentals of Database Systems. 7. ed. Boston: Pearson, 2016.

EXPO. Expo Router Documentation. [S.l.], 2024. Disponível em: https://docs.expo.dev/router/introduction/. Acesso em: 10 jun. 2026.

FACEBOOK. React Native: A Framework for Building Native Apps Using React. [S.l.], 2024. Disponível em: https://reactnative.dev/. Acesso em: 10 jun. 2026.

FERREIRA, E. S. et al. Mobile solution and chronic diseases: development and implementation of a mobile application and digital platform for collecting, analyzing data, monitoring and managing health care. BMC Health Services Research, 2024.

FEW, Stephen. Information Dashboard Design: Displaying Data for At-a-Glance Monitoring. 2. ed. Burlingame: Analytics Press, 2013.

FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. 2000. Tese (Doutorado em Ciência da Computação) — University of California, Irvine, 2000.

FOWLER, Martin. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002.

GRAPHQL FOUNDATION. GraphQL Documentation. [S.l.], 2025. Disponível em: https://graphql.org/. Acesso em: 10 jun. 2026.

GRPC AUTHORS. gRPC Documentation. [S.l.], 2025. Disponível em: https://grpc.io/docs/. Acesso em: 10 jun. 2026.

H2 DATABASE ENGINE. H2 Database Engine. [S.l.], 2024. Disponível em: https://www.h2database.com/html/main.html. Acesso em: 10 jun. 2026.

HL7 INTERNATIONAL. FHIR Release 5. Ann Arbor, 2023. Disponível em: https://hl7.org/fhir/. Acesso em: 29 jul. 2026.

JAKARTA EE. Jakarta Bean Validation Specification. Version 3.0. [S.l.], 2023. Disponível em: https://jakarta.ee/specifications/bean-validation/3.0/. Acesso em: 10 jun. 2026.

JAKARTA EE. Jakarta EE Platform. [S.l.], 2025. Disponível em: https://jakarta.ee/. Acesso em: 10 jun. 2026.

JONES, Michael; BRADLEY, John; SAKIMURA, Nat. JSON Web Token (JWT). RFC 7519. Fremont: Internet Engineering Task Force, 2015. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 10 jun. 2026.

KLEPPMANN, Martin. Designing Data-Intensive Applications: the big ideas behind reliable, scalable, and maintainable systems. Sebastopol: O'Reilly Media, 2017.

LARMAN, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3. ed. Upper Saddle River: Prentice Hall, 2004.

MERKEL, Dirk. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal, n. 239, 2014.

MICRONAUT FOUNDATION. Micronaut Documentation. [S.l.], 2025. Disponível em: https://docs.micronaut.io/. Acesso em: 10 jun. 2026.

MINIO, INC. MinIO Documentation. [S.l.], 2025. Disponível em: https://min.io/docs/. Acesso em: 10 jun. 2026.

MOREIRA, Vinícius Rodrigues dos Santos. Projeto Pronto Afeto API. 2024. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) — Instituto Federal da Bahia, Salvador, 2024. Disponível em: https://www.ads.ifba.edu.br/item1249. Acesso em: 14 jul. 2026.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). Security and Privacy Controls for Information Systems and Organizations. NIST Special Publication 800-53 Revision 5. Gaithersburg: NIST, 2020. Disponível em: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final. Acesso em: 10 jun. 2026.

ORACLE. Java Platform, Standard Edition 17 Documentation. [S.l.], 2025. Disponível em: https://docs.oracle.com/en/java/javase/17/. Acesso em: 10 jun. 2026.

ORACLE. MySQL 8.4 Reference Manual. [S.l.], 2025. Disponível em: https://dev.mysql.com/doc/refman/8.4/en/. Acesso em: 10 jun. 2026.

ORGANIZAÇÃO MUNDIAL DA SAÚDE (OMS). Doenças não transmissíveis (Noncommunicable Diseases). Genebra: World Health Organization, 2023. Disponível em: https://www.who.int/news-room/fact-sheets/detail/noncommunicable-diseases. Acesso em: 29 out. 2025.

PAUL, Margaret M. et al. The State of Remote Patient Monitoring for Chronic Disease Management in the United States. Journal of Medical Internet Research, v. 27, e70422, 2025. DOI: 10.2196/70422.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL Documentation. Version 17. [S.l.], 2025. Disponível em: https://www.postgresql.org/docs/current/. Acesso em: 10 jun. 2026.

PROVOS, Niels; MAZIÈRES, David. A Future-Adaptable Password Scheme. In: USENIX ANNUAL TECHNICAL CONFERENCE. Proceedings [...]. Monterey, California: USENIX Association, 1999.

QUEIRÓS, Alexandra; SILVA, Alexandre G.; ALVARENGA, Manuel; ROCHA, Nelson P. Remote Care Technology: A Systematic Review of Reviews and Meta-Analyses. Technologies, v. 6, n. 1, 2018. DOI: 10.3390/technologies6010022.

REACT NATIVE. React Native Documentation. [S.l.], 2025. Disponível em: https://reactnative.dev/. Acesso em: 10 jun. 2026.

RIES, Eric. A Startup Enxuta: Como os Empreendedores Atuais Utilizam a Inovação Contínua para Criar Negócios Extremamente Bem-Sucedidos. São Paulo: Leya, 2012.

SILVA, Alciene Pereira da. Prototipagem e validação de aplicativo móvel para avaliação e conduta da regulação emocional baseado no modelo de Calista Roy: estudo multimétodos. 2025. 187 f. Tese (Doutorado em Enfermagem e Saúde) — Universidade Federal da Bahia, Escola de Enfermagem, Salvador, 2025.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.

SPILCA, Laurentiu. Spring Security in Action. Shelter Island: Manning Publications, 2021.

SPRING. Spring Boot Reference Documentation. [S.l.], 2024. Disponível em: https://docs.spring.io/spring-boot/docs/current/reference/html/. Acesso em: 10 jun. 2026.

SPRING. Spring Data JPA Reference Documentation. [S.l.], 2025. Disponível em: https://docs.spring.io/spring-data/jpa/reference/. Acesso em: 10 jun. 2026.

SRIRAM, Venkat; RICHARDSON, Alison; MCCANN, Julie. Carers Using Assistive Technology in Dementia Care at Home: An Integrative Literature Review. BMC Geriatrics, v. 22, n. 1, 2022. DOI: 10.1186/s12877-022-02968-6.

STATCOUNTER GLOBAL STATS. Mobile Operating System Market Share Brazil. [S.l.], 2026. Disponível em: https://gs.statcounter.com/os-market-share/mobile/brazil. Acesso em: 28 jul. 2026.

TAN, S. Y. et al. A Systematic Review of the Impacts of Remote Patient Monitoring (RPM) Interventions on Safety, Adherence, Clinical and Quality-of-Life Outcomes. npj Digital Medicine, 2024.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE). IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998. New York: IEEE, 1998.

WORLD HEALTH ORGANIZATION (WHO). Ethics and Governance of Artificial Intelligence for Health. Geneva: World Health Organization, 2021. Disponível em: https://www.who.int/publications/i/item/9789240029200. Acesso em: 29 jul. 2026.