REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

# UTILIZANDO O MÉTODO PESQUISA-AÇÃO PARA DESENVOLVER UMA API PARA UM SISTEMA DE GESTÃO DE CLÍNICAS DE SAÚDE

## *USING THE ACTION RESEARCH METHOD TO DEVELOP AN API FOR A HEALTH CLINIC MANAGEMENT SYSTEM*

## *USO DEL MÉTODO DE INVESTIGACIÓN-ACCIÓN PARA DESARROLLAR UNA API PARA UN SISTEMA DE GESTIÓN DE CLÍNICAS DE SALUD*

Pedro Makson Fontes da Costa[1], Vitor Gabriel do Nascimento Silva[2], José de Anchieta da Silva Junior[3], Reudismam Rolim de Sousa[4]

e676670
https://doi.org/10.47820/recima21.v6i7.6670

PUBLICADO: 7/2025

**RESUMO**

As plataformas de *software* estão inseridas nos mais diversos contextos. No entanto, há ambientes que necessitam de soluções tecnológicas para apoio às suas operações. Neste contexto, este artigo apresenta o desenvolvimento de uma API REST para um sistema web voltado à gestão de clínicas de saúde, com ênfase na automação de processos administrativos e operacionais. A proposta surgiu a partir das necessidades identificadas em uma clínica especializada na realização de exames laboratoriais, que até então utilizava planilhas eletrônicas para organizar suas atividades, enfrentando dificuldades como falta de integração entre setores, retrabalho e riscos de erro. Para desenvolver a API, foi utilizado o método pesquisa-ação, que busca melhorar processos práticos. O sistema busca centralizar o gerenciamento da clínica por meio de uma API, que permite o cadastro de pacientes, controle de funcionários, cadastro de serviços e agendamento de exames. Como resultado, a API, uma vez integrada a módulos *front-end*, visa oferecer uma solução eficaz e personalizável, capaz de otimizar o fluxo de trabalho da equipe, reduzir falhas e elevar a qualidade dos serviços prestados aos pacientes.

**PALAVRAS-CHAVE**: API REST. Sistema de Gestão. Clínica de Saúde.

***ABSTRACT***

*Software platforms are used in a wide range of contexts. However, some environments compete for technological solutions to support their operations. In this context, this article presents the development of a RESTful API for a web system designed to manage health clinics, with a focus on automating administrative and operational processes. The proposal arose from the needs of a clinic specializing in laboratory tests, which at the time used spreadsheets to organize its activities, facing difficulties such as a lack of integration between sectors, rework, and a risk of error. To develop an API, the action research method was used, which seeks to improve practical processes. The system aims to centralize clinic management through an API, enabling patient registration, employee control, service registration, and exam scheduling. As a result, an API, once integrated with front-end modules, aims to provide an effective and customizable solution that optimizes the team's workflow, reduces errors, and improves the quality of services offered to patients.*

***KEYWORDS:*** *REST API. Management System. Health Clinic.*

---

[1] Graduando no Bacharelado Interdisciplinar em Tecnologia da Informação pela Universidade Federal Rural do Semi-Árido (UFERSA).
[2] Graduando no Bacharelado Interdisciplinar em Tecnologia da Informação pela Universidade Federal Rural do Semi-Árido (UFERSA).
[3] Profissional de Tecnologia da Informação pela Akag Digital/Software Developer Mid-Level.
[4] Doutor em Ciência da Computação pela Universidade Federal de Campina Grande (UFCG) Professor na Universidade Federal Rural do Semi-Árido (UFERSA).

**ISSN: 2675-6218 - RECIMA21**

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



***RESUMEN***
*Las plataformas de software se utilizan en una amplia gama de contextos. Sin embargo, existen entornos que requieren soluciones tecnológicas para respaldar sus operaciones. En este contexto, este artículo presenta el desarrollo de una API REST para un sistema web destinado a la gestión de clínicas de salud, con énfasis en la automatización de procesos administrativos y operativos. La propuesta surgió de las necesidades identificadas en una clínica especializada en la realización de análisis de laboratorio, que hasta entonces utilizaba hojas de cálculo electrónicas para organizar sus actividades, enfrentando dificultades como la falta de integración entre sectores, la repetición de trabajos y el riesgo de error. Para desarrollar la API, se empleó el método de investigación-acción, que busca mejorar los procesos prácticos. El sistema busca centralizar la gestión de la clínica mediante una API que permite el registro de pacientes, el control de empleados, el registro de servicios y la programación de exámenes. Como resultado, la API, una vez integrada con los módulos front-end, busca ofrecer una solución eficaz y personalizable, capaz de optimizar el flujo de trabajo del equipo, reducir errores y mejorar la calidad de los servicios prestados a los pacientes.*

***PALABRAS CLAVE****: API REST. Sistema de Gestión. Clínica de Salud.*

## 1. INTRODUÇÃO

Muitas clínicas de saúde ainda realizam a gestão de seus processos de forma manual, utilizando planilhas eletrônicas para o controle de dados. Esse método, embora comum, é suscetível a erros, pouco eficiente e apresenta limitações na atualização e compartilhamento das informações, principalmente quando depende de arquivos locais. Como apontado por Gonçalves (2023), a prática de utilizar planilhas compromete a confiabilidade das informações, exige verificações constantes e limita a capacidade de resposta das equipes diante das demandas cotidianas. Além disso, a ausência de integração entre setores e o retrabalho decorrente do controle descentralizado tornam a gestão clínica um processo moroso e vulnerável.

De acordo com Santos, Ebram e Silva (2022), a maioria das clínicas necessita de sistemas que permitam agendar consultas ou exames, gerenciar o estoque, controlar serviços prestados e manter registros organizados dos pacientes. Apesar disso, os sistemas disponíveis no mercado nem sempre atendem às demandas específicas de cada clínica, apresentando funcionalidades genéricas que muitas vezes são subutilizadas ou deixam de contemplar processos essenciais como a gestão de funcionários e o acompanhamento financeiro.

A proposta deste trabalho surgiu a partir da observação das dificuldades enfrentadas por um cliente real em uma clínica especializada na realização de exames laboratoriais. Inicialmente, o controle de informações era feito por meio de planilhas e, posteriormente, foi adquirido um sistema *desktop*. No entanto, o sistema implementado não atendeu às expectativas da equipe, apresentando limitações em funcionalidades fundamentais, como a integração de dados e o agendamento de exames. Tal cenário evidenciou a necessidade de uma solução personalizada, capaz de otimizar os fluxos de trabalho e elevar a qualidade do atendimento.

**ISSN: 2675-6218 - RECIMA21**

Nesse contexto, o presente artigo tem como objetivo desenvolver as *APIs* que compõem um sistema de gestão para clínicas, com o intuito de otimizar e automatizar processos administrativos e operacionais. Este sistema será projetado para atender às necessidades específicas de um cliente real que atualmente gerencia sua clínica por meio de planilhas eletrônicas. Utilizando o método pesquisa-ação, foi possível levantar e analisar os requisitos conforme as necessidades do cliente, realizar a modelagem do *software* e desenvolver a *API REST* no *back-end* para o sistema de gerenciamento de clínicas.

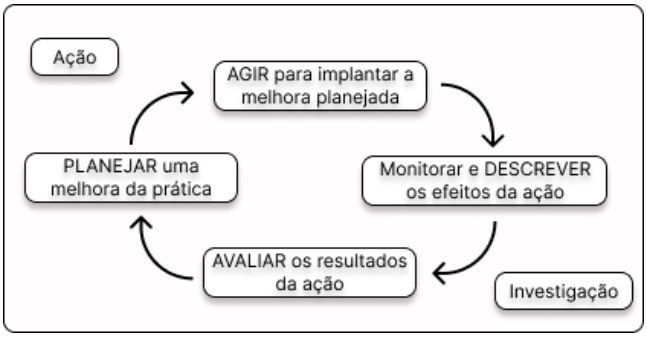
As contribuições do trabalho residem na busca por uma solução personalizada, voltada às reais necessidades da clínica em questão. A substituição do controle manual por um sistema *web* integrado permitirá centralizar as informações, reduzir falhas, agilizar tarefas e oferecer maior suporte à tomada de decisões. Dessa forma, espera-se que a implementação do sistema contribua significativamente para a melhoria do funcionamento da clínica, otimizando o uso de recursos e ampliando a qualidade dos serviços prestados aos pacientes.

## 2. MÉTODO

O método adotado neste trabalho baseou-se na pesquisa-ação, conforme proposta por Tripp (2005). Essa abordagem visa aplicar métodos consagrados da literatura para resolver um problema prático, envolvendo a colaboração entre pesquisadores e participantes no processo de diagnóstico, intervenção e avaliação. No presente caso, a pesquisa-ação foi aplicada com o objetivo de desenvolver uma *API REST* para otimizar o gerenciamento de uma clínica de exames laboratoriais, atendendo às necessidades reais de um cliente.

A pesquisa-ação seguiu quatro etapas principais: planejamento, implementação, monitoramento e avaliação, conforme o modelo de Tripp (2005), exposto na Figura 1. A representação do ciclo de pesquisa pode ser vista no Quadro 1.

**Figura 1 -** Etapas do método pesquisa-ação




**Fonte:** Melo *et al*., (2025), adaptada de Tripp (2005)

**ISSN: 2675-6218 - RECIMA21**

**Quadro 1 -** Representação do ciclo de pesquisa

| Etapa | Prática | Investigação |
|---|---|---|
| Planejamento | Levantamento dos requisitos junto ao cliente | (a) das necessidades operacionais da clínica<br>(b) das funcionalidades desejadas |
| Implementação | Desenvolvimento da *API REST* modular com tecnologias modernas | (a) do comportamento do sistema frente aos requisitos<br>(b) da escolha adequada das ferramentas |
| Avaliação | Apresentação da *API* com interface *Swagger* | (a) da aceitação do sistema pelo cliente<br>(b) da melhoria no processo de gestão da clínica |

**Fonte:** Autoria própria

No planejamento, foram realizados levantamentos de requisitos junto ao cliente, a fim de compreender as principais dificuldades enfrentadas na gestão da clínica. A fase de implementação consistiu no desenvolvimento da *API RESTful* com base nos requisitos coletados. Durante o monitoramento, foram analisados os testes e validações em ambiente de desenvolvimento. Por fim, a avaliação considerou os resultados obtidos com a aplicação prática da *API*.

As abordagens consagradas empregadas nesta pesquisa-ação consistiram da aplicação de práticas da Engenharia de *Software,* conforme sugerido por Pressman e Maxim (2021), com o objetivo de garantir robustez, manutenibilidade e segurança ao sistema.

O processo foi estruturado em três etapas principais: elicitação e análise de requisitos, modelagem do sistema e implementação da *API*. A integração dessas etapas formou um ciclo coeso, com foco em atender às necessidades reais de uma clínica especializada na realização de exames laboratoriais.

A especificação foi a primeira fase do processo, na qual os requisitos do software foram definidos com base nas necessidades e expectativas dos usuários finais. Essa etapa incluiu a identificação de funcionalidades e restrições que o sistema deveria atender, contemplando tanto os requisitos funcionais quanto os não funcionais. Os requisitos funcionais definem o comportamento esperado do sistema, enquanto os requisitos não funcionais tratam de aspectos como desempenho, segurança, confiabilidade e interoperabilidade (Pressman; Maxim, 2021).

A elicitação de requisitos foi realizada por meio de uma abordagem centrada no cliente, visando compreender de forma detalhada as operações cotidianas da clínica e as expectativas em relação ao sistema. Inicialmente, foi conduzida uma reunião presencial com o cliente real, na qual

ISSN: 2675-6218 - RECIMA21

foram investigadas as rotinas operacionais, os principais desafios enfrentados e as funcionalidades desejadas.

Durante esse encontro, foram levantadas informações referentes aos requisitos funcionais e não funcionais, com foco em desempenho, segurança, usabilidade e restrições técnicas. Reuniões subsequentes permitiram aprofundar e complementar as informações coletadas, assegurando que os requisitos identificados representassem com precisão as necessidades específicas da clínica.

Com a análise e validação dos requisitos, iniciou-se a fase de modelagem do sistema. Foram utilizados os recursos da *UML (Unified Modeling Language)* (Guedes, 2018), por meio da ferramenta *PlantUML*, permitindo a construção clara e estruturada dos modelos necessários.

Foram elaborados principalmente três diagramas: o diagrama de classes, responsável por representar a estrutura do sistema; o diagrama de casos de uso, que mapeia as interações dos usuários com o sistema; e o modelo entidade-relacionamento, essencial para o planejamento do banco de dados relacional.

A etapa de implementação foi conduzida com tecnologias modernas do ecossistema *JavaScript/TypeScript*. O desenvolvimento da *API REST* no *back-end* foi realizado utilizando o ambiente *Node.js*, em conjunto com o *framework NestJS*, que oferece uma arquitetura modular e escalável.

A aplicação foi desenvolvida com *TypeScript*, garantindo maior robustez e legibilidade do código por meio da tipagem estática. O banco de dados utilizado foi o *PostgreSQL*, acessado via *TypeORM*, que proporciona mapeamento objeto-relacional eficiente. A comunicação entre os módulos do sistema foi feita por meio de *APIs RESTful*, cuja documentação foi gerada automaticamente com o uso da ferramenta *Swagger*.

No que se refere à segurança, o sistema implementa autenticação baseada em *JSON Web Tokens (JWT)*, além da criptografia de senhas com o algoritmo *Argon2*, proporcionando proteção às informações sensíveis dos usuários.

## 3. MODELAGEM DO SISTEMA

Nesta seção, é exposto o processo de desenvolvimento do sistema de gerenciamento para clínicas. Iniciando com a elicitação e análise de requisitos e, por sua vez, a modelagem do sistema.

A união dessas etapas metodológicas forma um ciclo coeso que permite a criação do sistema, atendendo às necessidades específicas da clínica.

### 3.1. Requisitos do Sistema

A etapa de levantamento e definição de requisitos é fundamental no processo de desenvolvimento de sistemas, pois permite identificar e documentar as funcionalidades que a aplicação deverá atender para corresponder às necessidades reais do cliente. No contexto do

ISSN: 2675-6218 - RECIMA21

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



sistema de gerenciamento de clínicas desenvolvido neste trabalho, os requisitos foram elicitados com base nas demandas observadas em uma clínica real, especializada na realização de exames laboratoriais.

Os requisitos funcionais descritos a seguir representam as operações essenciais que o sistema deve oferecer aos seus diferentes perfis de usuários (pacientes, funcionários e administradores). Esses requisitos englobam desde o processo de autenticação até o controle de pacientes, exames, serviços, colaboradores, horários de funcionamento e gestão financeira.

A seguir, apresenta-se o Quadro 2 com a lista completa dos requisitos funcionais identificados para o sistema:

**Quadro 2**. Lista de Requisitos Funcionais

| Requisitos Funcionais (RF) | Descrição |
|---|---|
| **RF001 - Entrar no sistema** | Todos os usuários poderão acessar o sistema mediante autenticação com e-mail e senha. |
| **RF002 - Sair do sistema** | Todos os usuários poderão encerrar sua sessão de forma segura. |
| **RF003 - Cadastrar paciente** | Funcionários e Administradores poderão cadastrar novos pacientes com informações pessoais e clínicas. |
| **RF004 - Visualizar paciente** | Funcionários e Administradores poderão visualizar os dados detalhados de pacientes. |
| **RF005 - Listar pacientes** | Funcionários e Administradores terão acesso à lista de todos os pacientes cadastrados. |
| **RF006 - Editar paciente** | Funcionários e Administradores poderão editar os dados dos pacientes. |
| **RF007 - Cadastrar funcionário** | O Administrador poderá cadastrar novos funcionários com dados como nome, CPF, e-mail e cargo. |
| **RF008 - Visualizar funcionário** | O Administrador poderá visualizar os dados detalhados de um funcionário. |
| **RF009 - Listar funcionários** | O Administrador terá acesso à listagem completa dos funcionários cadastrados. |
| **RF010 - Editar funcionário** | O Administrador poderá atualizar as informações de um funcionário. |
| **RF011 - Excluir funcionário** | O Administrador poderá remover um funcionário do sistema. |

ISSN: 2675-6218 - RECIMA21

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



| | |
|---|---|
| **RF012 - Cadastrar serviço** | O Administrador poderá cadastrar novos serviços prestados pela clínica. |
| **RF013 - Visualizar serviço** | Funcionários e Administradores poderão visualizar os dados de serviços registrados. |
| **RF014 - Listar serviços** | Funcionários e Administradores terão acesso à listagem de serviços disponíveis. |
| **RF015 - Editar serviço** | O Administrador poderá alterar os dados de um serviço existente. |
| **RF016 - Cadastrar exame** | Funcionários e Administradores poderão cadastrar exames vinculando paciente, serviço, data e local. |
| **RF017 - Visualizar exame** | Todos os usuários poderão visualizar exames. Pacientes verão apenas seus próprios exames. |
| **RF018 - Listar exames** | Funcionários e Administradores poderão consultar todos os exames cadastrados. Pacientes visualizaram apenas os próprios. |
| **RF019 - Editar exame** | Funcionários e Administradores poderão editar os dados de exames agendados. |
| **RF020 - Excluir exame** | Funcionários e Administradores poderão excluir exames do sistema. |
| **RF021 - Realizar pagamento de exame** | Pacientes poderão registrar o pagamento de exames. Funcionários e Administradores poderão confirmar os registros. |
| **RF022 – Cadastrar receita/despesa** | Funcionários e Administradores poderão registrar entradas e saídas financeiras da clínica. |
| **RF023 - Visualizar título** | Funcionários e Administradores poderão visualizar os detalhes dos títulos financeiros. |
| **RF024 - Listar títulos** | Funcionários e Administradores terão acesso à lista completa de títulos registrados. |
| **RF025 - Editar título** | Funcionários e Administradores poderão modificar os dados de um título. |
| **RF026 - Excluir título** | Funcionários e Administradores poderão excluir um título do sistema. |
| **RF027 - Lançar pagamento** | Funcionários e Administradores poderão registrar o pagamento de despesas. |
| **RF028 - Lançar recebimento** | Funcionários e Administradores poderão registrar o recebimento de receitas. |

**ISSN: 2675-6218 - RECIMA21**

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



| | |
|---|---|
| **RF029 - Cadastrar horários de funcionamento** | Administradores poderão definir os horários de atendimento da clínica. |
| **RF030 - Visualizar horários de funcionamento** | Funcionários e Administradores poderão visualizar os horários disponíveis para atendimento. |
| **RF031 - Editar horário de funcionamento** | O Administrador poderá alterar os horários já cadastrados. |
| **RF032 - Excluir horário de funcionamento** | O Administrador poderá remover horários previamente definidos. |

**Fonte:** Autoria própria

Além disso, foram definidos requisitos não funcionais, que garantem a qualidade, segurança e confiabilidade da aplicação. Esses requisitos estabelecem critérios técnicos importantes, como a validação de dados inseridos, o controle de sessões, a verificação segura das credenciais de acesso e a obrigatoriedade de atualização periódica de senhas. Eles asseguram que o sistema funcione corretamente mesmo em situações críticas, promovendo uma experiência segura e estável aos usuários.

A seguir, apresenta-se o Quadro 3 com a lista dos requisitos não funcionais identificados para o sistema:

**Quadro 3.** Lista de Requisitos Não Funcionais

| Requisitos Não Funcionais (RNF) | Descrição |
|---|---|
| **RNF001 - Validar dados de entrada** | O sistema deve validar campos obrigatórios, formatos, tamanhos e caracteres, garantindo que apenas dados corretos e completos sejam aceitos. |
| **RNF002 - Validar credenciais de usuário** | As credenciais fornecidas devem ser verificadas com base nos dados armazenados, com mensagens claras em caso de erro. |
| **RNF003 - Controlar tempo limite de sessão** | A sessão do usuário será encerrada automaticamente após 3 minutos de inatividade. |
| **RNF004 - Alterar senha periodicamente** | O sistema exigirá que os usuários troquem suas senhas a cada 90 dias, reforçando a segurança. |

**Fonte:** Autoria própria

### 3.2. Diagrama de Casos de Uso

Os diagramas de caso de uso são artefatos fundamentais na modelagem de sistemas orientados a objetos. Eles descrevem, de forma clara e visual, as interações entre os atores e as funcionalidades oferecidas pelo sistema, denominadas casos de uso. Essa representação contribui

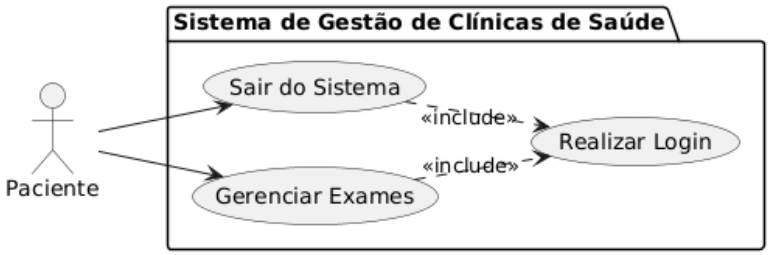
ISSN: 2675-6218 - RECIMA21

significativamente para a compreensão dos requisitos do sistema e para a comunicação entre desenvolvedores, analistas e demais *stakeholders.*

No contexto do sistema de gestão de clínicas de saúde proposto neste trabalho, foram elaborados três diagramas de caso de uso distintos, cada um representando as funcionalidades acessíveis a um perfil de usuário específico: Paciente, Funcionário e Administrador. Esses diagramas auxiliam na identificação e organização das funcionalidades que cada tipo de usuário pode acessar, além de destacar casos comuns, como o processo de autenticação, por meio da relação de inclusão entre os casos de uso.

Na Figura 2, apresenta-se o primeiro diagrama, que descreve as interações do perfil Paciente com o sistema.

**Figura 2 -** Diagrama de Casos de Uso (Paciente)




**Fonte:** Autoria própria

O diagrama de caso de uso do perfil Paciente representa as funcionalidades acessíveis aos usuários cadastrados como pacientes no sistema. O principal objetivo desse perfil é possibilitar a consulta e o acompanhamento de exames, além de oferecer recursos relacionados à visualização de informações pessoais e à interação com funcionalidades administrativas de forma segura e restrita.

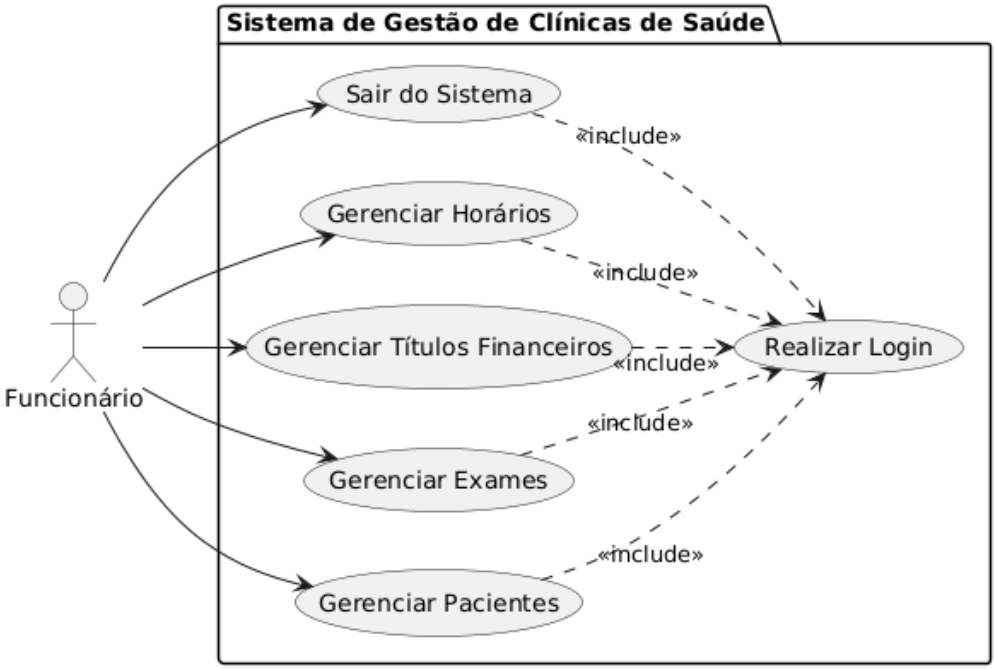
Na Figura 3, apresenta-se o segundo diagrama, que descreve as interações do perfil Funcionário com o sistema.

ISSN: 2675-6218 - RECIMA21

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



**Figura 3 -** Diagrama de Casos de Uso (Funcionário)



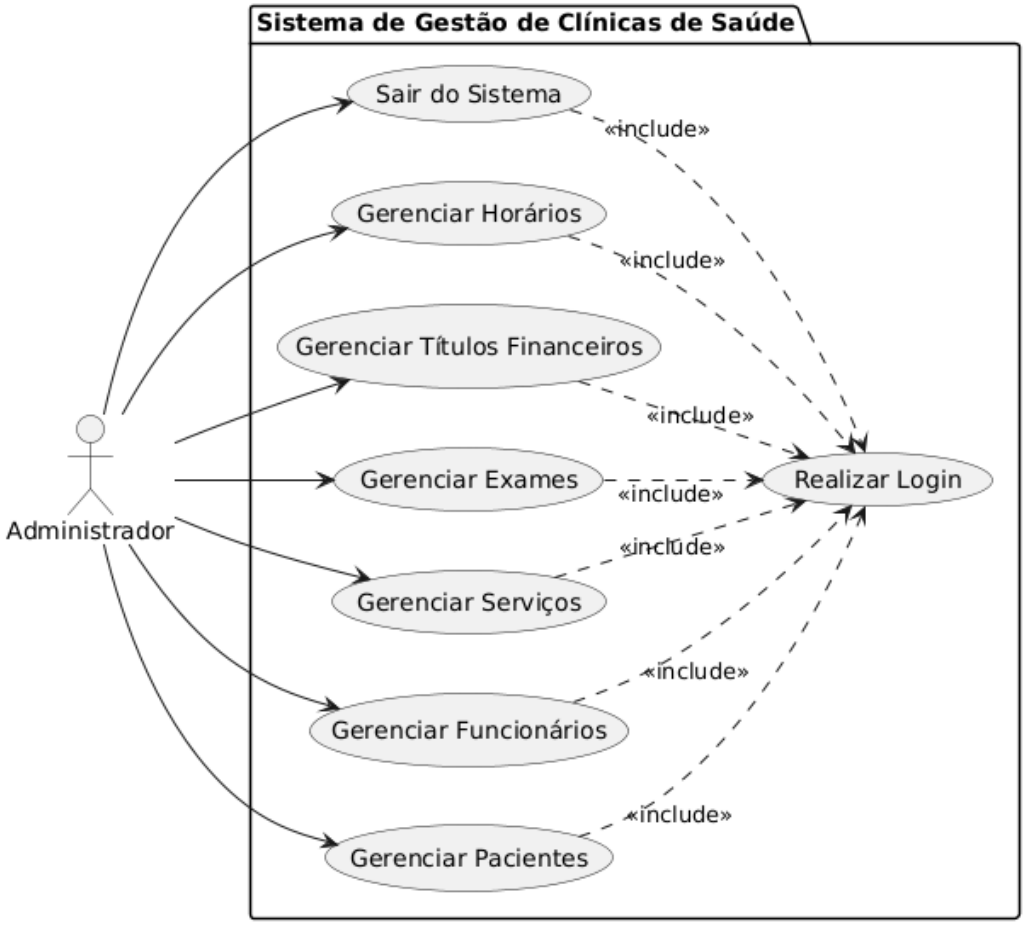

**Fonte:** Autoria própria

O diagrama de caso de uso do perfil Funcionário apresenta as funcionalidades disponíveis para os colaboradores da clínica com permissões operacionais dentro do sistema. Esses usuários têm acesso a operações relacionadas ao gerenciamento de pacientes, exames, títulos financeiros e horários de funcionamento.

Na Figura 4, apresenta-se o terceiro diagrama, que descreve as interações do perfil Administrador com o sistema.

ISSN: 2675-6218 - RECIMA21

**Figura 4 -** Diagrama de Casos de Uso (Administrador)




**Fonte:** Autoria própria

O diagrama de caso de uso referente ao perfil Administrador descreve as principais ações que esse tipo de usuário pode executar no sistema de gestão de clínicas de saúde. O administrador possui acesso completo às funcionalidades administrativas e operacionais, sendo responsável por gerenciar informações de pacientes, funcionários, serviços, exames, finanças e horários de funcionamento da clínica.

### 3.3. Diagrama de Classes

O diagrama de classes representa a estrutura estática do sistema, evidenciando as entidades principais, seus atributos e os relacionamentos entre elas. Ele é uma das peças fundamentais da modelagem orientada a objetos, sendo essencial para o entendimento e organização da arquitetura de dados da aplicação.

**ISSN: 2675-6218 - RECIMA21**

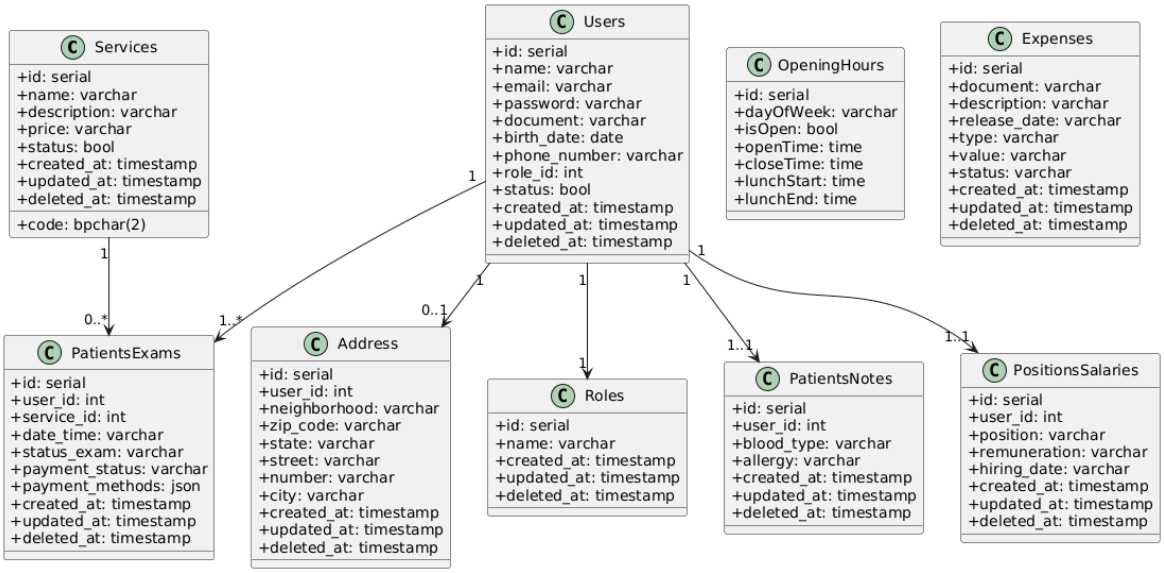
REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



No contexto do sistema de gestão de clínicas de saúde desenvolvido neste trabalho, o diagrama de classes foi construído com base nos requisitos levantados e modela as principais entidades envolvidas nos processos administrativos, operacionais e clínicos da instituição. Este modelo visa garantir uma estrutura clara e consistente para o desenvolvimento das *APIs*, refletindo as regras de negócio da clínica.

A seguir na Figura 5, apresenta-se o diagrama de classes, que descreve as interações das entidades do sistema.

**Figura 5 -** Diagrama de Classe




**Fonte:** Autoria própria

O diagrama apresentado na Figura 5 contempla as seguintes entidades principais:

- *Users:* representa os usuários do sistema, sejam pacientes, funcionários ou administradores. Cada usuário está vinculado a um papel definido pela tabela *Roles.*
- *Roles:* define os tipos de papéis existentes no sistema (ex.: paciente, funcionário, administrador), essenciais para o controle de permissões.
- *PatientsExams:* armazena os exames realizados pelos pacientes, vinculando cada exame a um usuário e a um serviço oferecido pela clínica.
- *Services:* registra os serviços prestados pela clínica, como os tipos de exames laboratoriais, contendo dados como nome, código, preço e descrição.
- *PatientsNotes:* contém informações complementares sobre os pacientes, como tipo sanguíneo e alergias, auxiliando no cuidado clínico individualizado.

ISSN: 2675-6218 - RECIMA21

- *Address:* representa o endereço associado ao usuário, com informações como bairro, cidade, rua e CEP.
- *PositionsSalaries:* armazena informações relacionadas ao vínculo empregatício dos funcionários, como cargo, salário e data de admissão.
- *Expenses: registra as movimentações financeiras da clínica, como despesas e receitas, com campos para valor, tipo, status e data de lançamento.*
- *OpeningHours:* define os horários de funcionamento da clínica, indicando dias da semana, turnos e intervalos de almoço.

Este diagrama fornece uma visão clara da estrutura e das regras de negócio do sistema, sendo fundamental para guiar o desenvolvimento das funcionalidades no *back-end*, garantindo organização, escalabilidade e coerência entre os dados.

### 3.4. Diagrama de Entidade Relacionamento

O Diagrama Entidade-Relacionamento (DER) tem como objetivo representar de forma visual a estrutura lógica do banco de dados, detalhando as entidades envolvidas, seus atributos e os relacionamentos entre elas. Essa modelagem é fundamental para garantir a integridade dos dados e servir como base para o desenvolvimento de sistemas bem estruturados e escaláveis.

No contexto deste projeto, o DER ilustra as principais entidades que compõem o sistema, tais como usuários, serviços, exames, endereços e papéis de acesso (*roles*). Ele revela como essas entidades se conectam, evidenciando, por exemplo, que um usuário pode estar associado a um ou mais exames, possuir um endereço cadastrado, ter anotações clínicas vinculadas e estar relacionado a uma função específica (*role*) no sistema.

Além disso, o modelo contempla informações complementares, como horários de funcionamento, gastos operacionais e informações salariais de funcionários, permitindo uma visão abrangente e integrada dos dados que o sistema precisa gerenciar. A modelagem também contempla atributos temporais, como datas de criação, atualização e exclusão lógica, promovendo rastreabilidade e controle de histórico dos registros.

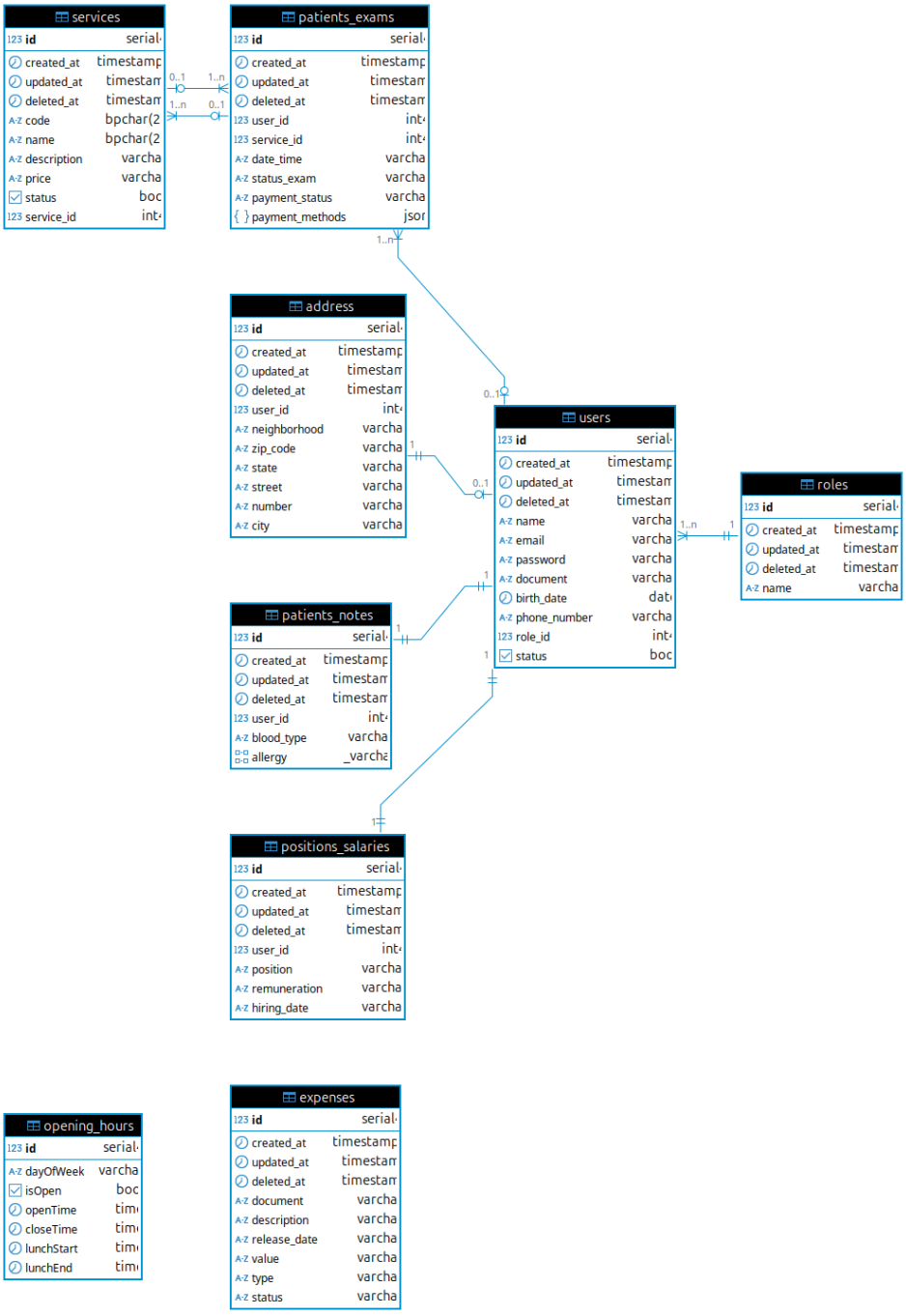
O diagrama apresentado na Figura 6 contempla as seguintes tabelas principais:

ISSN: 2675-6218 - RECIMA21

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



**Figura 6 -** Diagrama Entidade Relacionamento




**Fonte:** Autoria própria

Esse DER serviu como ponto de partida para a derivação do diagrama de classes e orientou a estruturação do banco de dados relacional utilizado no projeto.

**ISSN: 2675-6218 - RECIMA21**

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



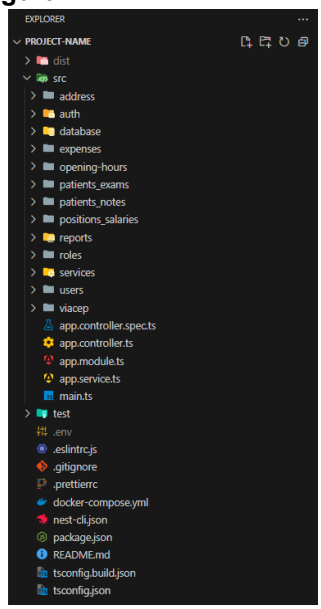
## 4. DESENVOLVIMENTO DA *API*

O desenvolvimento da *API REST* do sistema de gestão clínica foi baseado em uma arquitetura modular, escalável e segura, utilizando tecnologias modernas do ecossistema *JavaScript/TypeScript.* Segundo Felício (2022), uma *API* (Interface de Programação de Aplicações) tem como função permitir a comunicação entre diferentes sistemas de forma simplificada e estruturada, o que é essencial em ambientes que demandam integração de serviços, como é o caso de clínicas de saúde.

Para a construção do *back-end*, foi utilizado o *Node.js*, que permite o desenvolvimento de aplicações *server-side*. De acordo com Godinho (2024), o *Node.js* destaca-se por seu alto desempenho e compatibilidade com diversos sistemas operacionais. Além disso, como complementa Nunes (2018), sua leveza e escalabilidade o tornam ideal para aplicações que exigem grande volume de requisições.

Sobre esse ambiente, foi adotado o *framework NestJS*, que, conforme Mata, Lima e Mata (2023), oferece uma estrutura robusta baseada em módulos reutilizáveis, controladores e serviços. Os controladores recebem as requisições *HTTP*, enquanto os serviços concentram a lógica de negócio da aplicação. Isso permitiu organizar os módulos do sistema de forma clara, mantendo a separação de responsabilidades e facilitando a manutenção do código. Cada módulo da aplicação foi implementado como uma unidade funcional independente, incluindo os módulos de usuários, pacientes, exames, serviços, funcionários, financeiro e horários de funcionamento (Figura 7), permitindo organizar a lógica da *API* de forma coesa e reutilizável.

**Figura 7 -** Estrutura do sistema




**Fonte:** Autoria própria

**ISSN: 2675-6218 - RECIMA21**

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

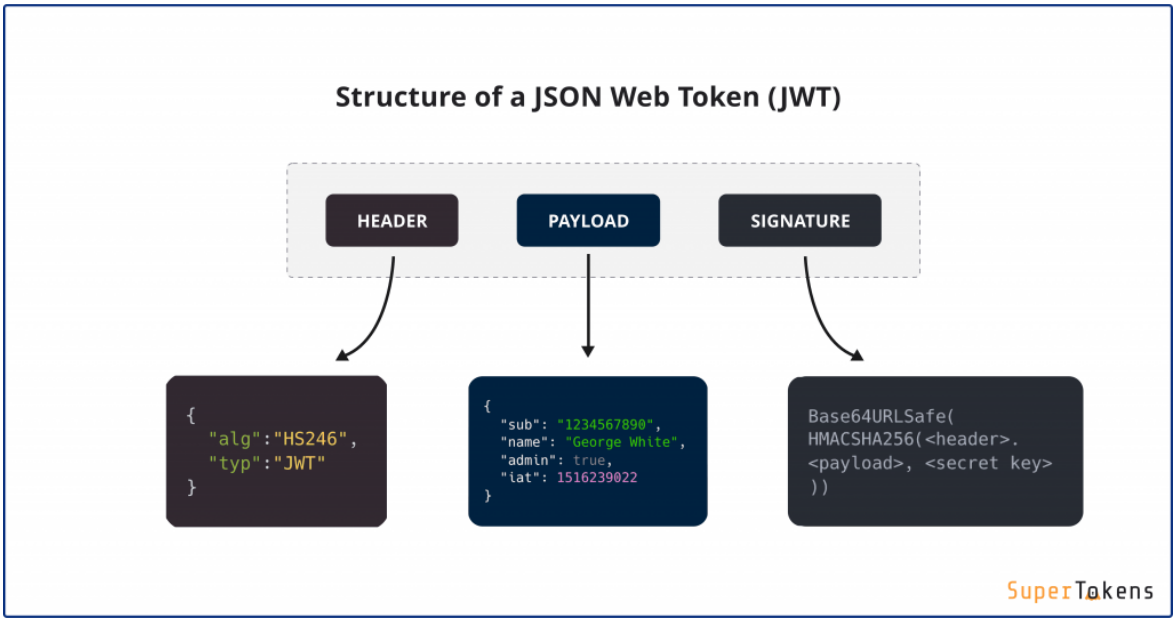


A linguagem utilizada no desenvolvimento foi o *TypeScript*, um superconjunto do *JavaScript*. Como ressalta Mendes (2022), o *TypeScript* adiciona tipagem estática ao *JavaScript*, aumentando a segurança e a padronização do código, o que foi fundamental na construção de uma aplicação robusta e confiável. O uso do *Express.js*, *framework* que simplifica o tratamento de rotas e requisições HTTP (Barbosa, 2023), também contribuiu para uma implementação ágil e eficiente das funcionalidades.

A persistência dos dados foi realizada utilizando o *PostgreSQL*, sistema de gerenciamento de banco de dados relacional de código aberto. De acordo com o PostgreSQL *Global Development Group* (2023), trata-se de uma solução robusta e aderente aos padrões *SQL*, sendo apropriada para aplicações que exigem alta confiabilidade e escalabilidade. A comunicação entre o sistema e o banco de dados foi feita por meio do *TypeORM*, um *framework* de mapeamento objeto-relacional (Mendes, 2022), que permitiu o uso de classes e entidades no acesso aos dados, promovendo maior integração com a modelagem do sistema.

No que se refere à segurança, foram implementadas estratégias de autenticação e proteção de dados sensíveis. A *API* utiliza autenticação por *JSON Web Token (JWT)*, padrão que permite a transmissão segura de informações por meio de objetos *JSON* assinados digitalmente (Barbosa, 2023). Para o armazenamento das senhas, foi adotado o algoritmo *Argon2*, vencedor da *Password Hashing Competition*, que oferece elevada proteção contra ataques de força bruta e otimiza o uso de memória (Gregório; Goya, 2019).

**Figura 8 -** Estrutura JSON Web Token




**Fonte:** Vantico (2025)

**ISSN: 2675-6218 - RECIMA21**

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



## 5. RESULTADOS

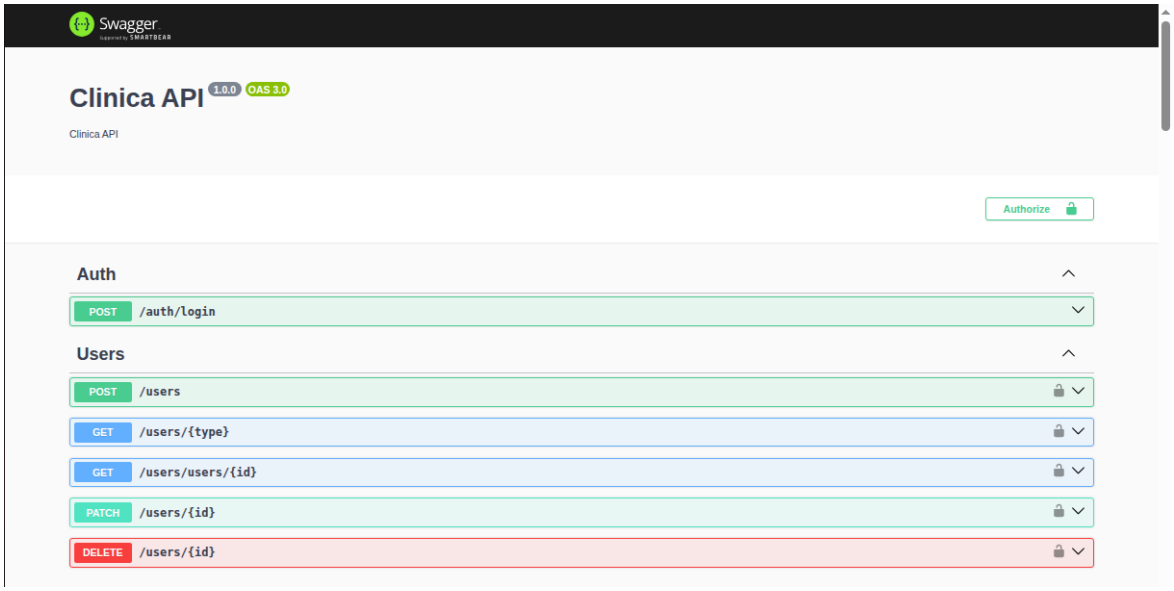
A fase de desenvolvimento resultou na construção de uma *API REST* funcional voltada à gestão de clínicas de saúde, implementada com o *framework NestJS*. A arquitetura adotada seguiu uma abordagem modular, com o uso de controladores, serviços e entidades, o que proporcionou clareza, escalabilidade e facilidade de manutenção do código, conforme discutido por Mata, Lima e Mata (2023).

Para evidenciar a operação da aplicação desenvolvida, foram utilizados os recursos da ferramenta *Swagger*, que gerou automaticamente uma documentação interativa da *API*. O *Swagger*, segundo Batista (2022), é baseado na especificação *OpenAPI* e permite que desenvolvedores explorem *endpoints*, parâmetros, modelos de dados e autenticação de forma intuitiva e prática.

Na Figura 9, é possível ver algumas imagens das rotas da *API* no *Swagger*, demonstrando os principais grupos de rotas implementados, como autenticação, cadastro de usuários, exames, serviços e horários. Cada rota possui descrições claras, métodos *HTTP* bem definidos e exemplos de entrada e saída, permitindo testes diretos na interface.

**Figura 9 -** Documentação da API via *Swagger*

(a) Tela contendo as seções *Auth* e *Users*

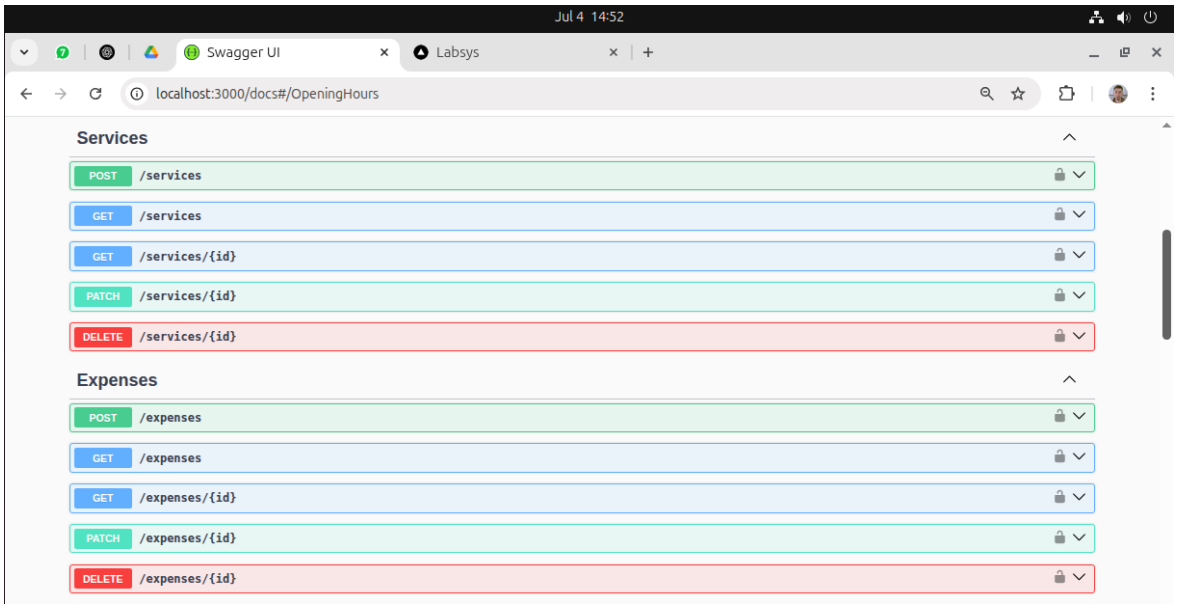



ISSN: 2675-6218 - RECIMA21

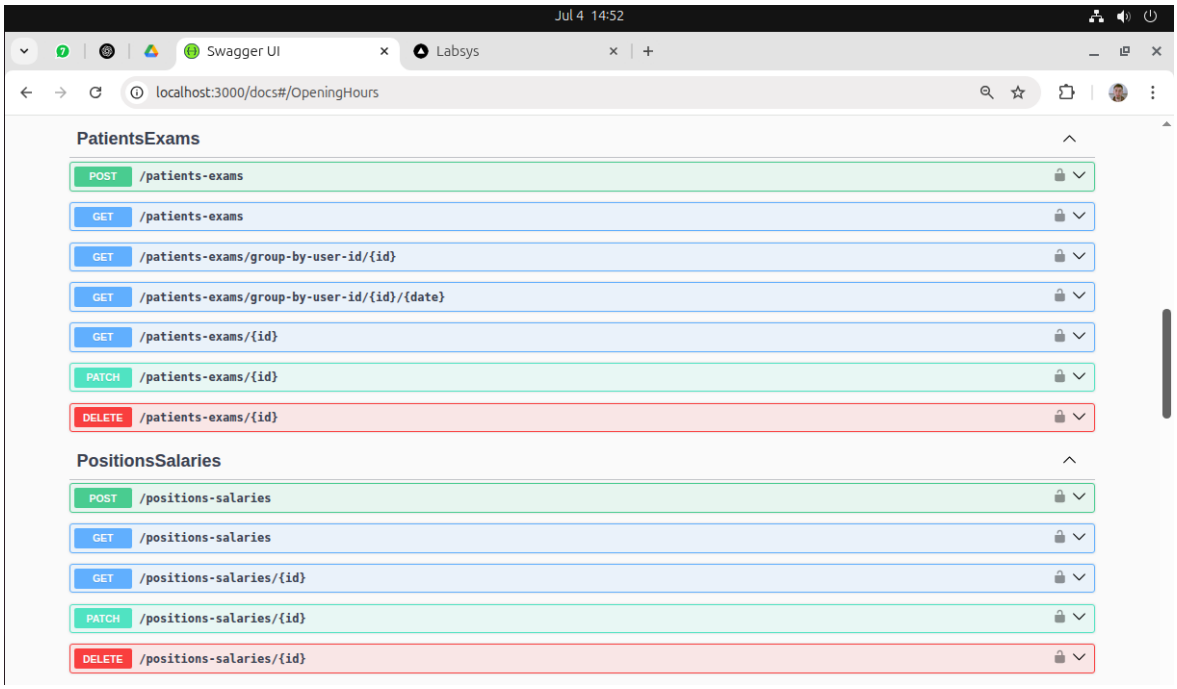
REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



(b) Tela contendo as seções *Services* e *Expenses*




(c) Tela contendo as seções *PatientsExams* e *PositionsSalaries*

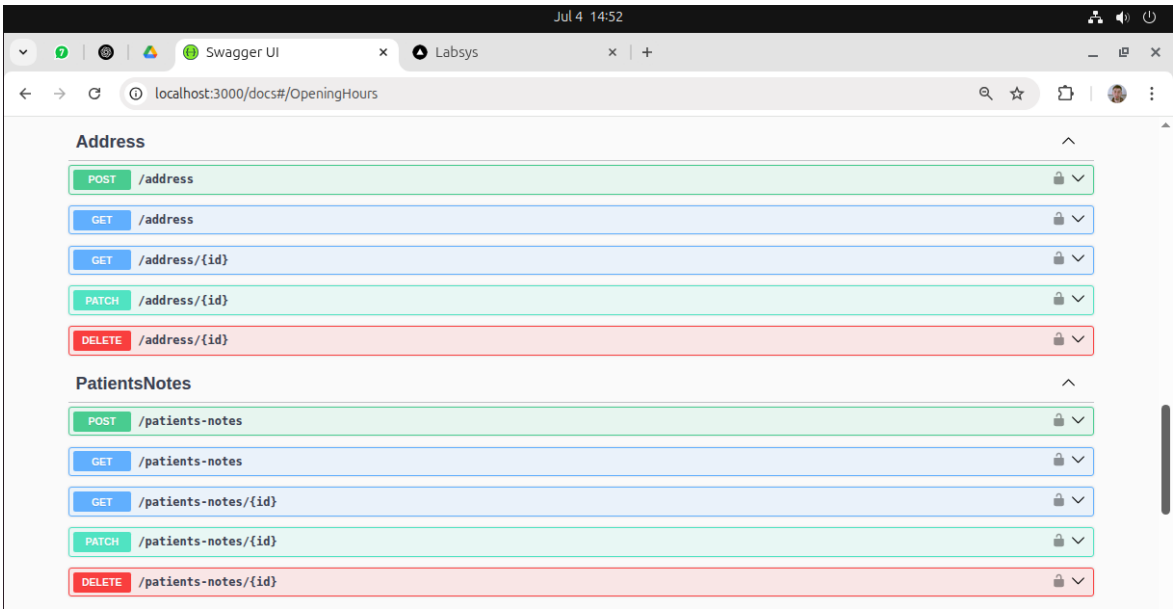



ISSN: 2675-6218 - RECIMA21

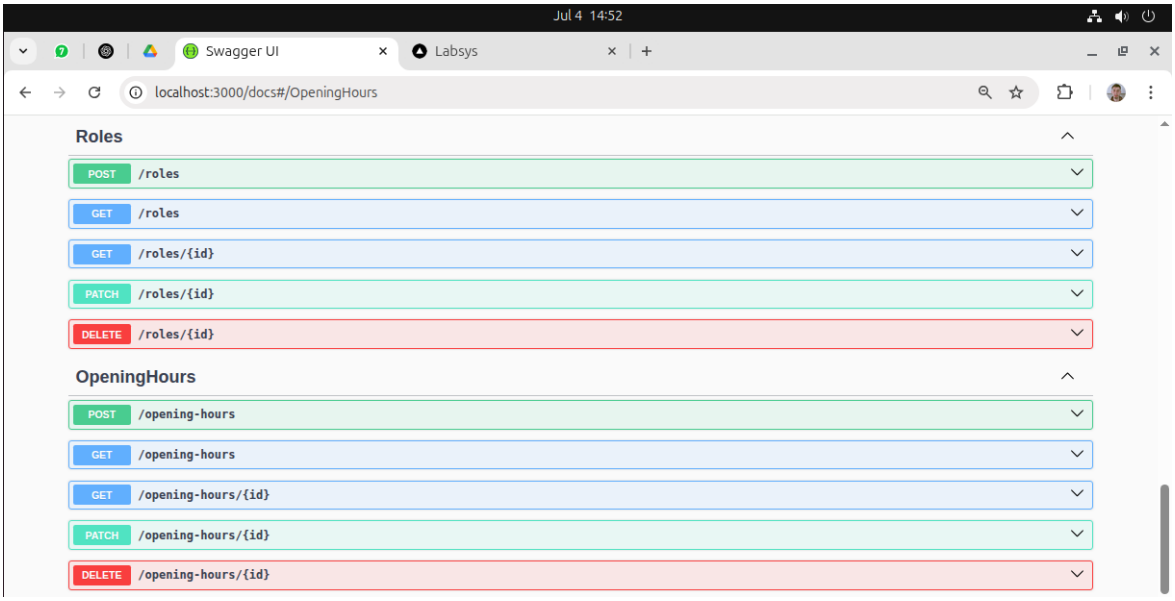
REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218



(d) Tela contendo as seções *Address* e *PatientsNotes*




(e) Tela contendo as seções *Roles* e *OpeningHours*




**Fonte:** Autoria própria

As imagens apresentadas demonstram que a *API* está devidamente estruturada, com rotas organizadas, descrições claras e suporte à autenticação via *JWT*, confirmando o alinhamento entre a documentação e as funcionalidades implementadas. Com isso, conclui-se que a *API* atende aos objetivos propostos, oferecendo uma base sólida e segura para a gestão de clínicas de saúde.

**ISSN: 2675-6218 - RECIMA21**

## 6. CONSIDERAÇÕES

O desenvolvimento da *API REST* apresentada neste trabalho buscou atender a uma demanda real de informatização de processos em uma clínica especializada em exames laboratoriais. Através da aplicação de boas práticas de engenharia de software e da utilização de tecnologias modernas como *Node.js*, *NestJS*, *TypeORM* e *PostgreSQL*, foi possível construir uma solução robusta, escalável e alinhada com as necessidades identificadas junto ao cliente.

A modelagem detalhada dos requisitos, aliada à representação visual por meio de diagramas e à adoção de padrões como *JSON Web Token (JWT)* e *Argon2*, contribuiu para a segurança, manutenibilidade e clareza na estrutura do sistema. Além disso, a geração automática da documentação com *Swagger* facilitou o entendimento da *API* por parte de desenvolvedores e validou a integridade das funcionalidades implementadas.

Os resultados demonstram que o sistema é capaz de centralizar operações essenciais da clínica, como gerenciamento de pacientes, exames, serviços, finanças e controle de horários, promovendo uma gestão mais eficiente e integrada.

## AGRADECIMENTOS

Agradecemos aos grupos LIS — Laboratório de Inovações em *Software* e LISA – Laboratório de Inovações em *Software* e Automação, pelo apoio, e à UFERSA pelo financiamento, por meio da Pró-Reitoria de Pesquisa e Pós-Graduação (PROPPG) através dos editais, PROPPG Nº12/2024, PROPPG Nº 21/2024 e PROPPG Nº 22/2024.

## REFERÊNCIAS

BARBOSA, F. J. B. **Ampliação e modernização do projeto meu-tcc**: uma abordagem para melhorar a eficiência e a usabilidade do back-end. 2023. Trabalho de Conclusão do Curso (Graduação em Tecnologias da Informação e Comunicação) – Universidade Federal de Santa Catarina, Campus Araranguá, SC. 2023.

BATISTA, S. S. **Desarrollo de un sistema de gestión de API's en la plataforma Moonshot.** [*S. l.: s. n.*], 2022. 67 p.

FELÍCIO, D. F. M. S. R. **RapiTest - Aplicação Web para testar API**. 2022. Dissertação (Mestrado) - Instituto Superior de Engenharia de Lisboa, Lisboa, 2022.

GODINHO, I. D. **Desenvolvimento de templates de aplicações com variantes node.js e reactjs**. 2024. Monografia (Graduação em Engenharia da Computação) - Instituto de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade, 2024.

GONÇALVES, W. O. **Recursos poderosos da linguagem de programação VBA utilizados em planilhas para gerenciamento financeiro**. 2023. Monografia (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Insitituto Federal de Educação, Ciência e Tecnologia de Goiás, Uruaçu, 2023.

GREGÓRIO, P.; GOYA, D. **Hash criptográfico sobre senhas e aleatoriedade do Argon2**. [*S. l.: s. n.*], 2019.

GUEDES, G. T. A. **UML 2:** Uma abordagem prática. 3. ed. São Paulo: Novatec, 2018. ISBN 978-8575226469.

MATA, A. S. da; LIMA, E. C. S.; MATA, A. S. da. Desenvolvimento de uma aplicação gamificada utilizando os frameworks Angular e NestJS para auxiliar no tratamento de crianças com TDAH. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 9, 2023.

MELO, T. R. S.; MORAIS, E. B. D.; SOUSA, R. R.; GONÇALVES, S. M. N. Utilizando o método pesquisa-ação para desenvolver um protótipo de um sistema para empréstimos de materiais para a UFERSA, Campus Pau dos Ferros. **Revista de Engenharia e Tecnologia**, v. 17, n. 1, 2025.

MENDES, M. S. **Proposta de nova implementação do sistema online de distribuição de disciplinas**. 2022. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Federal de Uberlândia, Uberlândia, 2022.

NUNES, G. N. As vantagens do Node.js. **REFAQI - Revista de Gestão, Educação e Tecnologia**, v. 4, n. 2, p. 2, dez. 2018.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **About PostgreSQL**. [*S. l.: s. n.*], 2023. Disponível em: https://www.postgresql.org/about/. Acesso em: 6 jul. 2025.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software.** 9. ed. Porto Alegre: AMGH, 2021. ISBN 978-6558040101.

SANTOS, Á. G. G. dos; EBRAM, V. E. C.; SILVA, V. R. B. da. **Sistema para clínica médica**. [S. l.]: Repositório Institucional do Conhecimento – RIC-CPS, 2022.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. **Educação e Pesquisa**, v. 31, p. 443-466, 2005.

VANTICO. JSON Web Token e a importância do Tempo de Expiração. **VANTICO**, 17 jan. 2025. Disponível em: https://vantico.com.br/json-web-token-jwt/. Acesso em: 6 jul. 2025.

**ISSN: 2675-6218 - RECIMA21**