CENTRO UNIVERSITÁRIO DE MACEIÓ

CURSO DE CIÊNCIA DA COMPUTAÇÃO

JOÃO BATISTA NETO

LEONARDO LIMA DE VASCONCELOS

LUCAS DANIEL ALVES CARNEIRO

Um Gerenciador de Medicamentos Inteligente utilizando a tecnologia a serviço da saúde

Maceió-AL
2025

JOÃO BATISTA NETO

LEONARDO LIMA DE VASCONCELOS

LUCAS DANIEL ALVES CARNEIRO

Um Gerenciador de Medicamentos Inteligente utilizando a tecnologia a serviço da saúde

Trabalho de Conclusão de Curso apresentado ao Centro Universitário de Maceió como um dos pré-requisitos para a obtenção de grau de Bacharel em Ciência da Computação.

Orientador: Esp. Victor Brunno Dantas de Souza Rosas

Coorientador: Esp. Icaro Santos Ferreira

Maceió-AL
2025

JOÃO BATISTA NETO

LEONARDO LIMA DE VASCONCELOS

LUCAS DANIEL ALVES CARNEIRO

Um Gerenciador de Medicamentos Inteligente utilizando a tecnologia a serviço da saúde

Monografia apresentada ao Centro Universitário de Maceió como um dos pré-requisitos para a obtenção de grau de bacharel em Ciência da Computação.

Aprovada em _____/_____/_____.

Banca Examinadora

Documento assinado digitalmente
gov.br VICTOR BRUNNO DANTAS DE SOUZA ROSAS
Data: 02/07/2025 12:27:57-0300
Verifique em https://validar.iti.gov.br

Esp. Victor Brunno Dantas de Souza Rosas

Centro Universitário de Maceió

Documento assinado digitalmente
gov.br ICARO SANTOS FERREIRA
Data: 03/07/2025 09:54:19-0300
Verifique em https://validar.iti.gov.br

Esp. Icaro santos ferreira

Centro Universitário de Maceió

Me. Izaac Duarte de Alencar

Centro Universitário de Maceió

Me. Marcos Vinicius Silva Bento

Centro Universitário de Maceió

Catalogação na fonte: Biblioteca do Centro Universitário de Maceió, Unima | Afya

V331g

Vasconcelos, Leonardo Lima de

Um gerenciador de medicamentos inteligente utilizando a tecnologia a serviço da saúde / Leonardo Lima de Vasconcelos, João Batista Neto, Lucas Daniel Alves Carneiro ; orientação [de] Victor Brunno Dantas de Souza Rosas ; coorientação [de] Icaro Santos Ferreira. – Maceió, 2025.

66 f. : il.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) -Centro Universitário de Maceió – Unima | Afya, Maceió, 2025.

Inclui Bibliografias: p. 61-65.

1. Gerenciamento de medicamentos. 2. Sistema web. 3. Saúde. I. Batista Neto, João. II. Carneiro, Lucas Daniel Alves. III. Rosas, Victor Brunno Dantas de Souza. (orient.). IV. Ferreira, Icaro Santos. (coorient.). V. Centro Universitário de Maceió. VI. Título.

CDU: 004

Bibliotecária responsável: Adriele da Silva Lima CRB-4/1898

DEDICATÓRIA

Dedicamos este Trabalho de Conclusão de Curso aos nossos pais, por todo o amor, apoio e força em cada etapa dessa jornada. Esta conquista é também de vocês, que estiveram sempre ao nosso lado, mesmo nos momentos mais difíceis.

Agradecemos igualmente aos nossos professores, que, com paciência e dedicação, contribuíram de forma essencial para a nossa formação. Cada ensinamento será levado conosco para além da sala de aula.

AGRADECIMENTOS

João

Agradeço, principalmente, aos meus pais, por todo o suporte oferecido durante todo o tempo de faculdade. Seu amor, dedicação e incentivo foram essenciais para que eu pudesse conquistar mais esta etapa da minha vida acadêmica. Agradeço também a todos que, de alguma forma, fizeram parte dessa trajetória. Meu sincero muito obrigado a todos.

Leonardo

Devo toda a minha carreira acadêmica aos meus pais, tenho toda gratidão por eles e dedico este e todos os trabalhos futuros a eles e aos amigos que me apoiaram nesta trajetória e me auxiliaram de alguma maneira, todos os elogios e críticas fizeram ser o que sou hoje e tenho orgulho disso, aos meus pais, família e amigos, dedico esse trabalho e essa formação a todos vocês.

Lucas

Agradeço, em especial, aos meus pais, pelo amor incondicional, apoio constante e por sempre acreditarem em meu potencial. Foram a base que sustentou minha caminhada acadêmica, oferecendo não apenas suporte material, mas também emocional, mesmo nos momentos mais desafiadores. Esta conquista é fruto da dedicação de vocês, que nunca mediram esforços para que eu pudesse alcançar meus objetivos. Meu mais profundo reconhecimento e gratidão.

RESUMO

Desenvolvimento de um sistema web para o gerenciamento e monitoramento seguro da administração de medicamentos em instituições de saúde, como casas de apoio e clínicas de longa permanência. A proposta surge diante dos desafios enfrentados por profissionais da área, como o esquecimento de doses, registros imprecisos, baixa rastreabilidade e o uso recorrente de métodos manuais, como planilhas físicas. A aplicação desenvolvida integra tecnologias modernas e acessíveis React.js no frontend e Supabase com PostgreSQL no backend com foco em automação, usabilidade e escalabilidade. Entre as funcionalidades implementadas, destacam-se o cadastro de pacientes e medicamentos, controle de horários de medicação, marcação de doses como administradas, alertas visuais e controle de estoque. A plataforma também conta com registros de auditoria e suporte a múltiplos usuários autenticados. Os testes demonstraram que o sistema apresenta desempenho eficiente e atende às necessidades de cuidadores e profissionais da saúde com diferentes níveis de familiaridade tecnológica. Conclui-se que a solução proposta contribui para a redução de erros humanos, otimiza processos internos e oferece potencial para uso em ambientes reais, com possibilidade de futuras expansões, como uma versão mobile e melhorias na interface.

Palavras-chave: Gerenciamento de medicamentos; Sistema web; Saúde; Supabase.

ABSTRACT

Development of a web system for the secure management and monitoring of medication administration in healthcare institutions, such as nursing homes and long-term care clinics. The proposal arose in response to the challenges faced by professionals in the field, such as forgotten doses, inaccurate records, poor traceability, and the recurrent use of manual methods, such as physical spreadsheets. An application developed integrates modern and easy-to-use React.js technologies on the frontend and Supabase with PostgreSQL on the backend, with a focus on automation, usability, and scalability. Among the innovative features, the following stand out: patient and medication registration, medication schedule control, dose marking as administered, visual alerts, and inventory control. The platform also has audit logs and support for multiple authenticated users. Tests demonstrated that the system performs efficiently and meets the needs of caregivers and healthcare professionals with different levels of technological familiarity. It was concluded that the proposed solution contributes to the reduction of human errors, optimizes internal processes, and offers potential for use in real environments, with the possibility of future expansions, such as a mobile version and interface improvements.

Keywords: *Medication management; Web system; Health; Supabase.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

1 INTRODUÇÃO

O gerenciamento adequado de medicamentos é um fator crítico para assegurar a eficácia dos tratamentos e a segurança dos pacientes em diversos contextos de cuidado à saúde. Com a crescente complexidade dos regimes terapêuticos, o uso simultâneo de múltiplos fármacos(polifarmácia) e a sobrecarga enfrentada por profissionais da saúde, surgem desafios significativos, como o esquecimento de doses, registros imprecisos, controle ineficiente de estoque e a dependência de métodos manuais para o monitoramento de administrações. Estudos recentes destacam que essas falhas podem acarretar complicações clínicas evitáveis, internações desnecessárias e maior vulnerabilidade para pacientes em situação de fragilidade (GUTTIER et al., 2023).

Ambientes como hospitais, clínicas, casas de apoio e instituições de longa permanência para idosos ainda recorrem com frequência ao uso de planilhas impressas ou anotações manuais, o que acentua os riscos associados à falta de digitalização. A ausência de sistemas informatizados e alertas automatizados compromete não apenas a rastreabilidade das doses aplicadas, mas também a organização do estoque e a agilidade no cuidado prestado.

Com base nesse cenário, este trabalho tem como objetivo propor o desenvolvimento de um sistema web voltado ao monitoramento e gerenciamento seguro de medicamentos. A proposta busca contribuir para a organização do fluxo de trabalho dos profissionais da saúde, promovendo automação de tarefas como o controle de horários, a marcação de doses administradas e a gestão de estoque, reduzindo a incidência de erros humanos.

A aplicação foi desenvolvida com tecnologias modernas e acessíveis, utilizando React.js no frontend e Supabase com banco de dados PostgreSQL no backend, garantindo robustez, escalabilidade e integração entre os módulos. A interface foi projetada com foco em simplicidade e usabilidade, visando atender tanto técnicos especializados quanto profissionais com menor familiaridade digital.

Este documento apresenta os fundamentos teóricos que embasam o projeto, descreve a metodologia utilizada durante o desenvolvimento, analisa os resultados obtidos a partir da aplicação prática da solução e propõe melhorias futuras com vistas à ampliação de seu escopo, incluindo a possibilidade de adaptação para dispositivos móveis.

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver um sistema web para o monitoramento e gerenciamento seguro de medicamentos, voltado para casas de apoio, cuidadores de idosos e instituições de saúde, incluindo a funcionalidade de marcar medicamentos como medicado.

2.2 Objetivos Específicos

- Desenvolver um sistema de cadastro e alerta de medicamentos utilizando banco de dados relacional (PostgreSQL) com suporte do Prisma ORM;
- Permitir o registro das doses administradas para reduzir falhas na aplicação de medicamentos;
- Estabelecer um mecanismo de controle de estoque para evitar a ausência de itens essenciais.

3 ASPECTOS TEÓRICOS

3.1 Necessidade de Soluções Tecnológicas para a Gestão de Medicamentos

A gestão inadequada de medicamentos, especialmente no ambiente doméstico, representa um risco significativo à saúde pública. O uso de medicamentos vencidos compromete sua eficácia terapêutica, pode causar efeitos adversos indesejados e aumentar o risco de intoxicações. Em razão disso, cresce a necessidade de soluções tecnológicas que auxiliem os usuários no controle seguro e eficiente de seus medicamentos.

Nesse contexto, o estudo de Ontoria (2023) propõe uma aplicação móvel que facilita a identificação dos medicamentos e suas datas de validade, alertando os usuários sobre o vencimento por meio de tecnologias de reconhecimento de imagem (OCR) e notificações automáticas. Essa abordagem tecnológica contribui para a precisão e agilidade na gestão de medicamentos, reduzindo erros humanos como a introdução manual de dados. O uso de plataformas em nuvem, como o Firebase, e a integração com bases de dados oficiais tornam a solução mais confiável, acessível e colaborativa, especialmente para públicos com dificuldades digitais.

A relevância desse tipo de iniciativa reside no seu potencial de ampliar o acesso à informação, promover maior autonomia no cuidado com a saúde e minimizar falhas comuns na administração de medicamentos. Além disso, tais soluções estão alinhadas com o avanço da transformação digital na área da saúde, que busca integrar tecnologias acessíveis ao cotidiano dos usuários, promovendo práticas mais seguras e responsáveis no uso de medicamentos em contextos não hospitalares.

3.2 Fatores Etários e Psicossociais na Não Adesão Medicamentosa

A compreensão dos fatores que levam à não adesão medicamentosa em diferentes faixas etárias é essencial para o desenvolvimento de estratégias eficazes de cuidado farmacoterapêutico. De acordo com Ge, Heng e Yap (2023), adultos mais velhos apresentam maior incidência de não adesão relacionada à memória, múltiplas comorbidades, efeitos colaterais e baixa motivação, ao passo que os mais jovens são mais afetados por falta de percepção da necessidade do tratamento e estilos de vida irregulares.

O estudo, de natureza transversal, comparou adultos jovens e idosos residentes na comunidade e evidenciou que intervenções devem ser segmentadas conforme o perfil etário e

psicossocial. Entre os idosos, o suporte familiar, lembretes tecnológicos e revisão do esquema terapêutico são estratégias que podem melhorar significativamente a adesão. O estudo também reforça que quanto maior a complexidade do regime e menor a literacia em saúde, maiores são os riscos de não adesão.

3.3 Erros na Administração de Medicamentos

A segurança na administração de medicamentos é um componente crítico na assistência hospitalar. Estudos internacionais revelam que falhas nesse processo comprometem diretamente a qualidade do cuidado e a segurança do paciente. Mohammed et al. (2022), em um estudo transversal hospitalar realizado com 423 enfermeiros de hospitais federais de Addis Ababa, Etiópia, identificaram que 59,9% dos profissionais cometeram pelo menos um erro de administração medicamentosa nos 12 meses anteriores à pesquisa. Os tipos mais comuns foram: administração em horário incorreto (56,8%), documentação inadequada (33,3%) e aconselhamento equivocado (27,8%).

Os principais fatores associados aos erros incluíram baixa experiência profissional, ausência de treinamentos regulares, indisponibilidade de diretrizes de administração de medicamentos e interrupções durante o processo de medicação. Tais condições refletem a necessidade de ambientes hospitalares mais estruturados, com protocolos bem definidos, treinamentos periódicos e apoio tecnológico, para mitigar falhas humanas e garantir a eficácia terapêutica.

Essa realidade reforça a importância de implementar sistemas que automatizam alertas, organizem rotinas e apoiem a tomada de decisão dos profissionais de saúde, contribuindo para a redução de danos evitáveis e para a melhoria dos desfechos clínicos.

3.4 Erros de Medicação e Farmacovigilância

A segurança do paciente é um dos pilares fundamentais na prestação de serviços de saúde de qualidade, especialmente no que se refere ao uso racional e seguro de medicamentos. Os erros de medicação, definidos como falhas que podem ocorrer em qualquer etapa do processo da prescrição à administração e representam um risco relevante para a saúde pública, com impacto direto sobre a morbimortalidade dos pacientes e os custos assistenciais. Segundo a Anvisa (2019), esses eventos são evitáveis e, por isso, demandam atenção estratégica no

contexto da farmacovigilância, que envolve a detecção, avaliação e prevenção de efeitos adversos e demais problemas relacionados ao uso de medicamentos.

Nesse contexto, a vigilância de erros de medicação busca não apenas quantificar e qualificar os eventos, mas também identificar causas e propor medidas de prevenção. A integração de protocolos padronizados, a capacitação contínua das equipes e o uso de tecnologias, como sistemas eletrônicos de prescrição e dispensação, são medidas essenciais para minimizar riscos. Além disso, adotar uma cultura institucional de notificação voluntária e não punitiva permite transformar erros em oportunidades de aprendizado, fortalecendo a melhoria contínua da assistência. O estudo de Almeida et al. (2022), realizado em um hospital universitário da Bahia, ilustra bem esse cenário: das 1.599 notificações analisadas, 56,2% foram quase-falhas e 43,8% erros sem dano, sendo a prescrição (31,3%) e a administração (30,8%) as etapas mais críticas do processo. Os tipos mais frequentes foram omissão de dose e administração de dose incorreta, com destaque para a sobrecarga de trabalho e interrupções como causas predominantes.

Assim, compreende-se que a farmacovigilância, associada à responsabilidade compartilhada entre profissionais, gestores e órgãos reguladores, é fundamental para garantir a segurança no uso de medicamentos. O desenvolvimento de estratégias de prevenção baseadas em evidências, aliado à valorização da cultura de segurança, consolida-se como ferramenta indispensável para qualificar os serviços de saúde.

3.5 Dificuldades no Uso de Medicamentos por Idosos

O uso correto de medicamentos por idosos representa um desafio amplamente documentado na literatura, especialmente diante do envelhecimento populacional e da alta prevalência de doenças crônicas nessa faixa etária. Um estudo de corte realizado com 1.161 idosos acompanhados pela Estratégia Saúde da Família no Sul do Brasil identificou que aproximadamente 15,5% dos participantes necessitam de auxílio para administrar corretamente seus medicamentos. Esses dados evidenciam um perfil de maior vulnerabilidade, que afeta diretamente a autonomia e a adesão ao tratamento. Entre os principais obstáculos relatados, destacam-se o esquecimento de doses, mencionado por 25,1% dos idosos, além de dificuldades físicas, como limitações para manusear embalagens e ler bulas ou rótulos, fatores que comprometem o uso seguro e eficaz dos medicamentos. Estima-se ainda que entre 5% e 6% das hospitalizações estejam relacionadas ao uso

inadequado de medicamentos, com taxas mais elevadas entre os idosos. Esses achados reforçam a necessidade de intervenções que considerem não apenas as dificuldades físicas e cognitivas dessa população, mas também os aspectos socioeconômicos que influenciam a adesão ao tratamento (GUTTIER et al., 2023).

3.6 Fatores Socioeconômicos e Saúde

Os determinantes sociais da saúde desempenham um papel crucial na capacidade dos idosos de gerenciar seus medicamentos. Indivíduos com menor escolaridade e em condições econômicas desfavoráveis demonstram maior necessidade de auxílio para a administração correta dos fármacos (GUTTIER et al., 2023). Além disso, a autoavaliação negativa da saúde também surge como um fator importante, visto que idosos que consideram sua saúde como ruim ou muito ruim tendem a apresentar maiores limitações, tanto físicas quanto cognitivas. Esse contexto exige a implementação de intervenções que contemplem não apenas as condições objetivas de saúde, mas também as percepções subjetivas dos idosos sobre seu bem-estar, além das condições socioeconômicas que os afetam.

3.7 Adesão Medicamentosa e o Monitoramento no Cuidado de Idosos

O uso adequado de medicamentos por idosos representa um dos principais desafios no contexto da assistência à saúde, especialmente em ambientes como casas de apoio e asilos, onde o monitoramento contínuo é essencial. A adesão ao tratamento medicamentoso entre idosos polimedicados é frequentemente comprometida por fatores como o esquecimento de doses, a complexidade das prescrições e dificuldades cognitivas e físicas, conforme demonstrado por Gomes et al. (2019), que observaram que grande parte dos idosos acompanhados apresentava dificuldades em seguir corretamente os esquemas terapêuticos, seja pela quantidade elevada de medicamentos(polifarmácia), seja por limitações no entendimento das orientações médicas. Esses dados reforçam a necessidade de estratégias de suporte, como sistemas digitais de monitoramento, capazes de organizar horários e doses de medicamentos de forma clara e acessível para cuidadores e pacientes.

Complementando essa perspectiva, pesquisa realizada por Lopes et al. (2013) destacou a influência dos fatores sociodemográficos e da ausência de suporte sistemático no manejo dos medicamentos por idosos da comunidade. Os autores propõem ações específicas de enfermagem para promover o uso racional de medicamentos, evidenciando a importância de

intervenções que facilitem a organização do tratamento no cotidiano dos idosos. No contexto de casas de apoio e lares para idosos, a implementação de sistemas de monitoramento eletrônico, como o proposto neste trabalho, surge como uma ferramenta promissora para aumentar a adesão medicamentosa, reduzir erros e contribuir para a melhoria da qualidade de vida dos residentes.

Assim, a criação de um site web destinado ao acompanhamento da administração de medicamentos em instituições de longa permanência é justificada não apenas pelas dificuldades práticas enfrentadas pelos idosos, mas também pelo respaldo científico que aponta para a necessidade de soluções inovadoras no apoio ao tratamento farmacológico.

3.8 Banco de Dados PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados relacional objeto-relacional de código aberto, desenvolvido para oferecer alta confiabilidade, robustez e aderência aos padrões da linguagem SQL. Sua arquitetura é baseada no modelo cliente-servidor e incorpora mecanismos avançados de controle de concorrência multiversão (MVCC), permitindo a execução de transações simultâneas sem conflitos de leitura, o que assegura maior desempenho em ambientes multiusuário (STONEBRAKER; Rowe, 1986).

Uma das características fundamentais do PostgreSQL é o suporte completo às propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), que garantem a integridade das transações e a consistência dos dados, mesmo em situações de falha do sistema). Além disso, o PostgreSQL distingue-se pela sua extensibilidade, possibilitando aos usuários criar novos tipos de dados, operadores, índices e linguagens de procedimento (MOMJIAN, 2001).

O sistema também suporta diversos padrões internacionais, como o SQL:2011, e apresenta recursos nativos para replicação de dados, particionamento de tabelas, execução de consultas paralelas e armazenamento de dados não estruturados por meio do suporte ao tipo JSON (GUPTA et al., 2014). Tais características tornam o PostgreSQL uma opção estratégica para aplicações que exigem alta disponibilidade, escalabilidade horizontal e integração com sistemas analíticos complexos.

3.9 Supabase

O Supabase é uma plataforma de desenvolvimento backend-as-a-service (BaaS) de código aberto que surgiu como uma alternativa ao Firebase, com o diferencial de utilizar o PostgreSQL como banco de dados relacional. Desenvolvido com o objetivo de simplificar o processo de desenvolvimento para aplicações web, o Supabase oferece uma série de funcionalidades integradas, como autenticação, armazenamento de arquivos, funcionalidades em tempo real e APIs RESTful e GraphQL geradas automaticamente, tudo em um único ambiente intuitivo e acessível (PIEROTTO, 2025).

Entre os principais diferenciais do Supabase, destacam-se a facilidade de configuração e a transparência proporcionada por sua arquitetura open-source. Enquanto plataformas como o Firebase exigem maior integração com soluções proprietárias, o Supabase permite que os desenvolvedores tenham controle total sobre os dados e a infraestrutura. Além disso, o suporte nativo ao PostgreSQL, com funcionalidades como Row Level Security (RLS), garante um alto nível de segurança e flexibilidade no gerenciamento de permissões, o que é essencial para aplicações que precisam de escalabilidade e controle rigoroso sobre o acesso aos dados (SUPABASE, 2022).

No contexto deste trabalho, o Supabase foi escolhido como solução de backend para armazenamento e gerenciamento de dados relacionados à administração de medicamentos, usuários e registros de monitoramento. Sua integração com bibliotecas JavaScript e o suporte nativo ao React possibilitaram uma comunicação fluida entre a interface desenvolvida e o banco de dados, otimizando o processo de desenvolvimento e garantindo a agilidade necessária para a implementação de uma aplicação moderna e interativa.

A plataforma também oferece um painel de controle completo, que permite a visualização dos dados, a execução de queries SQL diretamente pelo navegador e a criação de triggers para eventos personalizados. Esses recursos foram fundamentais para garantir a integridade dos dados e facilitar a manutenção da aplicação ao longo do tempo. A possibilidade de realizar ajustes finos nos dados, sem a necessidade de complicadas configurações de infraestrutura, é um dos motivos pelos quais o Supabase se consolidou como uma das soluções de backend mais adotadas por desenvolvedores de projetos open-source (WILSON, 2023).

Dessa forma, a adoção do Supabase no presente projeto não se justifica apenas por questões técnicas, mas também pelo alinhamento com boas práticas de desenvolvimento e sua

capacidade de atender às demandas específicas do sistema proposto, garantindo a flexibilidade, segurança e escalabilidade necessárias para o sucesso da aplicação.

3.9.1 Jwt token

O JWT (JSON Web Token) é um padrão aberto especificado pela RFC 7519, um documento técnico da IETF (Internet Engineering Task Force) que define normas para a criação e uso desse tipo de token, utilizado para definir um formato compacto e seguro para a transmissão de informações entre duas partes como um objeto JSON. O JWT tem como característica principal a assinatura digital das informações, o que assegura a integridade e autenticidade dos dados transmitidos. Isso significa que, uma vez que os dados são assinados, é possível verificar se o conteúdo não foi alterado durante a transmissão, garantindo a confiança entre as partes envolvidas na comunicação (JONES et al., 2015).

A estrutura do JWT é composta por três partes principais: o header, que especifica o algoritmo de assinatura utilizado; o payload, que carrega os dados propriamente ditos (como o ID do usuário e a data de expiração do token); e a assinatura, gerada com uma chave secreta que valida a integridade do token. Essa abordagem oferece uma autenticação sem estado (stateless authentication), o que significa que o servidor não precisa armazenar sessões do usuário, uma vez que todas as informações necessárias para verificar a identidade estão contidas no próprio token.

No contexto do Supabase, o JWT é utilizado como o mecanismo principal de autenticação. Quando um usuário realiza o login, o Supabase emite um token JWT, que é enviado a cada requisição subsequente feita pela aplicação. Isso permite que o sistema identifique o usuário sem a necessidade de revalidar suas credenciais constantemente, o que torna o processo mais eficiente e escalável (SUPABASE, 2022). Esse uso do JWT no Supabase se alinha com a tendência de adotar sistemas de autenticação modernos baseados em tokens, que são mais eficientes do que o uso de sessões tradicionais, especialmente em ambientes distribuídos e com escalabilidade exigente.

Além disso, o uso de JWT oferece benefícios significativos em termos de segurança e controle de acesso, já que ele pode ser configurado para expirar após um determinado período, minimizando o risco de uso indevido de tokens antigos ou comprometidos. Esse mecanismo é particularmente importante em aplicações que envolvem o gerenciamento de

dados sensíveis, como é o caso de sistemas de monitoramento de medicamentos, onde a proteção da privacidade dos usuários é crucial.

3.9.2 Row level security (RLS)

O Row Level Security (RLS) é um recurso avançado de controle de acesso a dados no PostgreSQL, que permite ao desenvolvedor definir políticas de segurança específicas de leitura e escrita com base em cada linha da tabela. Adotado pelo Supabase, o RLS oferece uma abordagem detalhada e flexível para restringir o acesso aos dados, garantindo que os usuários só possam visualizar ou modificar informações relacionadas a eles, de acordo com regras definidas no próprio banco de dados (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2025).

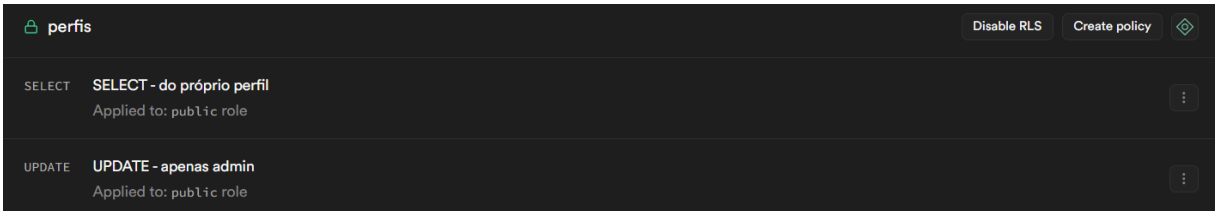
Esse controle é fundamental em ambientes onde múltiplos usuários interagem com a mesma base de dados, mas precisam de segurança granular em relação ao acesso às informações. Com o RLS, o desenvolvedor pode criar políticas que restrinjam o acesso às linhas da tabela com base em valores específicos armazenados na linha, como o user_id. Por exemplo, em uma aplicação, é possível configurar para que um usuário só tenha acesso às informações associadas ao seu próprio ID, baseado nas credenciais fornecidas pelo JWT (JSON Web Token) emitido pelo Supabase (SUPABASE, 2022).

Quando o RLS é ativado em uma tabela, nenhuma linha é acessível por padrão, o que garante que, sem a criação de políticas explícitas, o acesso aos dados seja totalmente restrito. As políticas de acesso precisam ser cuidadosamente criadas pelo desenvolvedor para definir regras claras sobre quem pode acessar ou modificar os dados. Essa abordagem stateless proporciona uma segurança adicional, pois as políticas de controle de acesso ficam encapsuladas diretamente no banco de dados, reduzindo a complexidade e a dependência de lógica adicional no frontend ou back-end da aplicação.

O uso de RLS é particularmente valioso em aplicações que lidam com dados sensíveis e requerem controle rigoroso sobre o acesso, como sistemas multiusuário e multitenancy. No contexto do sistema proposto neste trabalho, o RLS é essencial para garantir que apenas usuários autorizados, como cuidadores ou instituições, tenham acesso às informações dos pacientes sob sua responsabilidade. Por exemplo, ao aplicar RLS sobre a tabela de administração de medicamentos, cada usuário poderá acessar apenas os dados dos pacientes associados ao seu identificador, evitando a exposição indevida de informações confidenciais.

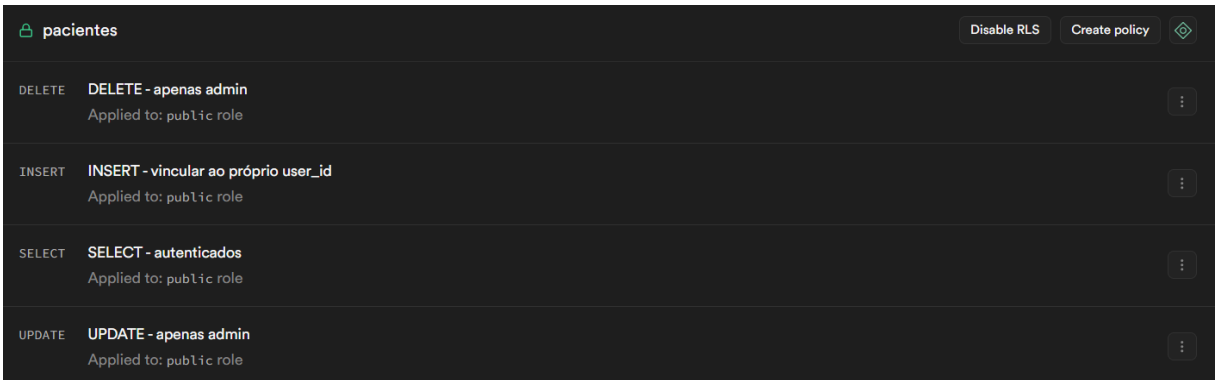
Combinado com o uso do JWT para autenticação, o RLS no Supabase oferece uma solução poderosa e segura para o controle de acesso refinado, permitindo que os dados sejam protegidos de forma eficiente e sem a necessidade de lógica extra no frontend, o que torna a aplicação mais segura e de fácil manutenção.

Figura 1 — RLS para tabela perfis



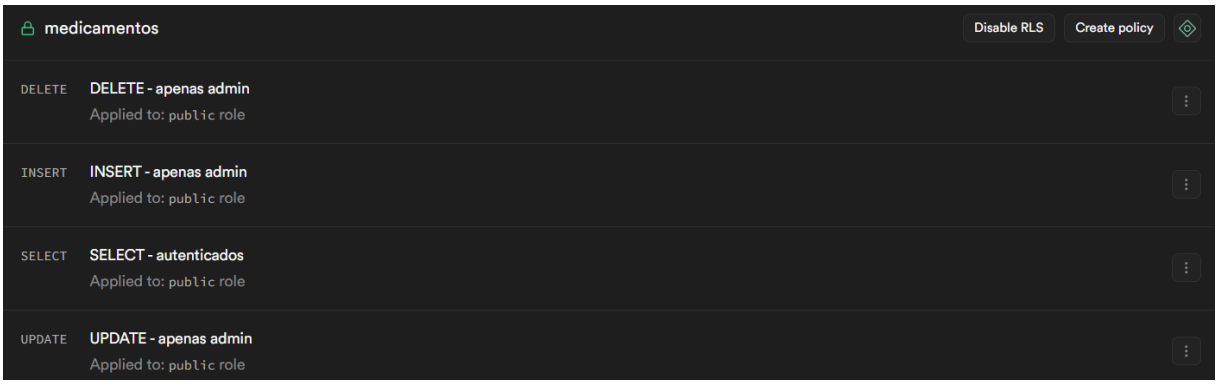
Fonte: Elaborado pelo autor, 2025.

Figura 2 — RLS para tabela pacientes



Fonte: Elaborado pelo autor, 2025.

Figura 3 — RLS para medicamentos



Fonte: Elaborado pelo autor, 2025.

3.9.3 Supabase auth e o gerenciamento de autenticação

O Supabase Auth é o serviço de autenticação nativo da plataforma Supabase, projetado para fornecer controle de acesso seguro e flexível em aplicações web. Baseado em protocolos modernos como JWT (JSON Web Tokens), ele permite implementar login por e-mail e senha, links mágicos, autenticação via provedores externos (OAuth) e OTP (One-Time Password) com validação por telefone,uma das vantagens do Supabase Auth é sua integração direta com o banco PostgreSQL, permitindo o uso de Row Level Security (RLS) para aplicar regras de acesso diretamente no nível do banco de dados. Isso significa que é possível restringir ou conceder acesso a determinados registros de acordo com o usuário autenticado, reforçando a segurança da aplicação.

Mesmo em seu plano gratuito, o Supabase fornece recursos como cadastro, login, gerenciamento de sessões, verificação de e-mail, redefinição de senha e controle de permissões, oferecendo uma estrutura completa para autenticação sem necessidade de serviços externos adicionais. Sua simplicidade de integração com aplicações em React e outras bibliotecas modernas também o torna uma escolha atrativa para aplicações escaláveis e seguras.

3.10 Prisma ORM

O Prisma ORM é uma ferramenta moderna de mapeamento objeto-relacional (ORM), desenvolvida para aplicações que utilizam as linguagens JavaScript e TypeScript. Ele adota uma abordagem baseada no padrão Data Mapper, em que o modelo de dados é centralizado em um arquivo de configuração declarativo chamado schema.prisma. Este esquema atua como a fonte única de verdade para a estrutura do banco de dados e para a geração automática de código tipado, eliminando inconsistências entre o modelo lógico e o banco de dados.

A arquitetura do Prisma é composta por três componentes principais: o Prisma Client, responsável por executar operações no banco de dados com segurança de tipos e autocompletar; o Prisma Migrate, utilizado para controlar a evolução do esquema com versionamento das migrações; e o Prisma Studio, uma interface gráfica que permite a visualização e edição dos dados de forma intuitiva (PRISMA, 2025a).

Um dos principais diferenciais do Prisma em relação a outros ORMs tradicionais é sua capacidade de garantir type safety, ou seja, segurança de tipos em tempo de desenvolvimento.

Ao gerar automaticamente um cliente com tipagem estática com base no esquema declarado, o Prisma reduz a possibilidade de erros em tempo de execução, aumentando a confiabilidade da aplicação. Outro aspecto relevante é sua portabilidade: o Prisma é compatível com diversos bancos de dados relacionais, como PostgreSQL, MySQL, SQLite e SQL Server, além de fornecer suporte ao MongoDB. Essa flexibilidade facilita sua adoção em diferentes cenários e projetos, sem a necessidade de grandes reestruturações no código (PRISMA, 2025b).

Com o lançamento da versão 6, o Prisma apresentou melhorias significativas no desempenho de consultas complexas, como JOINs e buscas aninhadas, que passaram a ser otimizadas no próprio banco de dados, e não mais apenas em nível de aplicação. Essa atualização aumentou a eficiência do ORM, especialmente em ambientes com grande volume de dados (BURK, 2024).

Em resumo, o Prisma ORM apresenta-se como uma solução robusta, segura e produtiva para o desenvolvimento de aplicações modernas que exigem acesso estruturado e seguro a bancos de dados, sendo especialmente relevante em projetos que valorizam a consistência dos dados, a agilidade no desenvolvimento e a escalabilidade da aplicação.

3.11 Axios e Express

O Express é um framework minimalista e flexível para o desenvolvimento de aplicações web e APIs no ambiente Node.js. Ele facilita a criação de servidores HTTP e o roteamento de requisições de maneira eficiente e escalável, permitindo a criação de endpoints que respondem a métodos HTTP padrão, como GET, POST, PUT e DELETE. Além disso, o Express adota um modelo baseado em middleware, o que oferece flexibilidade para implementar autenticação, validação, manipulação de erros e controle de acesso, sem comprometer a simplicidade do código. O framework é amplamente utilizado para a construção de APIs RESTful e aplicações escaláveis (HETT, 2020).

Por outro lado, o Axios é uma biblioteca JavaScript utilizada para realizar requisições HTTP de forma assíncrona, baseada no padrão Promise. Isso permite que o frontend interaja com APIs internas ou externas sem bloquear o fluxo da aplicação, o que melhora a experiência do usuário. Além da simplicidade na realização de requisições, o Axios permite o uso de interceptadores para tratamento de erros, autenticação e modificação de requisições ou respostas, o que facilita a integração com APIs RESTful e o controle centralizado das comunicações com servidores (AXIOS, 2023).

Combinados, Express e Axios oferecem uma solução robusta para o desenvolvimento de sistemas modernos, unindo a eficiência na criação de servidores e rotas com a flexibilidade na comunicação entre cliente e servidor.

3.12 Insomnia

O Insomnia é um cliente open-source e multiplataforma, projetado para o desenvolvimento, teste e documentação de APIs nos protocolos REST, GraphQL, gRPC, WebSocket e Server-Sent Events (SSE). Sua estrutura permite que desenvolvedores trabalhem com diversas interfaces de forma integrada, entre suas principais funcionalidades está o suporte a design de API com OpenAPI, oferecendo editor visual com pré-visualização, validação em tempo real das especificações e capacidade de importação/exportação de documentos OpenAPI diretamente para coleções de requisições (KONG, 2025).

O Insomnia também possui um motor de automação de testes (“collection runner”) que executa múltiplas requisições, permitindo o uso de scripts anteriores ou posteriores à resposta. Essa funcionalidade é essencial para configurar pipelines de CI/CD robustos, garantindo testes contínuos e padronizados nas APIs, adicionalmente, a ferramenta favorece o trabalho colaborativo por meio de sincronização via Git (Git Sync) e armazenamento em nuvem com criptografia ponta a ponta, assegurando segurança dos dados e rastreabilidade de alterações entre equipes, por fim, o Insomnia integra uma linha de comando (CLI Inso) que permite validação automática de specs, linting via OpenAPI, execução de testes e integração com CI/CD, favorecendo workflows automatizados e mantendo qualidade durante todo o ciclo de vida da API (INSOMNIA, 2025).

Essa combinação (suporte completo a múltiplos protocolos, design centrado no OpenAPI, testes automatizados, colaboração segura e automação via CLI) posiciona o Insomnia como uma plataforma completa e coerente para desenvolvimento de APIs, promovendo produtividade, consistência técnica e governança integrada.

3.13 Monitoramento de métricas com prometheus

Prometheus é uma plataforma open-source de monitoramento e armazenamento de métricas em séries temporais, criada originalmente pela SoundCloud em 2012. Ele coleta dados por meio de scraping ativo de endpoints HTTP expostos pelos serviços, grava localmente e permite consultas por meio da linguagem PromQL (CORALOGIX, 2024;

PROMETHEUS.IO, 2025). A arquitetura do Prometheus foca em confiabilidade, operando autonomamente e dispensando armazenamento distribuído, sendo eficiente mesmo em ambientes instáveis (SOUNDCLOUD, 2012).

A força do Prometheus reside em seu modelo dimensional de dados: cada métrica pode ter múltiplos rótulos (labels), como rota, status e instância, permitindo consultas refinadas por exemplo, taxa de requisições por segundo ou tempo médio de resposta e base para alertas automatizados (CORALOGIX, 2024). Sua linguagem PromQL possibilita filtros, agregações e transformações sofisticadas sobre séries temporais (PROMETHEUS.IO, 2025).

3.14 Visualização e monitoramento com grafana

O grafana é uma plataforma open-source líder em visualização de dados e monitoramento, utilizada para criar dashboards interativos que integram e exibem métricas, logs e traces de diferentes fontes (GRAFANA LABS, 2025). Seu design permite que equipes criem painéis personalizados com uma variedade de painéis (gráficos, heatmaps, tabelas), tornando os dados acessíveis e compreensíveis para todos, não apenas para operações técnicas (HELP NET SECURITY, 2024).

Uma das forças do Grafana é sua flexibilidade de integração: ele se conecta com mais de 100 fontes de dados, incluindo Prometheus, Loki, Elasticsearch e bancos SQL/NoSQL, sem necessidade de centralizar ou migrar dados (ZESTY, 2024; GRAFANA LABS, 2025). Além disso, oferece sistema de alertas embutido, com notificações via Slack, PagerDuty, e-mail, entre outros, fundamental para resposta proativa a eventos críticos.

No contexto do seu sistema na área da saúde, o Grafana pode ser utilizado para gerar dashboards que exibam:

- Taxa de aplicação de medicamentos por período;
- Latência e erros da API em tempo real;
- Alertas configuráveis para picos de falhas ou tempos de resposta elevados.

Essa visibilidade integrada promove transparência operacional, facilidade de auditoria e suporte à manutenção preventiva, fatores essenciais para garantir confiabilidade e segurança em instituições que trabalham com dados sensíveis.

3.15 Linguagem de programação

A linguagem de programação é uma ferramenta essencial para a construção de software, permitindo que os desenvolvedores se comuniquem com a máquina para realizar tarefas específicas. Essas linguagens podem ser divididas em linguagens de baixo nível (próximas do hardware) e linguagens de alto nível (mais próximas da linguagem humana), com as últimas sendo mais acessíveis e abstraídas das complexidades do hardware.

Linguagens de baixo nível são aquelas que exigem comandos mais próximos da linguagem da máquina, como Assembly. Por exemplo, para somar dois números em Assembly, o programador precisa especificar manualmente os registradores e a sequência de instruções. Essas linguagens permitem um controle detalhado sobre os recursos do hardware, porém são mais difíceis de escrever e compreender.

Linguagens de alto nível, como Python, Java e JavaScript, utilizam uma sintaxe mais próxima da linguagem humana, o que facilita o entendimento e o desenvolvimento de aplicações complexas. Por exemplo, em Python, uma soma pode ser realizada simplesmente com a instrução, sem a necessidade de controlar diretamente os registradores da máquina.

As linguagens de alto nível são mais fáceis de entender e escrever, permitindo ao desenvolvedor focar na lógica do problema, sem precisar de um conhecimento profundo sobre o funcionamento interno do hardware.

3.15.1 Framework

Um framework é um conjunto de bibliotecas e ferramentas que facilita o desenvolvimento de software, oferecendo uma estrutura pronta para criar aplicações. Ele inclui funcionalidades como gerenciamento de banco de dados, autenticação, roteamento de URLs e outras soluções essenciais que agilizam o processo de desenvolvimento, promovem a padronização do código e facilitam a manutenção, inclusive em projetos de grande escala (SENCHA, 2024).

A principal vantagem de utilizar um framework é a eliminação da necessidade de implementar funcionalidades básicas manualmente, o que reduz o tempo de desenvolvimento e melhora a escalabilidade das aplicações (EBA, 2023). Frameworks podem ser voltados para o frontend (como React e Angular), para o backend (como Django e Express) ou oferecer uma

abordagem full-stack, como o Next.js, que integra o desenvolvimento da interface com o do servidor.

3.15.2 Javascript

O JavaScript é uma linguagem de programação de alto nível, dinâmica e interpretada, amplamente utilizada no desenvolvimento de aplicações web. Criada originalmente por Brendan Eich em 1995, enquanto trabalhava na Netscape Communications, o JavaScript foi projetado para permitir a interação dinâmica com páginas web, adicionando funcionalidades que o HTML e o CSS não conseguiam fornecer (FLANAGAN, 2020).

Quando se diz que o JavaScript é dinâmico, significa que seu tipo de dados pode ser alterado em tempo de execução, ou seja, uma variável pode mudar de tipo conforme o programa é executado. Isso oferece flexibilidade ao programador, pois não é necessário definir rigidamente os tipos de dados, além disso, o JavaScript é uma linguagem interpretada, o que significa que seu código não precisa ser compilado antes de ser executado. O navegador lê e executa o código diretamente, linha por linha, permitindo atualizações rápidas e imediatas no comportamento da aplicação sem etapas intermediárias de compilação.

Segundo Flanagan (2020), uma das grandes forças do JavaScript é sua capacidade de ser executado diretamente no navegador, proporcionando interatividade e melhorando a experiência do usuário.

Algumas das principais vantagens do JavaScript incluem sua versatilidade, ampla adoção na indústria, suporte nativo nos navegadores e uma vasta gama de frameworks e bibliotecas que aceleram o desenvolvimento de aplicações complexas.

No quadro 1, são apresentadas as principais características do JavaScript que o tornam amplamente utilizado no desenvolvimento web.

Quadro 1 — Características do JavaScript

Execução no Lado do Cliente	Permite a criação de aplicações altamente interativas diretamente nos navegadores.
Multiplataforma	Funciona em praticamente qualquer dispositivo com um navegador moderno.
Extenso Ecossistema	Disponibiliza uma grande variedade de frameworks e bibliotecas que ampliam suas capacidades.

Fonte: Elaborado pelo autor, 2025.

3.15.3 React

O React é uma biblioteca JavaScript desenvolvida pelo Facebook para a criação de interfaces de usuário dinâmicas. Baseado em componentes reutilizáveis, ele facilita a manutenção e escalabilidade das aplicações. Um dos principais diferenciais do React é o uso do Virtual DOM, que melhora o desempenho ao atualizar apenas os elementos necessários da interface, em vez de recarregar toda a página. Além disso, sua abordagem declarativa e modular permite o desenvolvimento eficiente de interfaces complexas, com fácil integração a ferramentas como React Router, para navegação, e Redux, para gerenciamento de estado. Essas características tornam o React uma das tecnologias mais populares entre desenvolvedores de aplicações web modernas (REACT, 2025).

3.16 GitHub

O GitHub é uma plataforma de hospedagem de código-fonte baseada na web, amplamente utilizada no desenvolvimento de software colaborativo. Sua estrutura se fundamenta no sistema de controle de versão Git, criado por Linus Torvalds em 2005, permitindo o gerenciamento eficiente de alterações no código e a colaboração entre diversos desenvolvedores (Torvalds e Hamano, 2005).

O Git, sistema que serve de base para o GitHub, possui arquitetura distribuída, em que cada desenvolvedor mantém uma cópia completa do repositório. Isso possibilita o trabalho offline e sincronizações posteriores, favorecendo um desenvolvimento descentralizado e seguro. Operações como commits, branches e merges fazem parte da rotina de versionamento, oferecendo rastreabilidade e histórico detalhado (Chacon e Straub, 2014).

No contexto acadêmico, o uso do GitHub tem se mostrado eficaz na organização de projetos, especialmente em cursos de ciência de dados e estatística, por promover a

reprodutibilidade e boas práticas de versionamento (Beckman et al., 2020). A plataforma também integra ferramentas de automação, como o GitHub Actions, que executa automaticamente testes, builds e outras tarefas sempre que há alterações no repositório (Kinsman et al., 2021).

Além disso, o GitHub suporta arquivos CITATION.cff, que orientam a citação adequada de repositórios em trabalhos científicos, gerando automaticamente formatos como APA ou BibTeX, promovendo reconhecimento formal aos autores de código aberto (GitHub, 2021).

Funcionalidades colaborativas como pull requests, forks, issues e tags facilitam revisão e integração de alterações, fomentando um ambiente coletivo, transparente e controlado (Vasilescu et al., 2015). Esse modelo é essencial para o desenvolvimento do sistema de monitoramento farmacoterapêutico proposto, pois oferece controle de versões, histórico de alterações, colaboração entre integrantes e organização durante codificação e testes, o GitHub se apresenta como ferramenta estratégica na construção de soluções tecnológicas voltadas à saúde, garantindo rastreabilidade, colaboração eficiente e adesão a boas práticas de engenharia de software.

3.17 Vercel

O Vercel é uma plataforma de deploy e hospedagem voltada para aplicações web e sites estáticos, com foco em agilidade, escalabilidade e simplicidade. Criada pela equipe por trás do Next.js, a plataforma foi projetada para simplificar a implementação de aplicações frontend. Com integração direta a GitHub, GitLab e Bitbucket, a Vercel facilita o processo de deploy contínuo, permitindo que modificações feitas no código sejam automaticamente publicadas em produção sem a necessidade de intervenções manuais.

O Vercel destaca-se especialmente pela sua integração com o Next.js, um framework para React que permite o desenvolvimento de aplicações dinâmicas e sites estáticos. A plataforma otimiza o uso de renderização estática (Static Site Generation - SSG) e renderização server-side (SSR), técnicas que contribuem para a melhora no SEO (Search Engine Optimization) e no desempenho geral da aplicação, garantindo que o conteúdo seja rapidamente acessado pelo usuário (VERCEL, 2025).

Além disso, a escalabilidade da Vercel é baseada na tecnologia de edge computing, onde a aplicação é distribuída através de Content Delivery Networks (CDNs), permitindo que

o conteúdo seja servido de servidores localizados próximos ao usuário, o que reduz a latência e melhora a performance da aplicação em um nível global.

A plataforma também oferece funções serverless, que permitem a execução de código backend sem a necessidade de gerenciamento de servidores físicos ou virtuais. Esse modelo reduz custos operacionais e melhora a escalabilidade, pois o código é executado sob demanda, garantindo eficiência sem sobrecarga de infraestrutura.

Em resumo, a Vercel é uma solução completa e moderna para o deploy de aplicações web, oferecendo vantagens significativas em termos de performance, escala e agilidade no desenvolvimento. Sua integração com o Next.js e a otimização de renderização dinâmica tornam a plataforma uma escolha popular entre desenvolvedores que buscam criar experiências web rápidas e eficientes.

3.18 Docker

O Docker representa uma evolução no paradigma de implantação de software por meio da utilização de contêineres, unidades leves e isoladas do sistema operacional que encapsulam aplicações e suas dependências. Esse modelo proporciona portabilidade entre ambientes de desenvolvimento, teste e produção, minimizando o clássico problema do "funciona na minha máquina" (DOCKER, 2025).

A estrutura do Docker baseia-se em imagens, que são versões imutáveis construídas através de scripts chamados Dockerfile. Cada imagem pode ser instanciada como um contêiner, permitindo configuração padronizada, rastreabilidade e reprodutibilidade do ambiente de execução (DATACAMP, 2024). Tal abordagem facilita a automação de pipelines de CI/CD, pois garante que ambientes de build, teste e deploy utilizem exatamente as mesmas configurações.

Além disso, o Docker se integra plenamente a fluxos DevOps e plataformas de entrega contínua, já que permite que cada estágio do pipeline seja executado em contêineres idênticos. Isso aumenta a confiabilidade e diminui falhas por inconsistência de ambiente (DOCKER, 2025). Federationar serviços, bancos de dados e aplicações em múltiplos contêineres se tornou viável por meio do Docker Compose, ferramenta que simplifica a definição e o gerenciamento desses conjuntos com um único arquivo YAML, ainda que em produção, orquestradores como Kubernetes se façam necessários para garantir resiliência e escalabilidade.

Em suma, o Docker traz uma combinação de portabilidade, reprodutibilidade, automação e isolamento eficiente, que fundamenta práticas modernas de desenvolvimento e operação de software em escala.

3.19 Webhooks

Webhooks se destacam como uma solução eficaz para o modelo orientado a eventos. Ao invés de depender de técnicas ineficientes como polling em que um cliente precisa verificar periodicamente se há atualizações, os Webhooks possibilitam que um servidor envie automaticamente notificações para um endpoint previamente configurado quando determinado evento ocorre.

Segundo Biehl (2017), Webhooks funcionam como uma extensão natural das APIs REST, permitindo que aplicações se tornem reativas, isto é, capazes de responder em tempo real a eventos externos. Essa abordagem contribui para uma redução no consumo de recursos e maior escalabilidade dos sistemas, além de simplificar a lógica de integração entre serviços.

3.20 Aplicativos existentes para gerenciamento de medicamentos

O Medisafe é uma solução móvel que permite ao próprio paciente ou cuidador configurar lembretes de medicamentos, registrar o uso, receber alertas em tempo real e compartilhar informações com familiares. A interface é amigável e oferece recursos como gráficos de adesão ao tratamento. No entanto, seu foco é individual e não contempla o gerenciamento coletivo de pacientes ou controle de estoque.

O Pillboxie, disponível para dispositivos iOS, possui uma abordagem visual interativa, permitindo ao usuário organizar seus medicamentos em horários por meio de ilustrações intuitivas. Apesar de sua facilidade de uso, o aplicativo é voltado exclusivamente ao acompanhamento pessoal e não dispõe de funcionalidades como autenticação multiusuário, histórico consolidado ou integração com estoques.

4 METODOLOGIA

A aplicação destina-se a ambientes como instituições de longa permanência, casas de apoio e instituições de saúde, visando otimizar o controle de medicamentos, melhorar a segurança dos pacientes e reduzir riscos relacionados ao uso inadequado de medicamentos.

Além disso, foi realizado um levantamento bibliográfico abrangente, com a consulta a artigos científicos e materiais técnicos, com o intuito de fundamentar teoricamente o tema e embasar as necessidades identificadas. Este levantamento visou identificar soluções existentes, tecnologias aplicáveis e definir os requisitos necessários para a implementação da proposta, garantindo que a solução desenvolvida atenda às demandas específicas do público-alvo e contribua para a melhoria dos processos de gestão de medicamentos.

4.1 Métodos de Desenvolvimento

O desenvolvimento do projeto adotou uma abordagem incremental e iterativa, permitindo a construção progressiva da aplicação, com entregas parciais e melhorias contínuas a partir da análise de requisitos e dos testes realizados. Essa escolha metodológica favoreceu a flexibilidade no processo de desenvolvimento, possibilitando ajustes conforme novas necessidades foram identificadas ao longo do percurso.

Inicialmente, foi realizado o levantamento dos requisitos funcionais e não funcionais, seguido da escolha das tecnologias mais adequadas para a proposta. A partir disso, foi desenvolvido um protótipo funcional, o qual serviu como base para validar os fluxos principais do sistema e nortear a implementação das funcionalidades. A aplicação atualmente em desenvolvimento contempla um sistema completo de cadastro, edição, exclusão e visualização de dados tanto para pacientes quanto para medicamentos, garantindo o controle estruturado dessas informações.

Além disso, o sistema incorpora um controle de estoque de medicamentos, permitindo o gerenciamento das quantidades disponíveis e promovendo maior segurança no acompanhamento do tratamento dos pacientes. A tela inicial da aplicação foi desenhada para oferecer uma visão em tempo real do status da medicação de cada paciente, destacando aqueles que estão disponíveis para serem medicados, os que já foram medicados e os que apresentam atraso em suas administrações. Essa funcionalidade reflete o foco principal do

projeto, que é oferecer alertas inteligentes sobre os horários de medicação, promovendo organização, agilidade e precisão nos cuidados, especialmente em ambientes como casas de apoio, instituições de longa permanência e unidades de saúde.

Como se trata de um protótipo em desenvolvimento, o sistema vem sendo submetido a testes de validação funcional e de usabilidade, com o objetivo de refinar tanto a experiência do usuário quanto a eficiência dos processos internos. Essa abordagem tem possibilitado uma evolução consistente do projeto, mantendo alinhamento com os objetivos definidos e garantindo a viabilidade de aplicação em contextos reais de gestão medicamentosa.

4.2 Levantamento de Requisitos

O levantamento de requisitos foi essencial para identificar as necessidades do sistema proposto, voltado à gestão de medicamentos em ambientes institucionais. A partir da análise de práticas comuns em instituições de saúde e casas de apoio, foram definidos os principais requisitos funcionais, como cadastro e gerenciamento de pacientes e medicamentos, controle de estoque, alertas de horário de medicação, classificação do status dos pacientes e registro histórico das medicações administradas.

Foram também estabelecidos requisitos não funcionais relacionados à usabilidade da interface, desempenho do sistema, segurança e integridade dos dados, além da necessidade de acessibilidade para usuários com baixa familiaridade tecnológica. Esse mapeamento orientou o desenvolvimento técnico e garantiu que a aplicação atendesse às demandas reais do público-alvo. Abaixo listamos os requisitos no quadro 2:

Quadro 2 — Requisitos funcionais e não funcionais

Funcional	Cadastrar, editar, visualizar e excluir pacientes
Funcional	Cadastrar, editar, visualizar e excluir medicamentos
Funcional	Gerenciar o estoque de medicamentos, com controle de entrada e saída
Funcional	Emitir alertas sobre horários de medicação dos pacientes
Funcional	Classificar pacientes em: disponíveis para medicação, atrasados e já medicados
Funcional	Registrar histórico de medicamentos administrados a cada paciente
Funcional	Exibir tela inicial (home) com resumo do status de medicação

Funcional	Associar medicamentos aos respectivos pacientes
Não funcional	Interface amigável e de fácil usabilidade
Não funcional	Tempo de resposta rápido para ações comuns do sistema
Não funcional	Segurança no armazenamento dos dados dos pacientes
Não funcional	Capacidade de expansão para inclusão de novas funcionalidades
Não funcional	Integridade e consistência dos dados durante operações simultâneas
Não funcional	Acessibilidade para usuários com baixa familiaridade com tecnologia

Fonte: Elaborado pelo autor, 2025.

4.3 Processo de Desenvolvimento

A estruturação do projeto priorizou a organização em etapas lógicas, possibilitando o avanço contínuo e controlado das fases de codificação, testes e validações. Ao longo desse processo, foram incorporadas tecnologias modernas, que favoreceram a integração entre front-end, back-end e banco de dados, ao mesmo tempo em que garantiram segurança, escalabilidade e facilidade de manutenção.

A aplicação foi construída utilizando JavaScript com o framework React na interface do usuário, garantindo navegação fluida no ambiente web. No back-end, o Express foi utilizado para criação das rotas e gerenciamento das requisições, com Axios como intermediador das chamadas HTTP. O Prisma ORM foi adotado para facilitar a manipulação do banco de dados, promovendo eficiência na camada de persistência. Já o Supabase foi utilizado como solução completa de backend, oferecendo banco de dados relacional, autenticação via JWT (JSON Web Token) e controle de acesso com Row Level Security (RLS).

A aplicação foi conteinerizada com Docker, facilitando a padronização e a escalabilidade do ambiente. O repositório de código foi mantido no GitHub, com integração direta ao Vercel para deploy contínuo, o que permitiu que cada nova atualização fosse automaticamente publicada. Para validar as rotas da API e garantir o bom funcionamento da comunicação entre as camadas, foram utilizados testes com o Insomnia.

Essa combinação de tecnologias e práticas permitiu o desenvolvimento de uma aplicação moderna, funcional e alinhada aos requisitos levantados. O detalhamento de cada etapa do processo será apresentado nos subtópicos a seguir.

4.3.1 Modelagem e gerenciamento do banco de dados

A estrutura do banco de dados foi projetada para representar de maneira clara e funcional as relações entre pacientes, medicamentos, histórico de administração e controle de estoque. A modelagem foi desenvolvida em formato relacional, utilizando o Prisma ORM para facilitar o mapeamento entre as tabelas e o código da aplicação. O banco de dados em si foi implementado no Supabase, que oferece um ambiente compatível com PostgreSQL, integrado ao restante da infraestrutura do sistema.

No centro da estrutura encontra-se a entidade pacientes, que armazena dados como nome, data de nascimento, idade, quarto e uma foto identificadora. Cada paciente está associado a um usuário do sistema por meio da chave user_id, permitindo a vinculação entre registros clínicos e contas autenticadas.

A tabela medicamentos contém as informações essenciais dos fármacos cadastrados, como nome, descrição, dosagem (em mg) e possíveis ingredientes alergênicos. Associada a ela está a tabela estoque_medicamento, responsável por armazenar a quantidade disponível de cada medicamento, viabilizando o controle de estoque diretamente dentro da aplicação.

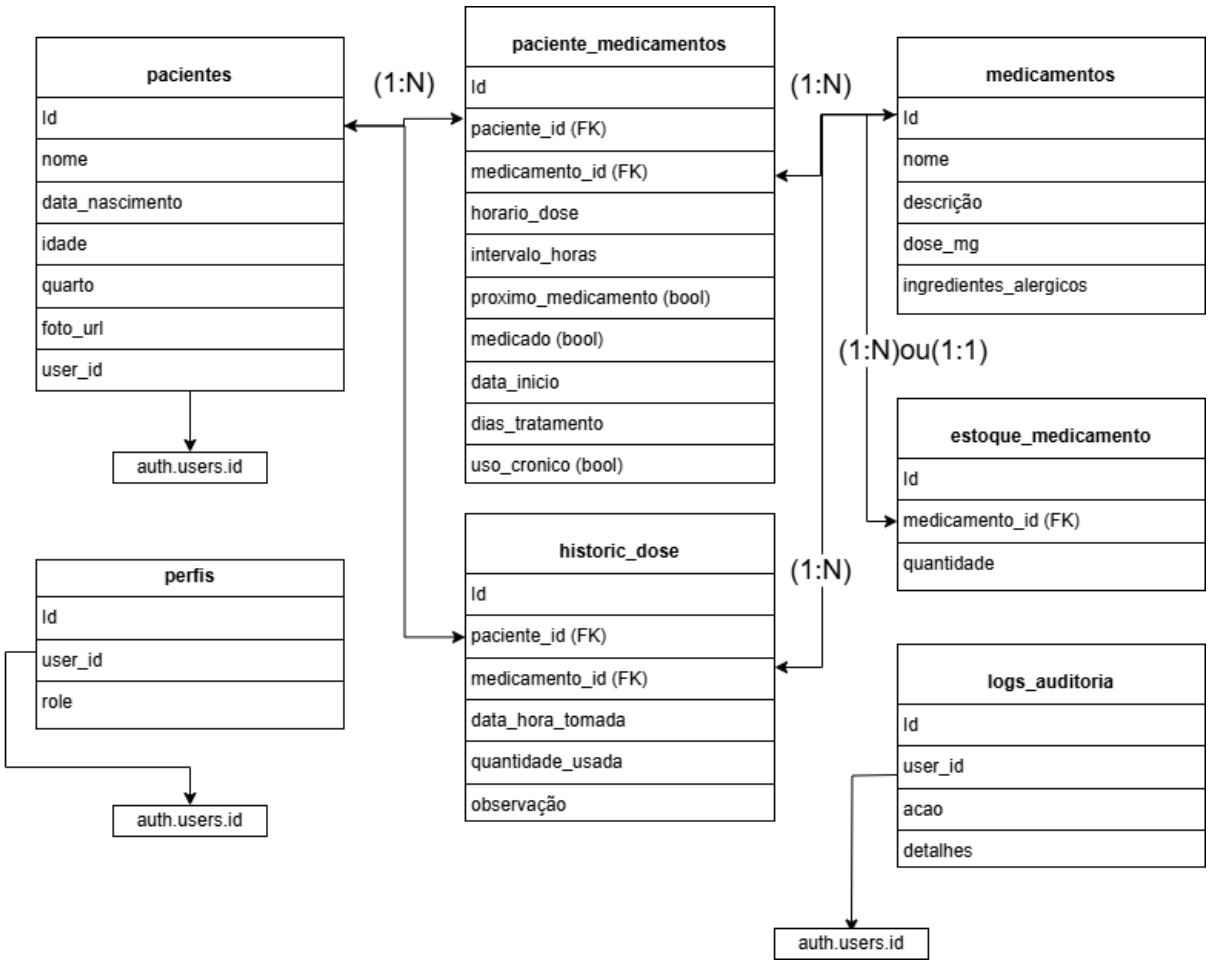
A relação entre pacientes e medicamentos é representada pela tabela paciente_medicamentos, que permite configurar o horário de administração, o intervalo entre doses, a data de início do tratamento, o número de dias e o uso contínuo. Essa tabela também registra se o paciente já foi medicado e se há dose pendente, dados importantes para o funcionamento dos alertas automatizados do sistema.

Para garantir rastreabilidade, a tabela historico_dose registra cada administração realizada, vinculando paciente, medicamento, data e hora da dose, quantidade utilizada e possíveis observações inseridas pelo responsável.

Por fim, a tabela perfis define o nível de permissão (role) atribuído a cada usuário, também vinculada ao sistema de autenticação, permitindo controlar o acesso às funcionalidades da aplicação conforme o perfil de uso.

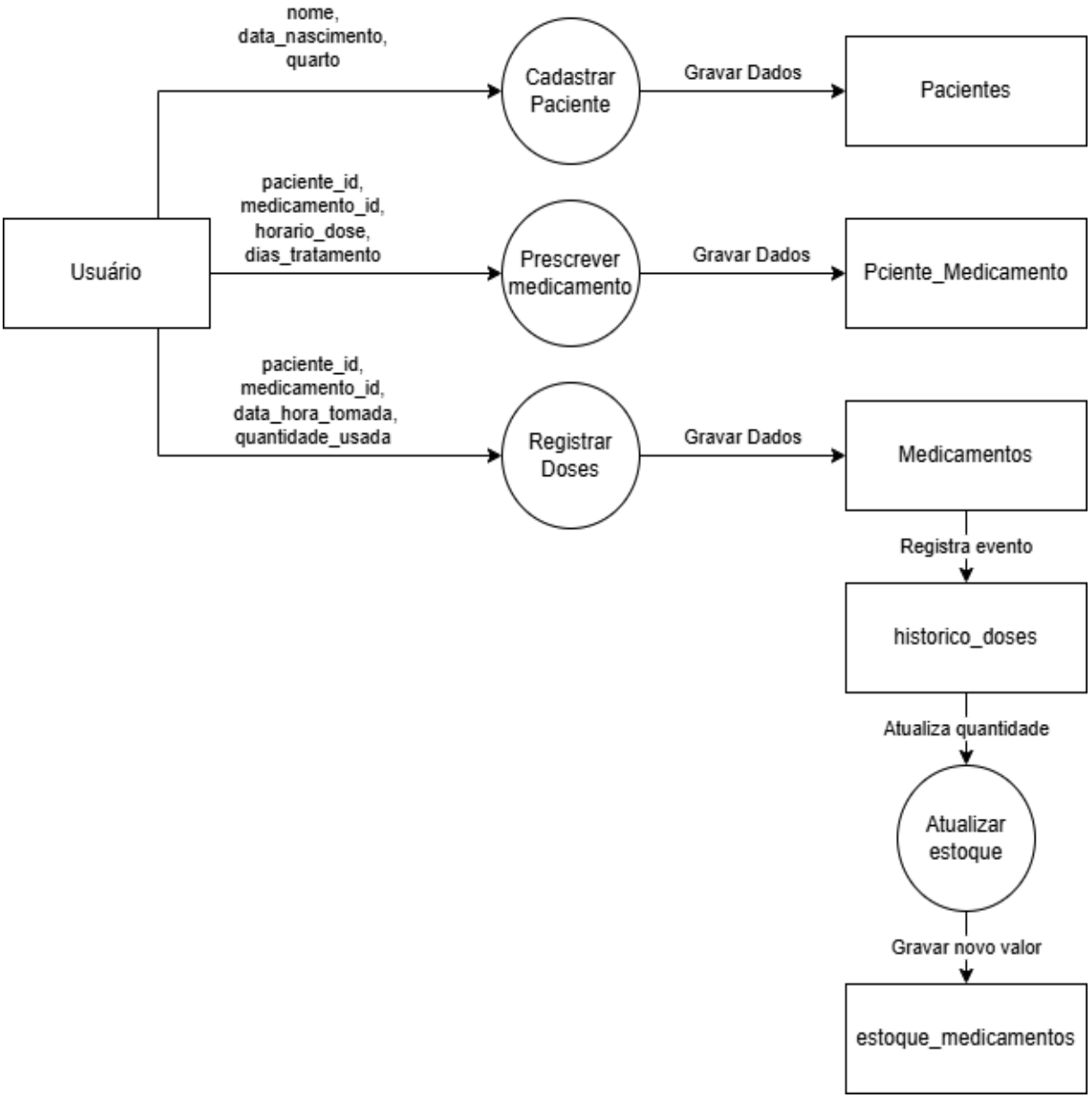
Essa estrutura foi pensada para garantir integridade dos dados, facilitar consultas e oferecer suporte à lógica de funcionamento da aplicação em tempo real.

Figura 4 — Diagrama Entidade-Relacionamento (ERD)



Fonte: Elaborado pelo autor, 2025.

Figura 5 — Fluxo de dados



Fonte: Elaborado pelo autor, 2025.

4.3.2 Segurança e controle de acesso: jwt e row level security

A segurança do sistema foi estruturada para garantir que os dados sensíveis principalmente relacionados a pacientes e medicamentos sejam acessados apenas por usuários devidamente autenticados e autorizados. Para isso, foram adotados dois recursos principais disponibilizados pela plataforma Supabase: o uso de tokens JWT (JSON Web Token) para

autenticação e a implementação de políticas de segurança a nível de linha, conhecidas como Row Level Security (RLS).

O Supabase oferece um sistema de autenticação integrado com suporte a login e registro de usuários, e a partir dessa base é gerado um token JWT a cada sessão autenticada. Esse token é transmitido em cada requisição à API, contendo as credenciais e permissões associadas ao usuário. No contexto da aplicação, o JWT é utilizado para validar o acesso ao sistema e garantir que apenas usuários autenticados possam consumir as rotas e consultar os dados vinculados às suas permissões.

Complementando esse processo, foi configurado o uso do Row Level Security (RLS), uma funcionalidade do PostgreSQL que permite restringir o acesso a linhas específicas das tabelas com base em regras definidas. No sistema, isso significa que cada usuário somente pode acessar os pacientes, medicamentos e históricos associados à sua própria conta, impedindo o acesso indevido a registros de terceiros. Essas regras foram aplicadas com base no campo user_id presente nas tabelas pacientes, perfis e nos relacionamentos derivados.

Essa combinação entre autenticação via JWT e políticas de RLS proporciona uma camada robusta de segurança, eliminando a necessidade de implementar controles manuais na aplicação e reduzindo significativamente os riscos de vazamento ou exposição de dados. Além disso, a segurança baseada no banco de dados contribui para que a lógica de autorização permaneça consistente e centralizada, independentemente da linguagem ou framework utilizado no back-end.

Com essa arquitetura, o sistema se torna apto a operar de forma segura em ambientes que lidam com informações sensíveis, como instituições de saúde, casas de apoio e unidades de longa permanência, assegurando conformidade com princípios de privacidade e proteção de dados.

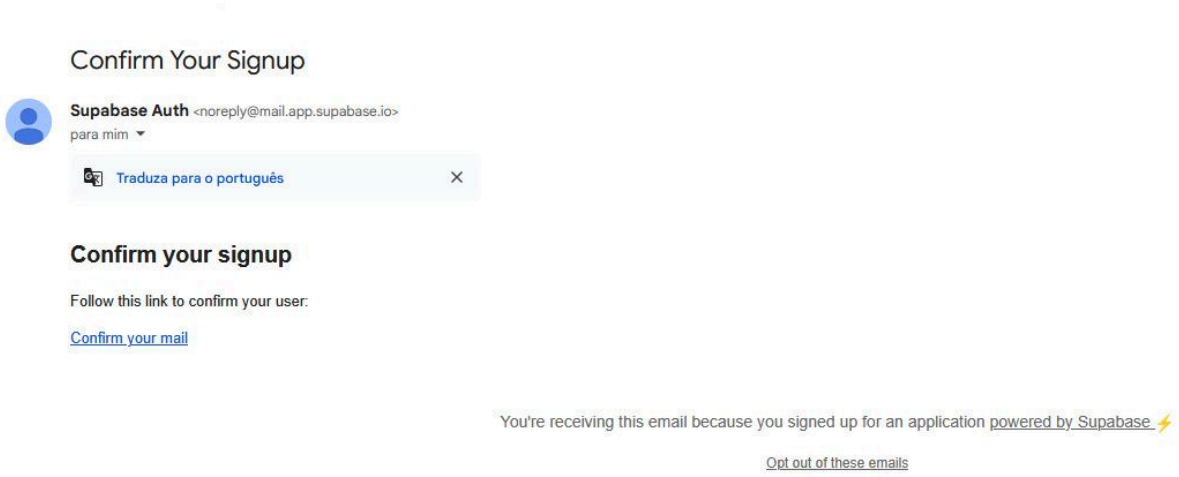
4.3.3 Implementação da Autenticação com Supabase

No desenvolvimento do sistema proposto, foi utilizada a funcionalidade de autenticação integrada do Supabase (Supabase Auth) para controlar o acesso dos usuários às funcionalidades da aplicação. Essa escolha se deu por sua facilidade de integração com o front-end em React e pelo fato de o plano gratuito atender plenamente às necessidades do projeto.

A autenticação foi configurada com suporte a e-mail e senha, incluindo recursos como confirmação por e-mail e redefinição de senha automatizada. Cada usuário autenticado passou a ter uma sessão única identificada por um token JWT, permitindo que o sistema aplicasse regras de segurança e controle de acesso com base no identificador do usuário.

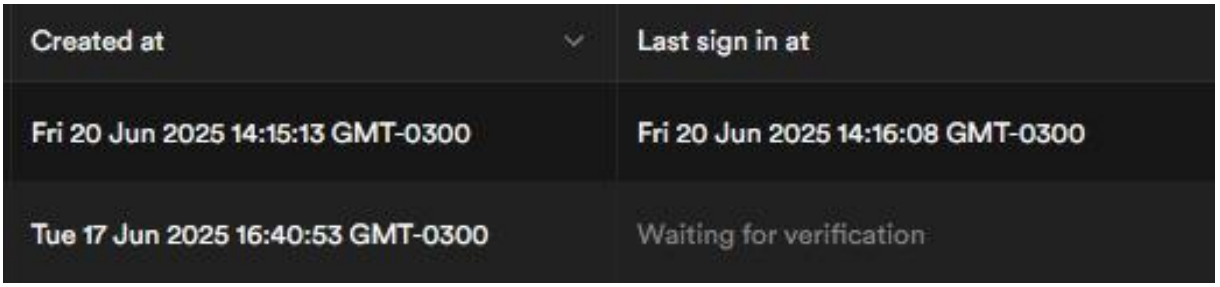
As permissões foram ajustadas no Supabase utilizando o recurso de Row Level Security (RLS), permitindo que os dados de pacientes e medicamentos fossem acessados somente pelo usuário responsável, garantindo privacidade e controle. Esse mecanismo dispensou a necessidade de implementar lógica complexa de autorização no backend, concentrando a segurança diretamente na camada de dados.

Figura 6 — Autenticação de e-mail



Fonte: Elaborado pelo autor, 2025.

Figura 7 — Status de verificação da conta do usuário



Fonte: Elaborado pelo autor, 2025.

4.3.4 Registro de logs

Durante o desenvolvimento do sistema, foi implementado um robusto mecanismo de registro de logs com o objetivo de monitorar, rastrear e documentar de forma sistemática todas as ações relevantes realizadas pelos usuários dentro da aplicação. Essa funcionalidade foi concebida como parte fundamental da arquitetura do sistema, visando garantir maior controle, transparência e segurança nas operações executadas.

Toda vez que um usuário autenticado interage com o sistema por meio de ações consideradas críticas como a aplicação de uma dose de medicamento, a edição de registros, o cadastro de um novo paciente ou qualquer modificação que impacte diretamente os dados clínicos, um registro detalhado é gerado e armazenado de forma automatizada.

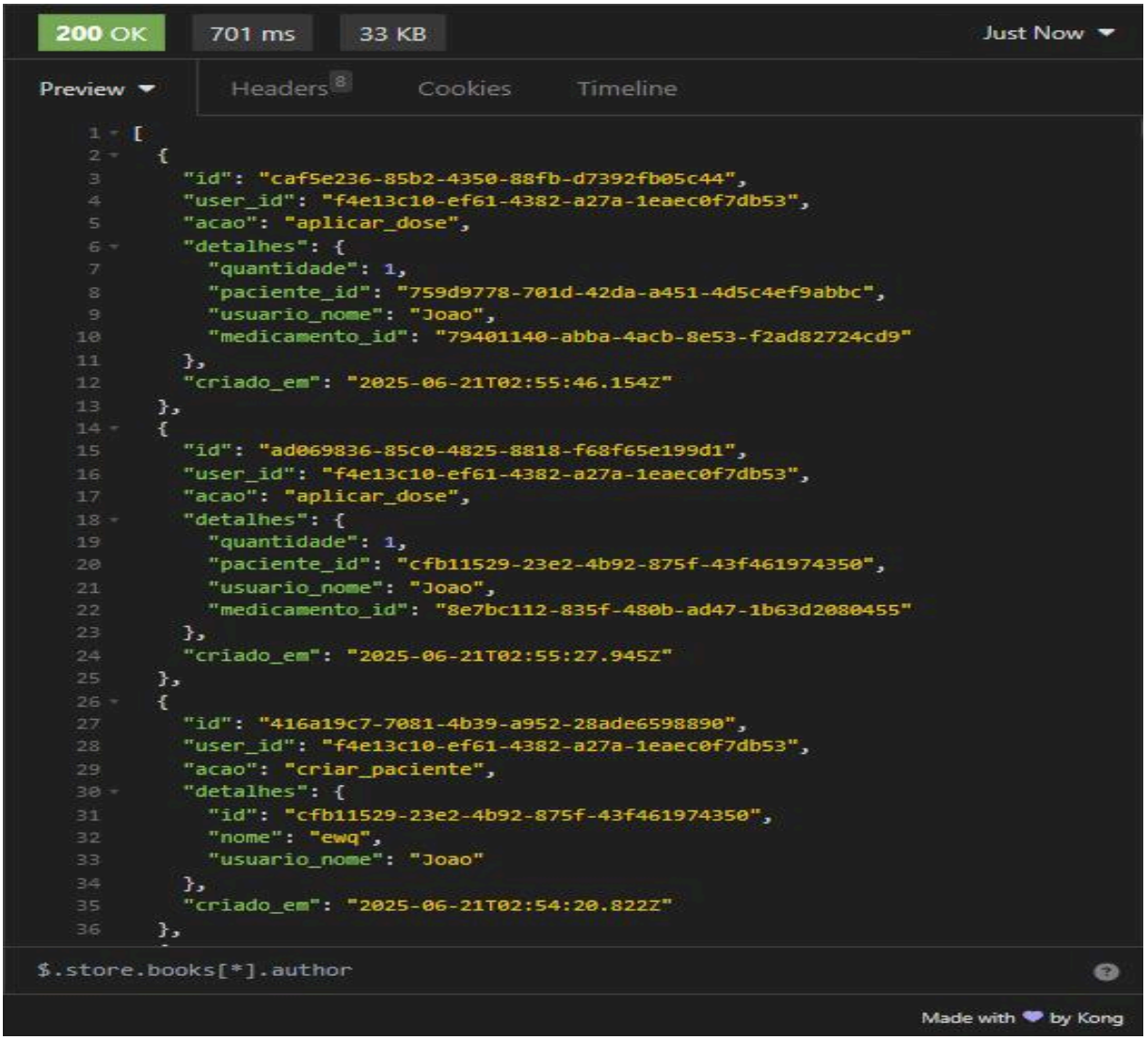
Esses registros contêm informações fundamentais para auditoria, como o identificador único da ação, o identificador do usuário que a executou, a descrição da operação (por exemplo, “aplicar_dose”, “editar_paciente” ou “criar_medicamento”), bem como dados complementares associados à ação, incluindo o nome do usuário envolvido, o ID do paciente, o medicamento relacionado e quaisquer parâmetros relevantes. Além disso, cada evento é acompanhado da data e hora exatas em que ocorreu, o que permite um rastreio cronológico completo das atividades.

Todas essas informações são persistidas no banco de dados por meio de uma estrutura organizada e acessível, permitindo sua consulta posterior de maneira estruturada e eficiente. Essa abordagem viabiliza não apenas a visualização simples dos registros, mas também análises mais complexas sobre o uso do sistema, o comportamento dos usuários e a detecção de padrões de acesso.

A adoção desse mecanismo de logging representa uma importante medida de segurança e governança da informação, sendo essencial para a rastreabilidade de ações críticas, a identificação de possíveis falhas ou comportamentos suspeitos, a realização de auditorias internas e o atendimento a eventuais exigências regulatórias ou legais relacionadas à conformidade com normas de proteção de dados e responsabilidade digital.

O acesso aos dados de log é restrito e realizado por meio de uma rota protegida da API, a qual exige autenticação e permissões específicas. Dessa forma, apenas usuários autorizados como administradores ou auditores do sistema possuem a capacidade de consultar os registros, garantindo a confidencialidade e o uso adequado das informações registradas.

Figura 8 — Registro de logs




```
[
  {
    "id": "caf5e236-85b2-4350-88fb-d7392fb05c44",
    "user_id": "f4e13c10-ef61-4382-a27a-1eaec0f7db53",
    "acao": "aplicar_dose",
    "detalhes": {
      "quantidade": 1,
      "paciente_id": "759d9778-701d-42da-a451-4d5c4ef9abbc",
      "usuario_nome": "Joao",
      "medicamento_id": "79401140-abba-4acb-8e53-f2ad82724cd9"
    },
    "criado_em": "2025-06-21T02:55:46.154Z"
  },
  {
    "id": "ad069836-85c0-4825-8818-f68f65e199d1",
    "user_id": "f4e13c10-ef61-4382-a27a-1eaec0f7db53",
    "acao": "aplicar_dose",
    "detalhes": {
      "quantidade": 1,
      "paciente_id": "cfb11529-23e2-4b92-875f-43f461974350",
      "usuario_nome": "Joao",
      "medicamento_id": "8e7bc112-835f-480b-ad47-1b63d2080455"
    },
    "criado_em": "2025-06-21T02:55:27.945Z"
  },
  {
    "id": "416a19c7-7081-4b39-a952-28ade6598890",
    "user_id": "f4e13c10-ef61-4382-a27a-1eaec0f7db53",
    "acao": "criar_paciente",
    "detalhes": {
      "id": "cfb11529-23e2-4b92-875f-43f461974350",
      "nome": "ewq",
      "usuario_nome": "Joao"
    },
    "criado_em": "2025-06-21T02:54:20.822Z"
  },
```


Fonte: Elaborado pelo autor, 2025.

4.3.5 Utilização do prisma orm e flexibilização da modelagem

Para facilitar a interação entre o código da aplicação e o banco de dados relacional utilizado no Supabase, foi adotado o Prisma ORM (Object-Relational Mapping). O Prisma permite mapear as tabelas do banco para objetos manipuláveis na aplicação, otimizando a leitura, escrita e manutenção dos dados de forma mais intuitiva e segura.

Além de oferecer uma interface de consulta declarativa e fortemente tipada, o Prisma possibilitou maior agilidade no desenvolvimento e organização da lógica de dados. Por meio de sua estrutura de modelos, foi possível estabelecer os relacionamentos entre entidades

(como pacientes, medicamentos e doses aplicadas) com clareza e controle, ao mesmo tempo em que se adotou uma estrutura de navegação mais próxima da fluidez oferecida por bancos NoSQL.

Embora o banco de dados utilizado seja relacional (PostgreSQL), a aplicação do Prisma permitiu aproximar a modelagem da experiência NoSQL em dois aspectos principais: primeiro, na forma como os dados são acessados com relacionamentos encadeados e projeções personalizadas (por exemplo, ao buscar um paciente já incluindo seus medicamentos e doses); segundo, na possibilidade de estruturar inserções e atualizações com objetos aninhados, eliminando a necessidade de múltiplas queries separadas.

Além disso, o Prisma facilitou a criação de migrates controladas, oferecendo versionamento da estrutura do banco, além de validação de dados, controle de integridade e consistência entre os ambientes de desenvolvimento, testes e produção.

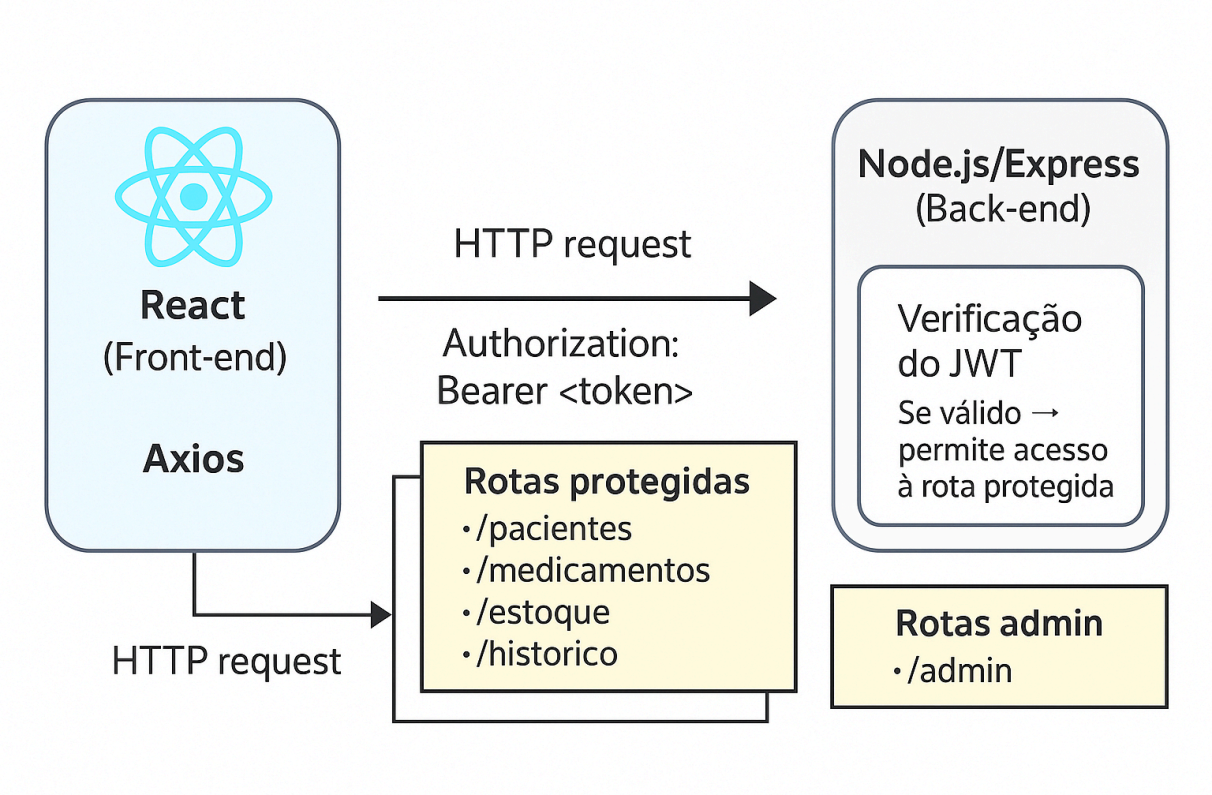
Essa abordagem híbrida utilizando um banco relacional com ferramentas que oferecem flexibilidade típica dos bancos NoSQL proporcionou ao sistema a robustez de uma modelagem relacional bem estruturada, aliada à agilidade de uma aplicação moderna, com manipulação de dados eficiente e segura.

4.3.6 Estrutura e lógica da aplicação (back-end)

A camada de back-end da aplicação foi desenvolvida utilizando o framework Express, que opera sobre a plataforma Node.js. Essa escolha se deu pela simplicidade, flexibilidade e ampla adoção da tecnologia no desenvolvimento de APIs RESTful. O Express permitiu estruturar rotas de forma modular, organizando as funcionalidades do sistema com clareza e separação de responsabilidades.

Foram criadas rotas específicas para cada entidade central do sistema, como pacientes, medicamentos, estoques, associações entre pacientes e medicamentos, além do registro histórico de doses administradas. Cada rota foi acompanhada de controladores responsáveis por aplicar a lógica de negócio, como validações de campos obrigatórios, verificação de vínculos entre entidades, atualização de status de medicação e controle de fluxo de horários.

Figura 9 — Requisições HTTP entre o Front-end e Back-end



Fonte: Elaborado pelo autor, 2025.

No lado do servidor, essas requisições são interceptadas e validadas. A lógica de autorização leva em conta o token enviado, garantindo que o usuário tenha permissão para acessar ou modificar os dados solicitados. A aplicação ainda verifica, em determinadas rotas, o role do usuário definido na tabela de perfis para permitir ou restringir funcionalidades específicas conforme seu nível de acesso (por exemplo, cuidador, administrador ou visitante).

Além disso, a arquitetura modular adotada no Express permitiu a separação de middlewares de autenticação, funções utilitárias, validações e tratadores de erro. Essa organização contribuiu para a manutenção do código, facilitando a leitura, os testes e futuras expansões da aplicação.

Com essa estrutura, o back-end da aplicação passou a oferecer uma API robusta, segura e adaptada às regras de negócio do sistema, servindo como elo entre a interface do usuário, o banco de dados e os mecanismos de autenticação e controle de acesso.

4.3.7 Desenvolvimento da interface do usuário (front-end)

A interface da aplicação foi desenvolvida utilizando a biblioteca JavaScript React, amplamente adotada no desenvolvimento de aplicações web modernas pela sua modularidade, desempenho e facilidade de manutenção. A estrutura do front-end foi organizada com base em componentes reutilizáveis, permitindo escalabilidade e padronização visual entre as diferentes telas da aplicação.

Um dos elementos centrais da interface é a tela inicial (home), que foi projetada para exibir de forma clara e objetiva a situação dos pacientes em relação à administração de medicamentos. A interface divide os pacientes em três categorias: “Pacientes disponíveis”, “Pacientes atrasados” e “Pacientes já medicados”. Cada seção é atualizada dinamicamente de acordo com o estado de medicação dos pacientes, com base nos horários registrados e nos status definidos no banco de dados.

Dentro de cada categoria, os pacientes são apresentados em cartões contendo informações relevantes como nome, idade, quarto, horário da dose, intervalo entre doses, data de nascimento, uso crônico e o status da última medicação. Também é exibida a foto do paciente, tornando a identificação visual mais rápida, especialmente útil em ambientes institucionais.

O layout foi desenvolvido com foco em clareza, garantindo boa apresentação em diferentes dispositivos. As cores utilizadas indicam o estado de medicação, ajudando o cuidador a identificar rapidamente quais pacientes necessitam de atenção. Toda a navegação foi pensada para ser intuitiva, com ações diretas e informações agrupadas de maneira lógica.

Durante o desenvolvimento do front-end, foram adotadas boas práticas como a componentização de elementos recorrentes (cartões, botões, labels), organização em pastas por funcionalidade, controle de estado com React Hooks (como useState e useEffect), além da comunicação com a API por meio do Axios. Essa abordagem garante consistência visual, reutilização de código e maior facilidade para manutenção e evolução da aplicação.

4.3.8 Ambiente de desenvolvimento e conteinerização

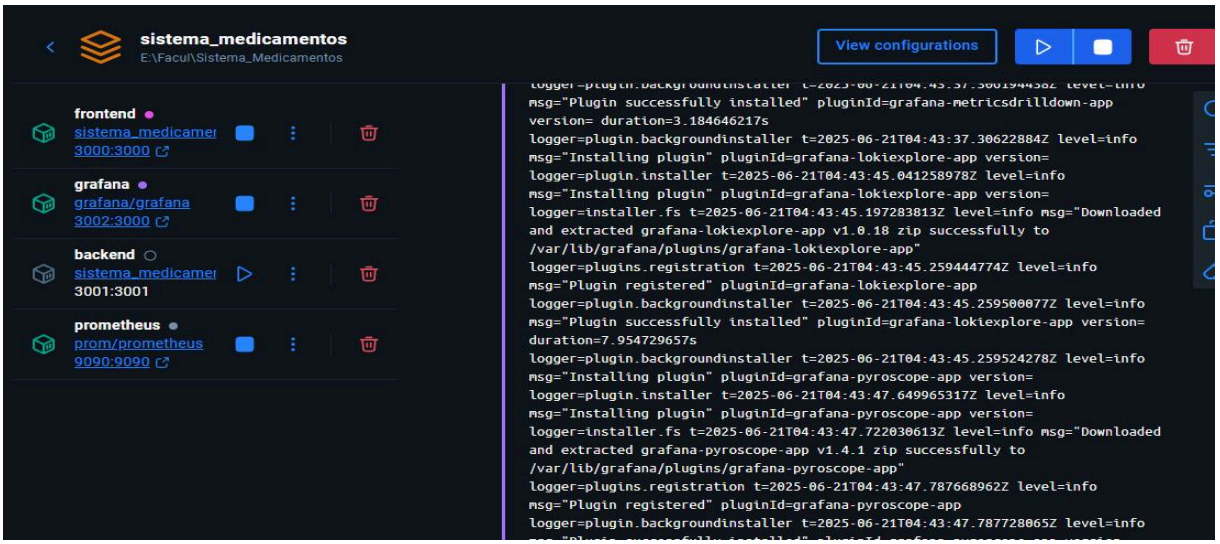
Com o objetivo de garantir padronização, portabilidade e agilidade no processo de desenvolvimento, foi adotado o uso do Docker como ferramenta de conteinerização do ambiente da aplicação. O Docker permite a criação de ambientes isolados e reproduzíveis,

evitando problemas relacionados a incompatibilidades entre sistemas operacionais, versões de dependências e configurações específicas de cada máquina desenvolvedora. A aplicação foi estruturada com base em quatro contêineres principais: o serviço de frontend, responsável pela interface desenvolvida em React e exposto na porta 3000; o backend, implementado em Node.js com Express e exposto na porta 3001, encarregado pela lógica de negócio e integração com o Supabase; o serviço de grafana, utilizado para monitoramento e visualização de dados, acessível pela porta 3002; e o prometheus, responsável pela coleta de métricas, operando na porta 9090.

Esses serviços foram configurados utilizando arquivos Docker e docker-compose, o que possibilitou inicializar todo o ambiente com um único comando, promovendo rapidez no processo de desenvolvimento e testes. As imagens dos contêineres foram baseadas em distribuições leves, com suporte a hot-reload, volumes persistentes e mapeamento de portas. A organização por contêineres específicos permitiu o isolamento funcional de cada parte do sistema, tornando o ambiente replicável e consistente para toda a equipe.

Durante o desenvolvimento, o Docker também facilitou a automação de tarefas como migração de banco de dados, seed de dados iniciais e uso de variáveis de ambiente para diferentes contextos (desenvolvimento, testes e produção). Essa abordagem ainda oferece uma base sólida para implantações futuras em serviços de nuvem compatíveis com Docker, garantindo portabilidade e escalabilidade. Em suma, a utilização do Docker trouxe previsibilidade, controle de dependências e facilidade de manutenção ao projeto, sendo um elemento central na estruturação da aplicação. A divisão dos serviços pode ser visualizada na Figura 10:

Figura 10 — Container docker



Fonte: Elaborado pelo autor, 2025.

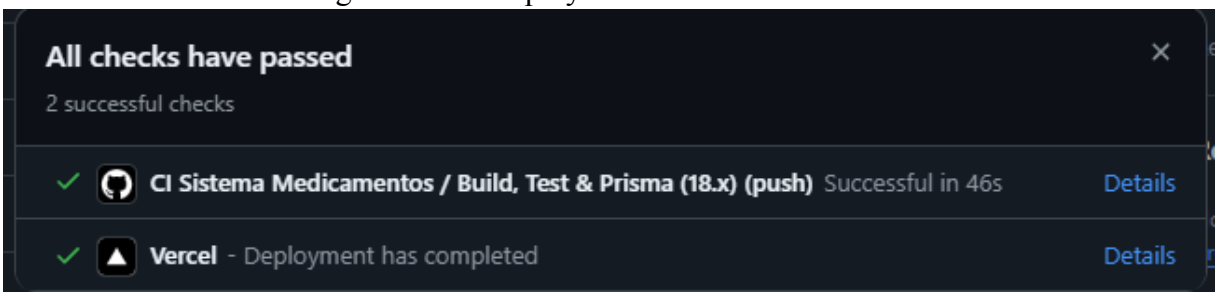
4.3.9 Versionamento de código e deploy contínuo

O versionamento de código da aplicação foi realizado por meio da plataforma GitHub, que atua como repositório remoto e ferramenta de controle de versão distribuída. A utilização do GitHub possibilitou o registro histórico de alterações, a organização das etapas de desenvolvimento por meio de branches e a colaboração entre desenvolvedores de forma segura e controlada. O projeto foi estruturado em uma branch principal (main), responsável por representar a versão estável da aplicação, e branches auxiliares destinadas a implementações específicas, correções ou testes. Esse modelo de versionamento contribuiu para o rastreamento de mudanças, facilitou revisões de código (pull requests) e minimizou conflitos entre funcionalidades desenvolvidas em paralelo.

Para publicação da aplicação em ambiente online, foi implementado um processo de deploy contínuo (CD – Deploy Contínuo) por meio da plataforma Vercel, a qual se conecta diretamente ao repositório do GitHub. A cada nova atualização enviada para a branch principal, a Vercel dispara automaticamente a etapa de build do front-end, resolve as dependências do projeto e realiza a publicação da nova versão da aplicação, sem necessidade de procedimentos manuais. Esse fluxo automatizado garante que todas as alterações aprovadas e integradas ao repositório principal sejam imediatamente disponibilizadas em ambiente de produção. Mais detalhes sobre o repositório e código-fonte estão disponíveis no Anexo A.

Além da publicação automática, a integração entre GitHub e Vercel permite a geração de links de pré-visualização para testes em pull requests, o que contribui para validações antecipadas e reduz a incidência de erros na produção. A adoção desse fluxo baseado em integração contínua trouxe como benefícios a agilidade nas entregas, a simplificação do processo de deploy, o aumento da confiabilidade do sistema e a possibilidade de rollback rápido em caso de falhas. Trata-se de uma prática moderna de desenvolvimento que reforça a eficiência operacional e a manutenção contínua da aplicação em ambientes sempre atualizados. Podemos ver isso na Figura 11 abaixo:

Figura 11 — Deploy contínuo no vercel e sistema CI



Fonte: Elaborado pelo autor, 2025.

4.3.10 Testes e validação da aplicação

A validação funcional da aplicação foi uma etapa essencial para garantir que cada funcionalidade implementada atendesse corretamente aos requisitos definidos. Os testes foram conduzidos de forma manual e exploratória, com foco principal na verificação do comportamento das rotas da API, consistência dos dados trafegados e integridade das ações executadas pelo sistema.

Para simulação de requisições HTTP e testes das rotas do back-end, foi utilizada a ferramenta Insomnia, amplamente adotada no desenvolvimento de APIs REST. Com ela, foi possível estruturar requisições completas (GET, POST, PUT, DELETE), configurar parâmetros, headers e autenticação via token JWT, além de acompanhar de forma clara as respostas do servidor. Isso permitiu validar o funcionamento das operações em diferentes cenários, como criação, atualização, busca e exclusão de registros de pacientes, medicamentos, estoque e histórico de doses.

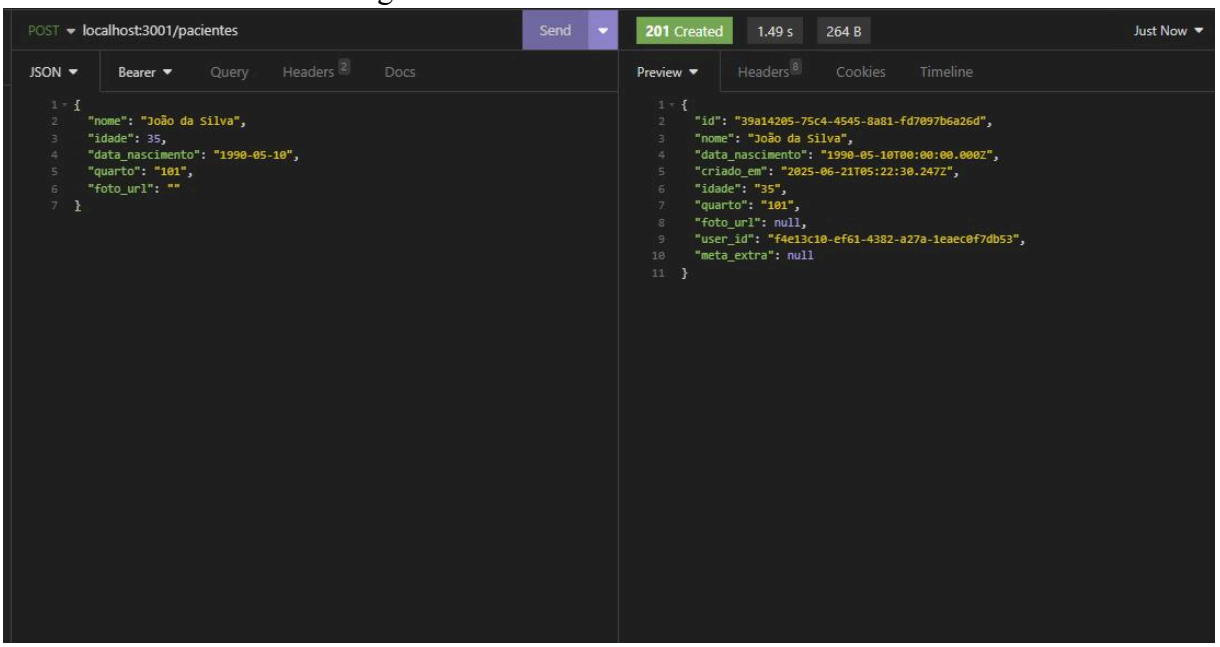
Os testes com o Insomnia também foram fundamentais para verificar o comportamento das regras de autorização e acesso. Foram realizados cenários com usuários

autenticados e não autenticados, além de tentativas de acesso a dados de outros usuários, a fim de comprovar o funcionamento correto das políticas de segurança e RLS (Row Level Security).

Além da camada de rotas, cada módulo funcional da aplicação foi testado diretamente na interface, após sua integração com o front-end. Isso incluiu a criação e edição de pacientes e medicamentos, movimentação de estoque, atualização de status de medicação, visualização em tempo real na tela inicial e registro correto no histórico. Durante esses testes, foram observados o fluxo de dados entre front-end e back-end, o retorno das mensagens de sucesso ou erro, e o impacto das ações no banco de dados.

O processo de validação foi iterativo, com ajustes pontuais realizados conforme eventuais inconsistências eram identificadas(Figura 12). Essa abordagem garantiu que o sistema entregue estivesse em conformidade com os objetivos propostos, funcionando de maneira estável, coerente e segura.

Figura 12 — Teste de rota no insomnia



Fonte: Elaborado pelo autor, 2025.

4.3.11 Envio de notificações via telegram

Durante o desenvolvimento do sistema, foi incorporado uma funcionalidade de envio de notificações em tempo real via Telegram, com o objetivo de alertar o responsável sobre situações críticas que exigem atenção imediata. Essa funcionalidade foi empregada tanto para

informar sobre níveis baixos de estoque de medicamentos quanto para avisar quando um paciente estava em atraso com seus compromissos agendados no sistema. A integração com a API do Telegram foi realizada em etapas práticas e bem definidas, permitindo um processo eficiente e confiável de comunicação entre o sistema e os usuários.

A primeira etapa consistiu na criação de um bot personalizado por meio do serviço oficial do Telegram, acessado através do contato com o @BotFather. Através do comando /newbot, foi possível definir um nome e um identificador único (@username) para o bot. Finalizada essa configuração inicial, o sistema retornou um *token* exclusivo, essencial para a autenticação e envio de mensagens.

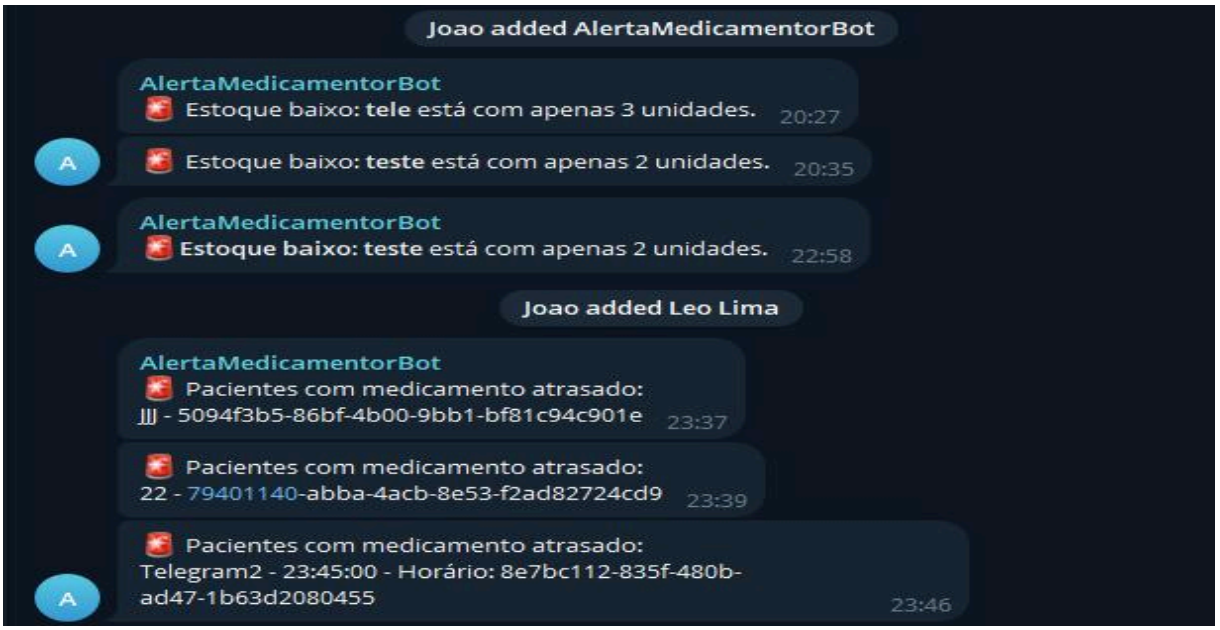
Em seguida, foi necessário identificar o chat ID do destinatário das notificações. Para isso, o usuário enviou uma mensagem ao bot, e, por meio da URL https://api.telegram.org/bot<TOKEN>/getUpdates, foi possível extrair o chat ID correspondente, tanto para conversas individuais quanto para grupos. Essa etapa garantiu que as mensagens fossem corretamente direcionadas ao canal desejado.

Com essas informações, desenvolveu-se uma função personalizada no backend da aplicação, utilizando Node.js e a biblioteca Axios, responsável por realizar as chamadas HTTP à API do Telegram. A função foi centralizada em um arquivo utilitário (telegram.js), onde o bot token e o chat ID foram referenciados a partir de variáveis de ambiente, garantindo segurança e flexibilidade na configuração.

O mecanismo de notificação foi acoplado a pontos estratégicos do sistema, especialmente em verificações automáticas. Sempre que a quantidade de um determinado medicamento cai abaixo do valor mínimo permitido, uma mensagem de alerta é enviada automaticamente ao Telegram(Figura 13), informando o nome do item e a quantidade disponível. Da mesma forma, quando um paciente ultrapassa o horário previsto para o atendimento, o sistema dispara um alerta informando o atraso, permitindo uma resposta rápida da equipe responsável. Todas as mensagens são formatadas de maneira clara e objetiva.

Por fim, o uso da biblioteca dotenv permitiu o carregamento das variáveis de ambiente necessárias para a autenticação do bot, tornando a aplicação mais segura e organizada. Essa abordagem assegura uma integração eficaz entre o sistema de gestão e o canal de comunicação, contribuindo diretamente para a agilidade na tomada de decisões.

Figura 13 — Webhooks via telegram

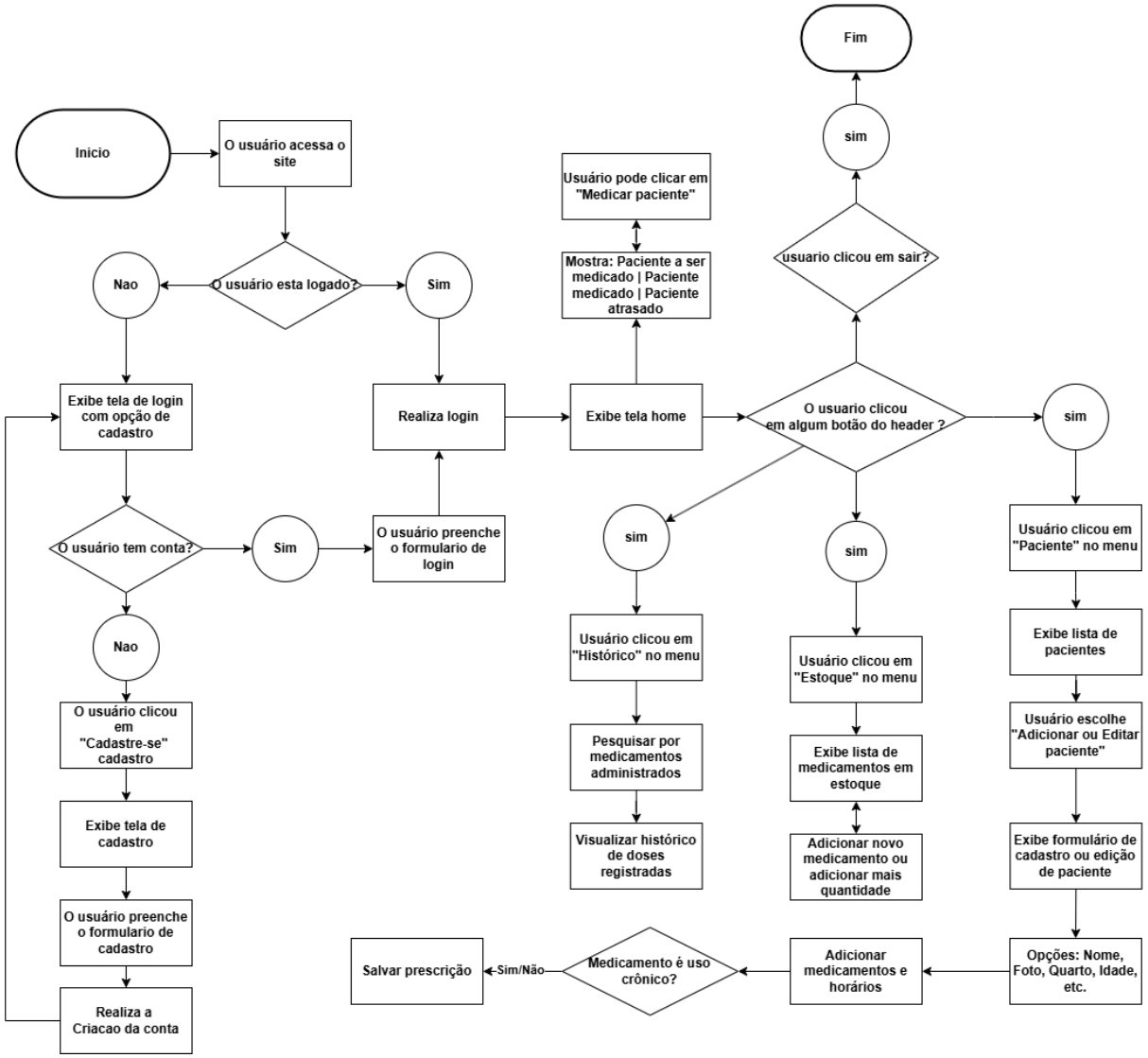


Fonte: Elaborado pelo autor, 2025.

4.4 Fluxo de navegação da aplicação

O Fluxograma da Aplicação Geral foi elaborado com a finalidade de demonstrar o fluxo de navegação e as ações realizadas pelo usuário dentro da aplicação(Figura 14). Este diagrama descreve, de maneira sequencial, o comportamento do sistema desde o acesso inicial até as funcionalidades principais, como o gerenciamento de pacientes, controle de estoque e visualização de histórico. Esse tipo de representação é fundamental para a definição da interface do usuário (UI) e da experiência do usuário (UX), além de auxiliar no planejamento das telas e nas transições entre funcionalidades ao longo do sistema.

Figura 14 — Fluxograma da Aplicação Geral



Fonte: Elaborado pelo autor, 2025.

4.4.1 Descrição do Fluxo

O fluxo de navegação da aplicação inicia-se quando o usuário acessa o sistema via navegador. A primeira verificação consiste em identificar se o usuário já está logado. Caso negativo, é exibida a tela de login, que também oferece a opção de criação de nova conta. Se o usuário ainda não possui cadastro, ele poderá clicar em “Cadastre-se”, sendo direcionado para o formulário de criação de conta. Após o preenchimento e envio dos dados, a conta é criada e o sistema o redireciona para a próxima etapa. Por outro lado, se o usuário já possui uma conta,

ele pode acessar diretamente o formulário de login, preenchê-lo e, ao autenticar-se corretamente, será redirecionado à tela principal (home) do sistema.

Ao acessar a tela home, o usuário visualiza as categorias de pacientes classificadas de acordo com o estado da medicação: pacientes disponíveis para medicação, pacientes atrasados e pacientes já medicados. A partir desta tela, é possível iniciar ações diretamente relacionadas à administração dos medicamentos por meio da opção “Medicar paciente”.

No topo da tela (header), há opções de navegação que direcionam o usuário para diferentes áreas do sistema. Se o usuário clicar em “Histórico”, será direcionado a uma página onde poderá pesquisar pelas doses administradas e visualizar o histórico completo de medicamentos já aplicados a cada paciente.

Caso o usuário clique na opção “Estoque”, o sistema exibe uma lista dos medicamentos disponíveis, permitindo a adição de novos itens ou atualização das quantidades em estoque.

Ao selecionar a opção “Paciente”, o sistema apresenta a listagem dos pacientes cadastrados. O usuário pode optar por adicionar um novo paciente ou editar os dados de um existente. Nestes casos, será exibido um formulário para preenchimento ou atualização das informações do paciente, como nome, idade, quarto e foto.

Durante o processo de adição de um novo medicamento para determinado paciente, o sistema permite configurar os horários e a frequência de administração, incluindo uma verificação sobre se o uso do medicamento será contínuo (uso crônico). Após o preenchimento dos dados, é possível salvar a prescrição, que será incorporada ao controle de doses futuras.

Por fim, o usuário tem a opção de encerrar a sessão clicando em “Sair” no menu do cabeçalho, o que encerra a navegação e finaliza o fluxo do sistema.

5 RESULTADOS E DISCUSSÕES

5.1 Avaliação da Solução Desenvolvida em Relação ao Mercado

Com base na análise de aplicativos existentes apresentada na seção 3.20, observa-se que as soluções atualmente disponíveis para o gerenciamento de medicamentos concentram-se majoritariamente no uso individual. Essas ferramentas oferecem funcionalidades como lembretes de medicação, registro de uso e gráficos de adesão ao tratamento, mas não contemplam necessidades específicas de contextos institucionais, como casas de apoio e instituições de longa permanência. Entre as limitações identificadas, destaca-se a ausência de controle coletivo de pacientes, gerenciamento de estoque e suporte a múltiplos usuários com diferentes níveis de acesso.

A aplicação desenvolvida neste trabalho foi concebida para atender essas demandas não supridas pelas soluções analisadas. O sistema propõe uma abordagem voltada à realidade de instituições que administram medicamentos para vários pacientes, permitindo o cadastramento de cuidadores e usuários vinculados, com autenticação individual por meio de token JWT. A plataforma oferece registro consolidado das doses administradas, histórico de medicação por paciente, e controle de estoque com alertas automáticos sobre a validade e a quantidade dos medicamentos. Além disso, a interface foi projetada com foco na usabilidade, visando facilitar a operação por profissionais da saúde, mesmo aqueles com pouca familiaridade com tecnologia.

Ao suprir essas lacunas, a solução apresentada neste trabalho diferencia-se das ferramentas já disponíveis, ampliando o alcance das funcionalidades e contribuindo para a segurança na administração dos medicamentos em ambientes institucionais. Essa proposta atende às exigências práticas do setor e busca garantir maior controle, rastreabilidade e eficiência no cuidado com os pacientes.

5.2 Comparação entre Soluções Existentes e a Aplicação Desenvolvida

Com base nas soluções analisadas na seção 3.20, observa-se que aplicativos como Medisafe e Pillboxie atendem prioritariamente ao público individual, oferecendo funcionalidades voltadas à autogestão da medicação. No entanto, tais soluções apresentam limitações relevantes para ambientes institucionais, como a ausência de controle de múltiplos pacientes, registro consolidado de doses, controle de estoque e autenticação multiusuário.

A aplicação desenvolvida neste trabalho foi projetada para suprir essas lacunas, com foco em instituições de saúde, clínicas e casas de apoio. Entre os principais diferenciais, destacam-se a autenticação segura com múltiplos perfis de usuários (por meio de JWT e RLS), controle centralizado de estoque, categorização automática de pacientes com base no status da medicação (disponíveis, atrasados, já medicados), e registro histórico de cada dose administrada. A estrutura web acessível e a integração com banco de dados relacional também garantem escalabilidade e confiabilidade na gestão dos dados.

Dessa forma, a aplicação proposta diferencia-se por atender demandas específicas do cuidado coletivo, com maior rastreabilidade, controle e segurança, aspectos fundamentais para ambientes institucionais, podemos comparar isso abaixo no quadro 3:

Quadro 3- Comparativo entre soluções existentes e o sistema proposto

Critério	**Medisafe**	**Pillboxie**	**Este trabalho(sistema proposto)**
Público-alvo	Usuário individual	Usuário individual	Instituições com multiplos pacientes
Plataforma	Android / iOS	iOS	Web
Controle de múltiplos pacientes	Não	Não	Sim
Registro de doses administradas	Parcial	Não	Sim
Controle de estoque	Não	Não	Sim
Relatórios em tempo real(webhooks)	Não	Não	Sim
Instalação local necessária	Sim (mobile)	Sim (iOS)	Não (acesso via navegador)
Autenticação e controle de acesso	Sim(básico)	Não	Sim(JWT + RLS)

Fonte: Elaborado pelo autor, 2025.

5.3 Tempo de respostas das apis

Com o objetivo de avaliar o desempenho do sistema desenvolvido, foram realizados testes de tempo de resposta em diversas funcionalidades essenciais para o funcionamento da aplicação. As medições foram feitas em ambiente local, utilizando a infraestrutura

conteinerizada por meio do Docker, com banco de dados operando no Supabase. As requisições foram executadas de forma sequencial e controlada, considerando-se o tempo decorrido entre o envio da solicitação e o recebimento da resposta por parte da API.

Foram consideradas para análise as operações de login de usuários, cadastro e exclusão de pacientes, consulta de dados de medicamentos, leitura dos registros de auditoria e coleta de métricas do sistema. A autenticação foi medida pelo tempo necessário para validar as credenciais e retornar o token de acesso. Já o cadastro de paciente envolveu a análise do tempo total entre o preenchimento do formulário e o armazenamento definitivo dos dados no banco. O tempo de exclusão considerou a identificação do registro e sua remoção. Para as consultas de medicamentos, foi analisado o tempo necessário para carregar os dados de medicamentos vinculados a um paciente. As operações de auditoria avaliaram a velocidade com que os logs de ações são acessados e exibidos, enquanto a coleta de métricas analisou a resposta das rotas que fornecem dados internos do sistema.

Os resultados médios obtidos para cada uma dessas ações estão organizados no Quadro 4, e evidenciam a capacidade do sistema de fornecer respostas rápidas e eficientes, mesmo em situações que envolvem múltiplas operações em banco de dados ou consultas encadeadas. Essa análise serve como base para validar a responsividade da aplicação, reforçando seu potencial de uso em ambientes reais com demandas por agilidade e confiabilidade.

Quadro 4 — Tempo de respostas da api

Rota da api	**Tamanho da resposta**	**Tempo da resposta**
Login/registro	3.3kb	entre 500ms e 2s
Criar paciente	200b	entre 700ms e 1.50s
Deletar paciente	11b	entre 2.20s e 3s
Registro de logs	variável	entre 300ms e 2s
Métricas	entre 5kb e 7kb	700ms
Estoque	600b	entre 100ms a 800ms

Fonte: Elaborado pelo autor, 2025.

5.4 Avaliação geral

A partir da análise dos dados coletados e do desempenho observado até o momento, é possível afirmar que a aplicação desenvolvida já cumpre com diversos dos objetivos inicialmente propostos. A integração entre frontend, backend e banco de dados foi estruturada de forma coesa, proporcionando uma base funcional que permite testes e uso real. Embora o projeto ainda esteja em fase de desenvolvimento e apresenta pontos que podem e devem ser aprimorados, os testes iniciais indicam que o sistema é capaz de operar de forma estável e eficiente.

A interface, mesmo em sua versão preliminar, mostrou-se acessível e funcional, permitindo que usuários com diferentes níveis de familiaridade tecnológica interajam com as funcionalidades essenciais do sistema. A robustez apresentada na manipulação de dados e no controle das operações básicas demonstra que há uma boa fundação para futuras melhorias.

Assim, mesmo com espaço considerável para evolução, o sistema já se apresenta como uma solução viável, que pode ser utilizada em contextos reais e aprimorada continuamente, conforme novas demandas e feedbacks forem sendo identificados.

5.5 Considerações finais

Durante o processo de implementação, foram enfrentados desafios relacionados principalmente à sincronização dos horários de medicação, à manipulação de fusos horários entre cliente e servidor, e à configuração inicial de ferramentas como Supabase, Prisma e Docker. No entanto, a estrutura modular do sistema permitiu a rápida identificação e correção desses pontos, com ganhos progressivos em performance e organização do código. A utilização de ferramentas modernas como o Supabase facilitou a autenticação de usuários e o armazenamento seguro de dados sensíveis, enquanto o Docker possibilitou a replicação do ambiente de desenvolvimento com consistência. A experiência de implementação também evidenciou a importância de práticas como logs de auditoria, deploy contínuo e monitoramento em tempo real, que foram fundamentais para garantir a confiabilidade e a rastreabilidade do sistema.

6 CONCLUSÕES

Como parte do encerramento deste trabalho, é possível afirmar que os objetivos propostos foram, em sua maioria, devidamente alcançados. O objetivo geral desenvolver um sistema web funcional voltado ao monitoramento e gerenciamento seguro de medicamentos em ambientes de cuidado, como casas de apoio e instituições de saúde foi atendido por meio da construção de uma aplicação completa, com backend, frontend e banco de dados integrados de forma coerente e funcional.

No que se refere aos objetivos específicos, observou-se o cumprimento de cada uma das metas estabelecidas. Foi desenvolvido um sistema de cadastro de medicamentos utilizando banco de dados relacional PostgreSQL, com o auxílio do Prisma ORM, o que facilitou a comunicação entre a aplicação e a base de dados. A interface visual construída permite o gerenciamento claro dos horários de medicação, facilitando o controle por parte dos usuários. Além disso, a funcionalidade de registrar medicamentos como “medicados” mostrou-se eficaz para garantir rastreabilidade e segurança no processo. Também foi incluído um controle de estoque básico, que alerta sobre a baixa quantidade de medicamentos, contribuindo para a prevenção de falhas no fornecimento.

Embora o sistema ainda esteja em desenvolvimento e apresente oportunidades de melhoria e expansão, os resultados obtidos até aqui demonstram o potencial da aplicação para ser utilizada em contextos reais. A conclusão é de que os objetivos do trabalho foram atingidos de forma satisfatória, ao mesmo tempo em que o projeto permanece aberto para novas funcionalidades e refinamentos futuros, como a melhoria da interface, a inclusão de relatórios mais avançados e a adaptação para dispositivos móveis.

7 SUGESTÕES PARA TRABALHOS FUTUROS

O sistema ainda possui potencial para aprimoramentos e funcionalidades adicionais que podem ser exploradas em trabalhos futuros. Entre as principais possibilidades de evolução estão:

- **Aplicativo mobile**: Desenvolver uma versão nativa para dispositivos móveis, com foco em praticidade e portabilidade no uso diário por cuidadores;
- **Internacionalização (i18n)**: Adicionar suporte a múltiplos idiomas, o que amplia a acessibilidade do sistema a usuários de diferentes regiões;
- **Integração com dispositivos IoT**: Explorar a possibilidade de conectar o sistema a dispositivos inteligentes, como dispensadores automáticos de medicamentos.

REFERÊNCIAS

AGÊNCIA NACIONAL DE VIGILÂNCIA SANITÁRIA (ANVISA). Boletim de Farmacovigilância nº 8: Erros de Medicação. Brasília: ANVISA, 2019. Disponível em: https://www.gov.br/anvisa/pt-br/centraisdeconteudo/publicacoes/monitoramento/farmacovigilancia/boletins-de-farmacovigilancia/boletim-de-farmacovigilancia-no-08.pdf. Acesso em: 22 jun. 2025.

ALMEIDA, G. Q.; SANTOS, G. B. N.; NOBLAT, A. C. B.; NOBLAT, L. A. C. B. Erros de medicação em um centro de farmacovigilância de um Hospital Universitário. Journal of Analysis and Pharmacovigilance, Salvador, v. 1, n. s2, p. 37–45, 2023. DOI: 10.22563/2525-7323.2022.v1.s2.p.37. Disponível em: https://ojs.jaff.org.br/ojs/index.php/jaff/article/view/545. Acesso em: 22 jun. 2025.

AXIOS. Axios: promise-based HTTP client for the browser and Node.js. 2023. Disponível em: https://axios-http.com/. Acesso em: 19 jun. 2025.

BANKS, Alex; PORCELLO, Eve. Learning React: functional web development with React and Redux. 2. ed. Boston: O'Reilly Media, 2017. Disponível em: https://www.oreilly.com/library/view/learning-react/9781491954614/. Acesso em: 02 jun. 2025.

BECKMAN, M. D.; DOGUCU, M.; BRAY, A. Implementing version control with Git and GitHub as a learning objective in statistics and data science courses. arXiv preprint arXiv:2001.01988, 2020. Disponível em: https://arxiv.org/abs/2001.01988. Acesso em: 19 jun. 2025.

BIEHL, Matthias. Webhooks – Events for RESTful APIs. API-University Press, 2017. Disponível em: https://books.google.com.br/books?id=5j64DwAAQBAJ. Acesso em: 20 jun. 2025.

BRASIL. Agência Nacional de Vigilância Sanitária. Erros de medicação. Boletim de Farmacovigilância, n. 8, ano 2019. Brasília: Anvisa, 2019. Disponível em: https://www.gov.br/anvisa/pt-br/centraisdeconteudo/publicacoes/monitoramento/farmacovigilancia/boletins-de-farmacovigilancia/boletim-de-farmacovigilancia-no-08.pdf. Acesso em: 01 jun. 2025.

BURK, Nikolas. Prisma 6: Better performance, more flexibility & type-safe SQL. Prisma Blog, 28 nov. 2024. Disponível em: https://www.prisma.io/blog/prisma-6-better-performance-more-flexibility-and-type-safe-sql. Acesso em: 19 jun. 2025.

CHACON, Scott; STRAUB, Ben. Pro Git. 2. ed. Berkeley: Apress, 2014. Disponível em: https://git-scm.com/book/en/v2. Acesso em: 19 jun. 2025.

CORALOGIX. Monitoring with Prometheus: Use Cases, Metrics, and Alternatives. 2024. Disponível em: https://coralogix.com/guides/prometheus-monitoring/. Acesso em: 20 jun. 2025.

DATA CAMP. Docker Compose Guide: Simplify Multi-Container Development. DataCamp, 26 mai. 2025. Disponível em: https://www.datacamp.com/tutorial/docker-compose-guide. Acesso em: 19 jun. 2025.

EBA. Framework SEO: Como eles afetam o desenvolvimento de sites e o SEO. Disponível em: https://ebaconline.com.br/blog/framework-seo. Acesso em: 19 jun. 2025.

DOCKER. What is Docker? Docker Docs, 2025. Disponível em: https://docs.docker.com/get-started/docker-overview/. Acesso em: 19 jun. 2025.

FLANAGAN, David. JavaScript: the definitive guide. 7. ed. Sebastopol: O'Reilly Media, 2020. https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016 Acesso em: 10 mai. 2025.

GE, Lixia; HENG, Bee Hoon; YAP, Chun Wei. Understanding reasons and determinants of medication non-adherence in community-dwelling adults: a cross-sectional study comparing young and older age groups. BMC Health Services Research, [S. l.], v. 23, n. 1, p. 905, 2023. DOI: 10.1186/s12913-023-09904-8. Disponível em: https://doi.org/10.1186/s12913-023-09904-8. Acesso em: 15 abr. 2025.

GITHUB. About CITATION files. GitHub Docs, 28 jul. 2021. Disponível em: https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-citation-files. Acesso em: 19 jun. 2025.

GITHUB. About version control and Git. GitHub Docs, 2025. Disponível em: https://docs.github.com/en/get-started/using-git/about-git. Acesso em: 19 jun. 2025.

GITHUB. Kong/insomnia: The open-source, cross-platform API client for GraphQL, REST, WebSockets, SSE and gRPC. GitHub, 2025. Disponível em: https://github.com/Kong/insomnia. Acesso em: 19 jun. 2025.

GOMES, D. G. dos S. et al. Adesão de pacientes idosos polimedicados: como eles se comportam frente à tomada de medicamentos? Revista Brasileira de Geriatria e Gerontologia, [S. l.], v. 27, e230211, 2024. Disponível em: https://www.scielo.br/j/rbgg/a/BDD4cghZgvwvwgGf9xY8QDc/. Acesso em: 11 jun. 2025.

GRAFANA LABS. Grafana: the open and composable observability platform. Grafana Labs, 2025. Disponível em: https://grafana.com/. Acesso em: 20 jun. 2025.

GUPTA, A.; LEE, J.; WILLIAMS, J. Database Systems for Advanced Applications: DASFAA 2014 International Workshops. Lecture Notes in Computer Science, v. 8421-8422. Berlin; Heidelberg: Springer, 2014. Disponível em: https://link.springer.com/book/10.1007/978-3-319-05810-8. Acesso em: 14 mai. 2025.

GUTTIER, M. C. R. et al. Dificuldades no uso de medicamentos por idosos acompanhados em uma coorte do Sul do Brasil. Revista Brasileira de Epidemiologia, [S. l.], v. 26, e230020,

2023. DOI: 10.1590/1980-549720230020. Disponível em: https://doi.org/10.1590/1980-549720230020. Acesso em: 10 mai. 2025.

HELP NET SECURITY. Grafana: open‑source data visualization platform. 20 maio 2024. Disponível em: https://www.helpnetsecurity.com/2024/05/20/grafana-open-source-data-visualization-platform/. Acesso em: 20 jun. 2025.

EXPRESS. Express.js: Node.js web application framework. 2020. Disponível em: https://expressjs.com/. Acesso em: 19 jun. 2025.

INSOMNIA. Introduction to Insomnia. Insomnia Docs, 2025. Disponível em: https://docs.insomnia.rest/insomnia/get-started. Acesso em: 19 jun. 2025.

JONES, Michael B.; BRADLEY, John; SAKIMURA, Nat. JSON Web Token (JWT). RFC 7519, IETF, mai. 2015. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 19 jun. 2025.

KINSMAN, Timothy; WESSEL, Mairieli; GEROSA, Marco A.; TREUDE, Christoph. How do software developers use GitHub Actions to automate their workflows? In: *Proceedings of the 18th IEEE/ACM International Conference on Mining Software Repositories* (MSR 2021), virtual, 17–19 maio 2021. p. 420–431. IEEE/ACM, 2021. DOI: 10.1109/MSR52588.2021.00054. Disponível em: https://doi.org/10.1109/MSR52588.2021.00054. Acesso em: 19 jun. 2025.

KONG. Announcing Insomnia 11 GA with vault integrations, multi‑tab support, and a new Git sync experience. Blog Kong, 18 mar. 2025. Disponível em: https://konghq.com/blog/product-releases/insomnia-11. Acesso em: 19 jun. 2025.

KONG. OpenAPI Design Tool – Kong Inc. Insomnia API Design. Kong Inc., 2025. Disponível em: https://konghq.com/products/kong-insomnia/api-design. Acesso em: 19 jun. 2025.

GAUTERIO, D. P. et al. Uso de medicamentos por pessoas idosas na comunidade: proposta de ação de enfermagem. Revista Brasileira de Enfermagem, [S. l.], v. 66, n. 5, p. 702–708, 2013. Disponível em: https://www.scielo.br/j/reben/a/TzScRWHYQQbHTtgyc5tCgvj/. Acesso em: 13 mai. 2025.

MEDISAFE. Medisafe Medication Management. Disponível em: https://www.medisafe.com. Acesso em: 27 abr. 2025.

MOHAMMED, Tihitena; MAHMUD, Sindew; GINTAMO, Binyam; MEKURIA, Zelalem Negash; GIZAW, Zemichael. Medication administration errors and associated factors among nurses in Addis Ababa federal hospitals, Ethiopia: a hospital‑based cross‑sectional study. BMJ Open, [S. l.], v. 12, n. 12, p. e066531, 2022. DOI: 10.1136/bmjopen‑2022‑066531. Disponível em: https://doi.org/10.1136/bmjopen‑2022‑066531. Acesso em: 28 mai. 2025.

MOMJIAN, Bruce. PostgreSQL: Introduction and Concepts. Boston: Addison-Wesley, 2001. Disponível em: https://momjian.us/main/writings/pgsql/other/bookfigs.pdf. Acesso em: 22 jun. 2025.

ONTORIA, Christian Toral. Desarrollo de una aplicación móvil para la alerta de caducidad de medicamentos. 2023. Trabajo de Fin de Grado (Graduação em Engenharia Informática) – Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Boadilla del Monte, 2023. Disponível em: https://oa.upm.es/75030/. Acesso em: 19 jun. 2025.

PIEROTTO, Quim. Supabase: a alternativa open-source ao Firebase. Cientistas Digitais, 2025. Disponível em: https://cientistasdigitais.com/desenvolvimento-web-e-apps/supabase-a-alternativa-open-source-ao-firebase/. Acesso em: 19 jun. 2025.

PILLBOXIE. Pillboxie App. Disponível em: https://pillboxie.com. Acesso em: 07 mar. 2025.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL: Row-Level Security. Disponível em: https://www.postgresql.org/docs/current/ddl-rowsecurity.html. Acesso em: 19 jun. 2025.

PRISMA. Is Prisma ORM an ORM? Prisma Documentation, 2025. Disponível em: https://www.prisma.io/docs/orm/overview/prisma-in-your-stack/is-prisma-an-orm. Acesso em: 19 jun. 2025.

PRISMA. Should you use Prisma ORM? Prisma Documentation, 2025. Disponível em: https://www.prisma.io/docs/orm/overview/introduction/should-you-use-prisma. Acesso em: 19 jun. 2025.

PROMETHEUS.IO. Visualizing metrics using Grafana. Tutorial oficial, 2025. Disponível em: https://prometheus.io/docs/tutorials/visualizing_metrics_using_grafana/. Acesso em: 20 jun. 2025.

SENCHA. Web Application Framework – A Comprehensive Guide. Sencha, 16 jul. 2024. Disponível em: https://www.sencha.com/blog/a-comprehensive-guide-to-web-application-frameworks/. Acesso em: 19 jun. 2025.

SOUNDCLOUD. Prometheus: Monitoring at SoundCloud. Blog SoundCloud, 26 jan. 2015. Disponível em: https://developers.soundcloud.com/blog/prometheus-monitoring-at-soundcloud. Acesso em: 20 jun. 2025.

STONEBRAKER, Michael; ROWE, Lawrence A. The design of POSTGRES. In: Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, New York, ACM, 1986. p. 340–355. Disponível em: https://doi.org/10.1145/16894.16888. Acesso em: 07 mar. 2025.

STUTTARD, Dafydd; PINTO, Marcus. The Web Application Hacker’s Handbook: finding and exploiting security flaws. 2. ed. Hoboken: Wiley, 2011. Disponível em: https://archive.org/details/dafydd-stuttard-marcus-pinto-the-web-application-hackers-handbook-finding-and-ex. Acesso em: 01 abr. 2025.

SUPABASE. Supabase Documentation. 2025. Disponível em: https://supabase.com/docs. Acesso em: 01 jun. 2025.

TORVALDS, Linus; HAMANO, Junio. Git: sistema de controle de versão distribuído. 2005. Disponível em: https://en.wikipedia.org/wiki/Git. Acesso em: 19 jun. 2025.

VASILESCU, B.; YU, Y.; WANG, H.; DEVANBU, P.; FILKOV, V. Continuous integration in a social coding world: empirical evidence from GitHub. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015), p. 805–816, 2015. DOI: 10.1145/2786805.2786850. Disponível em: https://doi.org/10.1145/2786805.2786850. Acesso em: 19 jun. 2025.

VERCEL. Getting started with Vercel. Vercel Docs, 23 maio 2025. Disponível em: https://vercel.com/docs/getting-started-with-vercel. Acesso em: 19 jun. 2025.

SUPABASE. The open-source Firebase alternative. Supabase, [S. d.]. Disponível em: https://supabase.com. Acesso em: 19 jun. 2025.

ZESTY. What is Grafana? Monitoring and Visualization Platform Explained. 2024. Disponível em: https://zesty.co/finops-glossary/grafana/. Acesso em: 20 jun. 2025.

ANEXO A — Link do Repositório no GitHub

O repositório com o código-fonte do sistema desenvolvido neste trabalho está disponível em: https://github.com/Joaobneto1/Sistema_Medicamentos. Acesso em: 22 jun. 2025.