UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

ROBSON MESQUITA GOMES

MORPHEUS HEALTHSYSTEM: IMPLEMENTAÇÃO DE API RESTFUL PARA TELEMEDICINA E DIAGNÓSTICO DE DISTÚRBIOS DO SONO COM MACHINE LEARNING

SOBRAL

2025

ROBSON MESQUITA GOMES

MORPHEUS HEALTHSYSTEM: IMPLEMENTAÇÃO DE API RESTFUL PARA TELEMEDICINA E DIAGNÓSTICO DE DISTÚRBIOS DO SONO COM MACHINE LEARNING

Trabalhoo de Conclusão de Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Me. David Nascimento Coelho

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G617m Gomes, Robson Mesquita.
MORPHEUS HEALTHSYSTEM: IMPLEMENTAÇÃO DE API RESTFUL PARA TELEMEDICINA E DIAGNÓSTICO DE DISTÚRBIOS DO SONO COM MACHINE LEARNING / Robson Mesquita Gomes. – 2025.
62 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Prof. Me. David Nascimento Coelho.

1. API de saúde. 2. Telemedicina. 3. Bruxismo do Sono. I. Título.

CDD 621.39

ROBSON MESQUITA GOMES

MORPHEUS HEALTHSYSTEM: IMPLEMENTAÇÃO DE API RESTFUL PARA TELEMEDICINA E DIAGNÓSTICO DE DISTÚRBIOS DO SONO COM MACHINE LEARNING

Trabalhoo de Conclusão de Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 05 de Agosto de 2025

BANCA EXAMINADORA

Prof. Me. David Nascimento Coelho (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Iális Cavalcante de Paula Júnior
Universidade Federal do Ceará (UFC)

Prof. Francisco Gean Dias da Silva Filho
Universidade Federal do Ceará (UFC)

À minha família, por acreditar que a educação era o caminho certo.

AGRADECIMENTOS

Agradeço ao Prof. Me. David Nascimento Coelho pela orientação dedicada ao longo deste trabalho de conclusão de curso, bem como pelo apoio constante durante minha jornada acadêmica em busca de conhecimento.

Ao Prof. Me. Rômulo Nunes de Carvalho Almeida, por me apresentar ao projeto que serviu de base para este trabalho e por sua inspiração e incentivo.

Ao Mestrando Francisco Gean Dias da Silva Filho, por ampliar minha visão sobre como a Engenharia pode ser aplicada em benefício da saúde humana.

Aos colegas e amigos Ismael Johnny Marques Ferreira e Waklio Xavier Neto do curso de Engenharia da Computação, Christian Farias de Oliveira do curso de Ciências Econômicas e Augusto Angelo Silva Mesquita do curso de Engenharia Elétrica da Universidade Federal do Ceará, pela parceria na equipe de desenvolvimento deste projeto e pelo apoio mútuo durante essa caminhada.

À minha amiga Me. Mariana Teixeira de Castro, pelo cuidado nas revisões deste trabalho e pela colaboração em diversos momentos ao longo do curso.

Ao Prof. Dr. Iális Cavalcante de Paula Júnior e à T.A.E. Michelle Fontenele, que, nos meus primeiros passos na universidade, atuando na coordenação do curso, me acolheram e me ajudaram a compreender aspectos fundamentais da vida acadêmica, em um momento em que tudo ainda era novo para mim.

E, com especial gratidão, à minha família, por acreditarem que a educação seria o melhor caminho e por fazerem não apenas o possível, mas também o que parecia impossível, para que eu pudesse continuar trilhando essa jornada.

“Milagres só acontecem com as pessoas que não desistem.”

(Eiichiro Oda)

RESUMO

A crescente demanda por soluções tecnológicas na área da saúde destaca a importância de ferramentas que facilitem o diagnóstico e o tratamento de doenças. Distúrbios como o bruxismo do sono afetam milhões de pessoas, com implicações significativas na saúde e qualidade de vida. Contudo, os métodos atuais de diagnóstico são frequentemente caros, demorados e inacessíveis. Este trabalho foca no desenvolvimento de uma API RESTful como infraestrutura central para uma plataforma inovadora, destinada à coleta, manutenção e processamento de dados clínicos de distúrbios do sono. A API foi projetada para ser o núcleo de integração entre diferentes componentes do ecossistema, como aplicações web, mobile e hardware de coleta, estabelecendo a base técnica para a futura implementação de modelos de pré-diagnóstico baseados em Inteligência Artificial. Ao criar uma arquitetura robusta e escalável, o projeto viabiliza a futura incorporação de análises automáticas, promove a interoperabilidade e alinha-se às demandas por tecnologias de saúde integradas e de baixo custo, buscando impactar positivamente a vida de milhões de pessoas.

Palavras-chave: API de saúde; Telemedicina; Bruxismo do sono

ABSTRACT

The growing demand for technological solutions in healthcare highlights the importance of tools that facilitate the diagnosis and treatment of diseases. Disorders such as sleep bruxism affect millions of people, with significant implications for health and quality of life. However, current diagnostic methods are often expensive, time-consuming, and inaccessible. This work focuses on the development of a RESTful API as the core infrastructure for an innovative platform, designed for the collection, maintenance, and processing of clinical data on sleep disorders. The API was designed to be the integration hub between different components of the ecosystem, such as web and mobile applications, and data collection hardware, establishing the technical foundation for the future implementation of pre-diagnostic models based on Artificial Intelligence. By creating a robust and scalable architecture, the project enables the future incorporation of automated analyses, promotes interoperability, and aligns with the demand for integrated, low-cost health technologies, aiming to positively impact the lives of millions.

Keywords: Health API; Telemedicine; Sleep bruxism.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* / Interface de programação de aplicativos
CRUD	*Create, Read, Update & Delete*
UML	*Unified Modeling Language* / Linguagem Unificada de Modelagem
ATM	Articulação Temporomandibular
BD	Bruxismo Diurno
BS	Bruxismo do Sono
CNS	Cartão Nacional de Saúde
CPF	Cadastro de Pessoa Física
DMPM	Distúrbio do Movimento Periódico das Pernas
EHR	*Electronic Health Record*
FHIR	*Fast Healthcare Interoperability Resources*
HTML	*Hypertext Markup Language*
HTTP	*Hiper Text Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IB	Índice de Bruxismo
JSON	*JavaScript Object Notation*
LGPD	Lei Geral de Proteção de Dados
REST	Representational State Transfer
SIS	Sistemas de Informação em Saúde
URI	Uniform Resource Identifiers
XML	*EXtensible Markup Language*

LISTA DE SÍMBOLOS

D	Deslocamento Elétrico
E	Campo elétrico
e	Constante de Stress
f	Força
k	Constante piezoelétrica
q	Carga
S	Tensão Longitudinal Aplicada
ε	Permissividade Elétrica

SUMÁRIO

1 INTRODUÇÃO

Os distúrbios do sono representam um problema de saúde pública global, afetando a qualidade de vida de milhões de pessoas em todo o mundo (FELLENDORF *et al.*, 2021). Entre os distúrbios mais comuns, o bruxismo do sono se destaca por suas consequências negativas para a saúde bucal e articular (ALMEIDA, 2016). Caracterizado pelo ranger ou apertar dos dentes durante o sono (YAP; CHUA, 2016), o bruxismo pode levar a diversos problemas, como dor de cabeça, dor na mandíbula, desgaste excessivo dos dentes e disfunção da Articulação Temporomandibular (ATM) (CARLSON; MCNAMARA, 2013; MACHADO *et al.*, 2015).

O diagnóstico do bruxismo do sono geralmente se baseia na história clínica do paciente e no exame físico, realizado por um dentista ou profissional especializado em distúrbios do sono (BRUXISM, 2021). No entanto, esses métodos podem apresentar limitações, como subjetividade e imprecisão (ALMEIDA, 2016). Essa dificuldade no diagnóstico preciso pode atrasar o início de um tratamento adequado, perpetuando os sintomas e suas consequências negativas para a saúde do indivíduo (ANDRADE *et al.*, 2018).

Diante dos desafios relacionados ao diagnóstico do bruxismo do sono, o desenvolvimento de uma ferramenta objetiva e confiável se torna crucial para auxiliar os profissionais de saúde. Essa ferramenta idealmente deveria ser capaz de identificar com precisão a presença do bruxismo, classificar sua severidade e fornecer informações adicionais que auxiliem no planejamento do tratamento individualizado (GONÇALVES *et al.*, 2015; MACHADO *et al.*, 2017).

A implementação de uma ferramenta de diagnóstico objetiva para o bruxismo do sono traria diversos benefícios, tanto para os profissionais de saúde quanto para os pacientes. Para os profissionais, a ferramenta permitiria um diagnóstico mais preciso e confiável, facilitando a escolha do tratamento mais adequado para cada caso (MACHADO *et al.*, 2017). Isso poderia levar a um melhor controle dos sintomas, à redução da dor e à melhora da qualidade de vida dos pacientes (ANDRADE *et al.*, 2018).

Para desenvolver essa ferramenta, um passo muito importante é o desenvolvimento de uma *Application Programming Interface* / Interface de programação de aplicativos (*API*) de forma a integrar um ecossistema de softwares e equipamentos de *hardware* que, por meio da telemedicina, possibilite a realização de exames remotamente.

1.1 Objetivos

O objetivo deste trabalho é desenvolver uma *API* de saúde para uma plataforma abrangente destinada à coleta, manutenção e processamento de dados clínicos relacionados ao bruxismo do sono. A partir deste objetivo geral, foram determinados os seguintes objetivos específicos:

- Permitir a integração e interação eficientes entre diferentes componentes da plataforma por meio da *API* desenvolvida.
- Estruturar a *API* para receber, armazenar e disponibilizar os dados clínicos e de anamnese necessários ao processo de pré-diagnóstico do Bruxismo do Sono (BS).
- Projetar a *API* para permitir a futura integração de modelos de Inteligência Artificial, validando o fluxo de dados com um módulo de análise simulado (Mock).
- Aplicar princípios de interoperabilidade no design da *API*, como a arquitetura RESTful e a documentação com OpenAPI, visando a futura conformidade com padrões de mercado como o *Fast Healthcare Interoperability Resources* (FHIR).

1.2 Justificativa

No cenário da saúde moderna, a busca por soluções inovadoras e eficientes se torna cada vez mais imperativa. A crescente complexidade das doenças, o aumento da expectativa de vida da população e a necessidade de otimizar os recursos disponíveis impulsionam a busca por ferramentas e tecnologias que auxiliem na prevenção, diagnóstico e tratamento de doenças.

Nesse contexto, a construção de plataformas de dados e informações para dispositivos médicos emerge como um tema de grande relevância (FELLENDORF *et al.*, 2021). Essas plataformas, ao permitirem a coleta, integração e análise de informações relacionadas à demonstração, avaliação e aplicação de projetos científicos e tecnológicos, desempenham um papel muito importante para a inovação e o desenvolvimento da indústria de dispositivos em saúde (WANG, 2021).

A relevância dessas plataformas reside em sua capacidade de oferecer suporte estratégico para o desenvolvimento de dispositivos médicos de alta qualidade e competitivos no mercado. Por meio da análise de dados coletados, é possível identificar oportunidades para aprimorar os dispositivos existentes, desenvolver novas tecnologias e otimizar os processos de produção e distribuição (COMMISSION, 2020).

Além disso, essas plataformas facilitam a colaboração e a interconexão entre as diversas partes interessadas na cadeia de valor da indústria de dispositivos médicos, como pesquisadores, empresas, profissionais de saúde e órgãos reguladores. Essa colaboração permite a troca de conhecimentos, experiências e recursos, acelerando o desenvolvimento de novas soluções e otimizando a utilização dos dispositivos existentes (GONÇALVES *et al.*, 2015).

O bruxismo do sono, distúrbio caracterizado pelo ranger ou apertar dos dentes durante o sono, afeta milhões de pessoas em todo o mundo, causando diversos problemas de saúde bucal e articular (ALMEIDA, 2016). Estima-se que cerca de 8% da população adulta sofra com o bruxismo, com prevalência ainda maior entre crianças e adolescentes (MACHADO *et al.*, 2015).

O diagnóstico preciso e precoce do bruxismo é fundamental para a implementação de um tratamento eficaz, minimizando seus impactos negativos na qualidade de vida dos pacientes. As principais técnicas de diagnóstico utilizadas atualmente incluem exames clínicos, questionários e monitoramento do sono (BRUXISM, 2021). No entanto, essas técnicas podem apresentar limitações, como subjetividade, imprecisão e alto custo (ALMEIDA, 2016).

É nesse contexto que surgiu a proposta deste trabalho: a construção de uma *API* de integração de sistemas em saúde, com foco no pré-diagnóstico do Bruxismo do Sono (BS). Através dessa plataforma, pretende-se auxiliar auxiliar profissionais da saúde na identificação precoce de pacientes com potencial risco de bruxismo, otimizando o processo de diagnóstico e contribuindo para o bem-estar da população.

No trabalho de Wang (2021), os autores desenvolveram uma plataforma de dados para instrumentos médicos inovadores com base na computação de serviço, visando a integração e análise de informações para a indústria de dispositivos em saúde de forma ampla. Diferente deste, o presente trabalho foca na construção de uma *API* especializada para o pré-diagnóstico do Bruxismo do Sono (BS), utilizando uma abordagem de telemedicina para integrar *softwares*, *hardwares* de coleta e futuros modelos de inteligência artificial, oferecendo uma solução direcionada a um distúrbio específico.

A *API* será capaz de executar modelos com técnicas de inteligência artificial e aprendizado de máquina, em especial um modelo já projetado (DIAS FILHO, 2022). Dessa forma, a aplicação será capaz de identificar padrões e características que indicam a presença do distúrbio, fornecendo aos profissionais de saúde uma ferramenta auxiliar valiosa para o pré-diagnóstico.

Por fim, este trabalho, ao contribuir para a construção de uma plataforma em saúde com foco no pré-diagnóstico do bruxismo do sono, trará benefícios significativos para a área da saúde, impactando positivamente a vida de milhões de pessoas em todo o mundo.

Para uma melhor compreensão da estrutura deste documento, os capítulos subsequentes estão organizados da seguinte forma: o Capítulo 2 apresenta a **Fundamentação Teórica**, detalhando os conceitos sobre o bruxismo do sono, as tecnologias envolvidas como *API*s RESTful e os aspectos legais pertinentes. O Capítulo 3 descreve os **Materiais e Métodos**, abordando o planejamento, o levantamento de requisitos, a modelagem do banco de dados e a construção da *API*. No Capítulo 4, são expostos os **Resultados e Discussões**, com a apresentação da *API* implementada, os testes de integração e a simulação do sistema. Finalmente, o Capítulo 5 apresenta as **Conclusões e Trabalhos Futuros**, consolidando os resultados alcançados e sugerindo os próximos passos para a evolução do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Todo projeto de *software* exige uma base teórica sólida para o seu desenvolvimento. Quando se trata de desenvolvimento de Sistemas de Informação em Saúde (SIS) não é diferente. A exigência da precisão e auto desempenho são constantes em cada etapa do processo, pois nesse tipo de sistema são trafegados dados sensíveis de diversos pacientes.

Este capítulo apresenta uma revisão da literatura sobre o bruxismo, abrangendo conceitos, tipologias, fatores de risco, consequências, diagnóstico e tratamento. Além disso, serão abordados os aspectos tecnológicos relacionados à plataforma desenvolvida, como a piezoeletricidade e a modelagem *Unified Modeling Language* / Linguagem Unificada de Modelagem (*UML*), e os aspectos legais envolvidos, com foco na Lei Geral de Proteção de Dados (LGPD).

2.1 Bruxismo

O bruxismo, caracterizado pelo ranger ou apertar dos dentes de forma involuntária, principalmente durante o sono, é um distúrbio comum que afeta milhões de pessoas em todo o mundo (OKESON, 2008). Essa condição pode causar diversos danos à saúde bucal e articular, incluindo desgaste dos dentes, fraturas dentárias, dor muscular e disfunção da Articulação Temporomandibular (ATM) mostrada na Figura 1 (MELLONIG J. C., 2009; MACHADO *et al.*, 2015).

O bruxismo pode ser classificado em duas categorias principais: Bruxismo do Sono (BS) e o Bruxismo Diurno (BD). O BS ocorre durante o sono, geralmente sem a consciência do indivíduo. É a forma mais comum de bruxismo e pode ser desencadeada por diversos fatores, como estresse, ansiedade, distúrbios do sono e certos medicamentos (ALMEIDA, 2016; MACHADO *et al.*, 2015). Já o Bruxismo Diurno (BD) ocorre durante o dia, geralmente em momentos de concentração, raiva ou frustração. É menos comum que o BS, mas também pode causar danos aos dentes e à ATM (MACHADO *et al.*, 2015; GONÇALVES *et al.*, 2015).

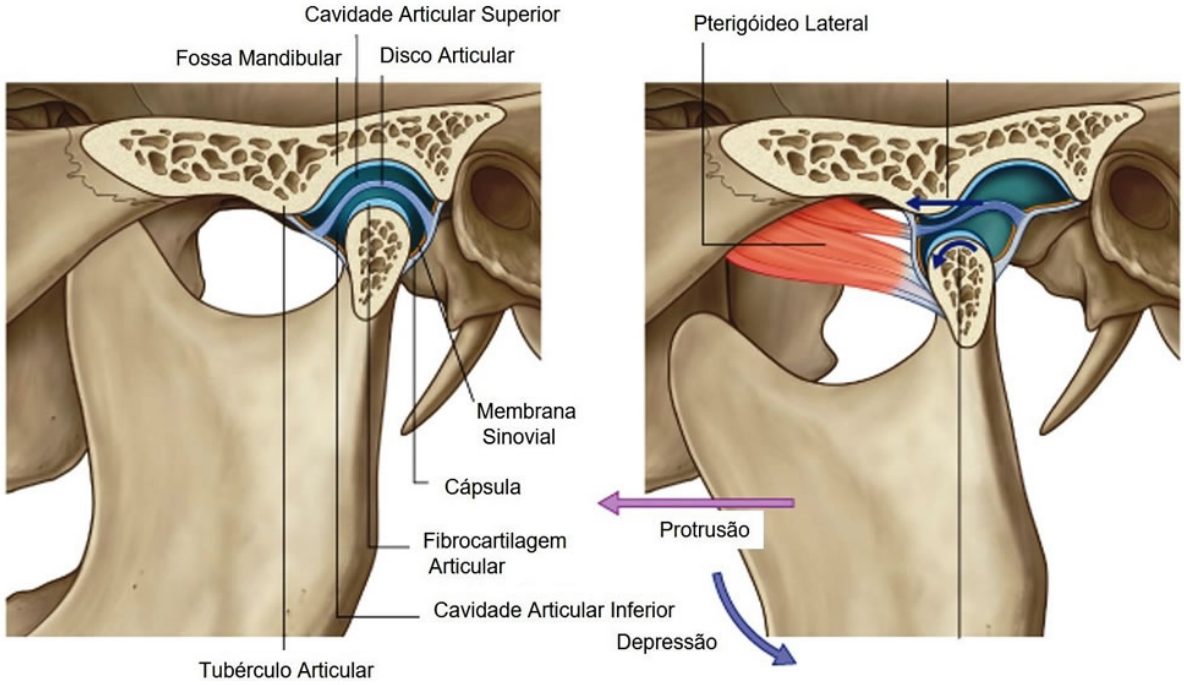
Em relação aos fatores de ricos, a etiologia[1] do bruxismo ainda não é completamente compreendida, mas segundo ALMEIDA (2016) diversos fatores podem aumentar o risco de desenvolver o distúrbio, incluindo:

- Genética: Estudos sugerem que o bruxismo pode ter uma predisposição genética.

[1] A **étiologia** é o ramo do conhecimento cujo objeto é a pesquisa e a determinação das causas e origens de um determinado fenômeno.

- Fatores Psicológicos: Estresse, ansiedade, depressão e transtornos de personalidade podem contribuir para o bruxismo.
- Distúrbios do Sono: Apneia do Sono[2], Distúrbio do Movimento Periódico das Pernas (DMPM)[3] e Narcolepsia[4] podem aumentar o risco de bruxismo.
- Medicamentos: Certos medicamentos, como antidepressivos e estimulantes, podem ser fatores desencadeantes do bruxismo.
- Fatores Oclusais: Má oclusão[5] dentária, mordida cruzada ou dentes apinhados podem contribuir para o bruxismo.

Figura 1 – Articulação Temporomandibular (ATM)



Fonte: Adaptado de https://www.auladeanatomia.com/sistemas/245/articulacao-temporo-mandibular

Ainda segundo ALMEIDA (2016) e MACHADO *et al.* (2015) o bruxismo pode causar diversos danos à saúde bucal e articular, incluindo:

- Desgaste dos Dentes: O ranger e o apertar dos dentes podem causar o desgaste da superfície dental, levando à sensibilidade, dor e até mesmo à perda de dentes.
- Fraturas Dentárias: O bruxismo pode aumentar o risco de fraturas dentárias, especialmente

[2] A **Apneia do Sono** é um distúrbio respiratório que provoca interrupções da respiração durante o sono. É uma doença grave que pode aumentar o risco de morte prematura.

[3] O **Distúrbio do Movimento Periódico das Pernas (DMPM)** é um transtorno do sono que se caracteriza por movimentos repetitivos nas pernas ou braços durante o sono.

[4] A **Narcolepsia** é uma doença neurológica crônica que provoca sonolência excessiva e episódios de sono involuntário. É um distúrbio do sono que afeta a capacidade de controlar os ciclos de sono-vigília.

[5] **Oclusão** é ato ou efeito de fechar, cerramento.

em pacientes com dentes enfraquecidos ou restaurações dentárias.

- Dor Muscular: O bruxismo pode causar dor nos músculos da face, mandíbula e pescoço, além de dores de cabeça.
- Disfunção da ATM: O bruxismo pode levar à disfunção da ATM, caracterizada por dor, estalos articulares, dificuldade de abrir e fechar a boca e tinnitus (zumbido nos ouvidos).

Geralmente, o diagnóstico do bruxismo é feito por meio de uma avaliação clínica, que inclui exame físico, histórico médico e questionários específicos. Em alguns casos, exames complementares, como polissonografia (para avaliar o sono) ou radiografias da ATM, podem ser necessários (ALMEIDA, 2016; MACHADO *et al.*, 2015).

O exame físico busca identificar sinais clínicos do bruxismo, como desgaste dental, hipersensibilidade dentária, dor muscular facial, estalos na ATM e bruxismo durante o sono. O profissional de saúde também pode palpar os músculos da face e da mandíbula para verificar a presença de dor ou tensão muscular.

Já o histórico médico é fundamental para identificar possíveis fatores de risco para o bruxismo, como estresse, ansiedade, depressão, distúrbios do sono, uso de medicamentos e histórico familiar de bruxismo.

Além disso, questionários específicos sobre bruxismo, como o Índice de Bruxismo (IB), podem ser utilizados para avaliar a severidade do distúrbio e identificar os padrões de bruxismo.

Por fim, em alguns casos, exames complementares podem ser necessários para confirmar o diagnóstico do bruxismo ou para avaliar a extensão do distúrbio. Os exames complementares mais utilizados incluem:

- Polissonografia: A polissonografia é um exame que monitora o sono do paciente durante a noite. O exame pode detectar episódios de bruxismo durante o sono e ajudar a identificar os fatores que desencadeiam o distúrbio.
- Radiografias da ATM: As radiografias da ATM podem mostrar danos à articulação causados pelo bruxismo, como desgaste da cartilagem ou deformidades ósseas.

A partir do diagnóstico, o tratamento do bruxismo depende da causa subjacente e da severidade dos sintomas. As opções de tratamento podem incluir:

- Terapia Comportamental: Técnicas de relaxamento, *biofeedback* e terapia cognitivo-comportamental podem ajudar a reduzir o bruxismo, especialmente em casos leves ou moderados.

- Placas de Mordida: Placas de acrílico personalizadas podem ser utilizadas para proteger os dentes e a ATM durante o sono. As placas de mordida podem ser eficazes para reduzir o desgaste dental e a dor muscular, mas não tratam a causa subjacente do bruxismo.
- Toxina Botulínica: A toxina botulínica pode ser injetada nos músculos da face e da mandíbula para relaxá-los e reduzir o bruxismo. Essa opção de tratamento é geralmente reservada para casos graves de bruxismo que não respondem a outras terapias.
- Tratamento Odontológico: Em alguns casos, o bruxismo pode causar danos aos dentes que necessitam de tratamento odontológico, como restaurações, coroas ou implantes dentários.
- Tratamento de Distúrbios do Sono: Se o bruxismo estiver relacionado a um distúrbio do sono, como apneia do sono, o tratamento do distúrbio do sono pode ajudar a reduzir os episódios de bruxismo.
- Medicamentos: Em alguns casos, medicamentos podem ser utilizados para controlar o bruxismo, especialmente se o distúrbio estiver relacionado à ansiedade ou estresse.

A seguir serão evidenciadas as tecnologias envolvidas na elaboração deste trabalho.

2.2 Piezoeletricidade

O fator primordial de um sistema de informação são os dados. Conhecer a origem dos dados auxilia o projetista de sistemas a otimizar sua utilização. No caso do bruxismo, deve-se detectar as vibrações causadas pelos movimentos dos musculos que movimentam a ATM. Uma das formas de detectar estas vibrações é através da piezoeletricidade.

A piezoelectricidade é uma combinação de efeitos do comportamento elétrico do material (REZENDE, 2004), que envolve a relação entre propriedades elétricas e mecânicas. Sua equação fundamental (Adaptada de Rezende (2004)) é:

$$D = \varepsilon E + eS, \tag{2.1}$$

onde D é o deslocamento elétrico, ε é a permissividade elétrica, E representa o campo elétrico, e representa a constante de stress e S é a tensão longitudinal aplicada.

A piezoeletricidade é um fenômeno amplamente utilizado em diversas aplicações práticas, incluindo a produção e detecção de som, a geração de tensões elevadas, a criação de frequências eletrônicas, o uso em microbalanças e a concentração ultrafina de conjuntos ópticos. Para observar as iterações piezoelétricas precisa-se de sensores específicos..

2.2.1 Sensor Piezoelétrico

Os sensores piezoelétricos são transdutores que medem parâmetros físicos, como tensão mecânica, convertendo-os em variações de cargas elétricas (KOUR *et al.*, 2021). Suas aplicações são vastas, incluindo o monitoramento não invasivo da pressão cardíaca (ZHANG *et al.*, 2024), a geração de ultrassom para visualização de órgãos (ZHOU *et al.*, 2014) e a detecção de ondas sonoras em microfones (KADLEC *et al.*, 2024). O princípio de funcionamento baseia-se no efeito piezoelétrico direto, onde a aplicação de uma tensão mecânica gera um potencial elétrico, e no efeito inverso, onde a aplicação de uma tensão elétrica provoca uma deformação física (REZENDE, 2004).

A carga induzida q em um material piezoelétrico é diretamente proporcional à força f aplicada, o que pode ser representado pela equação (Adaptada de Rezende (2004) e HBK (2025)):

$$q = kf, \tag{2.2}$$

onde k é uma constante piezoelétrica do material, que quantifica a sensibilidade do sensor (HBK, 2025).

Materiais piezoelétricos têm resistência de isolamento elevada, mas não infinita. Essa característica os torna inadequados para medições verdadeiramente estáticas, pois quando uma força constante é aplicada, a carga gerada vaza lentamente através da resistência interna do sensor e do circuito de medição. Como resultado, o sinal elétrico decai exponencialmente, tornando os sensores piezoelétricos ideais para a medição de fenômenos dinâmicos ou que variam no tempo (Wikipedia contributors, 2025).

2.3 Legislação e LGPD

Após a definição da origem e a aquisição de dados, a segurança destes é uma questão de grande importância a ser considerada na construção de qualquer sistema de informação. Por conta dos riscos inerentes à possíveis falhas nesses sistemas alguns países desenvolveram sua própria legislação para evitar que seus cidadãos não tenham seus direitos violados.

No Brasil temos a Lei Geral de Proteção de Dados (LGPD) (BRASIL, 2018) que arbitra sobre as condições e funcionalidades mínimas a serem oferecidas por um sistema ou empresa baseada em seus sete fundamentos.

> Art. 2º A disciplina da proteção de dados pessoais tem como fundamentos:
> I - o respeito à privacidade;
> II - a autodeterminação informativa;
> III - a liberdade de expressão, de informação, de comunicação e de opinião;
> IV - a inviolabilidade da intimidade, da honra e da imagem;
> V - o desenvolvimento econômico e tecnológico e a inovação;
> VI - a livre iniciativa, a livre concorrência e a defesa do consumidor; e
> VII - os direitos humanos, o livre desenvolvimento da personalidade, a dignidade e o exercício da cidadania pelas pessoas naturais.
> (BRASIL, 2018)

2.4 *Unified Modeling Language* / Linguagem Unificada de Modelagem (*UML*)

Uma parte fundamental para construção de sistemas complexos de software é o planejamento. Nessa etapa, erros são previstos e riscos são mitigados antes que exista a possíbilidade de causar algum prejuízo. Para planejar de forma precisa um sistema é necessário visualizá-lo antes de construí-lo e, durante sua construção, possuir meios de explicar o que deve ser construído para todos os desenvolvedores envolvidos. Nesse contexto a *UML* se torna uma ferramenta fundamental.

Segundo Larman (2004), a *UML* surge como uma ferramenta poderosa para a modelagem de sistemas de *software*, facilitando a comunicação entre desenvolvedores, analistas e outros *stakeholders*. Esta linguagem gráfica oferece um conjunto abrangente de diagramas para representar diversos aspectos do sistema, desde a estrutura e comportamento das classes até a interação entre os componentes e o fluxo de dados.

Através da *UML*, pode-se:

- Visualizar a arquitetura do sistema: Diagramas de classes, componentes e implantação permitem visualizar a estrutura do sistema, seus componentes, suas relações e como eles se distribuem em diferentes ambientes.
- Modelar o comportamento do sistema: Diagramas de sequências, atividades e casos de uso detalham o comportamento do sistema sob diferentes situações, desde a interação do usuário até o fluxo de dados entre os componentes.
- Documentar o sistema: A *UML* fornece uma documentação visual clara e concisa do sistema, facilitando a compreensão e a manutenção do mesmo.

2.5 *Application Programming Interface* / Interface de programação de aplicativos (*API*)

As *API*s são costumeiramente utilizadas para a conectividade de sistemas de informação. Esta sigla refere-se a um conjunto de regras e protocolos que permitem a comunicação entre diferentes sistemas de software. Em termos simples, uma *API* é uma interface que define como as funcionalidades de um *software* podem ser acessadas e utilizadas por outras aplicações. De acordo com FIELDING (2000), uma *API* pode ser vista como um intermediário que conecta usuários e sistemas, proporcionando uma interação estruturada e eficiente.

*API*s desempenham um papel fundamental no desenvolvimento moderno de software, pois permitem a criação de sistemas escaláveis e modulares. No contexto deste trabalho, uma *API* será usada para integrar e facilitar a comunicação entre os componentes de uma plataforma de saúde, permitindo o processamento e análise de dados clínicos de forma eficiente e segura. Essa abordagem reforça a importância das *API*s como facilitadoras da interoperabilidade entre sistemas distintos.

A utilização de padrões amplamente aceitos, como o *RESTful*, promove uma melhor padronização e uma curva de aprendizado mais suave para desenvolvedores, como indicado por Fielding (2000). Isso torna as APIs não apenas essenciais para a funcionalidade técnica, mas também estratégicas para o sucesso e a escalabilidade dos sistemas.

2.6 *RESTful*

RESTful refere-se a uma implementação de *API* baseada no estilo arquitetural Representational State Transfer (REST), proposto por Fielding (2000). Este estilo define um conjunto de restrições arquiteturais que enfatizam a escalabilidade, a simplicidade e a independência entre cliente e servidor. *API* RESTful são projetadas para permitir que diferentes sistemas interajam de forma eficiente por meio de chamadas *Hiper Text Transfer Protocol* (HTTP) padrão.

Um dos principais benefícios de usar *API*s RESTful é sua capacidade de suportar sistemas distribuídos de maneira escalável e flexível, facilitando a interoperabilidade. Essas *API*s utilizam recursos, que são identificados por Uniform Resource Identifierss (URIs), e manipulam representações desses recursos, geralmente no formato *JavaScript Object Notation* (JSON) ou *EXtensible Markup Language* (XML).

No contexto deste trabalho, a implementação RESTful será utilizada para garantir a padronização da *API* proposta, promovendo a facilidade de integração e a adoção de boas

práticas para o desenvolvimento de sistemas modernos.

2.7 Documentação de APIs RESTful

A documentação de uma API RESTful é o registro técnico de sua interface, detalhando recursos (*endpoints*), métodos HTTP, parâmetros de entrada/saída e formatos de dados. Em essência, trata-se de um contrato de comunicação entre sistemas, fornecendo aos desenvolvedores instruções claras sobre como utilizar cada serviço exposto (Document360, 2023; Amazon Web Services, 2024). Tal documentação especifica quais dados enviar em cada chamada e quais respostas esperar, funcionando como referência essencial para implementar e integrar componentes distintos.

No contexto de desenvolvimento distribuído (por exemplo, arquiteturas de microsserviços ou ambientes integrados), a documentação de API é fundamental para permitir que equipes independentes desenvolvam e integrem módulos de forma compatível, sem precisar conhecer a implementação interna umas das outras. Ela facilita o desacoplamento dos componentes e assegura interoperabilidade entre eles, pois todos se baseiam no contrato documentado. Nesse cenário, destaca-se o uso da especificação **OpenAPI**: trata-se de um padrão que descreve APIs RESTful de forma padronizada e legível tanto para humanos quanto para máquinas (Programae.dev, 2023b; Programae.dev, 2023a). A adoção do OpenAPI permite gerar automaticamente documentação interativa (por exemplo, via Swagger UI) e ferramentas de teste, promovendo consistência no processo.

2.8 *Hiper Text Transfer Protocol* (HTTP)

A integração entre sistemas é baseada em protocolos de comunicação. O *Hiper Text Transfer Protocol* (HTTP) é um protocolo de comunicação amplamente utilizado na internet para a transferência de informações entre clientes e servidores. Introduzido inicialmente em 1991, o HTTP foi projetado para ser simples e extensível, permitindo a troca de mensagens em formato texto e binário, como HTML, JSON, imagens e vídeos.

O protocolo HTTP segue um modelo baseado em requisições e respostas. O cliente, geralmente um navegador ou uma aplicação, envia uma requisição ao servidor, que responde com os dados solicitados. Estas mensagens incluem métodos específicos como `GET`, usado para recuperar informações, e `POST`, usado para enviar dados ao servidor. A Tabela 1 mostra os

métodos HTTP e suas funções.

No desenvolvimento de *API*s, o HTTP serve como base para a comunicação entre sistemas. Ele fornece recursos fundamentais, como cabeçalhos para metadados, códigos de status para indicar o resultado de operações e a capacidade de manipular conexões seguras usando *Hypertext Transfer Protocol Secure* (HTTPS).

Por sua robustez e ubiquidade, o HTTP é uma escolha natural para a construção de sistemas distribuídos e é essencial para a interoperabilidade em ambientes modernos (FIELDING *et al.*, 1999).

Tabela 1 – Métodos HTTP com suas descrições e usos principais

Método	**Descrição**	**Uso Principal**
`GET`	Recupera informações de um recurso especificado pelo URI	Obtenção de dados, como páginas HTML ou APIs
`POST`	Envia dados ao servidor para criar ou processar um recurso	Envio de formulários, criação de novos recursos
`PUT`	Atualiza ou substitui completamente o recurso no servidor	Atualização completa de informações
`DELETE`	Remove o recurso especificado no URI	Exclusão de recursos
`PATCH`	Aplica modificações parciais a um recurso	Atualização parcial de informações
`HEAD`	Recupera os cabeçalhos de um recurso sem o corpo da resposta	Verificação de informações sobre o recurso
`OPTIONS`	Obtém as opções de comunicação suportadas por um recurso	Descoberta de métodos suportados
`TRACE`	Realiza uma verificação de loop entre cliente e servidor	Depuração de conexões HTTP

Fonte: Adaptado de FIELDING *et al.* (1999) e DOCS (2023).

2.9 Uso de *API*s em Saúde

A utilização de *API*s tem se tornado uma ferramenta essencial no setor de saúde, especialmente devido à necessidade crescente de integração entre diferentes sistemas e plataformas. *API*s permitem a comunicação entre dispositivos, aplicações e bancos de dados, facilitando o compartilhamento de informações clínicas de forma segura e eficiente. A interoperabilidade é um dos maiores desafios na área da saúde, e as *API*s têm se mostrado soluções eficientes para superar essa barreira, permitindo que sistemas como prontuários eletrônicos (*Electronic Health Record* (EHR)) se comuniquem com outras aplicações e plataformas externas.

Uma das principais inovações que surgiram foi o padrão FHIR, desenvolvido para

melhorar a interoperabilidade dos sistemas de saúde. Esse padrão utiliza *API*s RESTful 2.6, permitindo que dados de saúde, como informações de pacientes, resultados de exames e históricos clínicos, sejam acessados e compartilhados de forma mais eficiente, promovendo uma troca rápida e segura de informações entre sistemas diversos (FOUNDATION, 2022).

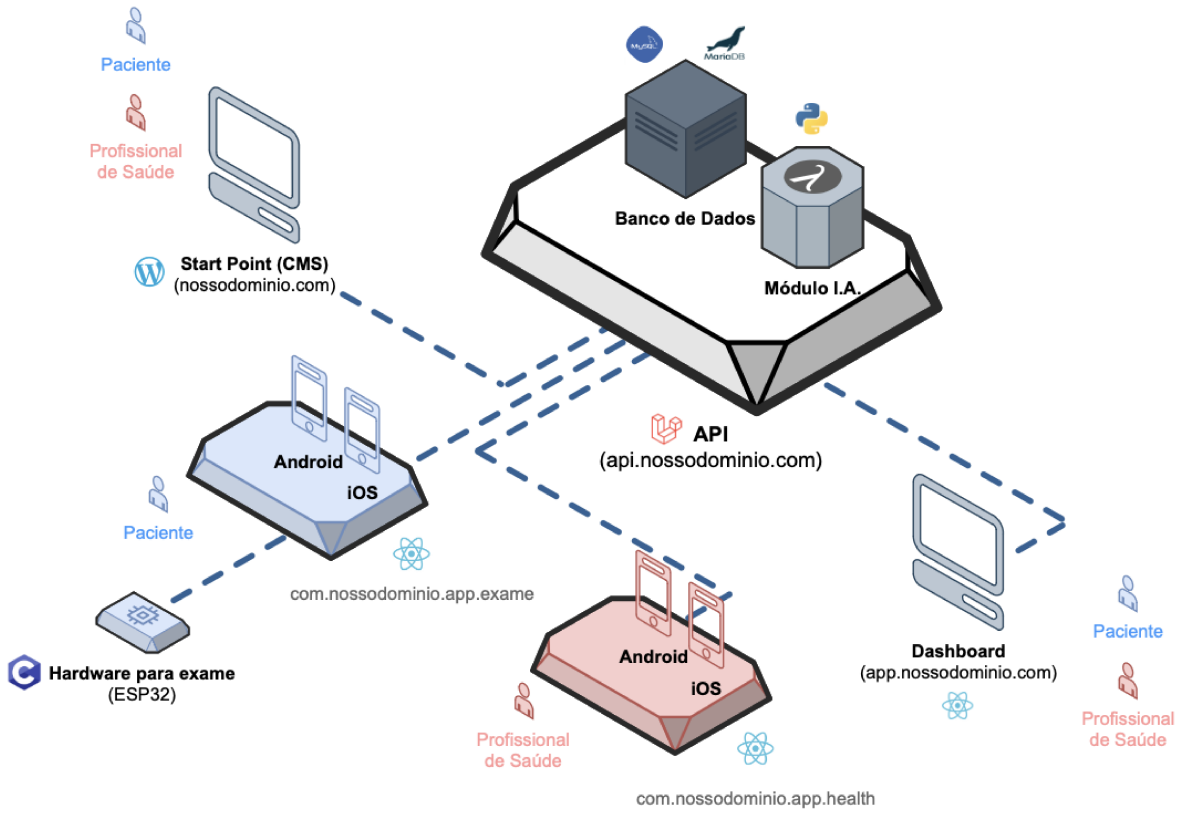
Em resumo, as APIs desempenham um papel fundamental no avanço da digitalização dos cuidados de saúde, promovendo a integração e a eficiência no uso de dados clínicos. Elas são ferramentas-chave não apenas para a modernização dos sistemas de saúde, mas também para a criação de soluções inovadoras que podem transformar a experiência dos pacientes e dos profissionais da saúde.

3 MATERIAIS E MÉTODOS

Com o conhecimento das estruturas e informações que permeiam o ambiente onde a aplicação será inserida se torna possível modelar uma solução para o problema proposto. A ideia do projeto, como citado várias vezes ao longo desse trabalho, é criar uma API que centralizará as informações coletadas e possibilitará a integração de todos os equipamentos e *softwares* necessários a realização do exame proposto.

O ecossistema onde a aplicação está inserida é aparesentado apresentada na Figura 2.

Figura 2 – Arquitetura do sistema onde a API será inserida



Fonte: Produção do autor (Apendice A)

A Figura 2 apresenta a arquitetura do ecossistema Morpheus HealthSystem, projetada para ser modular e escalável. No centro da arquitetura está a API RESTful, que atua como o núcleo de integração, centralizando toda a comunicação entre os componentes do sistema. A *API* é responsável por mediar o fluxo de dados entre as diversas aplicações cliente e os serviços de *backend*, que compreendem o Banco de Dados e o Módulo de Inteligência Artificial (I.A.), onde os dados clínicos são processados. O sistema serve a dois tipos de usuários: o paciente

e o profissional de saúde. O paciente utiliza um *hardware* de coleta de dados para o exame (ESP32) e um aplicativo móvel para o gerenciamento do processo. Já o profissional de saúde tem acesso aos resultados e análises por meio de um *dashboard web* e de um aplicativo de saúde complementar. Essa estrutura demonstra como a API possibilita a integração coesa de softwares e equipamentos de hardware, viabilizando a realização de exames remotos e a futura aplicação de diagnósticos assistidos por inteligência artificial.

3.1 Planejamento

3.1.1 Estudo do Problema

O primeiro passo no desenvolvimento do sistema é um estudo aprofundado do problema a ser resolvido. Nesta etapa, o foco foi entender as necessidades dos usuários e as características dos distúrbios do sono que foram abordados pelo sistema. A análise dos requisitos clínicos e a definição das especificações do sistema foram realizadas com base em literatura científica atualizada, junto a uma revisão dos métodos existentes para diagnóstico de distúrbios do sono, como o Bruxismo do Sono. Este estudo proporcionou uma base sólida para o desenvolvimento do modelo de software que será capaz de processar e interpretar os dados clínicos de forma eficaz.

3.1.1.1 Levantamento de Requisitos

Além do estudo, foi realizado um levantamento de requisitos do sistema, que envolve a definição das funcionalidades essenciais da plataforma, como a coleta e processamento de dados clínicos. Foi essencial um levantamento das tecnologias e protocolos necessários para garantir a interoperabilidade com outros sistemas.

3.1.2 Criação de um Módulo de Análise Simulado (Mock)

Para validar a arquitetura da API e o fluxo de dados de ponta a ponta, foi desenvolvido um módulo de análise simulado, ou Mock. Este componente imita o comportamento de um modelo de IA real, recebendo os dados de um exame e retornando uma resposta padronizada e fictícia. A criação deste mock foi crucial para testar a comunicação e a lógica de processamento da API antes mesmo da conclusão de um modelo de aprendizado de máquina definitivo.

3.1.3 Construção do modelo de Base de Dados Inicial

Foi realizada a construção do modelo de Base de Dados Inicial, que abrigará as informações necessárias para o treinamento do modelo de IA, bem como as consultas e análises clínicas realizadas pela *API*. Esse modelo de dados foi projetado para suportar as necessidades da aplicação e para garantir a escalabilidade e integridade dos dados.

3.1.3.1 Construção da Versão Básica da API

Para implementar a API RESTful do Morpheus HealthSystem, optou-se pelo framework Laravel (OTWELL *et al.*, 2024), principalmente devido às suas características maduras e bem estabelecidas para desenvolvimento web em PHP. O Laravel é moderno, de código aberto e mantém-se em evolução constante desde 2011, contando com ampla documentação e comunidade ativa. Ele segue o padrão arquitetural MVC, facilita a criação de APIs RESTful e oferece recursos prontos para tarefas comuns (autenticação, roteamento, *migrations*, etc.).

O **roteamento do Laravel** é bastante intuitivo e expressivo, favorecendo boas práticas. As rotas RESTful são definidas em arquivos dedicados (por exemplo, `routes/api.php`) usando declarações como `Route::get`, `Route::post` ou `Route::apiResource`. Este último, por exemplo, permite registrar múltiplos endpoints com uma única declaração. A Listagem 1 demonstra como todos os endpoints de ação *Create, Read, Update & Delete* (*CRUD*) para o recurso `pacientes` são criados.

Código-fonte 1 – Exemplo de definição de rotas de recurso na API.

```
1 // Em routes/api.php
2 Route::apiResource('pacientes', PacienteController::class);
```

A **comunidade global** ativa do Laravel e sua maturidade consolidada no mercado são diferenciais importantes para a manutenção e extensibilidade do sistema. Com um ecossistema rico (Laracasts, pacotes de terceiros, fóruns e eventos), o Laravel oferece apoio contínuo e soluções prontas para problemas comuns. Isso significa que, além dos recursos nativos, há muitas bibliotecas compatíveis e bons exemplos de uso, o que facilita agregar funcionalidades complexas com menor esforço de desenvolvimento.

Para resumir, o projeto adotou o padrão de **arquitetura MVC** e a estrutura recomen-

dada pelo Laravel. A seguir, destacam-se alguns pontos-chave do padrão implementado:

- **Controladores:** Cada recurso do sistema (p. ex., Paciente, Exame, Laudo) possui um controlador dedicado em `App`. Esses controladores foram criados via *Artisan* com a opção `-resource`, o que gera automaticamente métodos para as operações CRUD. Em controladores de ação única, utiliza-se o método `__invoke` quando apropriado.
- **Rotas RESTful:** Todas as rotas da API estão definidas em `routes/api.php`. Utilizou-se o método `Route::apiResource` para expor endpoints padrão (GET, POST, PUT/PATCH, DELETE) de forma concisa.
- ***Middlewares*:** O *middleware* de autenticação `auth:sanctum` é aplicado às rotas que exigem acesso autenticado. Outros middlewares podem ser usados conforme necessário para tarefas como validação de permissões ou logging.
- **Banco de Dados e Eloquent:** Foi utilizado **MySQL** como SGBD e definiu-se os modelos Eloquent para cada tabela. As migrations do Laravel gerenciam a criação e a evolução do esquema do banco de dados. O Eloquent ORM permite manipular dados via objetos PHP sem escrever SQL diretamente, o que acelera o desenvolvimento e reduz a probabilidade de erros.

Em suma, a construção da versão básica da API seguiu as boas práticas preconizadas pelo Laravel: uso do Eloquent ORM, *migrations* para controle de versão do banco de dados, roteamento RESTful, autenticação via Sanctum, estrutura modular com controladores e aplicação de middlewares para controle de acesso. Essa abordagem garante um *back-end* padronizado, escalável e de fácil manutenção, adequado aos objetivos de um projeto de engenharia da computação.

3.1.3.2 Documentação e Especificação da API

A especificação **OpenAPI** (antigo Swagger) foi escolhida por ser um padrão *amplamente adotado* e interoperável. A própria OpenAPI Initiative descreve o OAS como o "padrão de descrição de API mais usado no mundo" (OpenAPI Initiative, 2024a), definindo um *padrão formal para descrever APIs HTTP* (OpenAPI Initiative, 2024b). Isso garante compatibilidade com um vasto ecossistema de ferramentas e promove a *interoperabilidade* entre sistemas consumidores e provedores da API (Programae.dev, 2023c).

Para gerar a documentação, optou-se pelo pacote **Laravel OpenAPI (Scramble)**. Diferentemente de abordagens baseadas em anotações Swagger/PHPDoc, o *Scramble* realiza

análise estática do código-fonte para produzir a especificação OpenAPI sem anotações manuais (Sam Dark, 2024). Isso evita acoplamento desnecessário entre documentação e lógica de negócio. Segundo sua documentação oficial, manter um arquivo separado ou usar anotações manuais é "tedioso e pode resultar em documentação desatualizada" (Scramble Documentation, 2024). A geração automática garante que a documentação esteja sempre sincronizada com o código, reduzindo a chance de inconsistências.

Entre os principais benefícios dessa abordagem destacam-se: a *consistência e atualização automática* da documentação conforme o código evolui (Scramble.dev, 2024), a *redução de esforço manual* com a eliminação de anotações redundantes (BRANCO, 2024), e a *aderência a boas práticas* de desenvolvimento orientado a contratos, possibilitando também a geração de clientes e testes automatizados (Swagger.io, 2024).

Em projetos acadêmicos e com evolução contínua, como o *Morpheus HealthSystem*, essa estratégia se mostra especialmente vantajosa. Ela minimiza erros humanos, facilita a manutenção e permite que outras aplicações (como interfaces gráficas ou módulos de análise) consumam a API com maior previsibilidade e segurança. A documentação automatizada, baseada em OpenAPI, assegura rastreabilidade, reprodutibilidade e maior transparência ao processo de desenvolvimento.

3.1.4 Construção de Conexão Básica com Frontend Web e Aplicações Mobile

Com a versão básica da *API* em funcionamento, foi iniciada a construção da conexão entre o *backend* da *API* e o *frontend* da plataforma, tanto para a versão *web* quanto para as versões *mobile*. O estabelecimento dessa conexão permite que os usuários (profissionais de saúde) possam interagir com a plataforma de maneira eficiente, acessando dados e realizando análises em tempo real.

3.1.4.1 Adequação do software embarcado para transmissões básicas

Foi feita também nessa etapa a construção de conexão básica com aplicações *mobile*, permitindo que a plataforma seja acessada em dispositivos móveis, proporcionando maior flexibilidade para os profissionais de saúde.

3.1.5 Implementação da Interface de Comunicação com o Módulo de Análise

Nesta etapa, o foco foi construir a lógica dentro da API responsável por se comunicar com o serviço de análise. Isso incluiu a criação da rota que aciona o processamento, o tratamento dos dados do exame a serem enviados e a gestão da resposta recebida. Todos os testes desta interface foram realizados utilizando o módulo de análise simulado (Mock) descrito anteriormente.

3.1.6 Revisão de Bugs e Erros, Teste de Execução em Simulação

Finalmente, a fase de revisão de *bugs* e erros foi essencial para garantir que a plataforma funcionasse sem falhas. Através de testes rigorosos, foram identificados e corrigidos problemas que poderiam afetar a usabilidade ou segurança da aplicação. Adicionalmente, foram realizados testes de execução em simulação de ambiente real, onde a plataforma foi testada com dados clínicos simulados para verificar seu desempenho e garantir que a comunicação com o módulo de análise ocorre de maneira confiável e que a API está pronta para receber e processar os resultados do futuro modelo de IA definitivo.

4 RESULTADOS E DISCUSSÕES

Como resultado prático deste trabalho, foi desenvolvida a API RESTful do Morpheus HealthSystem, que atua como o núcleo para a integração dos componentes da plataforma. Conforme a metodologia descrita, a implementação utilizou o framework Laravel e a documentação seguiu o padrão OpenAPI 3.1.0, gerada automaticamente com o auxílio da ferramenta Scramble. As seções a seguir detalham a estrutura da API, o modelo de dados, os testes de integração e a discussão sobre os resultados alcançados.

4.1 Implementação da *API*

A *API* foi estruturada em módulos de recursos que atendem às diferentes necessidades do sistema. A implementação foi guiada por um modelo de dados robusto e versionado, e os *endpoints* foram projetados para expor as funcionalidades de forma segura e intuitiva. Uma lista exaustiva de todos os *endpoints* implementados, incluindo suas rotas e ações de controlador, pode ser encontrada no Apêndice C. As subseções seguintes apresentam uma visão geral dos principais grupos de funcionalidades.

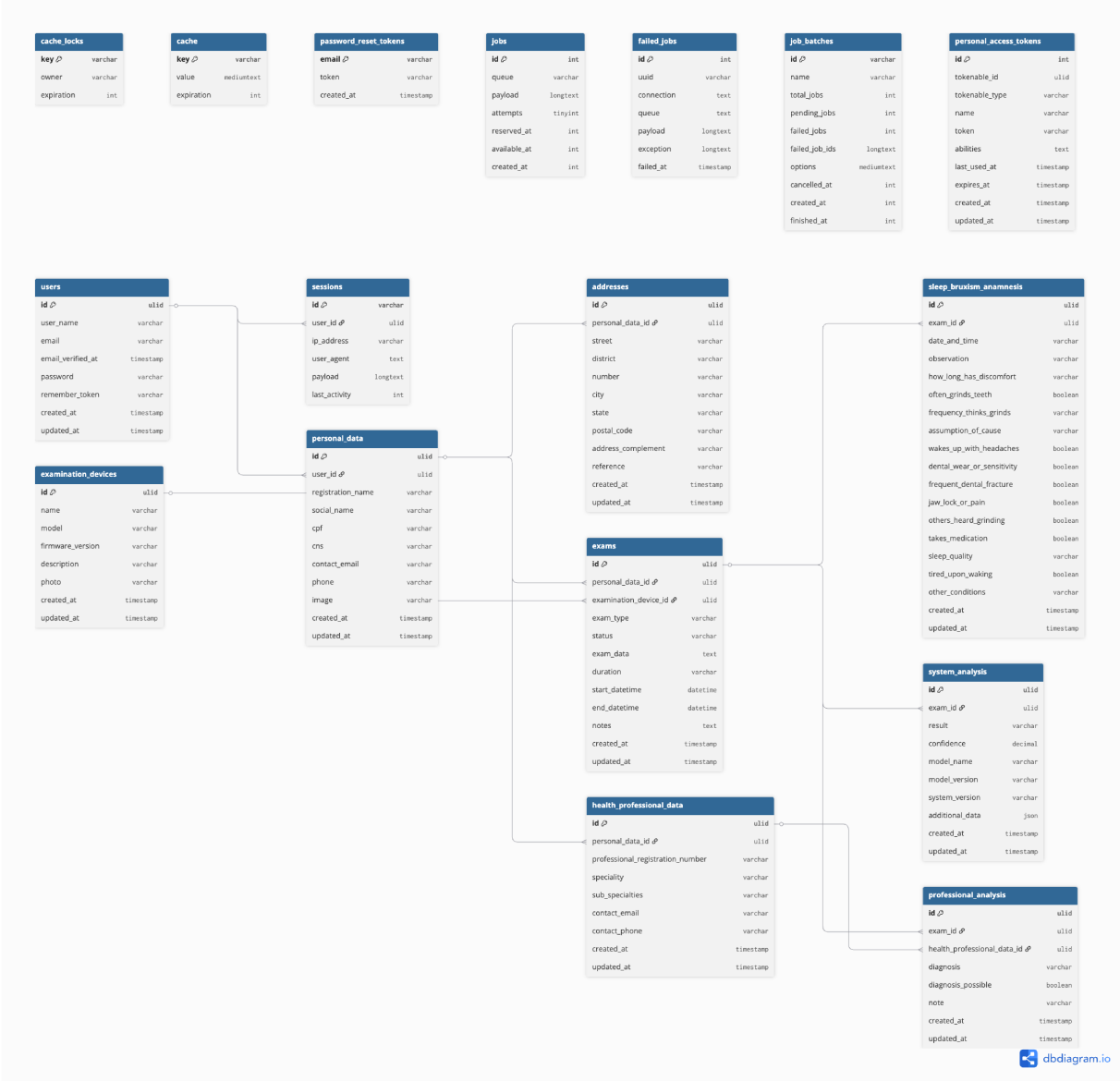
4.1.1 Modelo e Estrutura do Banco de Dados

A persistência dos dados da aplicação é gerenciada por um banco de dados relacional MySQL. A criação e a manutenção da estrutura do banco de dados foram realizadas por meio do sistema de *Migrations* do framework Laravel. Essa abordagem permite o versionamento do esquema do banco de dados, garantindo consistência e facilitando a colaboração e a implantação do sistema em diferentes ambientes.

O acesso e a manipulação dos dados na aplicação são abstraídos pelo *Object-Relational Mapper* (ORM) Eloquent, também do Laravel, que mapeia as tabelas do banco de dados para classes PHP, simplificando as operações de *CRUD*.

O diagrama entidade-relacionamento final, que inclui tanto as tabelas da aplicação quanto as tabelas padrão do framework, é apresentado na Figura 3 e também pode ser encontrodado no Apendice D. As principais entidades são descritas a seguir.

Figura 3 – Diagrama de Banco de Dados



Fonte: Produção do autor utilizando a ferramenta dbdiagram.io (Apendice D)

4.1.1.1 Entidades de usuários e autenticação

O núcleo de autenticação é formado pela tabela `users`, que armazena as credenciais de acesso (e-mail e senha *hash*). O Laravel gerencia automaticamente tabelas auxiliares como `password_reset_tokens` para a funcionalidade de redefinição de senha e `sessions` para o controle de sessões de usuários, que também foram incorporadas ao modelo final.

4.1.1.2 Entidades de dados clínicos e pessoais

A entidade central para os dados de indivíduos é a tabela `personal_data`, que armazena informações cadastrais como nome, número do Cadastro de Pessoa Física (CPF)

e número do Cartão Nacional de Saúde (CNS). Esta tabela possui uma relação de um-para-um com a tabela `users`, vinculando os dados de um paciente ou profissional à sua conta de acesso. Informações complementares, como endereços (`addresses`) e dados profissionais (`health_professional_data`), são armazenadas em tabelas separadas e relacionadas à `personal_data`, seguindo uma abordagem normalizada.

4.1.1.3 Entidades de Exames e Análises

O fluxo de exames, que representa a principal funcionalidade do sistema, é modelado por um conjunto de tabelas inter-relacionadas. A tabela `exams` é a entidade central, registrando cada procedimento realizado e seu estado atual (ex: *pending*, *analyzing*, *completed*). Cada exame está associado a um paciente (via `personal_data`) e a um equipamento (`examination_devices`).

Os resultados do exame são complementados por múltiplas fontes de análise:

- `dental_discomfort_anamnesis`: Armazena as respostas do questionário de anamnese preenchido pelo paciente.
- `system_analysis`: Tabela destinada a registrar o pré-diagnóstico que será gerado pelo módulo de Inteligência Artificial.
- `professional_analysis`: Armazena o diagnóstico e as notas inseridas por um profissional de saúde.

Essa estrutura modular permite que um único exame consolide diferentes perspectivas diagnósticas, enriquecendo o resultado final apresentado ao profissional de saúde.

4.1.2 Autenticação e Gestão de Usuários

A gestão de acesso é um requisito fundamental do sistema. Foram implementados *endpoints* para o registro, login e consulta de dados do usuário autenticado. A Tabela 2 resume essas rotas.

4.1.3 Gestão de Dados Pessoais e de Saúde

Para o correto funcionamento da plataforma, é essencial gerenciar os dados dos pacientes e dos profissionais de saúde. A Tabela 3 detalha os endpoints responsáveis por essas operações.

Tabela 2 – Endpoints para Autenticação e Gestão de Usuários.

Rota (Endpoint)	Método HTTP	Descrição da Funcionalidade
`/register`	POST	Realiza o cadastro de um novo usuário no sistema.
`/login`	POST	Autentica um usuário e retorna um token de acesso.
`/logout`	GET	Invalida o token de acesso do usuário autenticado.
`/user`	GET	Retorna os dados do perfil do usuário autenticado.
`/user`	PUT	Permite a atualização dos dados de perfil.

Tabela 3 – Endpoints para Gestão de Dados Pessoais e de Saúde.

Rota (Endpoint)	Método HTTP	Descrição da Funcionalidade
`/my/data`	GET/POST/PUT	Gerencia os dados pessoais do usuário autenticado.
`/personal-data`	GET/POST	Gerencia dados pessoais (acesso de administrador).
`/my/addresses`	GET/POST	Gerencia os endereços do usuário autenticado.
`/my/professional-data`	GET/POST/PUT	Gerencia os dados profissionais do usuário autenticado.

4.1.4 Gestão de Exames e Análises

Este é o núcleo do sistema, responsável por receber, processar e analisar os dados dos exames. A Tabela 4 resume as rotas implementadas.

Tabela 4 – Endpoints para Gestão de Exames e Análises.

Rota (Endpoint)	Método HTTP	Descrição da Funcionalidade
`/my/exams`	GET	Lista os exames do paciente autenticado.
`/my/exam`	POST	Permite o registro de um novo exame pelo paciente.
`/exam/{id}`	GET/PUT/DELETE	Permite a gestão de um exame por um profissional.
`/system-analysis/{examId}`	POST	Dispara a análise automática (IA) sobre os dados de um exame.
`/professional-analysis`	GET/POST	Permite que um profissional de saúde adicione sua análise.
`/my/anamnesis/dental-discomfort`	GET/POST	Gerencia o questionário de anamnese do paciente.

4.1.5 Detalhamento da requisição de anamnese

Para ilustrar a complexidade e a riqueza de dados que a *API* está preparada para gerenciar, a Tabela 5 detalha o corpo completo da requisição para o endpoint `POST` */my/anamnesis/dental-discomfort*. Este endpoint é responsável por submeter as respostas do questionário de anamnese, que são cruciais para o processo de pré-diagnóstico do bruxismo. Cada campo corresponde a uma pergunta específica feita ao paciente.

Tabela 5 – Estrutura de Dados para Registro de Anamnese (*DentalDiscomfortRegisterRequest*).

`Campo (Field)`	**Descrição da Pergunta**
`exam_id`	ID do exame ao qual a anamnese está associada. (Tipo de resposta: string)
`date_and_time`	Data e hora do preenchimento do questionário. (Tipo de resposta: datetime)
`observation`	Campo para observações gerais do paciente. (Tipo de resposta: string)
`how_long_has_mandibular _discomfort`	Há quanto tempo o paciente sente desconforto mandibular? (Tipo de resposta: string (Que é uma opção de uma escala))
`often_grinds_teeth`	O paciente range ou aperta os dentes? (Tipo de resposta: boolean)
`frequency_thinks_grinds_teeth`	Com que frequência o paciente acha que range os dentes? (Tipo de resposta: string, (Que é uma opção de uma escala))
`assumption_of_cause`	Qual a suposição do paciente sobre a causa do problema? (Tipo de resposta: string)
`wakes_up_with_headaches`	O paciente acorda com dores de cabeça? (Tipo de resposta: boolean)
`noticed_dental_wear_or _sensitivity`	O paciente notou desgaste ou sensibilidade nos dentes? (Tipo de resposta: boolean)

Continua na próxima página

Tabela 5 – (Continuação)

Campo (Field)	**Descrição da Pergunta**
frequent_dental_fracture_or_restoration	O paciente teve fraturas ou restaurações dentais com frequência? (Tipo de resposta: boolean)
jaw_locks_or_pain_when_opening_or_closing_mouth	A mandíbula do paciente trava ou dói ao abrir ou fechar a boca? (Tipo de resposta: boolean)
others_heard_grinding_teeth	Outras pessoas já ouviram o paciente ranger os dentes durante o sono? (Tipo de resposta: boolean)
takes_medication_related_to_symptoms	O paciente toma alguma medicação que possa se relacionar aos sintomas? (Tipo de resposta: boolean)
sleep_quality	Qual a autoavaliação do paciente sobre sua qualidade de sono? (Tipo de resposta: string, (Que é uma opção de uma escala))
tired_upon_waking	O paciente sente cansaço ao acordar? (Tipo de resposta: boolean)
other_conditions_related_to_symptoms	O paciente deseja descrever outras condições que julgue relacionadas aos sintomas? (Tipo de resposta: string)

A estrutura de dados para a anamnese foi projetada para ser simples e direta, ao mesmo tempo em que captura as informações essenciais para o pré-diagnóstico, alinhando-se aos questionários clínicos padrão para distúrbios do sono.

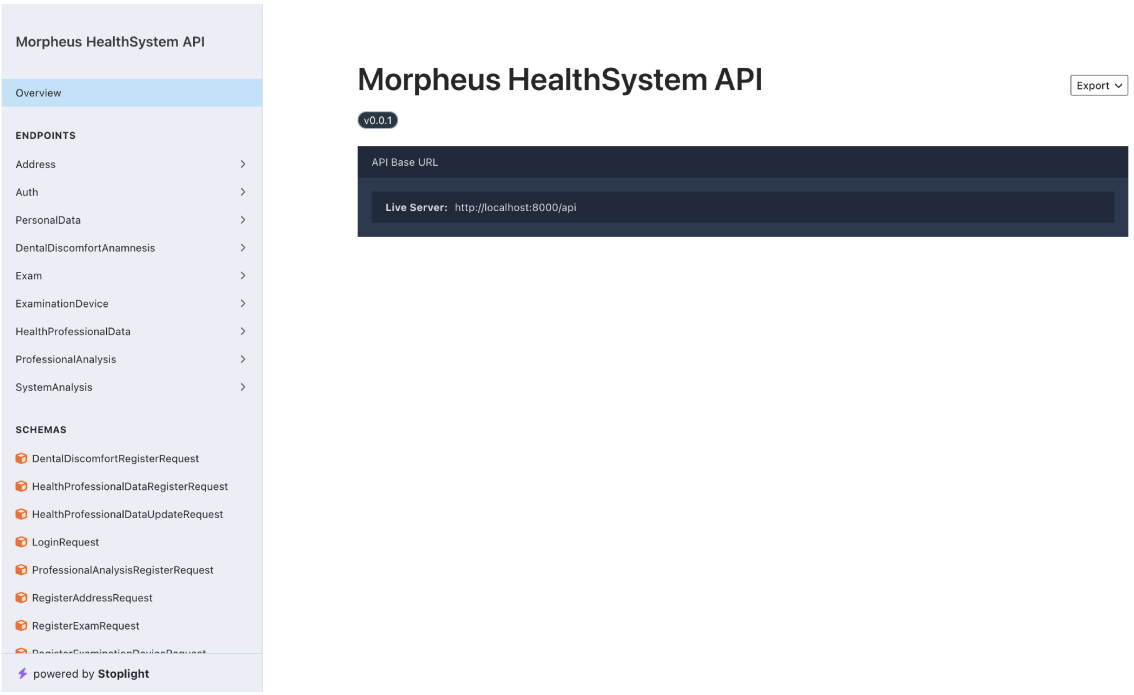
4.2 Integração do Sistema

A documentação gerada no padrão OpenAPI foi um pilar para a integração entre as diferentes frentes de desenvolvimento. Ela funcionou como um contrato claro, permitindo que as equipes de *frontend*, *mobile* e *hardware* desenvolvessem suas partes em paralelo, consumindo a *API* de forma padronizada.

4.2.1 Documentação Interativa como Ferramenta de Integração

A especificação OpenAPI permitiu a geração de uma documentação interativa, conforme ilustrado na Figura 4. Esta ferramenta foi essencial para que os desenvolvedores pudessem visualizar e testar cada *endpoint* diretamente no navegador, o que reduziu significativamente o tempo de depuração e alinhamento entre as equipes.

Figura 4 – Documentação interativa da *API* gerada via OpenAPI.



Fonte: Produção do autor

4.3 Testes e Simulação

Para garantir a robustez e a confiabilidade da *API*, foram realizados testes de validação e uma simulação do fluxo principal de análise de exame.

4.3.1 Testes de Validação de Dados

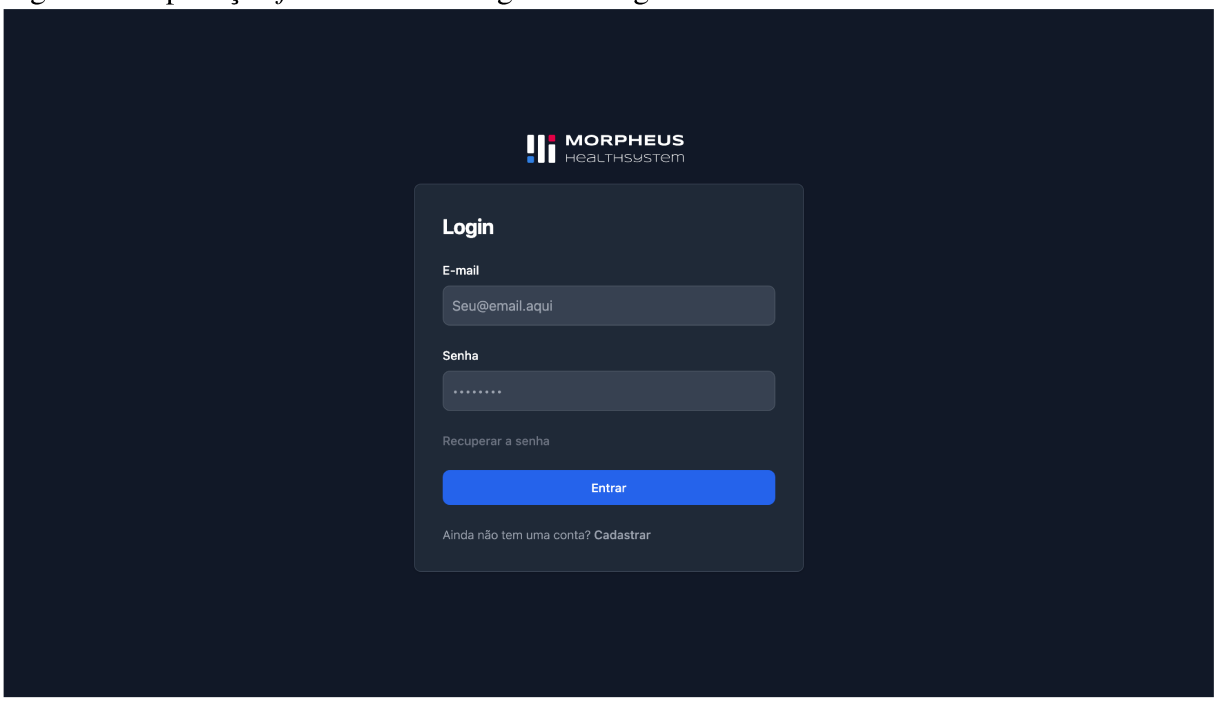
Foram executados testes de unidade e de integração para verificar as regras de validação em cada *endpoint*. Por exemplo, ao tentar registrar um usuário com um e-mail inválido, a *API* retornou corretamente o código de status HTTP 422 (Unprocessable Entity), junto com uma mensagem detalhando o erro, conforme o contrato definido na especificação OpenAPI.

4.3.2 Integração com Aplicação Frontend

Para validar a funcionalidade e a usabilidade da *API* em um cenário prático, foi desenvolvida uma aplicação web protótipo. A aplicação foi construída com a biblioteca React (Meta Platforms, Inc. and community, 2013-2025) para a interface de usuário, Vite como ferramenta de compilação e TypeScript para garantir a tipagem e a robustez do código. A comunicação com a *API* foi gerenciada pela biblioteca Axios, que realizou as chamadas HTTP para os endpoints RESTful.

O fluxo de interação do profissional de saúde na aplicação inicia-se na tela de autenticação, ilustrada na Figura 5. Nesta tela, as credenciais são enviadas via uma requisição POST para o endpoint `/login`. Em caso de sucesso, a *API* retorna um token de acesso que é armazenado localmente para autenticar requisições subsequentes.

Figura 5 – Aplicação *frontend web*: Página de Login



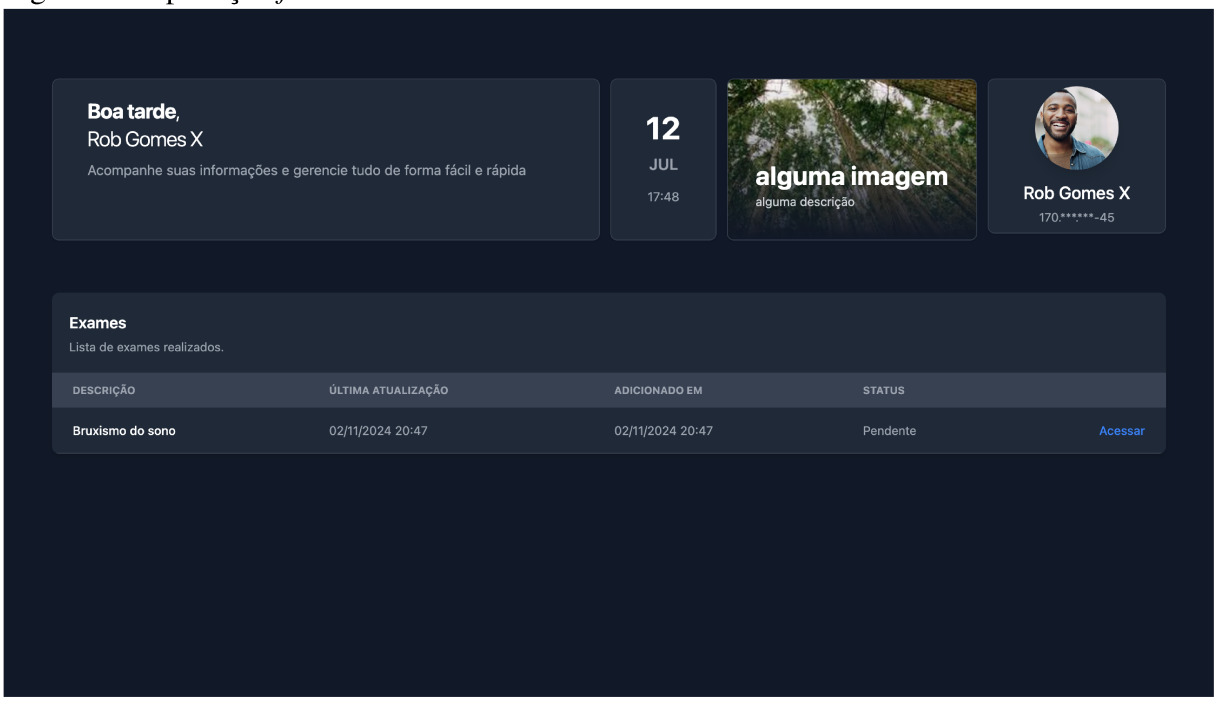
Fonte: Produção do autor

Após o login, o usuário é direcionado ao *dashboard* principal (Figura 6), que apresenta uma visão geral dos exames recentes e outras informações relevantes, obtidas através de chamadas GET para endpoints como `/exam`. A navegação entre as diferentes seções da aplicação foi implementada com a biblioteca React Router DOM (The Remix Team, 2014-2025).

A Figura 7 apresenta da tela de visualização de um exame específico com dados simulados. Nesta tela, os dados coletados pelo hardware seriam renderizados graficamente

para facilitar a análise pelo profissional, utilizando bibliotecas de visualização de dados como a ApexCharts. Esta tela demonstra a capacidade da aplicação em consumir e apresentar dados complexos provenientes da *API*.

Figura 6 – Aplicação *frontend web*: *Dashboard*



Fonte: Produção do autor

4.3.3 Simulação do Processo de Análise

Foi realizado um teste de simulação completo, no qual dados de um exame fictício foram enviados para o endpoint `/my/exam`. Posteriormente, a rota `/system-analysis/{examId}` foi acionada para iniciar o processamento. A *API* demonstrou-se capaz de receber os dados, armazená-los corretamente e enfileirar a tarefa para o módulo de Inteligência Artificial que, nesta fase do projeto, foi representado pelo módulo de análise simulado (Mock). Este teste validou com sucesso que a API pode se comunicar com um serviço de análise externo, enviando os dados no formato correto e processando a resposta, o que garante a prontidão da infraestrutura para a futura integração do modelo de IA definitivo, validando o fluxo de ponta a ponta.

4.4 Discussão dos resultados

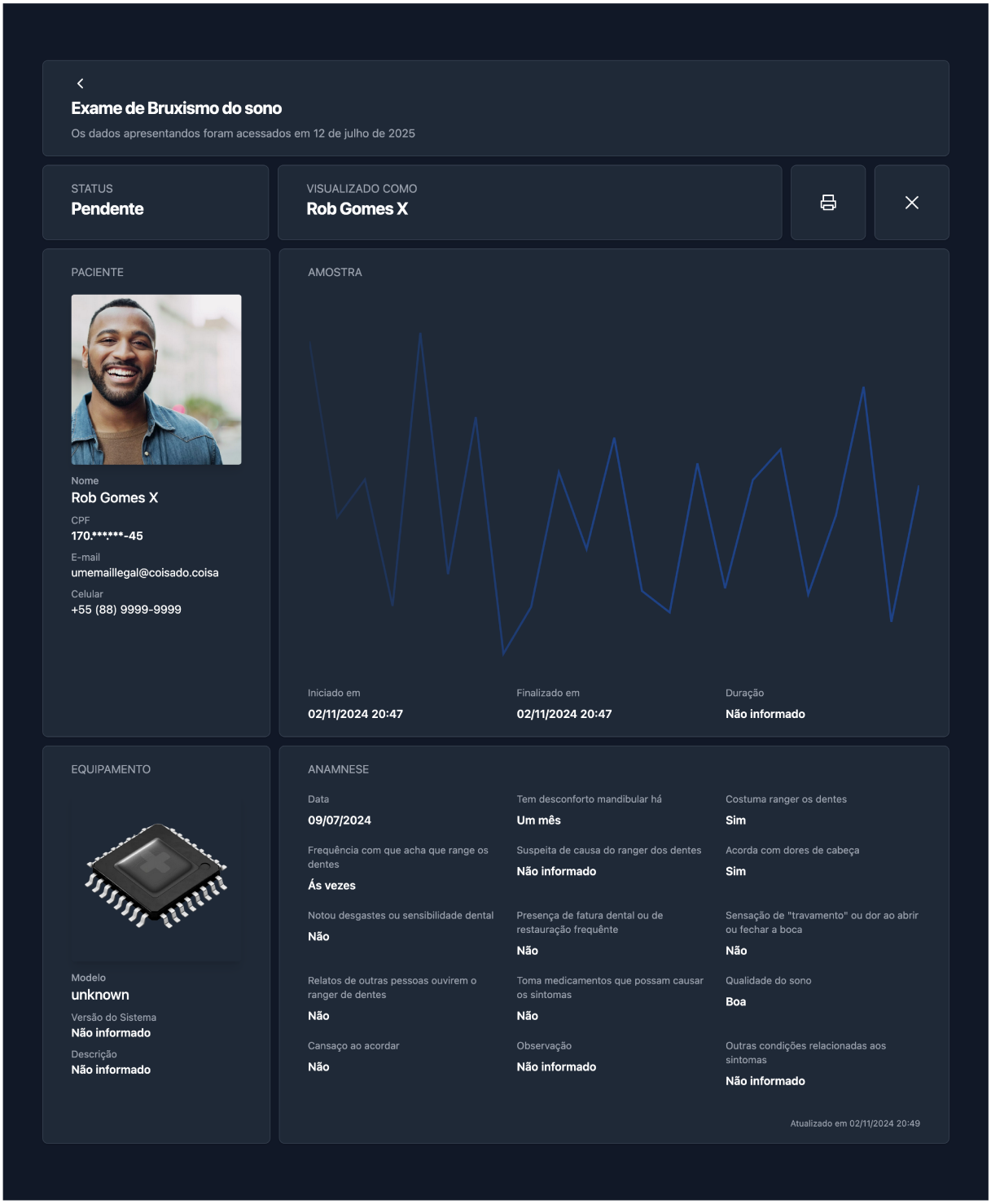
Os resultados obtidos demonstram que a *API* desenvolvida atende aos objetivos propostos, fornecendo uma plataforma centralizada e robusta para o ecossistema Morpheus

HealthSystem. A arquitetura RESTful, aliada à documentação OpenAPI, provou ser uma escolha acertada, facilitando a integração e a escalabilidade do sistema.

A estrutura de dados para exames e anamnese, embora funcional, representa uma primeira versão. A aderência a padrões de interoperabilidade em saúde, como o FHIR, não foi completamente implementada e se apresenta como uma clara oportunidade de melhoria em trabalhos futuros. A principal dificuldade encontrada foi a modelagem de dados de saúde, que exigiu um balanço entre a complexidade dos requisitos clínicos e a simplicidade necessária para uma *API* eficiente.

Conclui-se que o sistema é funcional e atinge seu propósito de protótipo viável, mas a validação com dados clínicos reais e a evolução para padrões de mercado são passos essenciais para sua maturidade.

Figura 7 – Aplicação *frontend web*: Exame *Mockup*



Fonte: Produção do autor

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho de conclusão de curso desenvolveu-se uma *API* RESTful para servir como núcleo de um ecossistema de telemedicina focado no diagnóstico de distúrbios do sono, com ênfase inicial no bruxismo. O objetivo principal de criar uma plataforma centralizada, robusta e escalável foi alcançado com sucesso através da implementação de uma *API* utilizando o *framework* Laravel, um banco de dados relacional MySQL e uma documentação padronizada com a especificação OpenAPI.

Os resultados apresentados demonstram a construção de um protótipo funcional, onde a API se mostrou capaz de gerenciar usuários, armazenar dados clínicos e se integrar com as aplicações de *frontend*, além de validar seu fluxo de comunicação com serviços de análise através de um módulo de IA simulado.

O projeto contribui com uma solução de baixo custo e alta acessibilidade para o monitoramento de distúrbios do sono, alinhada às crescentes demandas por inovação na área de saúde digital. Embora o sistema atual seja um protótipo viável, ele representa um passo significativo em direção a uma ferramenta que pode, futuramente, melhorar a qualidade de vida de pacientes ao facilitar um diagnóstico mais rápido e eficiente. As limitações identificadas, como a necessidade de validação com dados clínicos reais e a aderência a padrões de mercado, abrem caminho para as próximas etapas de desenvolvimento descritas a seguir.

A conclusão deste projeto marca o início de um ciclo de evolução contínua. Para que o Morpheus HealthSystem atinja seu pleno potencial, são propostas as seguintes linhas de trabalho futuro:

- **Desenvolvimento de um Módulo de Anamnese Dinâmica:** Uma das evoluções mais impactantes seria a criação de um módulo de anamnese inteligente. Em vez de um questionário estático, o sistema poderia adaptar as perguntas com base na patologia suspeita. Isso exigiria um trabalho de pesquisa interdisciplinar, envolvendo profissionais de saúde para mapear as perguntas mais relevantes para diferentes distúrbios (bruxismo, apneia do sono, etc.), tornando a coleta de dados subjetivos mais precisa e direcionada.
- **Adoção Completa do Padrão FHIR:** Embora a *API* tenha sido projetada com a interoperabilidade em mente, um próximo passo crucial é refatorar a estrutura de dados e os endpoints para aderir completamente ao padrão *Fast Healthcare Interoperability Resources* (FHIR). Isso garantiria a compatibilidade do Morpheus HealthSystem com sistemas de prontuários eletrônicos e outras plataformas de saúde globais.

- **Substituição do Módulo Simulado (Mock) e Validação do Modelo de IA Definitivo:** O passo mais crítico para a evolução do projeto será a substituição do módulo de análise simulado pelo modelo de aprendizado de máquina real, que está em desenvolvimento paralelo. Esta etapa envolverá não apenas a integração técnica, mas também um ciclo rigoroso de treinamento com dados clínicos reais e a validação da acurácia dos pré-diagnósticos gerados em comparação com os laudos de profissionais de saúde, estabelecendo a confiabilidade clínica da ferramenta.
- **Evolução do Ecossistema de Aplicações:** As aplicações de *frontend* e *mobile* podem ser aprimoradas com novas funcionalidades, como a visualização de dados históricos, a geração de relatórios em PDF para os pacientes, e a implementação de notificações em tempo real para alertar sobre o status dos exames.
- **Validação Clínica, Testes de Usabilidade e Testes de Aceitação:** Realizar um estudo clínico com pacientes e profissionais de saúde reais é fundamental. Isso permitiria não apenas validar a eficácia do sistema no diagnóstico, mas também coletar *feedback* sobre a usabilidade e nível de aceitação, garantindo que a tecnologia atenda às necessidades e expectativas dos seus usuários finais.

REFERÊNCIAS

ALMEIDA, M. A. C. Bruxismo do sono: uma revisão de literatura. **Revista Brasileira de Odontologia**, v. 73, n. 6, p. 770–778, 2016.

Amazon Web Services. **What is an API?** 2024. Acesso em: 28 jun. 2025. Disponível em: <https://aws.amazon.com/what-is/api/>.

ANDRADE, N. F. D.; LIMA, L. R. O. de; SOUSA, F. F. D. de; JATKOSKI, A. Sleep bruxism and quality of life: A systematic review and meta-analysis. **Sleep Medicine Reviews**, v. 38, p. 64–76, 2018.

BRANCO, D. **Documentação automática de APIs com OpenAPI**. 2024. Acesso em: 28 jun. 2025. Disponível em: <https://danielbranco.dev/posts/openapi-documentacao-automatica>.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. **Diário Oficial [da] União**, Brasília, DF: Presidência da República, 2018. ISSN 1677-7042. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>.

BRUXISM. **National Institute of Dental and Craniofacial Research**, 2021. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3081266/>.

CARLSON, K. R.; MCNAMARA, D. S. **Orofacial pain: A clinical primer**. [S.l.]: Elsevier Health Sciences, 2013.

COMMISSION, E. **Medical Devices Regulation (MDR)**. 2020. Disponível em: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32017R0745>.

DIAS FILHO, F. G. **Morpheus - ambiente de coleta, processamento de dados e pré-diagnóstico do bruxismo do sono**. 2022. Disponível em: <http://repositorio.ufc.br/handle/riufc/77978>.

DOCS, M. W. **HTTP request methods**. 2023. Acessado em: 18 de janeiro de 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>.

Document360. **What is API Documentation?** 2023. Acesso em: 28 jun. 2025. Disponível em: <https://document360.com/blog/api-documentation/>.

FELLENDORF, F. T.; SCHüTZ, C. G.; BAUER, M. F. P. M.; PLEWNIA, A. M.; HAUTZINGER, M. Monitoring Sleep Changes via a Smartphone App in Bipolar Disorder: Practical Issues and Validation of a Potential Diagnostic Tool. **Frontiers in Psychiatry**, v. 12, p. 641241, mar 2021. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/33841209/>.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. Tese (Doutorado) — University of California, Irvine, 2000. Disponível em: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

FIELDING, R. T. *et al.* **Hypertext Transfer Protocol – HTTP/1.1**. 1999. Internet Engineering Task Force (IETF). RFC 2616. Disponível em: <https://www.rfc-editor.org/rfc/rfc2616>.

FOUNDATION, F. **FHIR Overview**. 2022. Acesso em janeiro de 2025. Disponível em: <https://www.hl7.org/fhir/overview.html>.

GONÇALVES, L. C.; FREITAS, M. R. de; MACHADO, A. C. R.; JATKOSKI, A. Accuracy of electromyography and biofeedback for the diagnosis of sleep bruxism: A systematic review. **Sleep Medicine Reviews**, v. 24, p. 1–8, 2015.

HBK. **Selection guide for piezo sensors**. 2025. <https://www.hbkworld.com/en/knowledge/resource-center/articles/selection-guide-for-piezo-sensors>. Acessado em 06 de agosto de 2025.

KADLEC, F.; LELLOUCH, G.; AL-NAIMI, F.; DUFOUR, I.; ALCHEIKH, N. Piezoelectric micromachined microphone with high acoustic overload point and with electrically controlled sensitivity. **Micromachines**, v. 15, n. 7, p. 879, 2024.

KOUR, R.; ARYA, S. K.; KUMAR, V.; KUMAR, S.; KUMAR, R.; KUMAR, R. A comprehensive review of piezoelectric materials and their applications. **Applied System Innovation**, v. 4, n. 3, p. 52, 2021.

LARMAN, C. G. **UML e padrões de design: uma introdução à criação de sistemas de software objeto-orientados**. [S.l.]: Editora Elsevier, 2004.

MACHADO, A. C. R.; FREITAS, M. R. de; JATKOSKI, A. Sleep bruxism and temporomandibular disorders: A narrative review. **Sleep Medicine Reviews**, v. 25, p. 45–52, 2015.

MACHADO, A. C. R.; GONCALVES, L. C.; FREITAS, M. R. de; JATKOSKI, A. Diagnostic tools for sleep bruxism: An overview. **Sleep Medicine Reviews**, v. 31, p. 1–9, 2017.

MELLONIG J. C., A. M. A. . C. R. B. Bruxismo: uma revisão de literatura. **Revista de Odontologia da UNESP**, v. 38, n. 2, p. 117–124, 2009.

Meta Platforms, Inc. and community. **React**. 2013–2025. Disponível em: <https://react.dev>.

OKESON, J. P. **Tratamento das Desordens Temporomandibulares e Oclusão**. [S.l.]: Elsevier, 2008.

OpenAPI Initiative. **OpenAPI Initiative**. 2024. Acesso em: 28 jun. 2025. Disponível em: <https://www.openapis.org/>.

OpenAPI Initiative. **OpenAPI Specification (OAS)**. 2024. Acesso em: 28 jun. 2025. Disponível em: <https://spec.openapis.org/oas/v3.1.0>.

OTWELL, T. *et al.* **Laravel Framework**. GitHub, 2024. Disponível em: <https://github.com/laravel/framework/releases/tag/v11.45.1>.

Programae.dev. **OpenAPI e Swagger: entenda a diferença e aplicação**. 2023. Acesso em: 28 jun. 2025. Disponível em: <https://programae.dev/swagger-vs-openapi/>.

Programae.dev. **OpenAPI: O que é, para que serve e como utilizar**. 2023. Acesso em: 28 jun. 2025. Disponível em: <https://programae.dev/openapi/>.

Programae.dev. **OpenAPI: O que é, para que serve e como utilizar**. 2023. Acesso em: 28 jun. 2025. Disponível em: <https://programae.dev/openapi/>.

REZENDE, S. M. **Materiais e Dispositivos Eletrônicos**. S.l.: s.n., 2004. 474 p.

Sam Dark. **Laravel OpenAPI (Scramble) - GitHub**. 2024. Acesso em: 28 jun. 2025. Disponível em: <https://github.com/samdark/scramble>.

Scramble Documentation. **Scramble Docs - Laravel OpenAPI**. 2024. Acesso em: 28 jun. 2025. Disponível em: <https://scramble.dev/docs>.

Scramble.dev. **Why Scramble?** 2024. Acesso em: 28 jun. 2025. Disponível em: <https://scramble.dev/docs/why>.

Swagger.io. **Swagger Codegen: Building Clients from OpenAPI**. 2024. Acesso em: 28 jun. 2025. Disponível em: <https://swagger.io/tools/swagger-codegen/>.

The Remix Team. **React Router DOM**. 2014–2025. Disponível em: <https://reactrouter.com>.

WANG, Y. e. a. Innovative medical instruments data platform and information system construction based on service computing. **Journal of Healthcare Engineering**, v. 2021, 2021.

Wikipedia contributors. **Piezoelectric sensor — Wikipedia, The Free Encyclopedia**. 2025. <https://en.wikipedia.org/w/index.php?title=Piezoelectric_sensor&oldid=1234567890>. Acessado em 06 de agosto de 2025.

YAP, A. U.; CHUA, A. P. Sleep bruxism: Current knowledge and contemporary management. **Journal of Conservative Dentistry**, v. 19, n. 5, p. 383–389, sep 2016.

ZHANG, J.; LIU, Y.; WANG, Z.; LI, Y.; WANG, Y.; ZHANG, Y.; ZHANG, Y.; LI, Y.; LI, Y.; LIU, C. Frontiers in medical physics: material classification and blood pressure measurement of wearable piezoelectric sensors. **Frontiers in Physics**, v. 12, 2024.

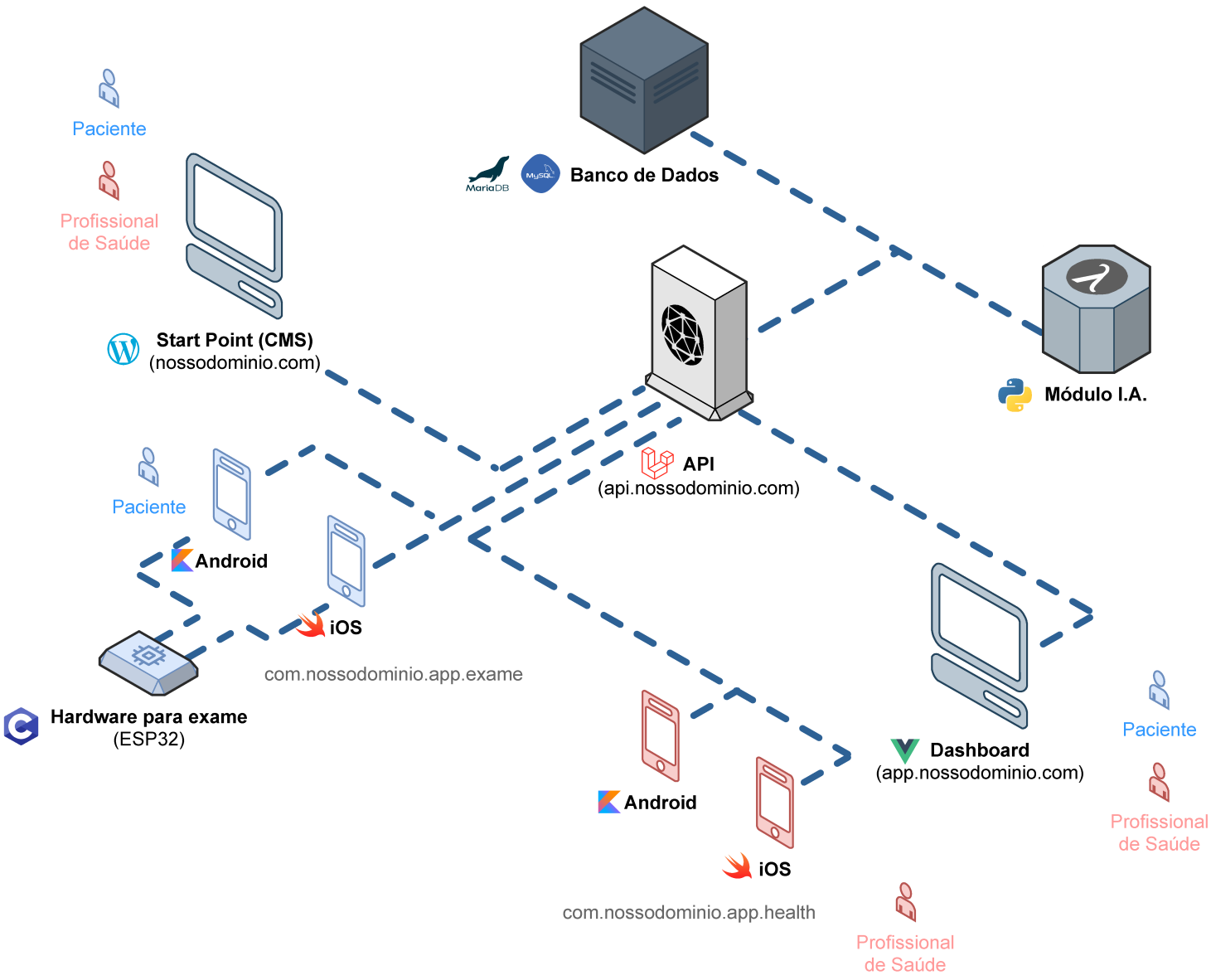
ZHOU, Q.; LAM, K. H.; ZHENG, H.; QIU, W.; SHUNG, K. K. Ultrasonic transducers for medical diagnostic imaging. **IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control**, v. 61, n. 10, p. 1556–1573, 2014. Publicado online em PMC em 2017.

APÊNDICE A – ARQUITETURA DO SISTEMA

Projeção da arquitetura do sistema, tanto a ideal quanto a do que se decidiu por ser equivalente a um protótipo viável.

System Architecture

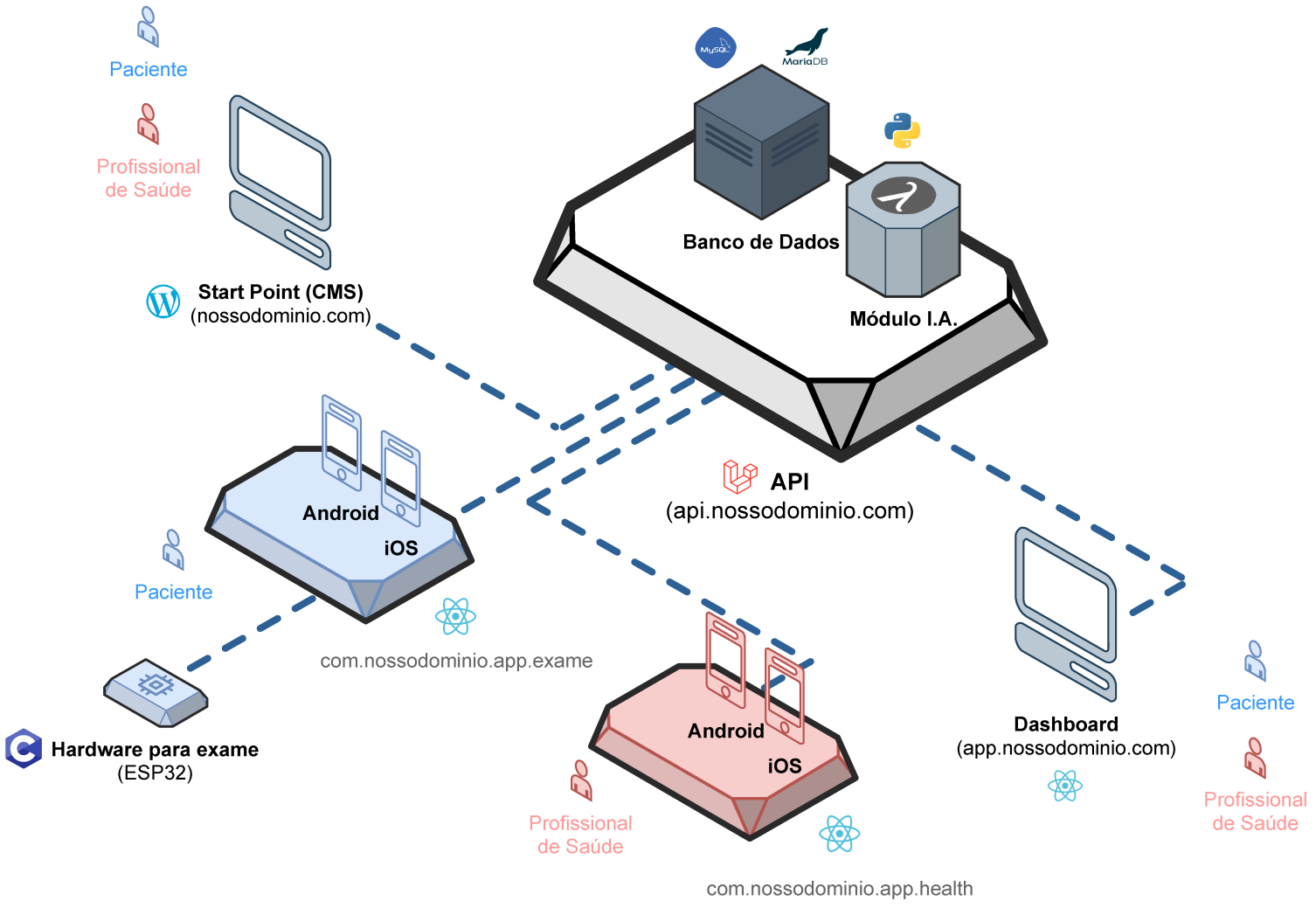
Projeto Morpheus - Modelo Ideal



Designed by Robson M. Gomes
contato@robsongomes.me

System Architecture

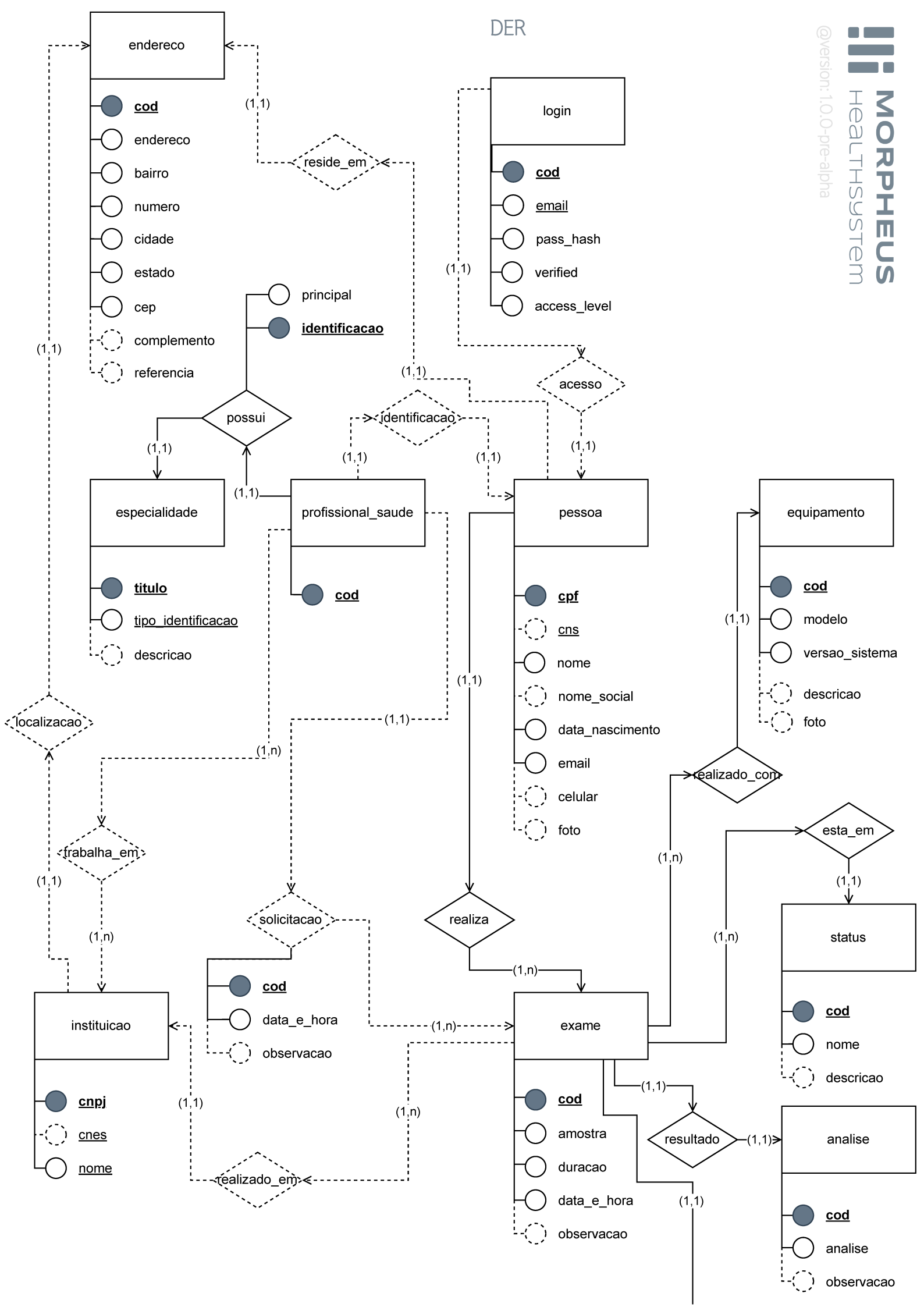
Projeto Morpheus - Protótipo Viável

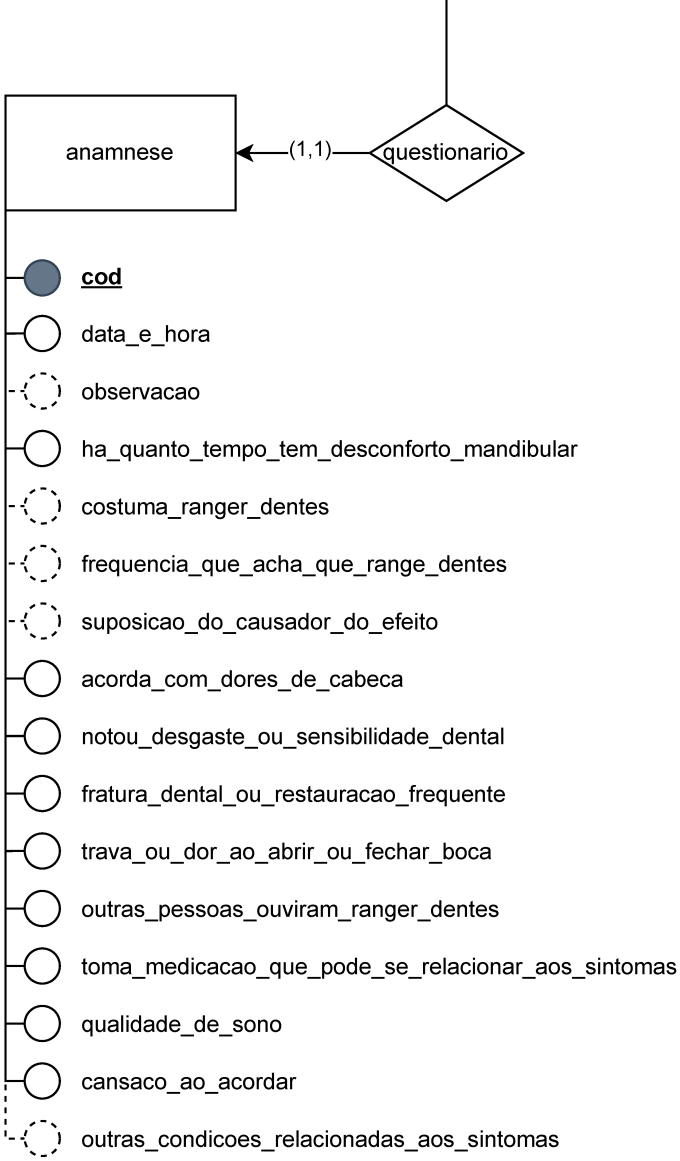


Designed by Robson M. Gomes
contato@robsongomes.me

APÊNDICE B – DIAGRAMA ENTIDADE E RELACIONAMENTO - V1.0.0-PRE-ALPHA

Diagrama entidade e relacionamento da versão 1.0.0-pre-alpha. A versão apresentada aqui teve sua última atualização adicionada em . Algumas entidades foram propositalmente descritas de maneira vaga, pois ocorrerão mudanças a depender de configurações do framework utilizado para a construção da API e também a depender de testes realizados com a aplicação mobile e o hardware desenvolvido para execução dos exames.


DER
MORPHEUS
HealTHsystem
@version:1.0.0-pre-alpha
endereco
cod
endereco
bairro
numero
cidade
estado
cep
complemento
referencia
reside_em
(1,1)
login
cod
email
pass_hash
verified
access_level
principal
identificacao
possui
acesso
identificacao
especialidade
titulo
tipo_identificacao
descricao
profissional_saude
cod
pessoa
cpf
cns
nome
nome_social
data_nascimento
email
celular
foto
equipamento
cod
modelo
versao_sistema
descricao
foto
localizacao
trabalha_em
realizado_com
esta_em
solicitacao
realiza
instituicao
cnpj
cnes
nome
cod
data_e_hora
observacao
exame
cod
amostra
duracao
data_e_hora
observacao
resultado
realizado_em
status
cod
nome
descricao
analise
cod
analise
observacao
(1,n)



APÊNDICE C – TABELA DE ROTAS DA API

A tabela a seguir detalha todos os endpoints implementados na *API* do Morpheus HealthSystem, excluindo as rotas internas do framework e de depuração. As rotas estão agrupadas por funcionalidade para facilitar a consulta.

Tabela 6 – Lista exaustiva de rotas da aplicação.

Método	**URI**
	Autenticação e Usuários
POST	</register>
POST	</login>
GET, HEAD	</logout>
GET, HEAD	</user>
PUT	</user>
DELETE	</user>
GET, HEAD	</unauthorized>
GET, HEAD	</my/roles>
GET, HEAD	</my/permissions>
	Dados Pessoais e Endereços
GET, HEAD	</my/data>
POST	</my/data>
PUT	</my/data>
GET, HEAD	</my/addresses>
POST	</my/address>
GET, HEAD	</my/address/{id}>
PUT	</my/address/{id}>
DELETE	</my/address/{id}>
GET, HEAD	</address>
POST	</address>
GET, HEAD	</address/{id}>
PUT	</address/{id}>

Continua na próxima página

Tabela 6 – (Continuação)

Método	**URI**
DELETE	</address/{id}>
GET, HEAD	</personal-data>
POST	</personal-data>
GET, HEAD	</personal-data/{id}>
PUT	</personal-data/{id}>
DELETE	</personal-data/{id}>
	Dados Profissionais
GET, HEAD	</my/professional-data>
POST	</my/professional-data>
PUT	</my/professional-data>
GET, HEAD	</professional-data>
POST	</professional-data>
GET, HEAD	</professional-data/{id}>
PUT	</professional-data/{id}>
DELETE	</professional-data/{id}>
	Exames e Análises
GET, HEAD	</my/exams>
POST	</my/exam>
GET, HEAD	</my/exam/{id}>
PUT	</my/exam/{id}>
DELETE	</my/exam/{id}>
GET, HEAD	</exam>
POST	</exam>
GET, HEAD	</exam/{id}>
PUT	</exam/{id}>
DELETE	</exam/{id}>
GET, HEAD	</my/anamnesis/dental-discomfort>
POST	</my/anamnesis/dental-discomfort>

Continua na próxima página

Tabela 6 – (Continuação)

Método	**URI**
GET, HEAD	</my/system-analysis>
POST	</my/system-analysis/{examId}>
GET, HEAD	</my/professional-analysis>
GET, HEAD	</professional-analysis>
POST	</professional-analysis>
GET, HEAD	</professional-analysis/{id}>
PUT	</professional-analysis/{id}>
DELETE	</professional-analysis/{id}>
GET, HEAD	</system-analysis>
POST	</system-analysis/{examId}>
	Dispositivos de Exame
GET, HEAD	</examination-devices>
POST	</examination-devices>
GET, HEAD	</examination-devices/{id}>
PUT	</examination-devices/{id}>
DELETE	</examination-devices/{id}>
	Rotas Auxiliares (Helpers)
GET, HEAD	</helpers/exam/status>
GET, HEAD	</helpers/exam/types>
GET, HEAD	</helpers/anamnesis/dental-discomfort/questions>
GET, HEAD	</helpers/examination-devices/models>

APÊNDICE D – DIAGRAMA ENTIDADE E RELACIONAMENTO COM ENTIDADES DO FRAMEWORK - V1.0.0-PRE-ALPHA

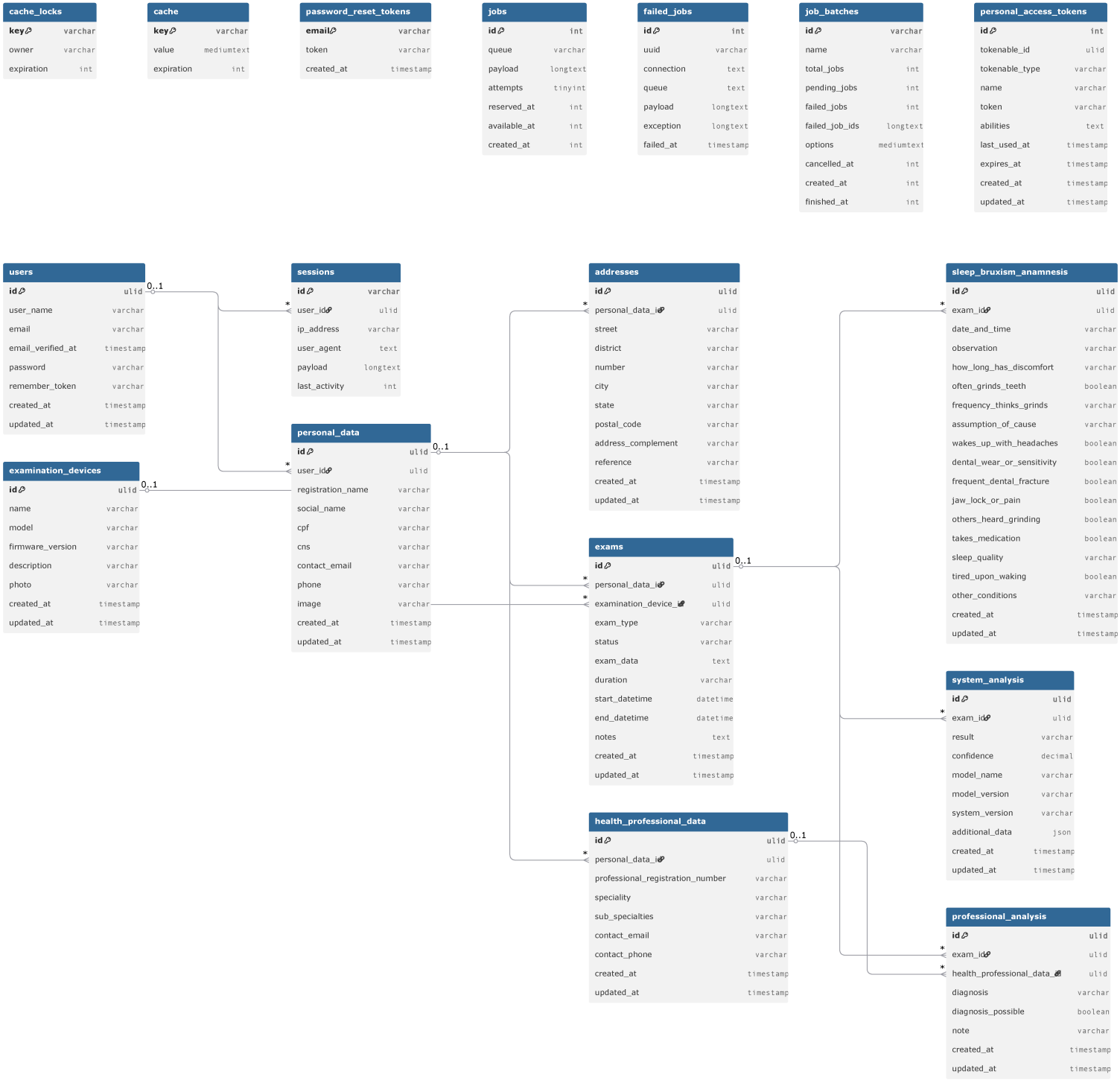
Diagrama entidade e relacionamento da versão 1.0.0-pre-alpha. A versão apresentada aqui teve sua última atualização adicionada em e inclui entidades específicas de funcionalidades do framework utilizado. Trata-se de uma boa representação da base de dados real do sistema.

Diagrama Entidade e Relacionamento (DER)

Inclui Entidades próprias do framework utilizado na aplicação backend

MORPHEUS HEALTHSYSTEM

@version 1.0.0-pre-alpha



Designed by Robson Gomes

robsongomes.me