INSTITUTO FEDERAL GOIANO – CAMPUS MORRINHOS

CURSO SUPERIOR DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PROTÓTIPO DE UMA API REST PARA GERENCIAMENTO E ENTREGA DINÂMICA DE ATIVOS 3D EM REALIDADE AUMENTADA

CARLOS EDUARDO DOMINGUES BONIFÁCIO CONRADO

MORRINHOS - GO
Fevereiro, 2026

CARLOS EDUARDO DOMINGUES BONIFÁCIO CONRADO

PROTÓTIPO DE UMA API REST PARA GERENCIAMENTO E ENTREGA DINÂMICA DE ATIVOS 3D EM REALIDADE AUMENTADA

Monografia apresentada ao Curso Superior de Bacharelado em Ciência da Computação do Instituto Federal Goiano – Campus Morrinhos, como requisito parcial para obtenção de título de Bacharel em Ciência da Computação.
Área de concentração: Ciência da Computação
Orientador: Dr. Alexandre Carvalho Silva

MORRINHOS - GO
Fevereiro, 2026

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema Integrado de Bibliotecas do IF Goiano - SIBi

Domingues Bonifácio Conrado, Carlos Eduardo
D671p Protótipo De uma API Rest Para Gerenciamento e Entrega Dinâmica de Ativos 3D em Realidade Aumentada / Carlos Eduardo Domingues Bonifácio Conrado. Morrinhos 2026.

50f. il.

Orientador: Prof. Dr. Alexandre Carvalho Silva.
Tcc (Bacharel) - Instituto Federal Goiano, curso de 0419204 - [MO.GRAD] Bacharelado em Ciência da Computação - Morrinhos (Campus Morrinhos).
1. API Rest. 2. Manufatura Enxuta. 3. Realidade Aumentada. 4. Indústria 4.0. I. Título.

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor: Carlos Eduardo Domingues Bonifácio Conrado

Matrícula: 2019104201940025

Título do trabalho: Protótipo De uma API Rest Para Gerenciamento e Entrega Dinâmica de Ativos 3D em Realidade Aumentada

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 05 / 03 / 2026

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☑ Sim ☐ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Morrinhos (Local) — 05 / 03 / 2926 (Data)

gov.br Documento assinado digitalmente
CARLOS EDUARDO DOMINGUES BONIFACIO CO
Data: 05/03/2026 12:18:28-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

gov.br Documento assinado digitalmente
ALEXANDRE CARVALHO SILVA
Data: 05/03/2026 13:58:58-0300
Verifique em https://validar.iti.gov.br

Assinatura do(a) orientador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 7/2026 - CCEG-MO/CEG-MO/DE-MO/CMPMHOS/IFGOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

ATA DE DEFESA DA BANCA DE EXAME DE TRABALHO DE CURSO POR VIDEOCONFERÊNCIA

Aos **2 3** dias do mês de fevereiro de **2026**, às **1 9** horas foi realizada a Banca de Exame, em formato remoto para a apresentação pública e defesa do trabalho de curso do discente CARLOS EDUARDO DOMINGUES BONIFÁCIO CONRADO intitulado como: PROTÓTIPO DE UMA API REST PARA GERENCIAMENTO E ENTREGA DINÂMICA DE ATIVOS 3D EM REALIDADE AUMENTADA **,** como requisito necessário para a conclusão do curso Bacharelado em Ciência da Computação - IFGoiano campus Morrinhos.

A Banca de Exame foi constituída pelos membros: **Prof. Dr. Alexandre Carvalho Silva (orientador)** , **Prof. Dr. Diogo Aparecido Cavalcante de Lima e Prof. Me. Angel Rodrigues Ferreira.** Após a análise, emitiram o seguinte resultado:

1 - (X) Aprovado

2 - ( ) Aprovado com ressalva

(A Banca Examinadora deve definir as exigências a serem cumpridas pelo aluno na revisão, ficando o orientador responsável pela verificação do cumprimento das mesmas.)

Observações:_____

3 - ( ) Reprovado com o seguinte parecer: _____ _____
_____

Morrinhos -GO, 23 de fevereiro de 2026.

Por ser verdade firmamos a presente:

(Assinado Eletronicamente)

Prof. Dr. Alexandre Carvalho Silva

Orientador(a)

(Assinado Eletronicamente)

Prof. Dr. Diogo Aparecido Cavalcante de Lima

Membro Externo - UFU

Documento assinado digitalmente
gov.br
DIOGO APARECIDO CAVALCANTE DE LIMA
Data: 04/03/2026 15:53:38-0300
Verifique em https://validar.iti.gov.br

(Assinado Eletronicamente)

Prof. Me. Angel Rodrigues Ferreira

Membro Interno
(Assinado Eletronicamente)

Documento assinado eletronicamente por:

- **Alexandre Carvalho Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 23/02/2026 20:08:44.
- **Angel Rodrigues Ferreira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 27/02/2026 11:44:15.

Este documento foi emitido pelo SUAP em 23/02/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 791849
Código de Autenticação: 1a96169de9

INSTITUTO FEDERAL GOIANO
Campus Morrinhos
Rodovia BR-153, Km 633, Zona Rural, SN, Zona Rural, MORRINHOS / GO, CEP 75650-000
(64) 3413-7900

DEDICATÓRIA

Dedico este trabalho a todas as pessoas que me apoiaram e acreditaram no meu potencial ao longo desta jornada. Em especial, à minha família, que sempre esteve ao meu lado com amor, paciência e incentivo. Agradeço também aos meus amigos, professores e orientador, que contribuíram com seu conhecimento, dedicação e sabedoria para a realização deste trabalho. Este trabalho de conclusão de curso é o reflexo de todos esses esforços, e cada passo dado foi possível graças ao apoio incondicional de todos que estiveram comigo.

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão a todas as pessoas que, de alguma forma, contribuíram para a realização deste trabalho.

Primeiramente, agradeço ao meu orientador, Alexandre, pela orientação, paciência e pelos valiosos conselhos ao longo de todo o processo de desenvolvimento deste TCC. Sua expertise foi essencial para o crescimento deste trabalho.

Agradeço também aos meus professores, que, ao longo da graduação, compartilharam seus conhecimentos e despertaram meu interesse pela área, além de oferecerem sempre apoio e incentivo.

A minha família, que foi meu alicerce durante toda essa jornada, merece um agradecimento especial. O apoio emocional, a compreensão e o amor incondicional que recebi de todos vocês foram fundamentais para que eu pudesse superar os desafios e conquistar mais essa etapa da minha vida.

Aos meus amigos, que me acompanharam nos momentos de alegria e nos de dificuldade, muito obrigado pelo apoio e pela amizade.

A todos que, de alguma forma, contribuíram para a realização deste trabalho, meu muito obrigado.

SUMÁRIO

RESUMO

Com a crescente competição no mundo corporativo, proveniente do advento da chamada Indústria 4.0 ou Quarta Revolução Industrial, viu-se uma necessidade de alavancar os processos industriais, fornecendo mecanismos tecnológicos que auxiliem e potencializem a otimização desses processos. É com base nisso que este trabalho se fundamenta, buscando através da junção da Manufatura Enxuta, que nada mais é do que a otimização de do serviço visando maximizar o valor para o cliente, e da I4.0 a implementação de uma tecnologia desenvolvida utilizando a interface API Rest aplicada a um sistema de Realidade Aumentada que possa auxiliar no controle e manutenção de equipamentos da Indústria Elétrica. Para isso, foi desenvolvido uma API Rest completa com ênfase na escalabilidade do projeto, onde primeiramente foi estabelecido as ferramentas a serem utilizadas, ferramentas totalmente de uso gratuito e a partir disso seguindo o passo a passo da modelagem à implementação do projeto. Para os testes da aplicação foi decidido a utilização do Teste de Caixa Preta, onde foi selecionado diferentes funcionalidades para se analisar as respostas esperadas e as respostas obtidas. Com a execução deste projeto ao final foi possível concluir a extrema praticidade e a adaptatividade dos sistemas que utilizam o estilo arquitetônico API Rest, possibilitando ao desenvolvedor uma dinamicidade frente a futuras manutenções e inovações no projeto.

Palavras-chave: API Rest, Manufatura Enxuta, Realidade Aumentada, Indústria 4.0.

ABSTRACT

With the growing competition in the corporate world, arising from the advent of so-called Industry 4.0 or the Fourth Industrial Revolution, there has been a need to leverage industrial processes by providing technological mechanisms that assist and enhance their optimization. Based on this context, this study is grounded in the integration of Lean Manufacturing—which is essentially the optimization of services aimed at maximizing value for the customer—and Industry 4.0, through the implementation of a technology developed using a REST API interface applied to an Augmented Reality system that can assist in the control and maintenance of electrical industry equipment. For this purpose, a complete REST API was developed with an emphasis on project scalability. Initially, the tools to be used were established, all of which were completely free, and from that point onward, the project followed a step-by-step process from modeling to implementation. For application testing, Black Box Testing was adopted, in which different functionalities were selected to analyze the expected and obtained responses. With the execution of this project, it was possible to conclude the high level of practicality and adaptability of systems that use the REST API architectural style, enabling developers to maintain flexibility when facing future maintenance and innovation demands.

Keywords: REST API, Lean Manufacturing, Augmented Reality, Industry 4.0.

1 – INTRODUÇÃO

A era das revoluções[1] foi um período que deixou marcas na história da humanidade. Não obstante foi a revoluções industriais[2] com suas fases que apresentou para a humanidade, inúmeras mudanças para a sociedade, no campo social, econômico e tecnológico. A primeira grande revolução foi marcada pela mudança do método de produção que passou de forma artesanal para a mecanizada, tendo adoção da máquina a vapor. Já a segunda revolução foi marcada pela chegada da eletricidade, da divisão de trabalho e da produção em massa no sistema produtivo. Por fim, após o final da Segunda Guerra Mundial têm-se o início da terceira revolução industrial, que também é conhecida como Revolução Técnico Científica Informacional, que se dá até os dias atuais.

Com a chegada dessa terceira revolução têm-se uma crescente competição do mundo das indústrias, proveniente do advento da chamada Indústria 4.0 — compreendida como a quarta revolução industrial, marcada pela integração de tecnologias digitais, físicas e biológicas aos processos produtivos, tais como Internet das Coisas (IoT), inteligência artificial, big data, computação em nuvem e sistemas ciberfísicos, promovendo maior automação, conectividade e eficiência — demandando a necessidade da criação de mecanismos que potencializem a otimização das atividades, tendo como premissa as atividades industriais. Essa abordagem teórica fundamenta-se nas contribuições de Hobsbawm (1962), ao abordar as transformações oriundas da Primeira e Segunda Revoluções Industriais; de Castells (1996), ao discutir a Revolução Técnico-Científica-Informacional; e de Schwab (2016), ao analisar os impactos da Indústria 4.0 no cenário produtivo contemporâneo.

---

[1] A **Era das Revoluções** é um período histórico que abrange aproximadamente os séculos XVIII e XIX, caracterizado por uma série de mudanças significativas e transformadoras em várias áreas, como política, economia, sociedade e tecnologia. Esse período foi marcado por várias revoluções que desafiaram as estruturas de poder existentes, promoveram novas ideias e alteraram profundamente as organizações sociais e políticas em muitas partes do mundo.

[2] **Revolução Industrial (século XVIII - XIX)**: Iniciada na Inglaterra, essa revolução foi marcada pela transição de uma economia agrícola e artesanal para uma economia baseada na produção em larga escala e no uso de máquinas. A introdução de novas tecnologias, como a máquina a vapor e o tear mecânico, transformou a produção e teve impactos profundos nas cidades, nas relações de trabalho e no modo de vida das pessoas.

Esta pesquisa tem como objetivo apresentar as vantagens da utilização de uma arquitetura baseada em API Rest no desenvolvimento de sistemas robustos e escaláveis. É a partir deste conceito que este trabalho se fundamenta, buscando a partir da elaboração de um protótipo de API Rest fornecer uma melhor praticidade na utilização de ferramentas de Realidade Aumentada — compreendida como uma tecnologia que integra elementos virtuais ao ambiente físico em tempo real, por meio de dispositivos como smartphones, tablets ou óculos inteligentes, permitindo a sobreposição de objetos digitais interativos ao mundo real — tendo em vista que a atividade de customização de um ambiente de RA acaba sendo uma atividade muito verbosa e trabalhosa que, quando aplicada à indústria, pode-se acarretar alguns gargalos, como a maior verbosidade na criação de cenários de Realidade Aumentada, exigindo múltiplas configurações, parametrizações e integrações técnicas para cada novo ambiente desenvolvido.

O modelo arquitetural API Rest se baseia na divisão de diferentes serviços que devem ser independentes e que em seu próprio processo possa garantir o funcionamento de todo um sistema. Para que isso ocorra de maneira correta a comunicação entre esses serviços é realizada através de comunicação HTTP (Hypertext Transfer Protocol), REST (Representational State Transfer) e Web APIs (Application Programming Interface) (VILLAÇA; PIMENTA JR; AZEVEDO; 2018). Na atual Revolução Industrial, expressões como Big Data — caracterizado pelo grande volume, variedade e velocidade de dados gerados e processados para extração de informações estratégicas —, descentralização — entendida como a distribuição da tomada de decisão e do processamento entre diferentes sistemas ou unidades autônomas —, virtualização — que consiste na criação de versões virtuais de recursos físicos, como servidores, máquinas e ambientes industriais —, digitalização — processo de conversão de informações e processos físicos em formatos digitais para otimizar, integrar e automatizar operações — e IoT Industrial — aplicação da Internet das Coisas no contexto industrial, permitindo a conexão de máquinas, sensores e dispositivos para coleta e troca de dados em tempo real — são muito comuns, e a maioria das tecnologias necessárias para a aplicação dela já existem (BIGHETI, et. al., 2020).

> O grande desafio, portanto, é promover a integração entre essas tecnologias, visando a obtenção de uma nova realidade produtiva, onde tudo estará conectado para que as melhores decisões de produção, custo e segurança sejam tomadas, sob demanda e em tempo real. (BIGHETI, el. al., 2020, p. 1).

Para realizar a aplicação deste conceito API Rest foi feita uma elaboração de um protótipo de serviço de cadastro e consultas de objetos de um ambiente de Realidade Aumentada, no exemplo apresentado neste projeto foi utilizado o cenário de uma subestação da indústria de energia elétrica. Para este serviço o usuário precisa se cadastrar no sistema para obter o seu Token (permissão de acesso ao sistema) e poder realizar as ações de cadastro e consulta de equipamentos presentes no serviço da subestação de energia elétrica, com isso será possível fornecer uma integração entre tecnologias de Realidade Aumentada e o sistema de controle da usina elétrica. Assim, o usuário final do projeto terá uma melhor imersão no ambiente industrial, potencializando e otimizando as atividades rotineiras. Dessa forma, o protótipo aqui apresentado corrobora com a dinâmica e necessidades da industria e aplicação serviços que se demanda na atualidade.

2 – OBJETIVOS

2.1 – OBJETIVO GERAL

O Objetivo Geral deste trabalho é desenvolver um protótipo de um serviço backend[3], utilizando o modelo arquitetural API Rest, que possibilite maior facilidade e eficiência no processo de customização de ambientes de Realidade Aumentada. Para o presente projeto foi tomado como exemplo prático a customização de um ambiente de uma subestação elétrica, onde o usuário poderá controlar e consultar informações importantes em relação ao seu maquinário

[3] O **backend** refere-se à parte do desenvolvimento de software que lida com a lógica de negócio, o processamento de dados e a interação com o banco de dados, e que não é diretamente acessada pelos usuários. Em outras palavras, é a camada de um sistema que fica "por trás" da interface visual (frontend) e é responsável por garantir que as solicitações feitas pelos usuários sejam atendidas corretamente.

2.2 – ETAPAS DO DESENVOLVIMENTO METODOLÓGICO

2.2.1 – Levantamento do caso de uso.

O levantamento de Casos de Uso é a etapa inicial do processo de definição dos requisitos de um sistema e antecede tanto a modelagem quanto o estudo detalhado dos Casos de Uso. Nessa fase, busca-se identificar, de forma ampla, quais funcionalidades o sistema deve oferecer e como os usuários irão interagir com ele.

Esse levantamento é fundamental, pois serve como base para todas as etapas seguintes do desenvolvimento, garantindo que o projeto comece com informações claras e bem estruturadas.

2.2.2 – Estudo do caso de uso.

Os estudos de Caso de Uso são uma etapa importante no processo de desenvolvimento de sistemas, pois permitem analisar, detalhar e validar como os usuários interagem com uma aplicação. Eles vão além da simples identificação das funcionalidades, descrevendo de forma organizada o comportamento do sistema em situações reais de uso.

2.2.3 – Elaboração de um diagrama de caso de uso.

A modelagem de Casos de Uso é uma etapa fundamental no desenvolvimento de uma API REST, pois ajuda a compreender, organizar e documentar como o sistema será utilizado pelos seus usuários ou por outros sistemas. Mesmo sendo uma técnica tradicional da engenharia de software, ela continua sendo muito importante no contexto atual de aplicações baseadas em serviços.

2.2.4 – Pesquisa de material para bibliografia correlata.

A pesquisa de material para bibliografia correlata é importante porque permite compreender o que já foi estudado sobre um determinado tema, evitando informações incorretas ou superficiais. Por meio dessa pesquisa, é possível conhecer conceitos, teorias e experiências que ajudam a construir um conhecimento mais sólido.

2.2.5 – Elaboração da parte prática.

A elaboração da parte prática, especialmente no desenvolvimento de uma API REST, é uma etapa fundamental, pois transforma o planejamento teórico em uma solução funcional. É nesse momento que os conceitos estudados são aplicados na criação de um sistema real, capaz de atender às necessidades dos usuários.

Inicialmente, essa fase envolve a definição da estrutura da API, com base nos requisitos levantados e nos Casos de Uso. A partir disso, são planejados os recursos, os endpoints, os métodos HTTP e os formatos de resposta. Esse cuidado inicial garante organização, padronização e facilidade de uso.

Em seguida, ocorre a implementação, na qual são utilizadas linguagens, frameworks e ferramentas adequadas ao projeto. Nessa etapa, são criadas as rotas, as regras de negócio, a conexão com o banco de dados e os mecanismos de autenticação e segurança. O objetivo é garantir que a API funcione de forma estável, segura e eficiente.

2.2.6 – Realização de Testes da parte prática.

A realização de testes na parte prática é muito importante no desenvolvimento de uma API REST, pois garante que o sistema funcione corretamente antes de ser utilizado pelos usuários. Por meio dos testes, é possível verificar se todas as funcionalidades estão respondendo como esperado.

Uma das principais importâncias dos testes é a identificação de erros. Durante o desenvolvimento, é comum que ocorram falhas na programação, nos dados ou na comunicação entre sistemas. Os testes permitem encontrar esses problemas de forma antecipada, evitando falhas em produção.

Além disso, os testes ajudam a garantir a qualidade do sistema. Ao simular diferentes situações de uso, é possível confirmar se a API trata corretamente dados inválidos, acessos não autorizados e exceções. Isso torna o sistema mais seguro e confiável.

Outro ponto relevante é a redução de retrabalho. Quando os erros são detectados cedo, eles podem ser corrigidos com mais facilidade, economizando tempo e esforço. Isso torna o processo de desenvolvimento mais eficiente.

Os testes também contribuem para a estabilidade da aplicação. Sempre que uma nova funcionalidade é adicionada ou modificada, é possível testar novamente o sistema para verificar se nada foi afetado. Dessa forma, evita-se que mudanças causem novos problemas.

De modo geral, a realização de testes é essencial para garantir que a API funcione bem, seja segura, apresente bom desempenho e atenda às expectativas dos usuários, aumentando a confiança no sistema desenvolvido.

3 – REFERENCIAL TECNOLÓGICO

Neste capítulo serão apresentados os tópicos referentes às tecnologias utilizadas para a elaboração deste projeto. A seleção destes visa trazer para o leitor um melhor entendimento frente ao desenvolvimento de sistemas backend, mostrando-lhes as tecnologias utilizadas para o desenvolvimento do protótipo apresentado neste trabalho. No total, serão 9 tópicos, que vão desde a linguagem de programação escolhida para o desenvolvimento, no caso o JavaScript, passando pelo ambiente de desenvolvimento, que por se tratar de um projeto para conclusão de curso e, portanto, sem fins lucrativos, foi escolhido o Visual Studio, que é uma IDE de uso gratuito disponibilizada e desenvolvida pela Microsoft, até as ferramentas, também chamadas de frameworks, utilizadas para acrescentar algumas funcionalidades ao nosso serviço final, a saber:

•NodeJS;
•DotEnv;
•Cors;
•Bcrypt;
•JWT;
•Postman;
• MongoDB.

3.1 – VISUTAL STUDIO

A IDE escolhida para o desenvolvimento do projeto foi o Visual Studio Code primeiramente por se tratar de um editor de código gratuito, fornecer diversas extensões que possibilitam um melhor controle do projeto e também pelo fato de suportar diversas linguagens como Python, Java, C++, JavaScript, dentre outras (VISUAL STUDIO CODE, 2023). O Visual Studio é um ambiente de desenvolvimento integrado (IDE, na sigla em inglês) criado pela Microsoft para o desenvolvimento de software. Ele é uma plataforma abrangente que oferece

diversas ferramentas e recursos para programadores criarem, depurarem e testarem aplicativos em diferentes linguagens de programação.

3.2 – JAVA SCRIPT

O Java Script foi criado em 1995 pelo programador Brendan Eich, que na época trabalhava para o Netscape. O JS primeiramente foi criado visando tornar a navegação na internet mais dinâmica (ROVEDA, 2021), por isso sua popularidade no desenvolvimento web. Porém, com o passar do tempo novas ferramentas para o JS foram sendo desenvolvidas, chamadas de frameworks, possibilitando assim que a linguagem expandisse os seus horizontes podendo a partir disso trabalhar tanto na parte de front-end, utilizando o framework Angular, por exemplo, como na parte de back-end utilizando o NodeJS.

3.3 – NODEJS

O JavaScript como é sabido, trata-se de uma linguagem client side, ou seja, toda a sua execução é realizada do lado do usuário, sem utilizar um processamento realizado em um servidor externo por exemplo (ROVEDA, 2021). Em contrapartida a isso, surge o NodeJS, que é um ambiente de execução de código JavaScript server-side, isso possibilita a criação de aplicações standalone, que são aplicações autossuficientes, em um servidor externo (BESSA, 2023).

> O Node.js é a ferramenta que vai nos entregar a capacidade de interpretar código JavaScript, de maneira bem similar ao navegador. Quando executamos um comando escrito em JavaScript, o Node.js interpreta esse comando e faz a sua conversão para a linguagem de máquina a ser executada pelo computador. Por esse motivo, o Node.js também pode ser referido como um JavaScript Runtime, ou um programa de execução do JavaScript (BESSA, 2023).

3.4 – DOTENV

O Dotenv é uma biblioteca que serve para gerenciar as variáveis de ambiente presentes em um projeto de NodeJS, o gerenciamento dessas variáveis é feito de forma com que as configurações das variáveis sejam armazenadas em um ambiente separado do código da aplicação (GOMES, 2023). Como as variáveis de ambiente guardam dados essenciais do sistema, utilizando um ambiente separado que armazene esses dados, que muitas vezes são chaves de acessos a serviços como banco de dados por exemplo ou APIs, é uma boa prática de programação e indispensável no desenvolvimento de aplicações reais.

3.5 – CORS

OO Cors é uma ferramenta utilizada em desenvolvimento de APIs que permitem adicionar cabeçalhos HTTP que informam os navegadores, permitindo que uma aplicação web seja executada em uma origem e acesse recursos de outra origem (HILLMAN, 2023). De forma mais detalhada, o CORS (Cross-Origin Resource Sharing) atua como um mecanismo de segurança implementado pelos navegadores, cujo objetivo é controlar o compartilhamento de recursos entre diferentes domínios, protocolos ou portas. Por meio da configuração adequada de cabeçalhos HTTP, como *Access-Control-Allow-Origin*, *Access-Control-Allow-Methods* e *Access-Control-Allow-Headers*, é possível definir quais origens externas estão autorizadas a consumir os recursos da API, quais métodos HTTP podem ser utilizados e quais informações podem ser transmitidas. Dessa maneira, além de viabilizar a comunicação entre aplicações distintas, o CORS também contribui para a proteção contra requisições maliciosas e acessos não autorizados, garantindo maior segurança e controle no desenvolvimento de sistemas distribuídos.

3.6 – BCRYPT

Tendo em vista que o para utilizar o sistema desenvolvido o usuário necessita realizar um cadastro de login, foi preciso pensar numa forma de tornar essas informações de login, como a senha, salva de maneira segura no banco de dados e a maneira escolhida foi a utilização da biblioteca bcrypt. O Bcrypt trata-se de um framework desenvolvido por Niels Provos e David Mazières que tem o propósito de “camuflar” senhas criadas por usuários, transformando o texto puro em dados indecifráveis através do algoritmo hash (KERLYN, 2019).

3.7 – JWT

Pelo fato de o usuário precisar realizar um login para ter acesso a determinados serviços da API, é necessário que este tenha a confirmação dessa permissão e isso é realizado utilizando a validação de token. E para isso foi escolhido o framework JWT

> Um JWT é um padrão para autenticação e troca de informações definido pela RFC7519. Nele é possível armazenar de forma segura e compacta objetos JSON. Este token é um código Base64 e pode ser assinado usando um segredo ou par de chaves privadas/públicas (SEGUINS, 2022).

3.8 – Postman

Como o presente trabalho trabalha com requisições REST é necessário que os end points do projeto sejam testados de alguma forma. Uma maneira possível de realizar esses testes é pelo próprio navegador utilizando o localhost e a porta pré-determinada, porém, essa não é maneira mais eficaz e correta de fazer isso. A forma mais correta é utilizar softwares de terceiros que possam realizar esses testes, e um desses é o Postman. O Postman é uma ferramenta que além de auxiliar o desenvolvedor a construir, testar, documentar e compartilhar as APIs ele também permite que as respostas das requisições HTTP e HTTPS sejam salvas de forma detalhada (URANO, 2022).

3.9 – MONGODB

Tendo em vista que os resultados obtidos do projeto não resultariam em um esquema fixo de dados, foi definido que seria utilizado um banco de dados NoSQL, que são bancos não relacionais e que têm os seus dados armazenados de forma diferente das tabelas relacionais (ORACLE, 2023), portanto o banco escolhido foi o MongoDB. O MongoDB diferencia dos outros sistemas tradicionais de banco de dados, onde ao invés de tabelas, linhas e colunas, a base para o armazenamento são documentos os quais são modelados utilizando a formatação JSON (LIMA, 2021).

4 – TRABALHOS RELACIONADOS

4.1 – AUGMENTED REALITY APPLICATION TO SUPPORT REMOTE MAINTENANCE AS A SERVICE IN THE ROBOTICS INDUSTRY

Neste artigo MOURTZIS, ZOGOPOULOS, VLACHOU (2017) nos apresenta uma plataforma de manutenção remota com Realidade Aumentada (RA) voltada para o Sistemas de Produto-Serviço (PSS) onde é combinado a tecnologia em nuvem e algoritmos de automação com intuito de aprimorar o serviço de manutenção durante o ciclo de vida dos produtos.

A Realidade Aumentada é trabalhada nesse artigo de forma que em seu projeto ela possa fornecer instruções visuais interativas, de forma que o suporte se torne mais acessível para os técnicos com diferentes níveis de experiência o que faz com que o tempo e também o custo da manutenção sejam reduzidos consideravelmente, tornando o comprador desse produto mais competitivo frente ao mercado.

O conceito de PSS é apresentado pelos autores como sendo uma estratégia de negócio onde uma empresa não oferecerá um único produto em si, mas também uma gama de serviços integrados com ele, combinando o produto com seus serviços a longo do seu ciclo de vida, ponto este que se faz necessário o estudo de uma possível utilização de API Rest, uma vez que apresentado no artigo a existência da complexidade na criação de cenas de RA, havendo a necessidade de reduzir a carga cognitiva dos designers de cena.

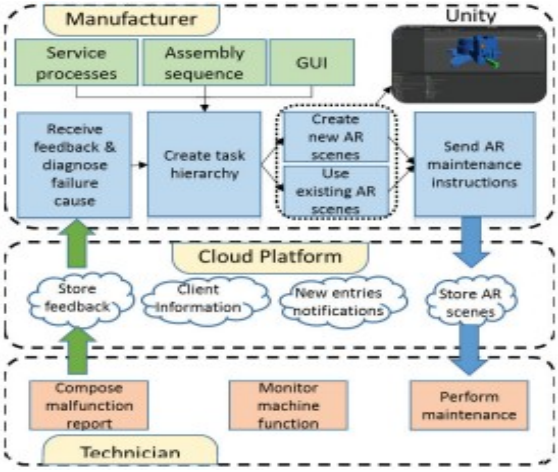

Figura 01 (Fonte: BIGHETI, 2020).

4.2 – PROPOSTA DE ARQUITETURA ORIENTADA A SERVIÇOS PARA APLICAÇÕES DE INTERNET DAS COISAS INDUSTRIAL

Neste trabalho BIGHETI, RISSO, CALDIERI (2020) começam apresentando uma revisão sobre a Arquitetura Orientada a Serviços e a sua aplicação na indústria. Após essa breve revisão os autores introduzem o leitor ao conceito de Microsserviços, mostrando essa arquitetura aplicada à Internet das Coisas Industrial (IIoT).

Como dito pelos autores o artigo apresenta uma nova proposta de arquitetura orientada a microsserviços para a aplicação de IIoT. Nesta arquitetura foi utilizado o framework Moleculer, trabalhando o desenvolvimento de diferentes atividades relacionadas à IIoT. O presente projeto busca fornecer os seguintes serviços relacionados a IioT:Aquisição de Dados (DAQ), Monitoramento Remoto(Dashboard), Otimização (KPIs), Controle de Processo, Identificação de Sistemas, Detecção de falhas ou anomalias e Mecanismo de Redundância. Além de apresentar essas propostas, os autores deixam claro que a arquitetura, justamente por estar baseada em microsserviços, pode ser facilmente expandida para permitir novas aplicações, como por exemplo a de virtualização, simulação de sistemas, segurança da informação, realidade aumentada e análise de dados utilizando técnicas de Big Data.

É apresentado a imagem (Figura 01) onde fica claro o principal componente operacional da arquitetura, o Service Broker, que se trata de um container de microsserviços responsável pela configuração dos nós como: nome, opção de comunicação, registros de microsserviços, descoberta automática de serviços e a configuração adicionais de trasnporter (a forma como esses microsserviços irão se comunicar) como endereço de IP e porta de comunicação. Cada serviço possuí os seguintes atributos que são o Name, Settings, Actions, Methos e Events, as mais importantes e que devem observadas com mais carinho pelo leitor são os atributos Settings e Actions, que serão responsáveis pela configuração das portas de comunicação e as ações ou funções que compõem o seu funcionamento, respectivamente.

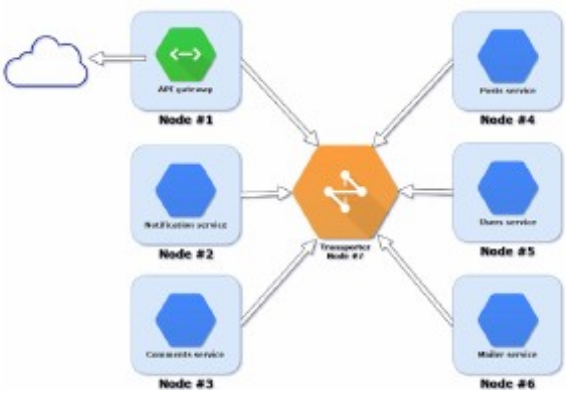

Figura 02 (Fonte: BIGHETI, 2020).

Cada serviço da arquitetura MOA oferecerá uma funcionalidade principal, que poderá ter outras diferentes funcionalidades relacionadas à sua principal. Bigheti, nos apresentam de forma resumida a funcionalidade de cada microsserviço presente na figura (Figura 02) com suas respectivas ações:

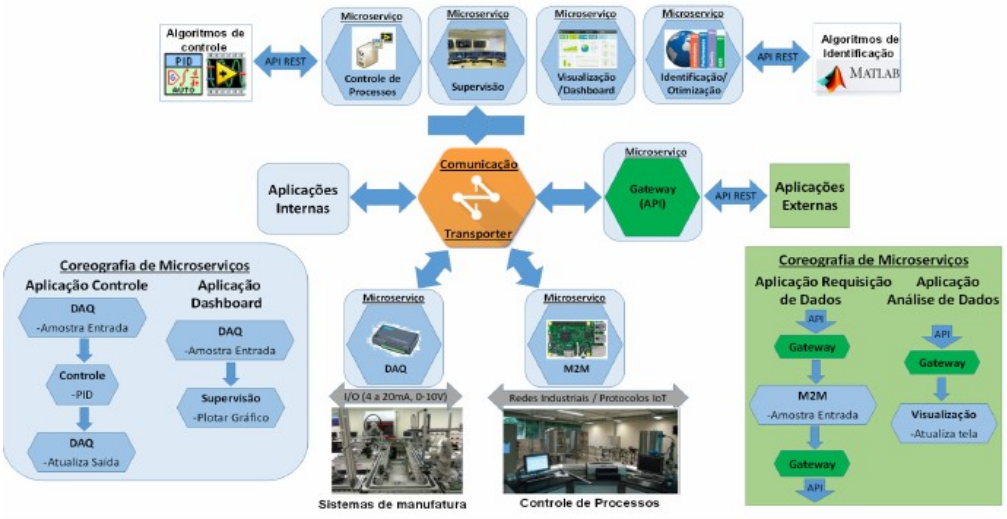

Figura 03 (Fonte: BIGHETI, 2020).

• **Serviço de Aplicação:** esse serviço será responsável por fornecer as funcionalidades principais do sistema como Monitoramento, Controle de Processos, Otimização e etc.

• **Serviço Gateway (API):** Este diferente dos outros é um serviço de conexão padronizado de aplicação/requisições externas ao microsserviços da arquitetura.

• **Serviço M2:** Este é o serviço de comunicação em rede com máquinas e sistemas. Semelhante a um middleware ele será responsável pela comunicação utilizando

diferentes redes e protocolos como Modbus, TCP/IP, CoAP, MQTT, etc. Portanto, este microsserviço disponibilizará duas ações, são elas: Aquisição de Dados de Entrada e a Atualização de Dados de Saída.

• **Serviço DAQ:** Este é um serviço de aquisição de dados de módulos de hardware alocados ao processo. Possuí assim como o serviço M2M duas ações, são elas: Aquisição de Dados de Entrada e Atualização de Dados de Saída.

Referente às aplicações internas, elas são desenvolvidas em Node.js e trabalha com os serviços de forma direta usando a comunicação via transporter. BIGHETI, RISSO, CALDIERI (2020) ressaltam que aplicações externas podem ser desenvolvidas em qualquer plataforma, porém devem sempre acessar e coreografar os serviços via comunicação REST (API) através do serviço Gateway.

Por fim, é apresentado que a aplicação trouxe diversos benefícios como:

• **Aumento da Resiliência:** Por se tratar de uma arquitetura MOA toda a aplicação é descentralizada e desacoplada em serviços que atuam de forma independente.

• **Escalabilidade:** Sendo um dos pilares da arquitetura de microsserviços cada serviço é um componente separado, permitindo desta forma expandir uma única funcionalidade ou serviço sem ter que modificar toda a aplicação.

• **Disponibilidade:** Outro grande pilar da arquitetura de serviços é a disponibilidade deles, neste exemplo os serviços críticos para a aplicação podem ser implementados em vários nós ou servidores para aumentar a disponibilidade e o desempenho sem afetar o mesmo dos outros.

• **Flexibilidade:** Com a adoção da arquitetura o desenvolvimento não fica limitado a uma única linguagem ou um único software, portanto, é possível escolher a melhor ferramenta para cada tarefa sem que haja problemas de incompatibilidades, justamente pelo fato de cada serviços trabalhar de forma independente.

• **Facilidade de Desenvolvimento e Modularidade:** A criação dos serviços se torna mais simples e mais rápida pelo fato de que cada serviço executar uma funcionalidade específica, desta forma ao adotar essa abordagem de

desenvolvimento a equipe de desenvolvimento pode ser dividida de acordo com cada serviço, tornando o processo mais ágil e fácil.

Com a apresentação de todos esses benefícios ao adotar a arquitetura de serviços para o desenvolvimento de um sistema para aplicações IIoT eu senti falta de algo que torne o processo mais fácil e intuitivo quando aplicado diretamente na indústria, possibilitando uma utilização mais lúdica e intuitiva por parte do trabalhador. É desta forma que a aplicação de um serviço voltado para a Realidade Aumentada serviria de forma complementar ao presente projeto, tirando a limitação da verbosidade da operação da mesma no ambiente de trabalho.

4.3 – PLATAFORMA DE E-COMMERCE INTEGRADA A MICRO SERVIÇOS

Neste trabalho o autor, ALMEIDA (2022) nos traz um exemplo da arquitetura de microsserviços aplicada à uma plataforma de e-commerce, buscando a partir disso apresentar ao leitor toda a fase de desenvolvimento de um software baseado em serviços, mostrando também como é feita a interação dos serviços, de uma forma que o sistema possa funcionar tanto em um ambiente de produção quanto num ambiente acadêmico. Para o desenvolvimento utilizou as tecnologias React e NextJS para o front-end e NodeJS para o back-end.

Logo no início de sua revisão bibliográfica o autor busca esclarecer os motivos para a adoção do NodeJS para o back-end de seu projeto, citando (GONZALEZ), “O NodeJS é o candidato perfeito para arquiteturas microsserviços por uma série de razões, conforme indicado na lista a seguir”:

- Fácil de aprender (embora possa ser difícil de dominar)
- Fácil de escalar
- Altamente testável
- Fácil de implantar
- Gerenciamento de dependências por meio de npm 1

• Existem centenas de bibliotecas para integrar com a maioria dos padrões protocolos"

Para o front-end utilizando o React junto com o NextJS utilizando nele o Typescript para que o código pudesse ser devidamente tipado ele justifica a utilização das mesmas pelo fato de o React ser uma tecnologia que possibilita uma melhor dinamicidade para o projeto, justamente pelo próprio motivo de sua criação, onde Almeida nos informa que uma equipe do Facebook em 2011, com o objetivo de otimizar a atualização e sincronização das atividades nos feeds de notícias da rede criaram o React.

Para o banco de dados foi escolhido banco de dados relacionais, o autor optou por escolher o banco de dados PostgreSQL por se tratar de um programa Open Source, no quesito de banco de dados, pelo fato dele ser Open Source o usuário tem acesso aprimorado aos números de benchmarking, que consiste no processo de busca das melhores práticas para melhorar o desempenho do serviço, acesso este que não é disponibilizado por empresas como a Oracle.

Assim sendo, por se tratar de um banco de dados relacional, o primeiro passo do autor foi a modelagem do modelo ralacional dos três diferentes bancos que podem ser visto nas figuras (Figura 03 - MODELO RELACIONAL) . A razão da modelagem de três diferentes bancos se dá pela necessidade de cada back-end possuir um banco próprio e sem conexões com os outros, conceito primordial da arquitetura de microsserviços.

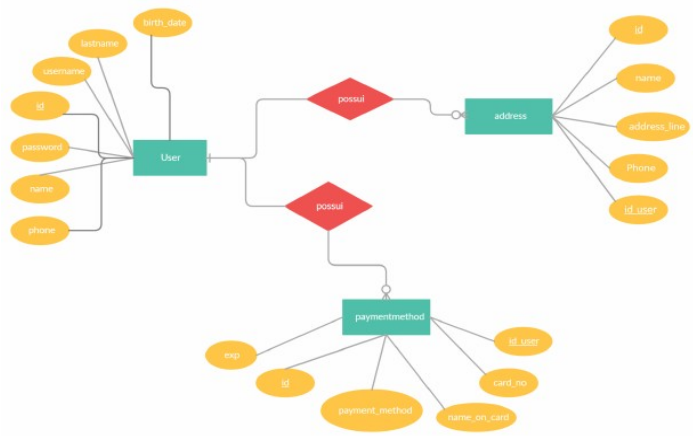

Figura 04 - Modelo relacional banco de dados de usuário (Fonte: ALMEIDA, 2022).

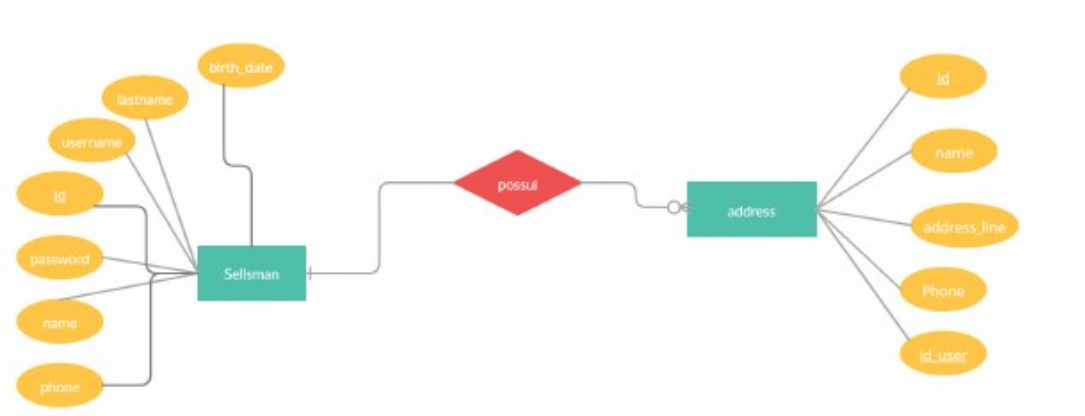

Figura 05 - Modelo relacional banco de dados de vendedor (Fonte: ALMEIDA, 2022).

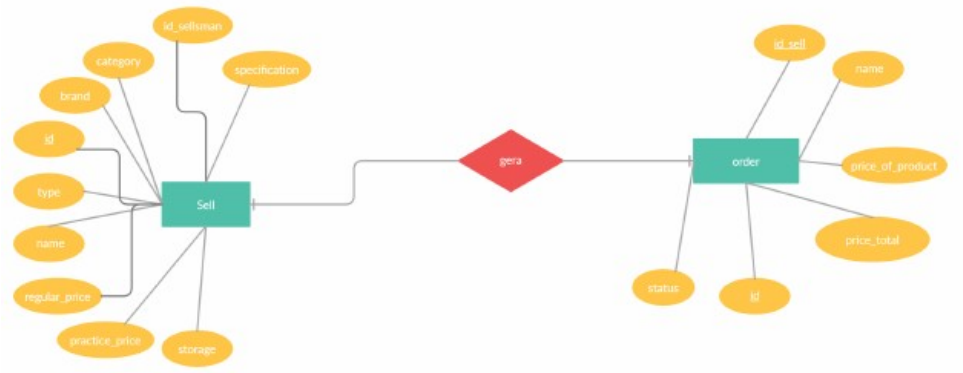

Figura 06 - Modelo relacional banco de dados de compras (Fonte: ALMEIDA, 2022).

Cada serviço utliza Endpoints para poderem se comunicar, dessa forma o front-end faz as requisições ao back-end que retorna as informações necessárias. Por se tratar de uma aplicação de e-commerce alguns end-points só poderão ser acessados mediante tokens de validação, tokens estes que serão disponibilizados para cada usuário, de forma encapsulada, através do login. Para realizar essa verificação do token são utilizados Middlewares, que são códigos de verificação que as requisições precisam passar antes de acessarem um endpoint específico, trazendo uma maior segurança e confiabilidade no processo da aplicação. Neste projeto o () criou um middleware chamado auth, que verifica se o usuário está logado ou não através do token de acesso enviado pelo header de cada requisição. Cada token é gerado pela biblioteca jsonwebtoken, que utiliza de uma palavra

secreta para gerar tokens de forma aleatória que têm duração de 24 horas após a sua autenticação.

O resultado obtido pelo autor foi satisfatório segundo o mesmo, conseguindo um bom acoplamento entre todas as tecnologias pois todas as funções do e-commerce funcionaram como esperando e de forma independente, sendo fiel ao conceito de microsserviços. O maior gargalo encontrado foi no deploy da aplicação em uma mesma máquina, onde os serviços de back-end e de front-end estão em um mesmo host, desta forma caso ele caia isso fará com que todos os serviços também caiam. Desta forma, têm-se a necessidade de separar os serviços de front-end e back-end em máquinas diferentes, pois desta forma, caso um serviço caia, os demais continuarão funcionando perfeitamente seguindo com o princípio principal de microsserviços.

5 – CONSIDERAÇÕES FINAIS

O desenvolvimento da API REST proposto neste trabalho permitiu aplicar, na prática, os conhecimentos relacionados à engenharia de software, levantamento de requisitos, modelagem de Casos de Uso, implementação e testes. A construção da solução possibilitou compreender, de forma concreta, os desafios e benefícios envolvidos no desenvolvimento de sistemas baseados em serviços, evidenciando a importância de um planejamento adequado e de boas práticas de programação.

Entre as principais potencialidades, destaca-se a facilidade de escalabilidade proporcionada pela arquitetura REST. Por se basear em princípios como stateless, separação entre cliente e servidor e uso de protocolos padronizados, a API permite que o sistema seja expandido de forma modular e flexível. Além disso, sua compatibilidade com diferentes plataformas favorece a integração com aplicações web, mobile e outros sistemas, ampliando as possibilidades de crescimento do projeto. O uso de endpoints bem definidos, versionamento de serviços e balanceamento de carga também contribui para a manutenção da performance mesmo com o aumento no volume de acessos.

Outro ponto positivo é a organização do sistema, proporcionada pela padronização das rotas, dos métodos HTTP e dos formatos de resposta, o que facilita a manutenção, a reutilização de código e a evolução contínua da aplicação. Essa estrutura torna o projeto mais preparado para futuras melhorias e adaptações.

Em relação às limitações, observa-se que o desenvolvimento de uma API REST exige um cuidado elevado com aspectos como segurança, autenticação, autorização e proteção de dados. A implementação de mecanismos como tokens, criptografia e controle de acesso pode aumentar a complexidade do sistema, especialmente em projetos com recursos limitados. Além disso, a dependência da conexão com a internet pode impactar o desempenho em ambientes instáveis.

Outra limitação está relacionada à gestão de erros, versionamento e compatibilidade com diferentes clientes. À medida que a API evolui, torna-se mais difícil manter a consistência entre versões, evitando que alterações prejudiquem sistemas já integrados. Também é necessário um esforço contínuo na

documentação e nos testes, para garantir que as mudanças não comprometam a estabilidade da aplicação.

Dessa forma, conclui-se que o desenvolvimento da API REST apresentou resultados positivos, principalmente no que se refere à escalabilidade, flexibilidade e organização do sistema. Ao mesmo tempo, evidenciou desafios técnicos que exigem planejamento, conhecimento especializado e manutenção constante. Mesmo com essas limitações, a solução desenvolvida demonstra grande potencial para expansão e aprimoramento, consolidando-se como uma base sólida para futuros desenvolvimentos.

6 – ARQUITETURA DO SISTEMA

O presente projeto tem por objetivo apresentar o conceito da arquitetura de microsserviços mediante o desenvolvimento de uma API Rest voltada para a aplicações de RA aplicadas à indústria. Este projeto em específico foi escolhido a aplicação de RA direcionada para um ambiente de controle de usinas elétricas, onde o funcionário poderá consultar e atualizar informações referentes ao maquinário da usina, trazendo uma melhor ludicidade e facilidade para o operador, tornando a atividade menos burocrática.

Apesar de a RA já trazer uma grande facilidade no que tange a manuseabilidade de ferramentas e máquinas a sua configuração e manutenção ainda se encontra muito verbosa. É nesse ponto que entra o meu projeto, pois com a API Rest será fornecido uma API onde estas aplicações de RA poderão a partir delas disponibilizar para o usuário uma maneira mais intuitiva de configurar e atualizar a sua aplicação.

Para o desenvolvimento do projeto foi separado dois serviços distintos, sendo eles o serviço de Equipamento e o serviço de Especificações. O motivo para essa separação é justamente a necessidade da desacoplação entre ambos, seguindo os princípios da arquitetura de Microsserviços. A estruturação dos mesmos pode ser facilmente visualizada no diagrama de classes na (figura – 06).

.

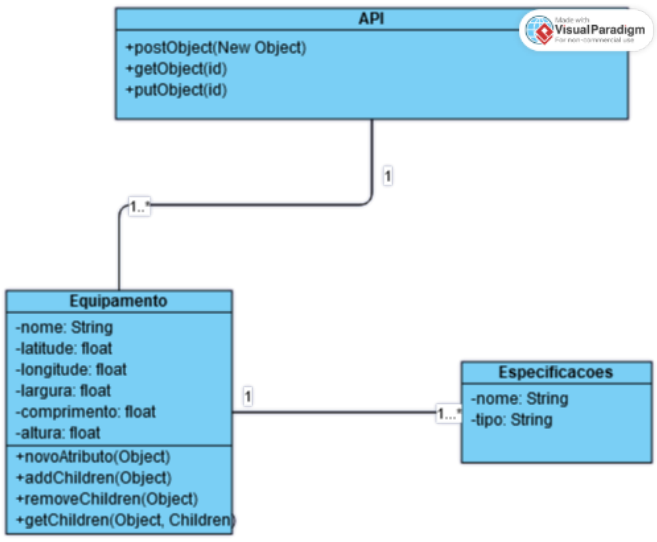

Figura 07 – Diagrama de Classes

Diagrama de Classes:

A Tabela a seguir apresenta a representação da classe API no diagrama de classes proposto para o sistema. Essa classe é responsável por centralizar as operações de manipulação dos objetos disponibilizados pelo serviço, definindo os principais métodos utilizados para criação, consulta e atualização dos dados. Por meio desses métodos, estabelece-se a comunicação entre o cliente e o backend da aplicação, garantindo a padronização das requisições e o correto funcionamento das operações realizadas via arquitetura REST.

API	
Métodos	**Descrição**
PostObject()	Método responsável por adicionar um novo objeto ao sistema.
GetObject()	Método responsável por retornar cada objeto a partir do id.
PutObject()	Método responsável por editar cada objeto.

Tabela 01 – Classe API

A Tabela a seguir apresenta a representação da classe Equipamento no diagrama de classes do sistema. Essa classe é responsável por modelar os objetos que compõem o ambiente da aplicação, definindo seus atributos estruturais e comportamentais. Por meio de seus atributos, são armazenadas as características físicas e identificadoras do equipamento, como dimensões e localização. Já seus métodos permitem a manipulação e o gerenciamento das especificações associadas ao objeto, possibilitando maior organização, modularidade e flexibilidade na estrutura do sistema.

Equipamentos	
Atributos	**Descrição**
nome : String	Referente ao nome do objeto.
latitude : Float	Atributo responsável por informar o tamanho latitudinal do objeto.
longitude : Float	Atributo responsável por informar o tamanho longitudinal do objeto.
largura : Float	Atributo responsável por informar o tamanho de largura do objeto.
altura : Float	Atributo responsável por informar o tamanho de altura do objeto.
Métodos	**Descrição**
novoAtributo()	Responsável por adicionar novo atributo ao objeto.
addChildren()	Responsável por adicionar uma especificação do equipamento.
removeChildren()	Responsável por remover uma especificação do equipamento.
getChildren()	Responsável por retornar uma especificação do equipamento.

Tabela 02 – Classe Equipamentos

A Tabela a seguir apresenta a representação da classe Especificações no diagrama de classes do sistema. Essa classe é responsável por definir e organizar as informações complementares associadas aos equipamentos, permitindo detalhar suas características específicas. Por meio de seus atributos, são descritos elementos que qualificam e categorizam o equipamento, contribuindo para uma modelagem mais estruturada e para uma melhor organização dos dados dentro da aplicação.

Especificações	
Atributos	**Descrição**
nome : String	Responsável para especificar o equipamento.
tipo : String	Responsável para informar o tipo do equipamento.

Tabela 03 – Classe Especificações

O usuário do sistema deste projeto poderá realizar algumas atividades após logado, sendo elas a de criar um objeto, analisar o objeto e também terá a funcionalidade de atualizar o objeto, podendo adicionar novos atributos ao mesmo

se necessário. A estruturação dessas atividades disponíveis para o usuário do sistema pode ser visualizada no diagrama de caso de uso presente na (figura – 07).

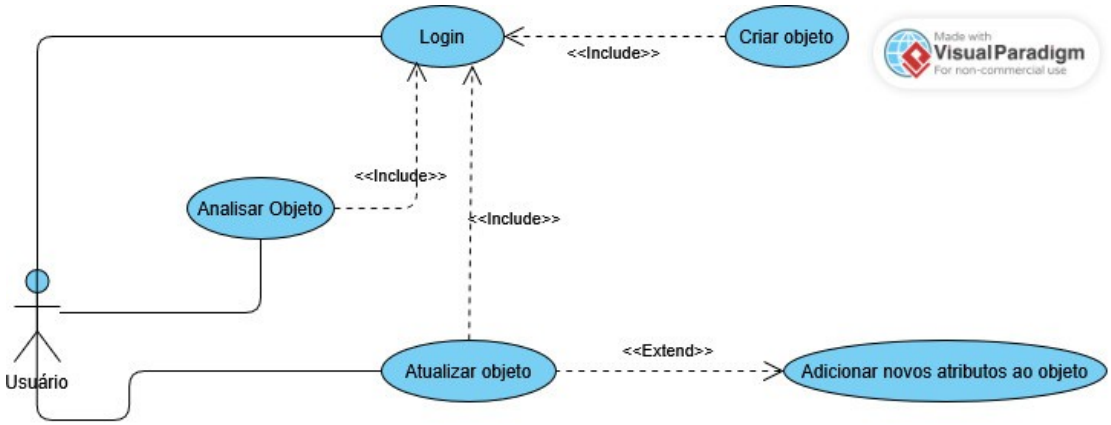

Figura 08 – Diagrama de Caso

6.1 – DESCRIÇÃO DO DIAGRAMA DE CASO

Caso de Uso	Login
Ator	Usuário
Descrição	Permite o cadastro de novo usuário, com e-mail e senha.
Pré Condição	Necessita ter e-mail e senha válidos.
Fluxo Normal	Acessar a rota de POST que permite cadastrar o usuário.
Fluxos Excepcionais	Inexistente
Pós Condição	Cadastra um novo usuário.

Tabela 04 - Descrição de caso de uso “Login”

Caso de Uso	Criar Objeto
Ator	Usuário
Descrição	Permite o usuário cadastrar novos equipamento no banco.
Pré Condição	É necessário que o usuário esteja logado
Fluxo Normal	Acessar a rota POST para cadastro de equipamento.
Fluxos Excepcionais	Informa que o usuário precisa estar logado.
Pós Condição	Adicionar equipamento ao banco.

Tabela 05 - Descrição de caso de uso “Criar Objeto”

Caso de Uso	Atualizar objeto
Ator	Usuário
Descrição	Permite o usuário atualizar dados referente a um equipamento já cadastrado no banco.
Pré Condição	O usuário precisa estar logado e o equipamento existir no banco.
Fluxo Normal	Acessar a rota PUT que permite o usuário atualizar os dados do equipamento.
Fluxos Excepcionais	Informa que o equipamento não existe ou que o usuário não tem acesso.
Pós Condição	Atualizar o equipamento..

Tabela 06 - Descrição de caso de uso “Atualizar Objeto”

Caso de Uso	Adicionar novos atributos ao objeto
Ator	Usuário
Descrição	Permite o usuário adicionar atributos referentes a um determinado equipamento.
Pré Condição	O usuário precisa estar logado e o equipamento existir no banco.
Fluxo Normal	Acessar a rota POST que permite o usuário atualizar os dados do equipamento.
Fluxos Excepcionais	Informa que o equipamento não existe ou que o usuário não tem acesso.
Pós Condição	Adicionar atributos ao equipamento.

Tabela 07 - Descrição de caso de uso “Adicionar novos atributos ao objeto”

Caso de Uso	**Analisar o equipamento**
Ator	Usuário
Descrição	Permite o usuário consultar um equipamento específico e analisar os seus atributos.
Pré Condição	O equipamento deve existir no banco.
Fluxo Normal	Acessar a rota GET que permite consultar um determinado equipamento cadastrado no banco.
Fluxos Excepcionais	Informa que o equipamento não existe.
Pós Condição	Consulta os atributos do equipamento.

Tabela 08 - Descrição de caso de uso “Analisar o equipamento”

7 – IMPLEMENTAÇÃO E FUNCIONAMENTO DO SISTEMA

Neste projeto foi desenvolvido um sistema back-end voltado para gerenciar o cadastro, consulta e atualização de objetos presente numa aplicação de RA de uma usina elétrica. Para isso foi adotado o modelo arquitetural API Rest, a fim de proporcionar uma maior escalabilidade e desacoplamento do sistema, permitindo assim que o mesmo esteja apto a passar por atualizações e melhorias no decorrer de seu ciclo de vida.

A linguagem escolhida foi o Javascript, por conta de sua popularidade o mesmo permite uma maior gama de suporte para o desenvolvedor, seja com sua própria documentação ou pelo apoio que a comunidade exerce em todo o mundo. Justamente o Javascript foi necessário utilizar algumas ferramentas para o desenvolvimento dos serviços de back-end, são elas:

- Express
- Nodemon
- Mongoose
- Dotenv
- Jsonwebtoken
- Bcrypt
- Cors

A comunicação entre os serviços foi feita através do protocolo HTTP utilizando API RESTFUL, já que por se tratar de um sistema menor e com menos verbosidade a adoção desta comunicação viu-se mais produtiva e vantajosa. (Figura 08 Rotas do Sistema)

Cada serviço foi dividido de uma forma que cada método ficasse separada exclusivamente de acordo com sua funcionalidade, permitindo em atualizações futuras uma maior facilidade, impedindo possíveis gargalos.

```
//PRIVATE ROUTE
routes.get("/equipamento/:_id", checkToken, equipamentoController.find); //busca um equipamento especifico
routes.get("/equipamento", checkToken, equipamentoController.list); //lista todos os equipamentos

routes.post(
  "/user/post/equipamento/:_id",
  checkToken,
  equipamentoController.add
);

//USERS ROUTES
routes.post("/user/register", userController.post); //Registra User

routes.post("/user/login", userController.login); //Efetua o login do usuário

routes.get("/user/account/:_id", checkToken, userController.find); //busca o usuário pelo id

//ESPECIFICACAO ROUTES
routes.post("/especificacao/add", checkToken, especificacaoController.post); //Adiciona uma funcionalidade à um equipamento

routes.get("/especificacao/find", checkToken, especificacaoController.list); //lista todos os equipamentos em uso, com suas especificações

routes.get("/especificacao/one/:_id", especificacaoController.find); //busca a especificação de um equipamento pelo seu respectivo id

module.exports = routes;
```

Figura 09 – Rotas do Sistema

7.1 – SERVIÇO DE EQUIPAMENTOS

Na figura 09 – Equipamentos Service pode-se ver como foi desenvolvido cada método do serviço dos equipamentos, juntamente com sua comunicação ao banco de dados.

```
module.exports = {
  async list(req, res) {
    const equipamento = await Equipamento.find();
    res.json(equipamento);
  },

  async find(req, res) {
    const { _id } = req.params;
    const equipamento = await Equipamento.findOne({ _id }); //seleciona o equipamento correspondente ao ID
    res.json(equipamento);
  },

  async add(req, res) {
    const { nome, latitude, longitude, largura, comprimento, altura } =
      req.body;

    let dataCreate = {};

    dataCreate = {
      nome,
      latitude,
      longitude,
      largura,
      comprimento,
      altura,
    };

    const equipemento = await Equipamento.create(dataCreate);
    res.json(equipemento);
  },
};
```

Figura 10 – Equipamentos Service

Os métodos a seguir compõem o conjunto de operações responsáveis pelo gerenciamento dos dados de Equipamentos no sistema, permitindo a interação direta com o banco de dados por meio da arquitetura REST. Essas funções possibilitam a listagem completa dos registros, a consulta individual de itens específicos e o cadastro de novos equipamentos, garantindo assim o controle e a manipulação adequada das informações armazenadas na aplicação.

• list(): O método list é responsável por retornar todos os itens presentes no banco de dados de Equipamentos.

• find(): O método find é responsável por retornar um item específico fazendo a busca pelo id deste produto retornando todos os seus dados. O id será informado no parâmetro da url.

• add(): O método add é responsável por realizar o cadastro do equipamento no banco de dados, seus dados serão passados através de um body em formato json.

```
module.exports = {
  async list(req, res) {
    const especificacao = await Especificacao.find();
    res.json(especificacao);
  },

  async find(req, res) {
    const { _id } = req.params;
    const especificacao = await Especificacao.findOne({ _id });
    res.json(especificacao);
  },

  async post(req, res) {
    const { nome, tipo, equipamentoId } = req.body;

    let dataCreate = {};

    dataCreate = {
      nome,
      tipo,
      equipamentoId,
    };

    const especificacao = await Especificacao.create(dataCreate);
    res.json(especificacao);
  },
};
```

Figura 11 – Especificação Service

• list(): O método list é responsável por retornar todas as especificações de cada equipamento, pois cada equipamento terá uma especificação diferente dependendo de sua função.

• find(): Irá retornar um a especificação especifica de um determinado equipamento.

• add(): O método add é responsável por realizar o cadastro da especificação de um determinado produto, dando-lhe as informações de atribuição deste produto.

7.2 – MÉTODOS DO USUÁRIO:

7.2.1 – REGISTER:

Método POST para realizar o cadastro de usuários no sistema. À esquerda o formato JSON que deve ser seguido e à direita como é salvo no banco de dados. (Figura 11 – Cadastro)

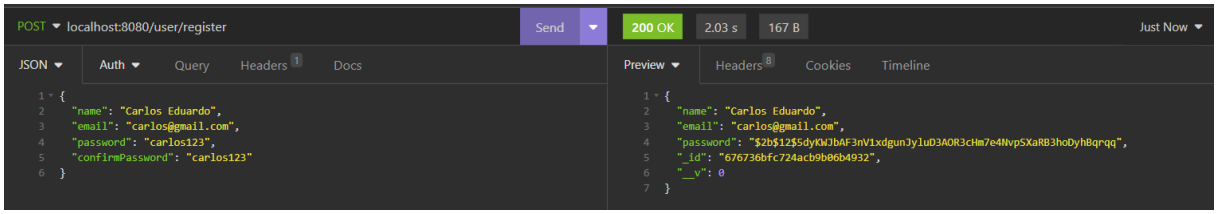

Figura 12 – Cadastro

7.2.2 – LOGIN:

Método POST responsável por realizar o login do usuário no sistema, para isso tem que ser passado o e-mail e a senha que foram cadastrados. O método após isso retorna um token de acesso para o usuário poder acessar rotas privadas exclusivos para quem está cadastrado no sistema. (Figura 12 – Login)

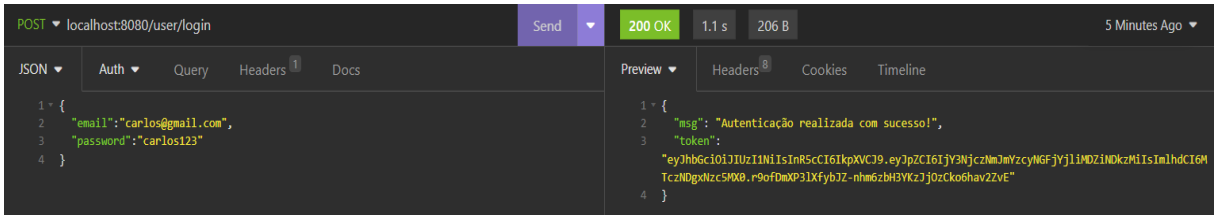

Figura 13 – Login

7.2.3 – ACCOUNT:

O método account é responsável por retornar os dados de um usuário específico, estes dados serão retornados a partir do id passado como parâmetro via url.

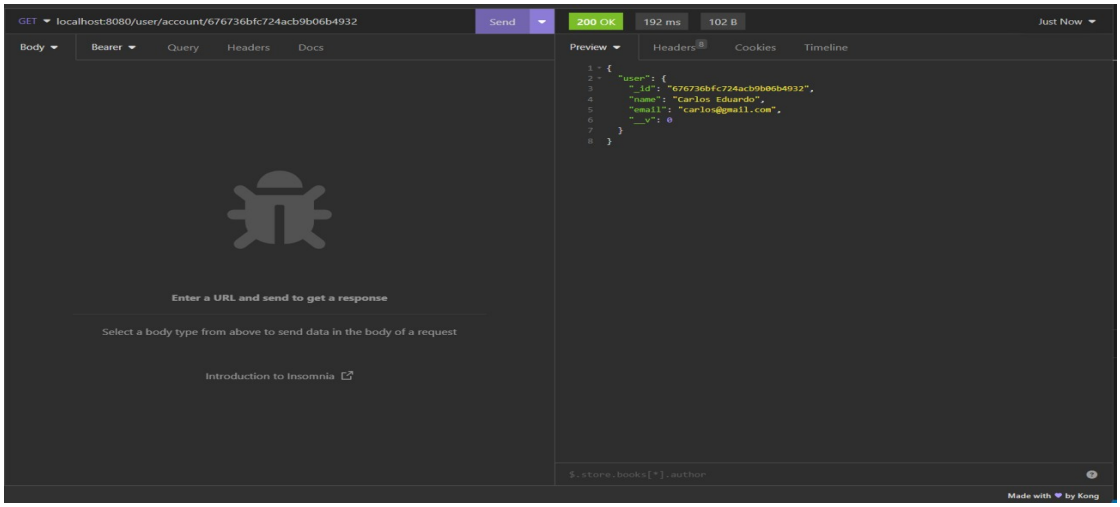

Figura 14 – Get User

8 – MÉTODOS DE EQUIPAMENTO:

8.1 – ADD:

O método add de Equipamentos é responsável por adicionar novos equipamentos ao sistema. Por se tratar de uma rota privada ela só poderá ser usada caso o usuário esteja logado e com seu Token de autorização que passado no cabeçalho da requisição.

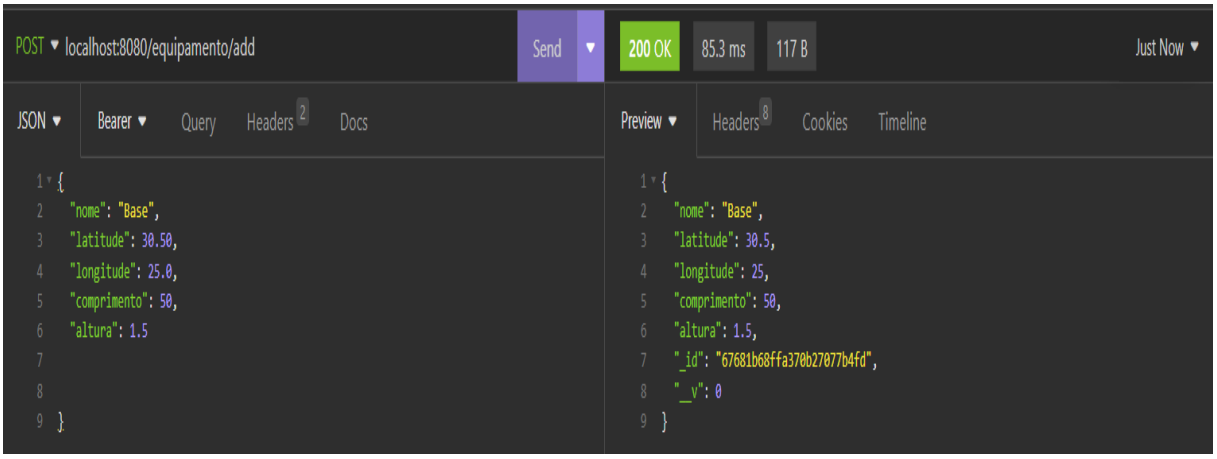

Figura 15 – Post Equipamento

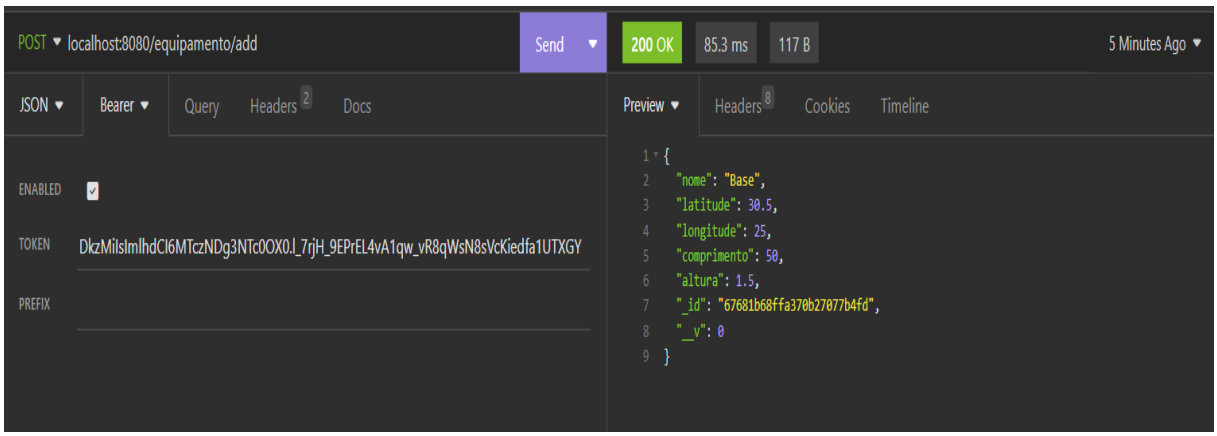

Figura 16 – Requerimento de Token para o Post

8.2 – Informações do equipamento

A rota equipamentos/info/:_id é responsável por retornar informações referentes a um equipamento específico que é determinado pelo id informado como parâmetro da rota. Esta rota por se tratar de uma rota privada, para ser usada é necessário estar logado no sistema, sendo passado o Token de autorização no cabeçalho da requisição.

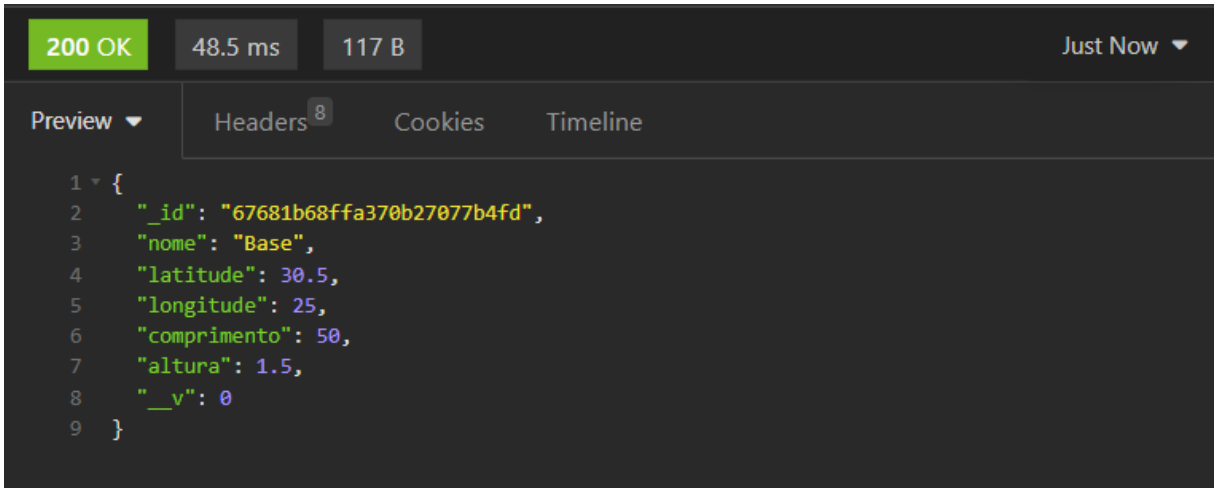

Figura 17 – Equipamentos Info

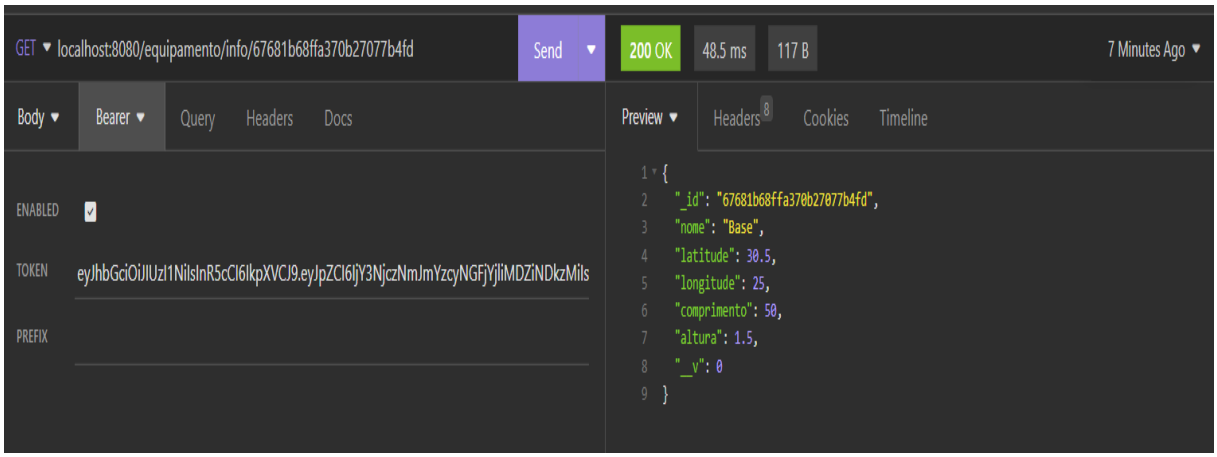

Figura 18 – Requerimento e Token para Equipamentos info

8.3 – Listagem de todos os equipamentos

A rota /equipamento/all é responsável por retornar todos os equipamentos presentes no bando de dados de Equipamento. Por ser um acesso a informações sensíveis está rota é privada, sendo possível acessá-la somente mediante login do usuário no sistema.

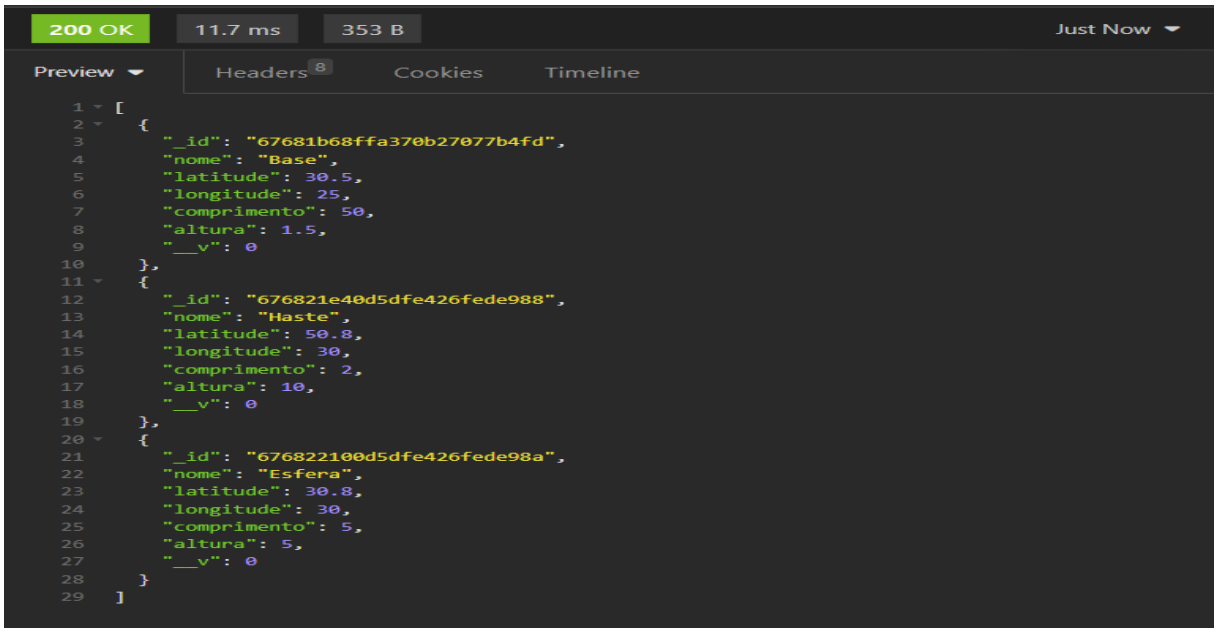

Figura 19 – Todos equipamentos cadastrados

9 – TESTE DO SISTEMA

9.1 – CONSIDERAÇÕES INICIAIS

Este capítulo tem como objetivo apresentar os procedimentos adotados para a realização dos testes do sistema desenvolvido, com foco na validação de suas funcionalidades, desempenho e confiabilidade. A etapa de testes é fundamental no processo de desenvolvimento de software, pois permite identificar falhas, verificar o atendimento aos requisitos e garantir a qualidade da solução antes de sua utilização em ambiente real.

Além disso, essa metodologia contribui para a detecção de inconsistências, falhas de validação e problemas de integração, auxiliando na melhoria contínua do sistema. Dessa forma, os testes realizados fornecem subsídios para analisar a estabilidade, a confiabilidade e a adequação da API às necessidades definidas no projeto, servindo como base para as análises apresentadas nas seções seguintes.

9.2 – DEFINIÇÃO DO TESTE DE CAIXA PRETA

Para a avaliação do sistema, foi adotada a metodologia de Teste de Caixa Preta, na qual o sistema é analisado a partir de suas entradas e saídas, sem considerar sua estrutura interna ou a forma como foi implementado. Nesse tipo de teste, o foco está no comportamento externo da aplicação, simulando a experiência do usuário ou de sistemas consumidores da API.

A escolha do Teste de Caixa Preta justifica-se por sua eficiência na verificação das funcionalidades previstas nos Casos de Uso e nos requisitos levantados, permitindo avaliar se as respostas fornecidas pela API estão de acordo com o esperado. Por meio dessa abordagem, é possível validar operações como cadastro, consulta, atualização e exclusão de dados, além do tratamento de erros e exceções.

9.3 – VALIDAÇÃO DAS CLASSES E CASOS DE TESTE

A seguir, serão apresentadas as classes de teste utilizadas na aplicação da metodologia de Teste de Caixa Preta, responsáveis por validar o funcionamento das funcionalidades do sistema a partir da análise de suas entradas e saídas. Essas classes foram desenvolvidas com o objetivo de simular diferentes cenários de uso, verificando se a API responde corretamente às requisições realizadas.

Por meio dessas classes, foi possível testar as principais operações do sistema, incluindo situações normais de funcionamento, casos alternativos e cenários de erro. Dessa forma, buscou-se avaliar o comportamento externo da aplicação, sem considerar sua estrutura interna ou sua implementação.

A apresentação dessas classes permite compreender como os testes foram organizados e executados, evidenciando os critérios adotados para validação das funcionalidades e a verificação dos resultados obtidos. Além disso, essa estrutura contribui para a confiabilidade do sistema, servindo como base para futuras validações e manutenções.

9.3.1 – LOGIN

Caso de Uso	Condição de Entrada	Classe Válida	Classe Inválida
Login	Acessar rota de login.	1. Inserir e-mail correspondente ao regex e a senha.	2. Inserir e-mail não corresponde(@). 3. Inserir e-mail que não existe.

Tabela 09 – Teste Login

Condição de Entrada	Classe	Resposta
Acessar rota de login	1	Apresentou o comportamento esperado.
	2	Informou que o e-mail era inválido.
	3	Informou que o e-mail não existia.

Tabela 10 – Resultado Teste Login

9.3.2 – CRIAR OBJETO

Caso de Uso	Condição de Entrada	Classe Válida	Classe Inválida
Criar Objeto	Acessar rota POST de equipamento..	1. Inserir equipamento com todos os atributos obrigatórios.	2. Inserir equipamento com atributos obrigatórios não preenchidos.
			3. Não estar logado no sistema.

Tabela 11 – Teste Criar Objeto

Condição de Entrada	Classe	Resposta
Acessar rota POST de equipamento.	1	Apresentou o comportamento esperado.
	2	Informou o campo obrigatório que não estava preenchido.
	3	Informou que o usuário não estava logado no sistema.

Tabela 12 – Resultado Teste Criar Objeto

9.3.3 – ANALISAR OBJETO

Caso de Uso	Condição de Entrada	Classe Válida	Classe Inválida
Analisar Objeto	Acessar rota GUET de um equipamento específico.	1. Fornecer o id correspondente do equipamento.	2. Fornecer um id não correspondente ao equipamento desejado.
			3. Fornecer um id inexistente.

Tabela 13 – Teste Analisar Objeto

Condição de Entrada	Classe	Resposta
Analisar Objeto	1	Apresentou o comportamento esperado.
	2	Apresentou um equipamento com seus atributos, porém, não correspondente ao desejado.
	3	Informou que não existe nenhum equipamento correspondente ao id fornecido.

Tabela 14 – Resultado Teste Analisar Objeto

10 – RESULTADOS E DISCUSSÕES

Conforme os testes de caixa preta, foi possível notar que o serviço é eficaz na entrega de respostas, possibilitando a flexibilidade na atualização dos dados. Logo, possíveis atualizações futuras não encontrarão gargalos em sua implementação.

Primeiramente, a escalabilidade foi uma das características mais notáveis durante os testes. A API Rest permite que diferentes componentes do sistema sejam dimensionados de maneira independente, o que se mostrou essencial para garantir a performance sob diferentes cenários de uso.

Além disso, a flexibilidade na manutenção e atualização do sistema foi outro ponto positivo. A arquitetura de microsserviços possibilitou a realização de modificações em componentes específicos sem comprometer o funcionamento global da aplicação. Durante os testes, observou-se que mudanças em um uma categoria de dados não gerou impactos negativos nos demais, proporcionando uma experiência de manutenção ágil e sem interrupções significativas no serviço. Essa característica também facilita a identificação e resolução de falhas, pois os problemas podem ser isolados a um único serviço, o que torna o processo de depuração e correção mais eficiente.

A independência que a API Rest propicia, também trouxe benefícios em relação à atualização contínua do sistema. A capacidade de implementar novas funcionalidades ou corrigir falhas em um serviço sem afetar a continuidade dos demais serviços permite uma evolução constante da aplicação de forma controlada e sem downtime, o que se alinha às boas práticas de DevOps e integração contínua.

Por outro lado, é importante destacar que a arquitetura de microsserviços, embora traga várias vantagens, também impõe desafios relacionados à complexidade de gerenciamento e comunicação entre os serviços.

Em termos gerais, os resultados obtidos indicam que a adoção da arquitetura trouxe ganhos consideráveis em termos de escalabilidade e manutenção. Os benefícios a longo prazo são evidentes, principalmente quando se considera o crescimento da aplicação e a necessidade constante de atualizações e melhorias.

Portanto, os resultados deste trabalho confirmam que o modelo de arquitetura API Rest é uma escolha estratégica eficaz para sistemas que demandam alta escalabilidade, flexibilidade nas manutenções e capacidade de evolução contínua, sendo altamente recomendada para cenários que exigem resiliência e performance constantes.

11 – CONCLUSÃO

Este trabalho demonstrou de maneira clara e eficaz os benefícios da API Rest, evidenciando a sua capacidade de proporcionar escalabilidade, flexibilidade e manutenção aprimorada em sistemas complexos. A análise realizada ao longo do desenvolvimento e testes confirmou a facilidade na gestão do sistema, como também, o controle mais granular sobre o desempenho e a operação, sem comprometer a continuidade dos serviços. A implementação e manutenção se mostrou eficiente, com a possibilidade de evoluir e corrigir falhas sem impactar negativamente os demais componentes do sistema.

Contudo, embora os resultados alcançados até o momento sejam promissores, a arquitetura, como esperado, apresenta desafios, principalmente relacionados à comunicação entre outros serviços. Esses desafios, no entanto, podem ser atenuados com a utilização de ferramentas especializadas e boas práticas de desenvolvimento e operação.

O próximo passo para o avanço deste trabalho será a implementação de um front-end robusto, que irá interagir de maneira fluida e eficiente com os microsserviços já implementados. A criação de uma interface de usuário amigável e responsiva, integrada com os serviços de back-end, será essencial para proporcionar uma experiência completa e acessível aos usuários finais. O front-end, ao ser projetado com as melhores práticas de desenvolvimento, garantirá que a comunicação entre o cliente e os microsserviços seja eficiente e eficaz, consolidando a arquitetura como uma solução completa e escalável. Este novo desenvolvimento ampliará ainda mais as capacidades do sistema, permitindo que os usuários aproveitem todo o potencial da arquitetura de microsserviços de maneira intuitiva e interativa.

Portanto, a conclusão deste trabalho marca um avanço no entendimento e na aplicação da arquitetura de microsserviços, e os próximos passos, com a implementação do front-end, prometem transformar o sistema em uma solução ainda mais poderosa e pronta para suportar o crescimento contínuo e as necessidades dinâmicas do mercado.

12 – REFERÊNCIAS BIBLIOGRÁFICAS

BIGHETI, Jeferson; RISSO, Segio; CALDIERI, Marcos. Proposta de Arquitetura Orientada a Microsserviços para Aplicações de Internet das Coisas Industrial. Anais do XXII Congresso Brasileiro de Automática , Lençóis Paulista - SP, 1, 2, 04/2020.

CASTELLS, Manuel. *A sociedade em rede*. São Paulo: Paz e Terra, 2000.

GOMES, (2023) “Dotenv: gerenciando variáveis de ambiente”. IETF. Disponível em: Dotenv: gerenciando variáveis de ambiente | Alura . Acessado em: 20 de nov. 2023.

HILLMAN, 2023 “O que é CORS e como resolver um erro de Cross- Origin Resource Sharing (CORS)?”. IETF. Disponível em:CORS: o que é e como resolver esses casos de erro | Alura . Acessado em: 20 de nov. 2023.

HOBSBAWM, Eric. *A era das revoluções: 1789–1848*. Rio de Janeiro: Paz e Terra, 2012.

KERLYN, 2019 “Uma breve introdução sobre BCrypt”. IETF. Disponível em:Uma breve introdução sobre BCrypt | by Kerlyn | reprogramabr | Medium . Acessado em: 21 de nov. 2023.

LIMA, 2021“MongoDB: O banco baseado em documentos”. IETF. Disponível em: MongoDB: O banco baseado em documentos | Alura . Acessado em: 22 de nov. 2023.

ORACLE, 2023 “O que é NoSQL?”. IETF. Disponível em:O que é NoSQL (Banco de Dados Não Relacionais)? | Oracle Brasil . Acessado em: 22 de nov. 2023.

ROVEDA, (2021) “JavaScript: O que é, para que serve e como funciona o JS”. IETF. Disponível em: JavaScript: o que é, para que serve e como funciona o JS? (kenzie.com.br) . Acessado em: 18 de nov. 2023.

SCHWAB, Klaus. *A quarta revolução industrial*. São Paulo: Edipro, 2016.

SEGUINS, 2022 “O que é JSON Web Tokens?”. IETF. Disponível em: O que é JSON Web Tokens? | Alura . Acessado em: 21 de nov. 2023.

VISUAL STUDIO CODE, (2023) “Learn to code with Visual Studio Code ”. IETF. Disponível em: Get Started with Visual Studio Code . Acessado em: 18 de nov. 2023.

URANO, 2022 “Postman: saiba como instalar e dar seus primeiros passos”. IETF. Disponível em:Postman: saiba como instalar e dar seus primeiros passos | Alura . Acessado em: 22 de nov. 2023.