UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FRANCISCO WERLLEY GONÇALVES PONTE

SUPERVISÃO DE DADOS UTILIZANDO NODE-RED, NODE.JS E REACTJS PARA ADAPTAÇÃO DE SISTEMAS LEGADOS AO CONTEXTO DA INDÚSTRIA 4.0

SOBRAL

2022

FRANCISCO WERLLEY GONÇALVES PONTE

SUPERVISÃO DE DADOS UTILIZANDO NODE-RED, NODE.JS E REACTJS PARA ADAPTAÇÃO DE SISTEMAS LEGADOS AO CONTEXTO DA INDÚSTRIA 4.0

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Reuber Régis de Melo

SOBRAL

2022

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

P857s Ponte, Francisco Werlley Gonçalves.
Supervisão de Dados Utilizando Node-RED, Node.Js e ReactJS para Adaptação de Sistemas Legados ao Contexto da Indústria 4.0 / Francisco Werlley Gonçalves Ponte. – 2022.
75 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia Elétrica, Sobral, 2022.
Orientação: Prof. Dr. Reuber Régis de Melo.

1. Node-RED. 2. Node.Js. 3. ReactJS. 4. Indústria 4.0. 5. Sistema de supervisão. I. Título.
CDD 621.3

FRANCISCO WERLLEY GONÇALVES PONTE

SUPERVISÃO DE DADOS UTILIZANDO NODE-RED, NODE.JS E REACTJS PARA ADAPTAÇÃO DE SISTEMAS LEGADOS AO CONTEXTO DA INDÚSTRIA 4.0

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Reuber Régis de Melo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. M. Sc. Rômulo Nunes de Carvalho Almeida
Universidade Federal do Ceará (UFC)

Enga. Iara Barbosa de Sousa
Analista de Projetos da TRINERGY

B. Sc. Joaquim Osterwald Frota Moura Filho
Universidade Federal do Ceará (UFC)

Dedico este trabalho a meus pais e irmãos, que estiveram sempre comigo e acreditaram em mim no decorrer dessa jornada.

AGRADECIMENTOS

Primeiramente a Deus, por me dar o dom da vida e proporcionar todos os dias a possibilidade de me tornar uma pessoa melhor.

À minha família, meus pais, Francisca Antônia Gonçalves Ponte e Francisco Françuá Albuquerque Ponte, e irmãos, Francisca Diana Meire Gonçalves Ponte e Francisco Wescley Gonçalves Ponte, por serem minha maior fonte de inspiração, me acompanharem e apoiarem durante toda minha vida e trajeto no curso.

Ao Prof. Dr. Reuber Régis de Melo por todo apoio e orientação nestes anos de curso. Um profissional competente e disciplinado que admiro muito e o tenho como referência.

À minha querida Mara Lívia de Sousa Gomes, que esteve ao meu lado fornecendo apoio e incentivo fundamentais para concluir as últimas etapas do curso.

Às pessoas importantes que fizeram parte desta minha jornada: Joaquim Moura, Alyson Rodrigo, Ana Lyvia, Alan Martins, Bruna Sousa, Caio Timbó, Danillo Fernandes, Danyela de Souza, Darlan Castro, Erick Medeiros, Ermeson Bezerra, Felipe Bruno, Heitor Vasconcelos, Ianna Kelly, Iara Sousa, Jadir Dias, Jair Santos, Jennifer Maria, Joham Lucas, Jonas Silva, Junior Oliveira, Kayo Igor, Keyvilania Freitas, Lucas Aguiar, Mailson Neres, Mairla Gomes, Manoel Erick, Marcelo Guedes, Matheus dos Anjos, Michel Bernardo, Rebeca Aguiar, Samelius Silva, Tácila Magalhães, Vinícius Souza, Vitor Manoel, Weder Mendes, Willian Withi.

Aos membros do PET e todas as turmas de alunos que tive o prestígio de recepcionar enquanto membro do programa. Foi uma honra fazer parte da história de vocês.

Aos meus queridos amigos da empresa e família Ágil Engenharia: Alan Araújo, Lorenna Araújo, Jeann Teixeira, Douglas de Paiva, Vanessa Alves, Camila Kelly, Pedro Holanda, Danilo Araújo, Willian Praciano, Felipe Barros, Heli Ribeiro e Alice Sousa. Obrigado por todo apoio.

Aos professores e funcionários da Universidade Federal do Ceará, Campus de Sobral, por me proporcionarem uma educação de excelência.

Em especial aos professores Euclimar Passos, Fábio Passos, Jermana Lopes, Eber Diniz e Márcio Amora. Todos marcaram de forma única meu trajeto na universidade, e sou profundamente grato.

"Seu destino é apenas o de um homem, mas suas aspirações devem ser as de um deus."

(Ovídio, Poeta Romano)

RESUMO

Neste trabalho foi desenvolvido um ecossistema ciberfísico como proposta para pequenas e médias automações, adaptando sistemas legados aos conceitos de Indústria 4.0. O controlador lógico programável da Siemens é utilizado em comunicação com *gateway* usando a ferramenta Node-RED para enviar os dados obtidos à aplicação desenvolvida com Node.js, que por sua vez é hospedada no serviço de *cloud* da Amazon. Para leitura dos dados foi desenvolvido um sistema de supervisão WEB privado em ReactJS capaz de apresentar de forma gráfica as amostras obtidas em tempo real, assim como realizar o *download* do seu histórico. De modo a aferir informações das amostras coletadas, foi utilizado o classificador k-vizinhos mais próximos (do inglês, *K-Nearest Neighbors* – KNN) para diagnosticar as falhas que possam ocorrer no processo estudado, de maneira a alertar o usuário. Por fim, o ecossistema foi aplicado em um dispositivo do laboratório de controle da universidade, e utilizando um conjunto de 82 amostras obtidas a partir de um *dataset* foi possível executar com êxito o ecossistema proposto. Após a execução do sistema, foi observado que todas as 82 amostras adquiridas foram coletadas e armazenadas sem nenhuma perda e o classificador de falhas apresentou 85% de acurácia. O acompanhamento dos parâmetros do sistema aconteceu de forma fácil, prática e intuitiva por meio do sistema de supervisão desenvolvido sem nenhum investimento financeiro em licenças de *softwares* externos ou treinamentos.

Palavras-chave: Node-RED. Node.Js. ReactJS. Indústria 4.0. Sistema de supervisão.

ABSTRACT

In this paper a cyber ecosystem was developed as a proposal for small and medium automations, adapting legacy systems to concepts of Industry 4.0. The Siemens Programmable Logic Controller is used in gateway communication using the Node-RED tool to send data obtained to the application developed with Node.Js, which in turn is hosted on the Amazon Cloud service. To read the data, a private web supervision system has been developed in ReactJS able to graphically display the samples obtained in real time, as well as downloading your history. In order to assess information from the collected samples, the K - Nearest Neighbors (KNN) classifier was used to diagnose failures that may occur in the process studied, so as to alert the user. Finally, the ecosystem was applied to a device in the university's control laboratory, and using a set of 82 samples from a dataset it was possible to successfully run the proposed ecosystem. After the system was executed, it was observed that all 82 purchased samples were collected and stored without any loss and the failure classifier showed 85% accuracy. The follow -up of system parameters happened easily, practical and intuitively through the supervision system developed without any financial investment in external software licenses or training.

Keywords: *Node-RED*. *Node.Js*. *ReactJS*. *Industry 4.0*. *Supervision System*.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AWS	*Amazon Web Services*
PLC	*Programmable Logic Controller*
SCADA	*Supervisory Control and Data Aquisition*
IoT	*Internet Of Things*
TCP	*Transmission Control Protocol*
IIoT	*Industrial Internet Of Things*
MQTT	*Message Queuing Telemetry Transport*
KNN	*K-Nearest Neighbors*
IA	Inteligência Artificial
M2M	*Machine to Machine*
IoMT	*The Internet of Medical Things*
HTTP	*Hypertext Transfer Protocol*
HTML	*HiperText Markup Language*
SPA	*Single Page Application*
Amazon EC2	*Amazon Elastic Compute Cloud*
TIA *Portal*	*Totally Integrated Automation Portal*
OPC	*OLE for Process Control*
ORM	*Object-Relational Mapper*
CORS	*Cross-Origin Resource Sharing*
JSON	*JavaScript Object Notation*

SUMÁRIO

1 INTRODUÇÃO

A terceira revolução industrial, também conhecida como revolução digital, é caracterizada pelo surgimento de equipamentos eletrônicos como *Programmable Logic Controller* (PLC)s, microcontroladores e sistemas *Supervisory Control and Data Aquisition* (SCADA), explica Júnior (2019). Já a quarta revolução industrial, chamada também de Indústria 4.0, é caracterizada pela inclusão de tecnologias como sistemas ciberfísicos e *Internet Of Things* (IoT) nos processos produtivos, acarretando maior autonomia na tomada de decisões e transparência nas relações entre humanos e máquinas (PEREIRA, 2018).

De acordo com Silva (2016) a concepção dos sistemas SCADA é de permitir que processos produtivos e instalações físicas sejam monitorados por sensores, fornecendo dados que podem ser armazenados, processados, analisados, manipulados e apresentados ao usuário. Com isso é possível o controle e monitoramento de diversos ramos da indústria.

Sistemas de supervisão vêm se tornando cada vez mais indispensáveis no ambiente fabril. Isso pode ser visto em Andrade (2018), que apresenta a importância e o diferencial da aplicação desses sistemas na supervisão de processos industriais, principalmente no aspecto econômico quando comparado a outras opções por ela citadas, como manter colaboradores apenas para supervisionar processos, em que estão mais suscetíveis a falhas do que sistemas autônomos.

Para Schroeder (2018) o conceito de sistemas ciberfísicos vem ganhando importância, pois, devido suas características de união dos sistemas computacionais e físicos, são fundamentais para a nova revolução industrial, marcada pela descentralização dos controles de processos e crescimento do uso de dispositivos inteligentes interconectados.

No Brasil, o desenvolvimento industrial baseado nos pilares da indústria 4.0 ainda ocorre de forma lenta e segundo Rodrigues (2021) esta transição está ocorrendo apesar das constantes crises econômicas no país. Ainda de acordo com Rodrigues (2021), um dos principais desafios vem sendo o alto investimento necessário para agregar os recursos da Indústria 4.0, e apesar dos benefícios comprovados as empresas que investem fazem isso com cautela.

1.1 Justificativa

A indústria vem passando por grandes transformações no decorrer da história, e nesta quarta revolução marcada pelo desenvolvimento tecnológico, os equipamentos como sensores, atuadores, transdutores e similares, assim como seus devidos sistemas, vêm desempenhando papel fundamental, sendo cada vez mais indispensáveis no cenário fabril.

Nesse contexto, parte dos equipamentos e *softwares*, em sua maioria, são importados e o alto custo resulta na dificuldade de automatizar pequenas e médias indústrias onde o gerenciamento estratégico financeiro é indispensável e o alto investimento nesses recursos por vez acaba não se justificando, cenário esse que carece de soluções tecnológicas eficientes, implementação fácil e principalmente baixo custo.

Em Vicente e Mendes (2021) é apresentada algumas soluções conhecidas para sistemas de controle e supervisão, sendo eles: MicroSCADA Pro, ABBsistema®; AggreGate SCADA / HMI, Tibbo SYSTEMS®; CIMPLICITY, GE Digital®; CREW, Esa Automation®; GENESIS64, ICONICS®; Ignition 8, Indutive Automation®; FACTORYTALK VIEW MACHINE EDITION, Rockwell Automation ®; e SIMATIC WinCC V7 SCADA, Siemens®. Onde cada uma delas possuem sua própria tecnologia, seja de infraestrutura ou comunicação, o que acaba por elevar bastante o custo não apenas de implementação, mas também manutenção e com licenças periódicas para utilização, o que para pequenas e médias automações pode acabar não sendo tão atrativas.

Com base nessa problemática, este trabalho traz uma solução capaz de atender a necessidade dessas empresas implementando um sistema de supervisão simples, de fácil manuseio e baixo custo, de modo a entrarem no contexto de Indústria 4.0. Essa solução também visa ser uma alternativa a sistemas legados.

1.2 Estado da Arte

A literatura aborda a utilização do Node-RED no ambiente fabril em diferentes processos, como em Cruz *et al.* (2021) onde é proposto o desenvolvimento de uma aplicação referente aos conceitos de Indústria 4.0 utilizando IoT e *Cloud Computing*. Nessa aplicação foi implementado o monitoramento de dados gerados por uma caldeira que alimenta linhas de vapor de fábrica, sem fazer alterações de *hardware* no PLC escolhido. Para isso, foi utilizado o Node-RED por meio do protocolo de comunicação Modbus *Transmission Control Protocol*

(TCP)/IP para comunicar com serviço de *broker*, obtendo como resultado uma *dashboard* capaz de mostrar os dados da caldeira em tempo real. Tal trabalho mostra a eficiência do Node-RED em coletar e comunicar os dados coletados eficientemente.

Já em Silva (2021) é realizado, também, um sistema de controle e monitoramento de um forno industrial, coletando dados de temperatura em tempo real para poderem ser tratados posteriormente. Após a coleta, os dados são passados para o computador Raspberry PI contendo Node-RED instalado, responsável por gerar a *dashboard* que irá exibir informações do forno. O projeto alterou a sistemática do trabalho neste setor, onde agora a terceirizada responsável pode usufruir de um sistema de monitoramento remoto da temperatura do forno permitindo antecipar e corrigir eventuais falhas. Ainda no trabalho de Silva (2021) é possível evidenciar a portabilidade do Node-RED que pode ser adaptado para redes locais, sistemas em nuvem e até mesmo *hardwares* de baixo custo como Raspberry PI.

Vale ressaltar a compartibilidade do Node-RED em trabalhar com diversas plataformas, como é mostrado em Pinto (2021), ao elaborar um sistema de manutenção preditiva de falhas de válvulas para um processo industrial. Ele desenvolveu uma plataforma baseada em Node-RED para coletar dados com um *framework* de criação para páginas WEB em Python chamado Flask, que resultou na análise do desempenho de até 3000 válvulas.

Com inúmeras aplicações sendo implementas, o Node-RED vem sendo foco de estudos e suas perspectivas no aspecto industrial a respeito da *Industrial Internet Of Things* (IIoT). Um exemplo disso é o trabalho abordado por Nitulescu e Korodi (2020), que desenvolveu um aplicativo que monstra o uso de protocolos como Modbus TCP interagindo com dispositivos industriais via *Message Queuing Telemetry Transport* (MQTT) e criando estruturas de supervisão bem próximas de soluções SCADA já conhecidas. O resultado de Nitulescu e Korodi (2020) foi um sistema supervisório capaz de receber dados e apresentar informações pertinentes em pouco tempo e utilizando baixa largura de banda.

1.3 Objetivo Geral

O objetivo geral do trabalho é apresentar uma solução alternativa capaz de atender a necessidade de pequenas e médias automações, adaptando sistemas legados baseados em PLCs da SIEMENS aos conceitos de Indústria 4.0.

1.3.1 Objetivos Específicos

1. Entender como se acessa dados de um PLC por meio do Node-RED;
2. Utilizar o Node-RED para armazenar dados em nuvem;
3. Propor uma solução com sistema de supervisão WEB de fácil manuseio e baixo custo utilizando Node-RED, Node.Js e ReactJS;
4. Apresentar a possibilidade de agregar novas ferramentas no sistema desenvolvido utilizando classificador de falhas como exemplo;
5. Implementar e avaliar um estudo de caso com a solução proposta.

1.4 Organização do Trabalho

Este trabalho está organizado em cinco capítulos. O capítulo um, traz uma breve introdução da temática abordada, assim como sua justificativa, seguido do estado da arte e objetivos a serem alcançados.

O capítulo dois apresenta a fundamentação teórica, trazendo conceitos importantes para o trabalho como Indústria 4.0, IoT, Sistemas Legado, Node.js, Node-RED, ReactJS e *K-Nearest Neighbors* (KNN).

No capítulo três é apresentada a metodologia abordada para construção e validação do trabalho, assim como a especificação da estrutura lógica a ser seguida em cada procedimento, materiais e métodos necessários para que o projeto proposto seja construído e resultados sejam obtidos.

O capítulo quatro traz a aplicação da metodologia abordada na sessão anterior, apresentando os resultados encontrados.

Já o quinto e último capítulo apresenta a conclusão do trabalho, consolidando os resultados obtidos e trazendo sugestões para trabalhos futuros.

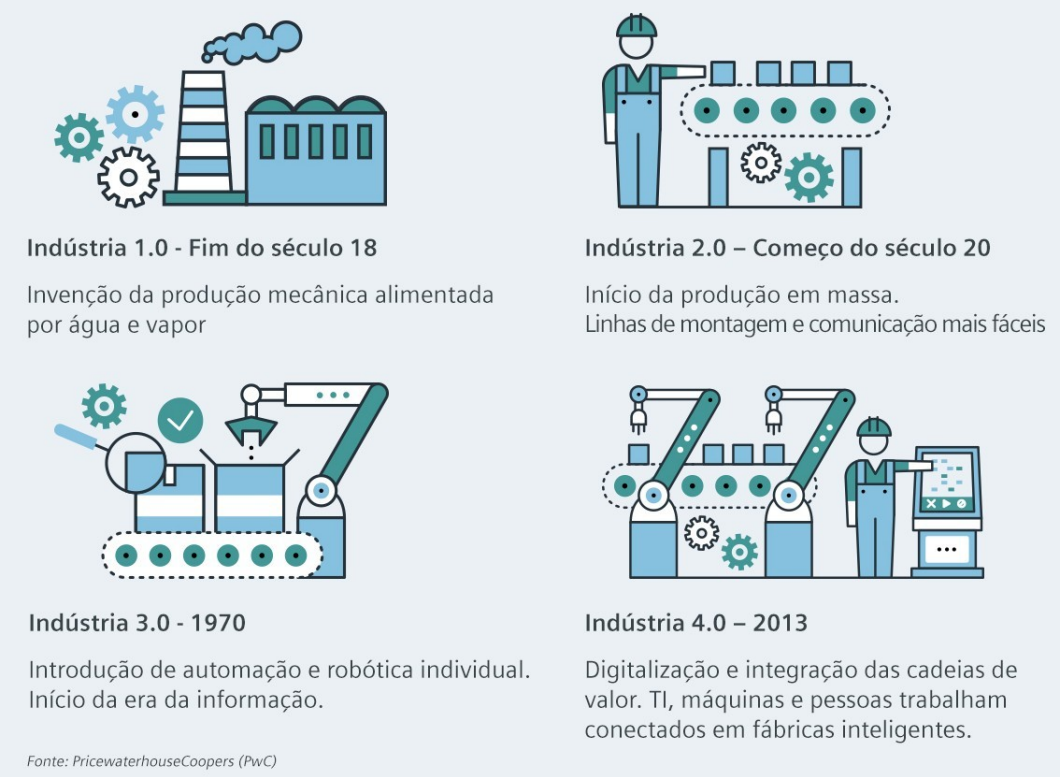
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado tópicos importantes para a compreensão do trabalho. Temas como Indústria 4.0, IoT, sistemas legados, Node.js, Node-RED, ReactJS e KNN serão abordados.

2.1 Indústria 4.0

A primeira revolução industrial aconteceu há mais de 250 anos, e os métodos de produção passaram por grandes mudanças, onde as máquinas a vapor começaram a substituir os animais. Com a entrada dos motores elétricos, o aço e as linhas de produção projetadas por Henry Ford, a segunda revolução industrial alcançou seu ápice. No final da década de 1970, os primeiros processos automatizados começaram a surgir. Com a popularização dos computadores e internet, se tornou visível que os produtos eletrônicos tornaram-se protagonistas da terceira revolução industrial. Alguns detalhes podem ser observados na Figura 1.

Figura 1 – Evoluções industriais no decorrer da história.



Fonte: SIEMENS (2021).

O conceito de Indústria 4.0, ou Quarta Revolução Industrial como também é conhecida, apresentou-se pela primeira vez na Feira de Hannover em 2011, caracterizada pela integração da automação industrial e soluções tecnológicas, como Inteligência Artificial (IA), IoT, robótica e *Cloud Storage*. Hoje, quando falado sobre manufatura avançada e fábricas inteligentes, está sendo abordada a quarta revolução industrial, que permitirá usar um novo conjunto de tecnologias inteligentes para integrar o mundo físico, digital e biológico, criando sistemas integrados de produção, trazendo maior eficiência e segurança aos mais diversos setores industriais (SIEMENS, 2021).

Ao ser apresentada para o mundo em 2011, algumas ferramentas e conceitos essenciais foram propostos dentro da Indústria 4.0, sendo eles (SIEMENS, 2021):

– **Operação em Tempo Real** , coleta e análise instantâneas de dados viabilizando tomadas de decisão;
– **Virtualização** , com cópias virtuais das fábricas, *Digital Twins*, permitindo simulações, rastreamento e monitoramento dos processos utilizando sensores, atuadores, transdutores, chipes e similares;
– **Orientação a Serviços** , utilizando arquiteturas de *software* sob demanda voltadas a encontrar soluções;
– **Descentralização** , devido à tomada de decisões poder partir das próprias máquinas inteligentes que passam a se comunicar entre si;
– **Modularidade** , com a possibilidade de produção sob demanda, permitindo flexibilidade da entrada e retirada de módulos de um sistema de produção; e
– **Interoperabilidade** , com comunicação constante entre máquinas, dispositivos e sistemas que dialogam entre si, em processos *Machine to Machine* (M2M).

Partindo dos conceitos supracitados, segundo Indústria (2020) a implementação de robótica avançada, sitemas M2M, IoT, sensores, atuadores e transdutores possibilitam que as máquinas se comuniquem e permita inferir informações de diversas etapas da produção fabril, apresentando novos fluxos de projetos, produção e até pós-venda. Dentre as principais tecnologias da Indústria 4.0, podem ser citadas:

1. **Inteligência Artificial:** aplicação de análises lógicas envolvendo aprendizagem de máquina de modo a inferir informações de eventos e analisar tendencias do sistema, apoiando decisões sobre o processo.

2. **Computação em nuvem:** é a utilização de serviços de computação, como armazenamento de dados, inferência e análise de informações, pela internet hospedados em *Datacenter*, tornando possível recursos flexíveis e economia escalável. Esta tecnologia permite o acesso a recursos computacionais dos mais diversos dispositivos remotos, evitando assim o desperdício econômico em equipamentos e permitindo que o chão de fábrica foque em suas atividades principais.
3. ***Big Data*:** é caracterizado pela presença da grande quantidade e variabilidade de dados, com velocidades de aquisição cada vez maiores, devido à crescente presença de sensores, atuadores e transdutores no ambiente fabril. O grande volume de dados resulta na ineficiência de *softwares* tradicionais, fazendo-se necessário a utilização de técnicas estatísticas e de aprendizagem de máquina para gerenciá-los, de modo a inferir informações e tendências não visíveis em análise puramente humana.
4. ***Cyber* segurança:** é o grupo de infraestruturas de *hardware* e *software* que garantem a proteção de informações, prevenindo ameaças que podem pôr em risco as informações que serão processadas, transportadas e armazenadas pelos sistemas e processos interligados.
5. **Internet das coisas:** caracterizada pela interconexão de equipamentos por meio de eletrônica, *softwares*, sensores, atuadores, transdutores e similares, com a capacidade de se comunicarem em rede podendo ser monitorados e controlados de forma remota afim de proporcionar ganhos na eficiência do processo.
6. **Robótica avançada:** se trata de equipamentos que interagem fisicamente com as pessoas ou processos em que estão inseridos, capazes de alterar o comportamento com base em dados e sensores, podendo agir ou não de forma autônoma.
7. **Manufatura digital:** consiste na utilização de sistesmas integrados que possam simular, analisar e aplicar ferramentas com a finalidade de criar definições de processos e produtos de forma simultânea.
8. **Manufatura aditiva:** é caracterizado pela fabricação de peças sob demanda por meio de impressoras 3D, podendo abranger diversos tipos de materiais como plástico, metal, ligas metálicas, cerâmicas, etc.
9. **Integração de sistemas:** se trata da comunicação entre diferentes sistemas e aplicações que possam atuar de forma coordenada possibilitando a transação de informações. Sendo estas utilizadas para influenciar na tomada de decisões no processo produtivo.

10. **Sistemas de simulação:** consiste no uso de softwares e técnicas computacionais para propor modelos digitais que simulem processos reais, resultados da interação complexa de várias variáveis dentro de um sistema. Também conhecidos como *Digital Twins*.
11. **Digitalização:** caracterizado pelo uso de tecnologias capazes de digitalizar processos de produção, desenvolvimento de produtos e modelos de negócios de forma a otimizá-los.

2.2 Internet das Coisas

A IoT caracteriza-se por uma rede de equipamentos, sejam eles sensores, atuadores, *softwares* e similares, capazes de se conectar e trocarem informações entre si. Hoje, este conceito está bastante presente no cotidiano, seja em objetos domésticos comuns até ferramentas industriais robustas. De acordo com Oracle (2014) há mais de 7 bilhões de dispositivos IoT conectados, com a perspectiva de crescimento para 22 bilhões em 2025.

Segundo Institute (2020), a IoT se tornou possível devido ao crescente desenvolvimento tecnológico de sensores, atuadores, transdutores e sistemas embarcados capazes de se conectar com a rede, favorecendo principalmente sistemas de controle e automação. Neste sentido, equipamentos são capazes de compartilhar dados entre si, utilizando a rede de forma com que atenda as necessidades decisórias sob a qual determinado processo está submetido, permitindo com que certas tarefas se tornem mais eficientes, de maneira automatizada, tendo como ênfase atividades demoradas, repetitivas e por vezes até perigosas.

Ainda de acordo com Institute (2020), é possível listar algumas aplicações comuns, sendo elas:

1. ***Consumer Applications*:** Há um uso amplo de IoT por consumidores em geral, o que inclui veículos conectados, automação residencial como sistemas de iluminação e alto-falantes, recursos de monitoramento remoto, entre outras funcionalidades características de uma *Smart Home*.
2. ***Smart Home Applications*:** Sistemas de iluminação, temperaturas ambiente, assim como mídias e segurança estão presentes no conceito de *Smart Home*, que se utiliza amplamente de IoT. Tais ferramentas devidamente aplicadas podem contribuir bastante na gestão financeira da casa, além de proporcionar maior facilidade no quesito controle e acessibilidade devido ser facilmente implementado junto a *smartphones*, tablets, computadores e correlatos.

3. ***Care Applications*:** Neste modelo de aplicação dispositivos são conectados com a internet e entre si para proporcionar assistência a idosos e/ou pessoas com deficiência de forma a melhorar suas qualidades de vida.
4. ***Medical and Healthcare Applications*:** Referente a presença de IoT em coleta e análise de dados para monitoramento e pesquisa de pacientes, usado para diferentes fins médicos e de saúde, conhecido também como *The Internet of Medical Things* (IoMT).
5. ***Industrial Applications*:** Este tipo de aplicação atua permitindo que dados sejam coletados e compartilhados entre equipamentos e tecnologias industriais, atualizando automaticamente informações e mantendo a eficiência do processo afim de evitar perdas de tempo de produção e, de certa forma, perdas financeiras. Tal aplicação também é conhecida como IIoT.
6. ***Agriculture Applications*:** As aplicações de IIoT na agricutura consiste na coleta de dados referente as condições de solo, clima, pragas, entre outros, afim de contribuir na automatização de técnicas agrícolas e tornar seus processos mais eficientes.
7. ***Military Applications*:** Neste contexto a utilização de IIoT se caracterizam pela vigilância, reconhecimento e várias outras formas de fornecer dados do campo de batalha, podendo incluir sensores e atuadores em munições, veículos, robôs, tecnologias vestíveis, entre outras aplicações que torne o exército mais eficiente. Tal aplicação também é conhecida como *Internet of Military Things*.

2.3 Sistemas Legado

De acordo com Tebet (2021), sistemas legados são aqueles considerados desatualizados que vêm sendo utilizados há muito tempo em processos internos, são geralmente projetados para ter uma longa vida útil, mas devido a diversos avanços da tecnologia acabam não se adequando devidamente à nova dinâmica da organização, como um *software* desenvolvido a bastante tempo, por exemplo.

Para Feathers (2004), partindo do exemplo anterior, um *software* legado pode ser assim considerado devido a sua obsolescência causada pela falta de testes, sem eles se torna impossível verificar se o comportamento do código está ficando melhor ou pior no decorrer do tempo em que está presente em suas aplicações. Tal descrição pode ser aplicada a qualquer tipo de sistema legado, sendo eles aqueles que tendem a ficar ultrapassados com o passar o tempo devido à falta de testes que visam validar suas eficiências no ambiente de produção. Em Feathers

(2004) também é apresentado que estes sistemas são caracterizados por dependerem bastante de intervenção humana e a presença de suas configurações que deixam a desejar.

Na perspectiva de Tebet (2021) um sistema legado pode ser observado na presença de características bem distintas, sendo elas: *software* desatualizado, tecnologia obsoleta, falta de mobilidade, falta de colaboradores treinados, falta de escalabilidade, falta de suporte, incompartibilidade com sistemas modernos e *software* hospedado em servidores físicos. Todos estes são fortes indícios de que o processo está ultrapassado.

2.4 Node.js

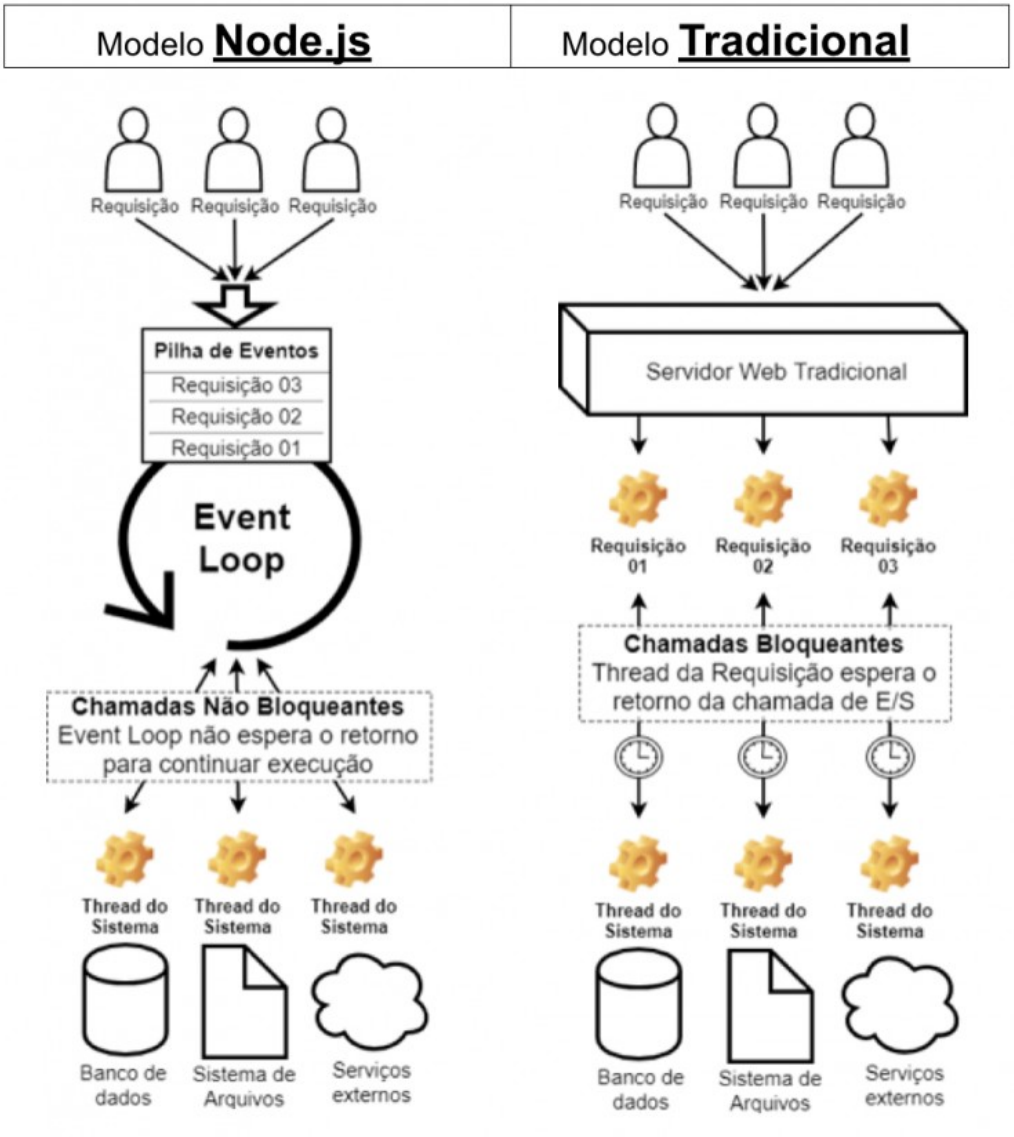
Node.js é um ambiente de programação em JavaScript que permite ao seu desenvolvedor criar ferramentas e aplicações que funcionem de forma autônoma, independente de qualquer navegador. Desta forma, é ideal para desenvolver as mais variadas aplicações WEB, desde APIs, sites e até servidores.

Segundo Melo (2021), Node.js teve sua criação no ano 2009 pelo engenheiro de *software* chamado Ryan Dahl como proposta alternativa ao *Apache Hypertext Transfer Protocol (HTTP) Server*, o mais popular servidor WEB até então. Criada em 2019, a *OpenJS Foundation* vem mantendo e promovendo a utilização do Node.js até os dias atuais.

Na Figura 2 a seguir é possível observar a comparação de funcionamento entre um servidor Node.js e um tradicional. Como bem explicado por Lenon (2018), no servidor em Node.js o *Event Loop* presente na Figura 2 é a única tarefa que trata as requisições, enquanto no modelo de servidor tradicional uma tarefa seria iniciada para cada requisição realizada. De forma assíncrona, as operações de entrada e saída são designadas pelo *Event Loop* às tarefas do sistema e continua processando as demais solicitações presentes na pilha de eventos, enquanto as tarefas do modelo tradicional permanecem aguardando a finalização de cada operação de entrada e saída, consumindo recursos computacionais durante esta espera.

Sendo um ambiente de programação usado por empresas como Uber, LinkedIn, Netflix, PayPal, entre outras, vem ganhando espaço entre os programadores pela sua arquitetura, baixo custo, flexibilidade e principalmente sua escalabilidade, assim, ideal para as aplicações deste trabalho.

Figura 2 – Diferença de funcionamento entre um servidor web tradicional e um servidor em Node.js



Fonte: Lenon (2018).

2.4.1 Métodos de requisição HTTP

De acordo com Contributors (2022), *HTTP* é um protocolo de camada de aplicação projetado para transmitir documentos em hipermídia, como HTML. É usado para comunicar servidores e navegadores da web, entre outros meios, como aplicativos móveis. Nesse protocolo, o cliente abre a conexão com um servidor, realiza a solicitação, ou emissão, de dados e aguarda uma resposta.

O protocolo *HTTP* utiliza métodos específicos, em que cada um atende diferentes finalidades, sendo alguns deles: GET, onde os parâmetros são informados no cabeçalho da requisição, usado bastante para consultar informações; POST, diferente do método anterior, este

passa os dados no corpo da requisição, geralmente utilizado para submeter informações a qual deseja-se armazenar; PUT, similar ao POST, é utilizado para submeter informações, no entanto, para substituir dados anteriores já armazenados sobre determinada informação; e DELETE, método responsável por deletar dados. Neste projeto foram utilizados os métodos GET, POST, PUT e DELETE, de forma com que atendessem todas as necessidades do sistema proposto.

2.5 Node-RED

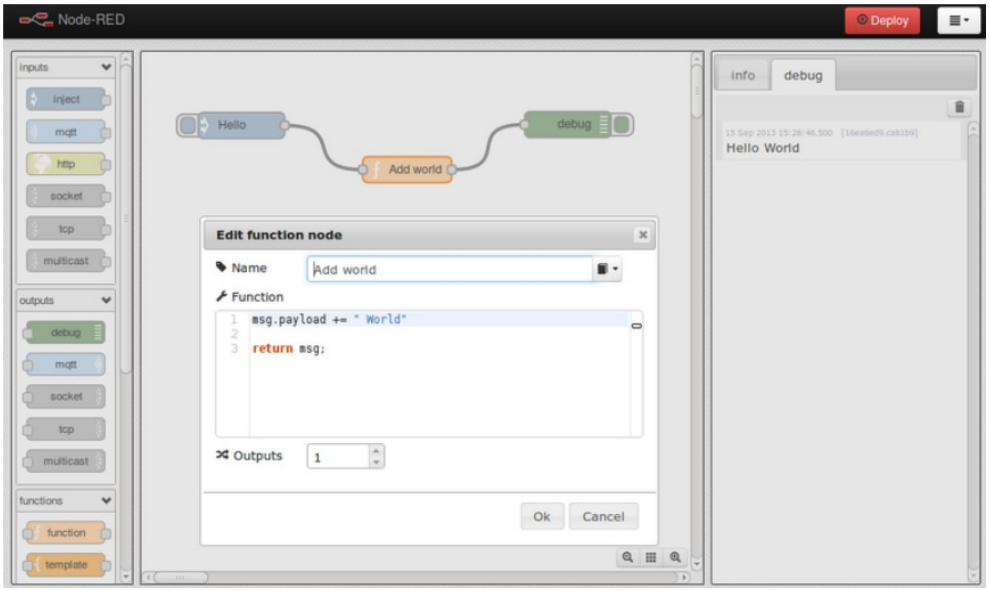
Node-RED é uma ferramenta de programação originalmente desenvolvida pela IBM, hoje disponibilizada pela *OpenJS Foundation*. Segundo Foundation (2021), o programa foi inventado por J. Paul Morrison na década de 1970 com o intuito bem definido de desenvolver um programa baseado em *flows* capaz de transportar dados e informações pelo que é pontuado como "nós", ou *nodes*. Visando bastante a representação visual, o programa se tornou acessível por grande quantidade de usuários, podendo não só interpretar mas também programar atividades sem necessariamente ter que entender o que acontece individualmente no código de cada nó.

Node-RED é um programa baseado em *JavaScript* desenvolvido e utilizado em Node.JS, aplicação responsável por executar o programa e junto a um navegador de internet viabilizar um ambiente adequado para que seja utilizado pelo usuário. De modo visual, fica fácil e prático observar e manipular os "nós" referente as tarefas do programa, podendo facilmente não apenas manipular mas também gerar dados e compartilhá-los via requisições WEB ou arquivos JSON, uma forma de notação para objetos em *JavaScript* que proporciona um padrão para trafegar informações entre aplicações, por exemplo.

Hoje, se trata de uma ferramenta tão prática que pode ser instalada e utilizada das mais diversas maneias. Em Foundation (2021) é abordado na documentação métodos de instalação em Raspberry PI, Docker, BeagleBone Boards, Android, e até sistemas em nuvem da *IBM Cloud*, Microsoft Azure e AWS.

Segundo Calabrez (2019), os blocos arredondados característicos no *layout* da plataforma de programação, como visto na Figura 3, são chamados de nodes, sendo estes classificados como elementos de *input*, *output* ou *function*. Ainda na Figura 3 é possível observar um exemplo básico de como se torna simples a utilização da aplicação, onde um node *input* contendo "*Hello*" conecta-se com outro node *function* responsável por conter a função em Javascript que acrescenta a palavra "*World*", resultando no parâmetro de saindo "*Hello World*" presente no node *output* chamado *Debug*.

Figura 3 – Função de "*Hello World*" no Node-RED.



Fonte: Calabrez (2019).

2.5.1 node-red-contrib-s7

Criada pelo time de desenvolvedores da ST-One[1] para interagir diretamente com os PLCs s7 da Siemens, de acordo com Cittolin (2022) a biblioteca *node-red-contrib-s7* fornece *nodes* como *s7 in*, *s7 out*, e *s7 control*, que permitem a conversação entre dispositivos S7 e o Node-RED, gerenciando a comunicação em baixo nível e fazendo com que a leitura e escrita de *tags*, variáveis do PLC, sejam realizadas em *JavaScript*, linguagem de alto nível, de forma prática e dinâmica. A biblioteca usa o protocolo de comunicação Siemens S7 ISO-on-TCP (RFC1006).

2.6 ReactJS

ReactJS é uma das bibliotecas baseadas em JavaScript mais populares da atualidade, ideal para construir sites e sistemas WEB. Considerado uma biblioteca para desenvolvimento *front-end*, interface gráfica do usuário, o ReactJS apresenta a capacidade de criar e manipular elementos visuais reutilizáveis, trazendo velocidade, simplicidade e escalabilidade para seus projetos.

Diferente dos sites tradicionais que solicitam e carregam várias páginas em *HiperText Markup Language* (HTML), ReactJS tem a capacidade de renderizar diversos elementos em tela utilizando apenas um único documento HTML, explorando bastante o conceito de *Single Page Application* (SPA).

[1] https://st-one.io/

Desenvolvido pelo Facebook e lançado como código aberto em 2013, tem sido utilizado por grandes empresas como Netflix, eBay, Instagram, Airbnb, etc. desde então, o que aumenta cada vez mais a credibilidade e, principalmente, a sua comunidade de desenvolvedores que fornecem várias ferramentas e componentes que compartilham entre si.

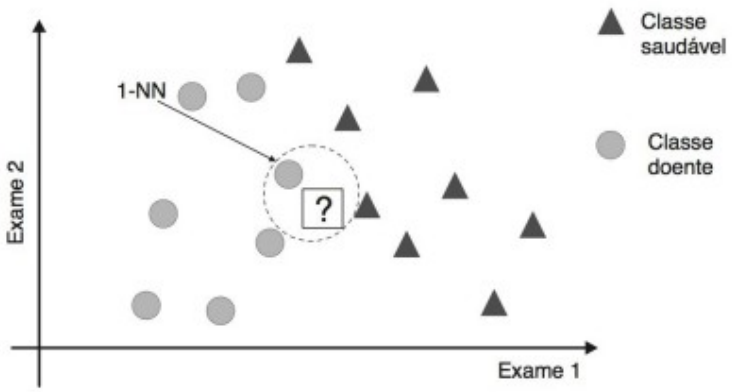
2.7 KNN

O KNN é um algoritmo de *machine learning*, bastante utilizado em problemas de classificação. Considerado um algoritmo preguiçoso, este método não aprende um modelo compacto para os dados, apenas se utiliza dos objetos de treinamento, no entanto, sua vantagem se deve ao fato de poder ser usado diretamente tanto para problemas de classificação como regressão sem grandes mudanças.

Em Faceli *et al.* (2011), é apresentado que neste algoritmo cada objeto passa a representar um ponto no espaço caracterizado por seus atributos, e ao definir uma forma de medida neste espaço é possível encontrar a distância entre eles. Dentre as métricas podem ser citadas: Euclidiana, Manhattan, Minkowski, Chebyshev, Sorensen, Canberra, entre outras.

Ainda em Faceli *et al.* (2011), é explicado que o KNN na fase de treinamento decora os exemplos rotulados para então poder classificar um exemplo ainda não rotulado com base em seus parâmetros de entrada, associando-o a uma quantidade K de exemplos do treinamento que estejam mais próximos. Na Figura 4 é apresentado um exemplo da aplicação do algoritmo para K=1, no qual os triângulos representam pessoas saudáveis e os círculos pessoas doentes, e o espaço de entrada é definido por dois atributos, representando o resultado de dois exames. Onde se encontra a "?" é o elemento que se deseja rotular, mais próximo de um círculo, doente, logo, para K=1 o elemento também se classificará como doente.

Figura 4 – Exemplo ilustrativo do algoritmo 1-KNN.



Fonte: Faceli *et al.* (2011).

Retratado por Castro e Ferrari (2016), o pseudocódigo deste algoritmo pode ser observado na Figura 5.

Figura 5 – Pseudocódigo do algoritmo KNN.

```
Entrada
   k : número de vizinhos
   data : base de dados com n objetos e m atributos (n x m)
   classe : vetor contendo a classe de cada objeto da base (n x 1)
   obj : objeto que deve ser classificado (1 x m)
Saída
   C : rótulo indicativo da classe do objeto
Passos
   // Calcular a distância entre a base de dados e o objeto D(n x 1)
   D = dist(data,obj);

   objs = ø;
   // Determinar os k objetos mais próximos
   Para i=1:k Faça
   {
      aux = D[1];
      pos = 1;
      Para j=2:n Faça
      {
         Se (aux > D[j]) e (j ∩ objs == ø) Então
         {
            aux = D[j];
            pos = j;
         }
      }
      objs.Add(pos);
   }

   // Pegar a classe dos k objetos mais próximos
   Ck = classe[objs];

   // Determinar a classe mais frequente
   C = moda(Ck);
```

Fonte: Castro e Ferrari (2016).

2.7.1 Distâncias

Para utilizar o KNN, definir adequadamente o método de cálculo da distância a ser utilizado é crucial para o bom desempenho do classificador. Em Hurtado (2022) é abordada algumas distâncias, onde para requisito de análise será utilizada apenas três distâncias neste projeto, sendo elas: Distância Euclidiana, Distância de Soergel e Distância de Manhattan, Equação 2.1, Equação 2.2 e Equação 2.3, respectivamente. Para ambas x_i e x_j são objetos representados por vetores no espaço R^d, e x_i^l e x_j^l são os respectivos elementos destes vetores correspondendo a sua posição l. A distância entre eles é dada por:

$$d_{euc}(x_i, x_j) = \sqrt{\sum_{l=1}^{d} (x_i^l - x_j^l)^2}. \quad (2.1)$$

$$d_{soe}(x_i, x_j) = \frac{\sum_{l=1}^{d} |x_i^l - x_j^l|}{\sum_{l=1}^{d} max(x_i^l, x_j^l)} \quad (2.2)$$

$$d_{manh}(x_i, x_j) = \sum_{l=1}^{d} |x_i^l - x_j^l| \quad (2.3)$$

2.7.2 Acurácia

Para medir a eficiência de classificadores como o KNN, são necessárias algumas métricas, sendo a acurácia uma delas. De acordo com Filho (2021), essa métrica constrói a relação entre o número de elementos classificados corretamente e o número total de elementos da classificação, apresentando como resultado valores sempre entre 0 e 1. Desta forma, quanto mais próximo de 1, melhor o desempenho do classificador.

Com isso, em Faceli *et al.* (2011) é abordado um método de analisar a eficiência do classificador para problemas com duas classes baseando-se na matriz confusão da Tabela 1 e Equação (2.4), em que, conforme a classificação das amostras: VP, verdadeiro positivo; VN, verdadeiro negativo; FP, falso positivo; e FN, falso negativo.

Tabela 1 – Matriz confusão para um problema com duas classes.

	Classe Positiva (Prevista)	Classe Negativa (Prevista)
Classe Positiva (Real)	VP	FN
Classe Negativa (Real)	FP	VN

Fonte: Adaptado de Faceli *et al.* (2011).

$$Acurácia\ Total = \frac{VP + VN}{VP + FP + VN + FN} \quad (2.4)$$

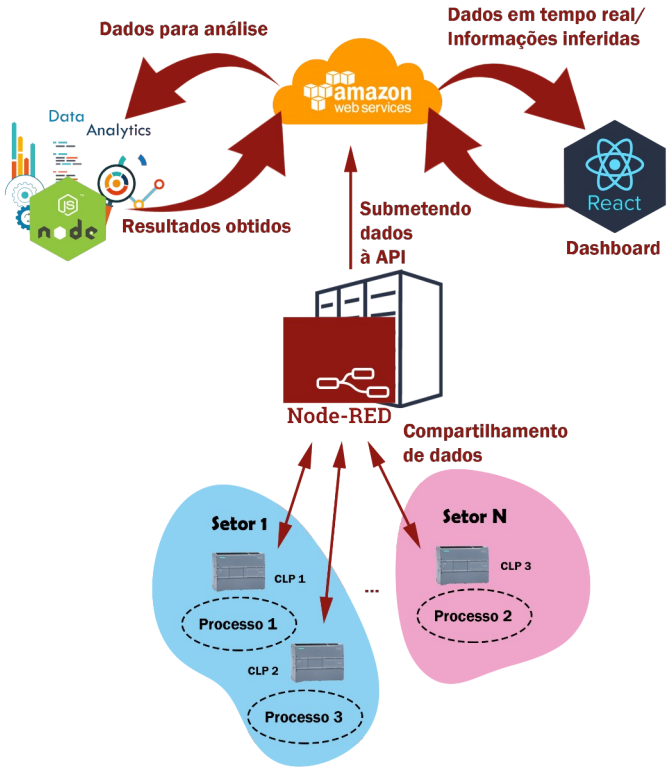
3 METODOLOGIA

Neste capítulo será apresentada a metodologia utilizada no desenvolvimento de uma solução baseada em Node-RED para adaptar equipamentos legados de pequenas e médias automações ao contexto da Indústria 4.0. Para demonstração da solução, foi realizado um estudo de caso referente ao funcionamento de uma máquina industrial e suas possíveis situações de falha. A seção 3.1 apresenta de forma geral a modelagem do fluxo proposto para o desenvolvimento do projeto. A seção 3.2 apresenta de maneira mais detalhada os passos necessários para a construção do trabalho seguindo o fluxo apresentado na seção 3.1. Por fim, a seção 3.3 apresenta os materiais e métodos utilizados no decorrer do projeto proposto.

3.1 Visão Geral

A Figura 6 apresenta a visão geral da solução proposta por este trabalho. A ideia básica é ter um "ecossistema ciberfísico" capaz de viabilizar que sistemas legados sejam adaptados aos conceitos de Indústria 4.0. A solução visa atender pequenas e médias automações.

Figura 6 – Ecossistema ciberfísico proposto.



Fonte: Próprio autor (2022).

Nesse “ecossistema ciberfísico” o fluxo de informação ocorre a partir de um conjunto de PLC’s com seus processos interconectados em setores, os quais por sua vez passam a se conectar com outros setores por meio do Node-RED. O Node-RED, no que lhe concerne, será responsável por fazer com que estes modelos de controladores mantenham contato entre si e enviem dados para aplicação em *cloud* da AWS, de forma com que sejam tratados e informações pertinentes dos processos a qual estão sendo analisados possam ser inferidas.

3.2 Procedimentos metodológicos

A construção da solução proposta é dividida em seis partes. A primeira é a configuração de um *gateway* com Node-Red. A segunda, configuração de um PLC para comunicação com Node-RED. A terceira, desenvolvimento de uma API para recebimento, armazenamento e disponibilidade de dados em nuvem. A quarta parte é a análise de dados. A quinta, desenvolvimento de um sistema de supervisão usando o ReactJS. Por fim, a última parte consiste na hospedagem da API e sistema de supervisão na AWS. Como demonstrado na Figura 7 a seguir.

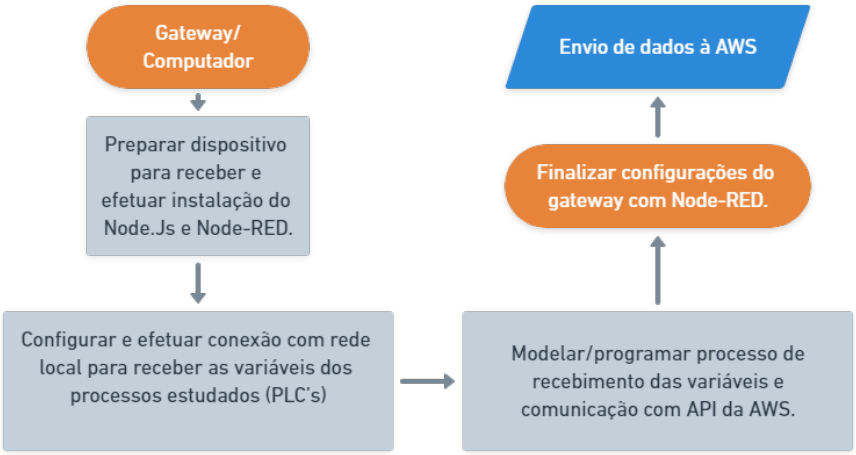
Figura 7 – Diagrama das etapas adotadas no desenvolvimento da solução proposta.

Fonte: Próprio autor (2022).

3.2.1 Configuração do gateway com Node-RED

O Node-RED foi instalado em um computador para que funcione como gateway na solução. Antes de começar a configurar as tarefas no Node-RED, é importante assegurar que os dispositivos os quais deseja mapear estejam visíveis na mesma rede que o *gateway*, computador, de forma com que as variáveis possam ser encontradas. Com isso, é possível configurar o projeto em Node-RED de forma com que receba as variáveis, *tags*, e submeta suas informações via *HTTP Request* utilizando método *POST* à API na AWS. Tais processos podem ser observados no diagrama da Figura 8.

Figura 8 – Diagrama para configurar o computador como *gateway*.

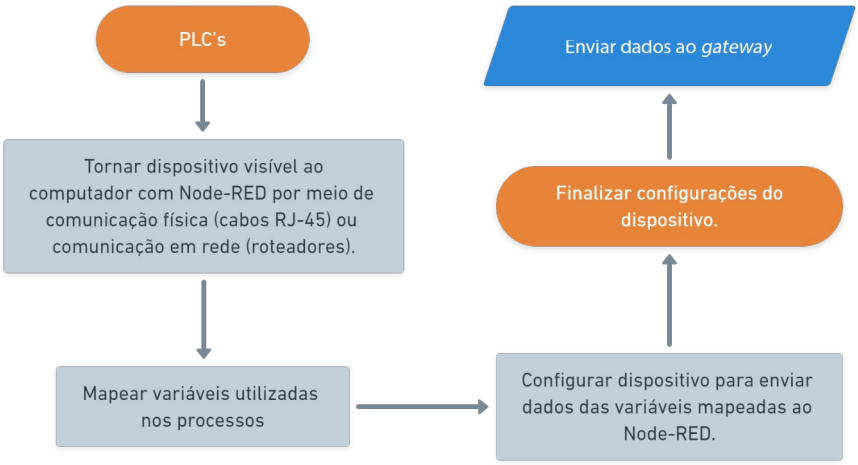


Fonte: Próprio autor (2022).

3.2.2 Configurando a comunicação entre PLC e o gateway

O PLC neste trabalho foi utilizado para demonstrar um dispositivo presente nos sistemas legados. A princípio é necessário assegurar que o dispositivo esteja visível ao computador com Node-RED, seja por meio físico com cabos RJ-45 ou via rede utilizando roteadores, e com isso dar início ao processo de configurar o controlador, visto que já esteja ajustado para a tarefa a qual está executando, para mapear as variáveis utilizadas no PLC e ajustar a rotina no Node-RED para então iniciar o compartilhamento de informações. Os passos comentados podem ser observados no diagrama presente na Figura 9.

Figura 9 – Diagrama dos passos a serem adotados afim de deixar o PLC apto ao desenvolvimento do projeto.

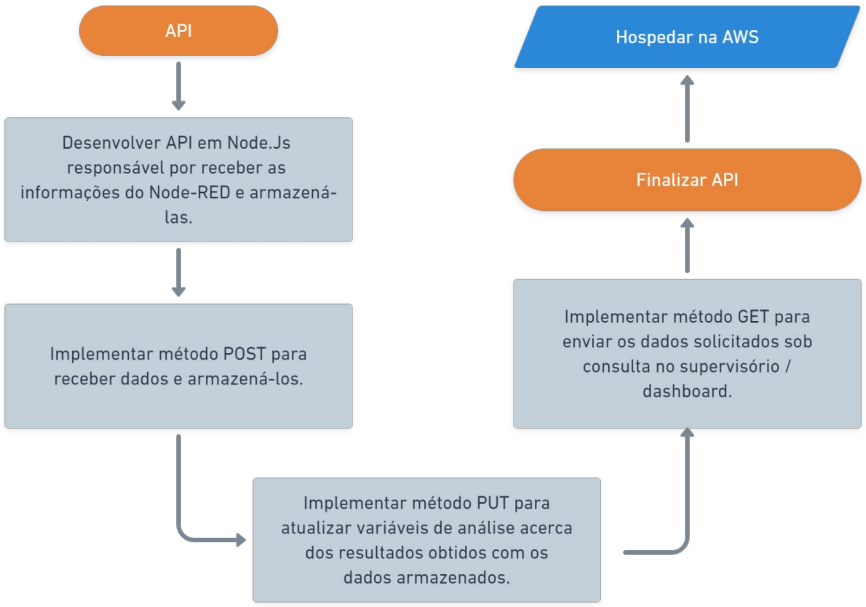


Fonte: Próprio autor (2022).

3.2.3 Desenvolvimento da API

A API é responsável por gerenciar a comunicação entre os dados recebidos dos PLC's e seu armazenamento, para disponibilizá-los quando solicitado. Existem várias linguagens e formas de desenvolver esta parte do processo, como PHP, Ruby, Python, mas neste projeto será utilizado Node.Js, pois, também está presente em outras partes deste trabalho. É essencial que a API receba essas informações e seja capaz de armazená-las em um banco de dados, para isto o método *POST* deve ser implementado e consumido pelo Node-RED. Tais dados armazenados são passíveis de atualização, não só referente aos processos enviados pelos PLC's mas também pela rotina de análise abordada no parágrafo subsequente, criando-se assim o método *PUT*. As informações. já tratadas e armazenadas, precisam estar disponíveis para consulta no sistema de supervisão, e para isso é preciso implementar o método de requisição chamado *GET*. Tais etapas podem ser observadas no diagrama da Figura 10.

Figura 10 – Diagrama dos passos a serem adotados afim de criar a API utilizada no desenvolvimento do projeto.

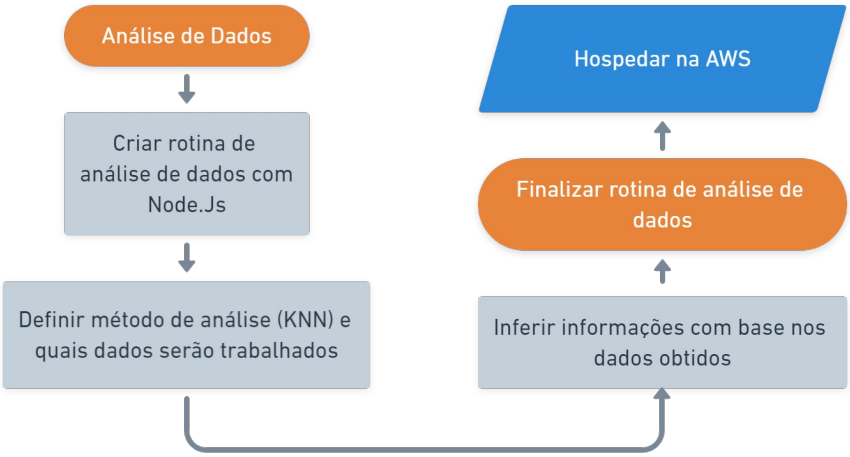


Fonte: Próprio autor (2022).

3.2.4 Procedimento para a análise de dados

O armazenamento de dados é importante, no entanto, o real valor está na capacidade de inferir informações a partir desses dados. O processo de análise é utilizado para as mais diferentes finalidades, e no ambiente industrial é frenquentemente encontrado no contexto de manutenção e eficiência de processos, e pode ser realizado de várias formas, métodos e ferramentas. Neste trabalho, variáveis de um processo industrial estudado foram analisadas utilizando o método de classificação KNN. Novamente com base em Node.Js, a rotina de análise é definida utilizando métodos de *HTTP Request*, retornando os devidos resultados. Na Figura 11 o diagrama refente a este parágrafo traz de forma resumida os passos supracitados.

Figura 11 – Diagrama dos passos a serem adotados afim de criar a rotina de análise de dados utilizada no desenvolvimento do projeto.



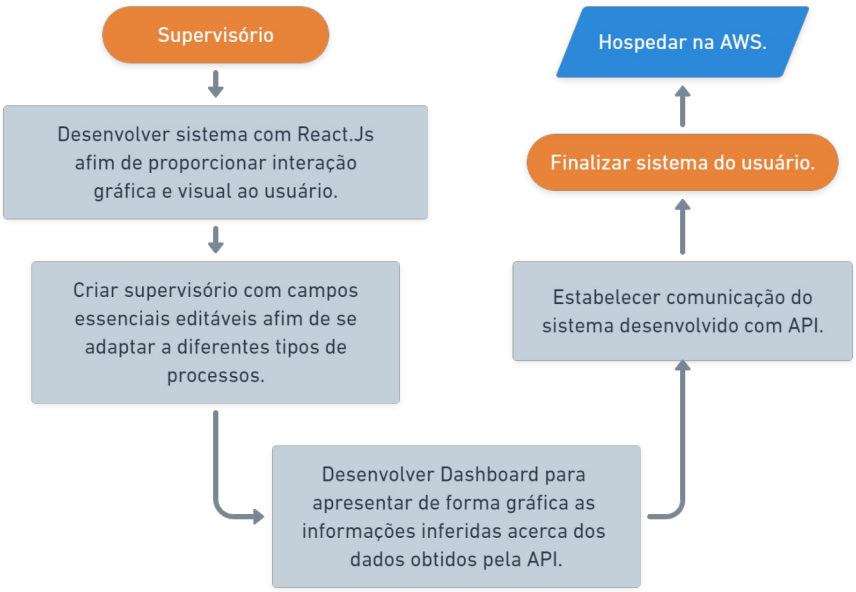
Fonte: Próprio autor (2022).

3.2.5 Desenvolvimento do sistema de supervisão

Após a coleta e armazenamento, os dados estão prontos para serem consultados, no entanto, para melhor experiência do usuário que deseja acompanhar o processo monitorado é essencial que haja tratamento visual das informações recebidas periodicamente, um sistema supervisório. O seu desenvolvimento pode ser realizado de várias formas e maneiras, utilizando linguagens como Java, PHP, Python, etc. Neste trabalho optou-se por utilizar ReactJS, por também funcionar no ambiente de desenvolvimento Node.Js utilizado em outras etapas do projeto. O supervisório deve ter variáveis capazes de mudar seus valores conforme os dados obtidos da API, assim como o *layout* deve ser dinâmico para adaptar-se a diferentes processos,

sob critério do usuário caso deseje que o método de análise seja modificado. Por fim, o supervisório também irá contar com um campo de *Dashboard* capaz de interpretar de forma gráfica as informações obtidas da análise de dados apresentada em 3.2.4, estando assim hábil a ser hospedado na AWS e completar o desenvolvimento do projeto proposto. Os passos descritos aqui podem ser visualizados sucintamente na Figura 12 .

Figura 12 – Diagrama dos passos a serem adotados afim de criar o supervisório utilizado no desenvolvimento do projeto.

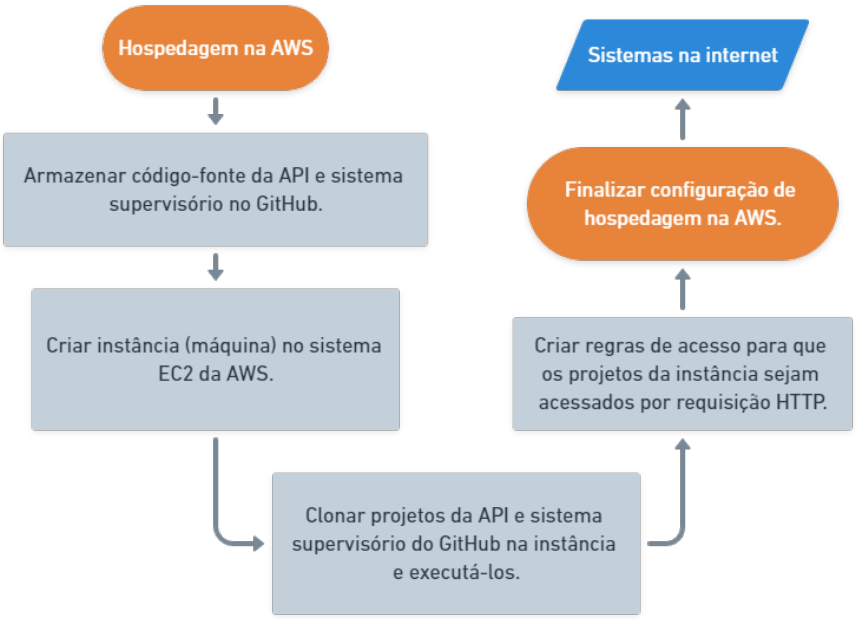


Fonte: Próprio autor (2022).

3.2.6 Hospedagem na AWS

A API e o sistema supervisório desenvolvidos são enviados de forma privada ao GitHub, plataforma que armazena códigos-fonte online, e para torná-los acessíveis a demais dispositivos com acesso à internet, é utilizado sistema de computação em nuvem chamado *Amazon Elastic Compute Cloud* (Amazon EC2), serviço disponibilizado pela AWS, onde é criada instância (máquina) que executará os projetos. As etapas são brevemente apresentadas na Figura 13.

Figura 13 – Diagrama das etapas a serem seguidas para hospedar os projetos na AWS.



Fonte: Próprio autor (2022).

3.3 Materiais e Métodos

Nesta seção será apresentado os materiais e métodos utilizados para executar os procedimentos apresentados de 3.2.1 a 3.2.6, de forma com que atenda aos critérios de desenvolvimento necessários para que o projeto proposto seja construído e resultados sejam obtidos.

3.3.1 Instalando e configurando Node-RED

Assim como as demais etapas que exigem programação apresentadas nos tópicos a seguir, nesta etapa foi utilizado um computador de uso pessoal para instalação do Node-RED. Suas especificações podem ser observadas na Tabela 2 a seguir.

A princípio, foi realizado a instalação do Node.Js[2] no computador, onde é necessário escolher a versão conforme as especificações da máquina. Neste trabalho foi utilizado o instalador .msi para Windows, versão 16.13.0, 64-bit, como pode ser observado na Figura 14. Após o *download* a instalação pode ser realizada normalmente seguindo os passos do arquivo executável.

[2] https://nodejs.org/pt-br/download/

Tabela 2 – Especificações do computador utilizado para desenvolver o supervisório.

Nome	Especificação
Marca	ACER
Modelo	Aspire 3
Armazenamento	512GB
Memória	12GB
Processador	Ryzen 5
Software	Visual Studio Code V1.62.3
Navegador	Microsoft Edge V96.0.1054.41

Fonte: Próprio autor (2022).

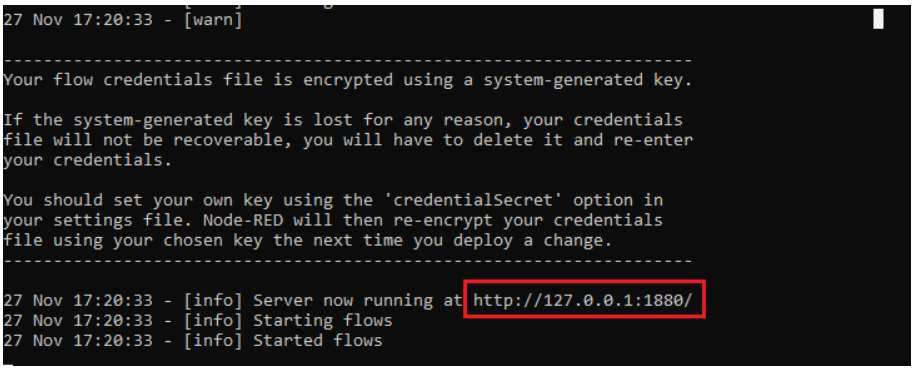
Figura 14 – Versão do Node.Js escolhida para o projeto.

Fonte: Próprio autor (2022).

Com a instalação do Node.js concluída no computador, é possível instalar o node-RED. Utilizando o *prompt* de comandos o Node-RED pode ser instalado executando o comando "npm install -g –unsafe-perm node-red". Após a instalação finalizada o comando “node-red” é usado para executar o projeto, e o endereço de acesso será apresentado na tela para poder utilizar a plataforma no navegador de sua preferência, como destacado na Figura 15. Para mais detalhes sobre instalação e execução, a documentação[3] pode ser consultada.

[3] https://nodered.org/docs/

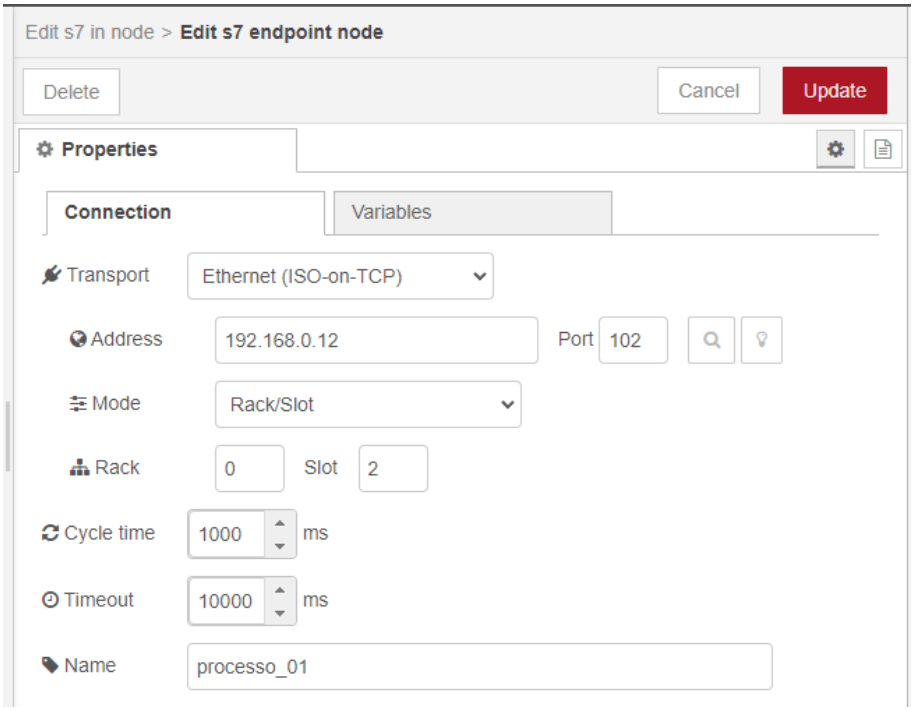
Figura 15 – Execução do Node-RED no prompt de comandos do Windows.



Fonte: Próprio autor (2022).

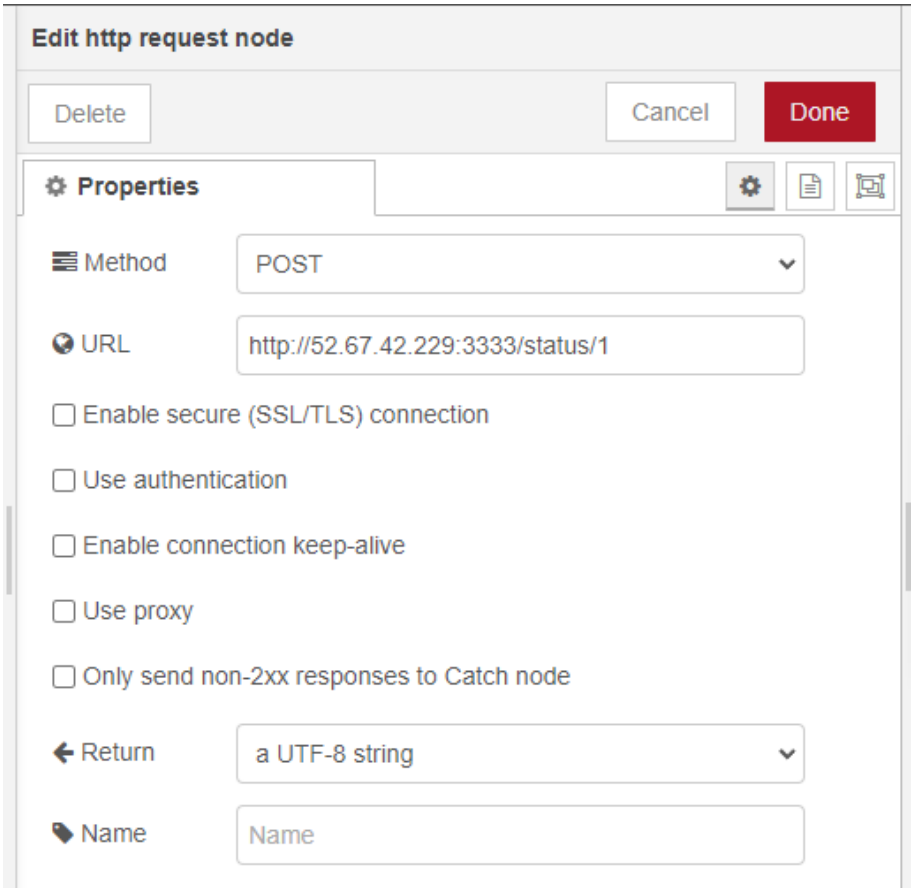
Com o projeto Node-RED iniciado, na plataforma é instalada a biblioteca "node-red-contrib-s7" que viabilizará a criação do *flow* responsável por comunicar-se com o PLC, a qual realizará a leitura das *tags* pré programadas a cada 1000ms. Sua configuração está presente na Figura 16, onde o IP do PLC será 192.168.0.12, usando porta 102, no *slot* 2 e ciclo de aquisição de dados com valor de 1000ms. Logo então fará a requisição via *POST* usando o *node HTTP request* enviando as informações para a API com a rota para atualização de *status* "http://52.67.42.229:3333/status/1", com sua configuração detalhada na Figura 17. O resultado final pode ser observado na Figura 18, demonstrando de forma visual como é fácil, rápido e prático utilizar o Node-RED.

Figura 16 – Configuração de conexão com *node* node- red-contrib-s7.



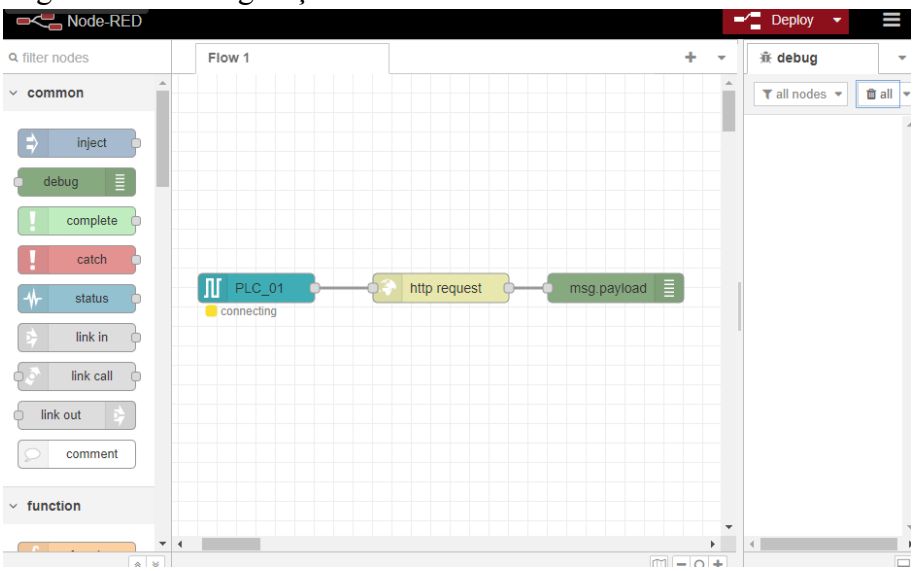
Fonte: Próprio autor (2022).

Figura 17 – Configuração do *node http request* .



Fonte: Próprio autor (2022).

Figura 18 – Configuração do Node-RED.



Fonte: Próprio autor (2022).

3.3.2 Configurando PLC

Neste projeto é utilizado o PLC SIMATIC S7-1200 1214c ac/dc/rly , da fabricante Siemens, disponível no laboratório de controle da Universidade Federal do Ceará campus Sobral. Este modelo possui quatorze entradas digitais, dez saídas digitais e duas entradas analógicas. O controlador pode ser observado na Figura 19.

Figura 19 – Controlador Lógico Programável utilizado.

SIEMENS

Data sheet **6ES7214-1BG40-0XB0**

SIMATIC S7-1200, CPU 1214C, compact CPU, AC/DC/relay, onboard I/O: 14 DI 24 V DC; 10 DO relay 2 A; 2 AI 0-10 V DC, Power supply: AC 85-264 V AC at 47-63 Hz, Program/data memory 100 KB

General information	
Product type designation	CPU 1214C AC/DC/relay
Firmware version	V4.5
Engineering with	
• Programming package	STEP 7 V17 or higher
Supply voltage	
Rated value (AC)	
• 120 V AC	Yes
• 230 V AC	Yes
permissible range, lower limit (AC)	85 V
permissible range, upper limit (AC)	264 V

Fonte: Siemens (2021).

A programação deste dispositivo é realizada por meio da linguagem Ladder utilizando o período gratuito de vinte e um dias do *software Totally Integrated Automation Portal* (TIA *Portal*) V14, também da Siemens. De acordo com Siemens (2019), nesta plataforma é possível realizar as seguintes funções para automação:

– Configuração e parametrização de *hardware*;

– Determinação da comunicação;

– Programação;

– Teste, comissionamento e assistência técnica com as funções de operação/diagnóstico;

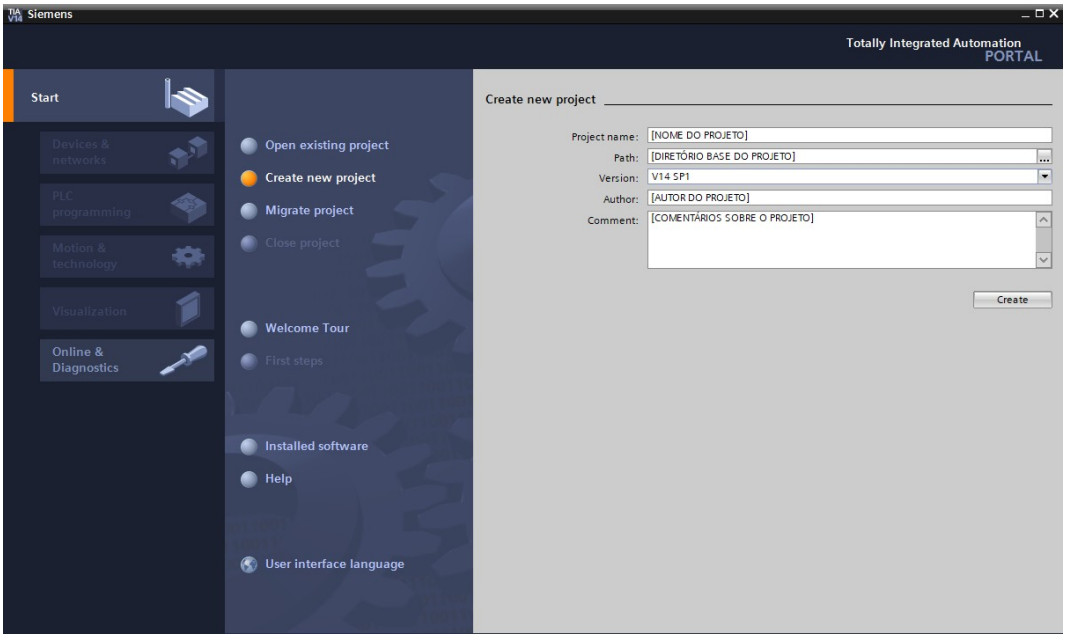
– Documentação;

– Criação das visualizações *SIMATIC Basic Panels* com o *WinCC Basic* integrado;

– Com outros pacotes WinCC também podem ser criadas soluções de visualização para PCs e outros *Panels*.

A interface inicial do *software* é apresentada na Figura 20 onde é especificado o nome do projeto, diretório base para ser salvo, versão do TIA *Portal* utilizado, autor do projeto e comentário. Para a criação do novo projeto foi dado o nome “projeto_tcc”, e após clicar no botão *create* é apresentado as opções de dispositivo, o qual receberá o nome “PLC_1” e os parâmetros selecionados estão presentes na Figura 21, com *firmware* v2.2, versão que não apresenta suporte para comunicação *OLE for Process Control* (OPC), tornando-o um sistema legado perfeito para o estudo de caso deste trabalho. É preciso atenção nas configurações iniciais, será criado uma nova *subnet* e o endereço IP receberá o valor "192.168.0.12", como apresentado na Figura 22, responsável por tornar o PLC visível na rede para o Node-RED. Logo então, a programação em Ladder referente ao processo pode ser realizada normalmente. Nota-se que neste projeto não foi utilizado roteador para realizar comunicação, mas há a possibilidade para este procedimento preenchendo o campo *use router* com o endereço IP do dispositivo.

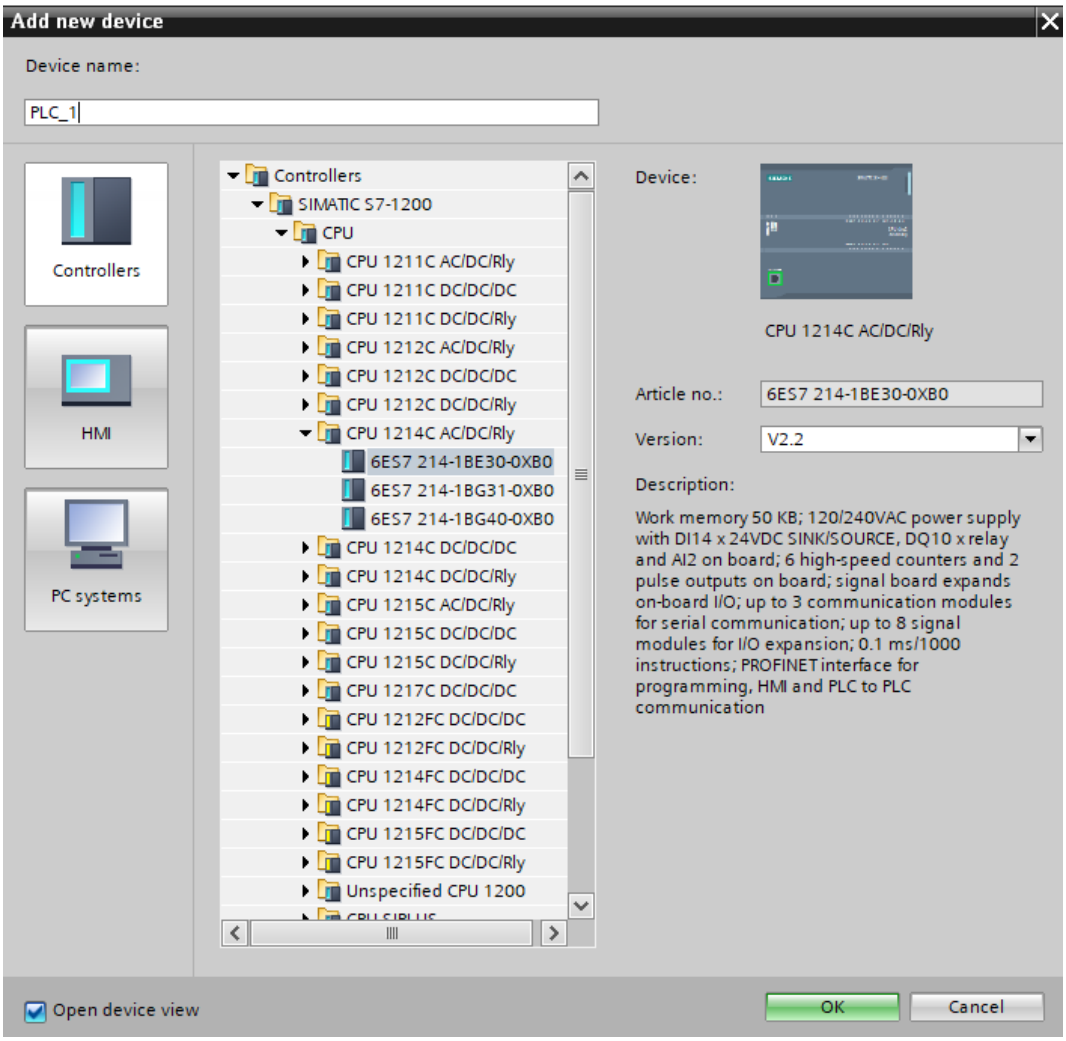
Figura 20 – Interface inicial do software TIA Portal V14 .



Fonte: Próprio autor (2022).

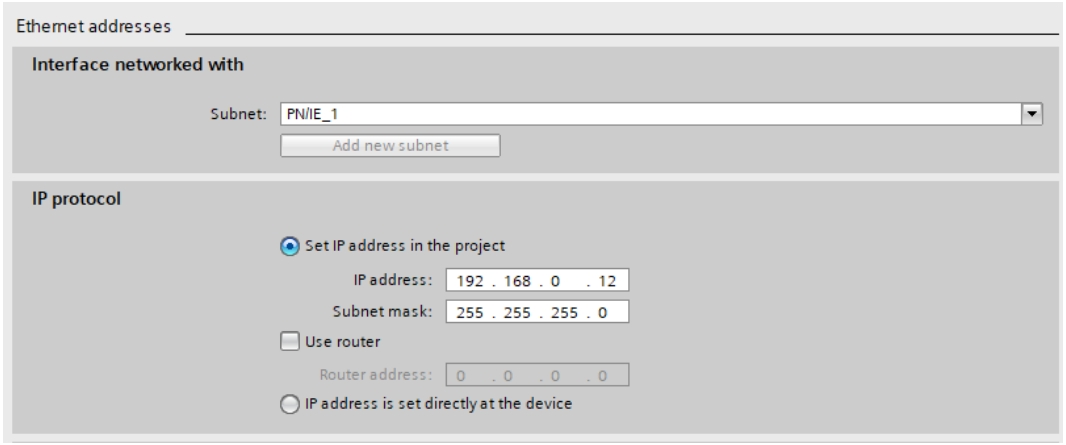
A requisito de análise, o projeto irá simular um processo que apresenta a interação entre as informações recolhidas de uma máquina industrial adquiridas do *dataset* “*AI4I 2020 Predictive Maintenance Dataset Data Set*”, desenvolvido em Matzka (2020) , utilizado para a análise de dados presente na sessão 3.3.4. As variáveis utilizadas podem ser acompanhadas na Tabela 3.

Figura 21 – Parâmetros do dispositivo PLC selecionado.



Fonte: Próprio autor (2022).

Figura 22 – Configurações de *Ethernet addresses*.



Fonte: Próprio autor (2022).

Tabela 3 – Detalhes das variáveis utilizadas para simular um processo industrial.

Endereço	Tipo	TAG	Descrição
MW1	Int	air_temperature_k	Estado responsável por informar temperatura do ar no processo em Kelvin.
MW4	Int	tool_wear_min	Responsável por informar qualidade da ferramenta no processo.
MW8	Int	rotational_speed_rpm	Atributo que informa a velocidade de rotação em rpm.
MW12	Int	process_temperature_k	Variável que irá informar a temperatura do processo em Kelvin.
MW14	Int	torque_nm	Estado responsável por informar o torque em Nm.
I0.0	Bool	gatilho	Variável de entrada para disparo manual de informações.

Fonte: Próprio autor (2022).

Vale ressaltar que, para fins de simulação, neste projeto, com exceção do *gatilho*, apenas variáveis de memória foram utilizadas, no entanto, quaisquer outras variáveis são aceitáveis.

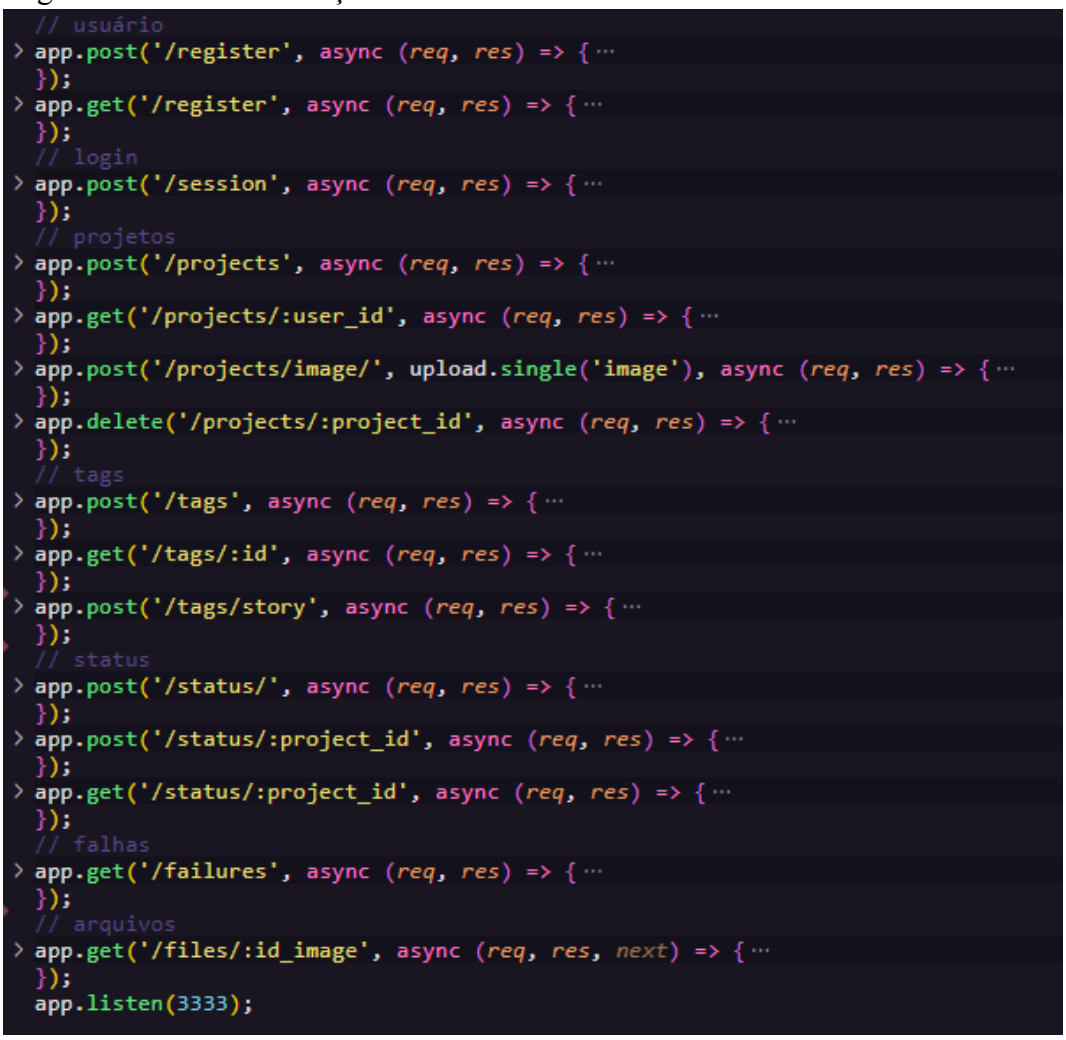
3.3.3 API

Construída em Node.JS, a API recebeu o nome de "projectapi" e as seguintes bibliotecas foram instaladas: *express*, para gerenciar diferentes requisições *HTTP*; *sequelize*, estrutura *Object-Relational Mapper* (ORM) para dar suporte a comunicação com banco de dados utilizando javascript; e o *Cross-Origin Resource Sharing* (CORS), biblioteca que permite um site ter acesso a recursos de outro com origens diferentes. Com a finalidade de armazenar as informações recebidas do Node-RED, foi utilizado o sistema de gerenciamento de banco de dados de código aberto MySQL, que se utiliza da linguagem SQL como interface.

Criadas utilizando o *express*, as rotas são responsáveis por distinguir cada ação que será requisita à API, sendo elas presentes na Figura 23 e descritas na Tabela 4.

Em analogia as rotas supracitadas, no banco de dados chamado "projectdata" foram criadas as tabelas: *apparentfailures*, *projects*, *status*, *tags* e *users*. As especificações de seus atributos são descritas na Tabela 5.

Figura 23 – Rotas/endereços da API.



```
// usuário
app.post('/register', async (req, res) => {···
});
app.get('/register', async (req, res) => {···
});
// login
app.post('/session', async (req, res) => {···
});
// projetos
app.post('/projects', async (req, res) => {···
});
app.get('/projects/:user_id', async (req, res) => {···
});
app.post('/projects/image/', upload.single('image'), async (req, res) => {···
});
app.delete('/projects/:project_id', async (req, res) => {···
});
// tags
app.post('/tags', async (req, res) => {···
});
app.get('/tags/:id', async (req, res) => {···
});
app.post('/tags/story', async (req, res) => {···
});
// status
app.post('/status/', async (req, res) => {···
});
app.post('/status/:project_id', async (req, res) => {···
});
app.get('/status/:project_id', async (req, res) => {···
});
// falhas
app.get('/failures', async (req, res) => {···
});
// arquivos
app.get('/files/:id_image', async (req, res, next) => {···
});
app.listen(3333);
```

Fonte: Próprio autor (2022).

Tabela 4 – Detalhes das rotas criadas na API.

Endereço/rota	**Métodos**	**Descrição**
/register	POST/GET	Criar novos usuários/Consultar informações sobre o usuário.
/session	POST	Responsável por validar *login* e iniciar sessão no sistema.
/projects	POST/GET/DELETE	Criar novos projetos/Consultar informações de projetos/Deletar projeto
/projects/image	POST	Enviar nova imagem base para projeto
/tags	POST/GET	Adicionar novas *tags*/Obter informações de *tags* existentes
/tags/story	POST	Especificar histórico de status da *tag* informando intervalo entre datas
/status	POST/GET	Adicionar novo valor de uma *tag*/Obter lista de ultimos estados de determinada *tag*.
/failures	GET	Submeter dados obtidos com Node-RED e obter informações sobre o tratamento dos dados.
/files	GET	Obter acesso ao arquivo da imagem utilizada em dado projeto

Fonte: Próprio autor (2022).

Tabela 5 – Detalhes das tabelas utilizadas no banco de dados.

Tabela	Atributo	Tipo
users	id	inteiro
	name	string
	email	string
	password	string
projects	id	inteiro
	user_id	inteiro
	name	string
	description	string
tags	id	inteiro
	project_id	inteiro
	name	string
status	id	inteiro
	project_id	inteiro
	tag_id	inteiro
	value	string
apparentfailures	id	inteiro
	project_id	inteiro
	datavalues	text
	failed	boolean

Fonte: Próprio autor (2022).

3.3.4 Método usado na análise de dados

A análise consiste em tratar os dados obtidos, a fim de proporcionar informações úteis. No desenvolvimento do projeto optou-se pelo método KNN, responsável por classificar amostras de dados conforme as características de seus K vizinhos mais próximos, como visto em 2.7. Com a escassez de bibliotecas para Node.Js utilizando KNN e as poucas existentes não apresentando características satisfatórias, optou-se por implementar o código para executar esta atividade. O código pode ser observado no Anexo A, desenvolvido dentro da API sob a rota "/failures", com o auxílio de bibliotecas como *ml-distance* [4] para obter as distâncias e *simple-statistics* para encontrar a moda, irá receber como parâmetros de entrada o valor de K, a amostra de dados a ser analisado e o banco dados utilizado para treinamento, retornando assim "*Will fail soon!*" caso a máquina esteja apresentando sinais de falha aparente ou "*No failure.*" caso se encontre em perfeito estado de funcionamento. Vale ressaltar que para otimizar o algoritmo, consumindo menos processamento da máquina, e não deixar a grandeza dos atributos influenciar na sua importância, os dados foram normalizados entre 0 e 1, de forma a manter a mesma escala, como pode ser observado entre as linhas 20 e 58 do Anexo A.

[4] https://github.com/mljs/distance

3.3.4.1 Conjunto de amostras utilizadas

Obter e principalmente publicar um conjunto de dados reais sobre manutenção preventiva geralmente é difícil, dessa forma, foi utilizado o *dataset* "*AI4I 2020 Predictive Maintenance Dataset Data Set*", desenvolvido em Matzka (2020) e disponibilizado em Dua e Graff (2017) que contém um conjunto de dados sintéticos que retratam informações reais encontradas na indústria sobre a "falha de uma máquina".

No *dataset* é apresentado seis atributos, sendo eles:

– **ID do produto:** identificador único presente no produto, contendo variantes das letras L, M e H, que representam sua qualidade;
– **Temperatura do ar:** dados obtidos em Kelvin com desvio padrão entre 2K e 300K;
– **Temperatura do processo:** dados obtidos em Kelvin e normalizados para um desvio padrão de 1K, adicionados à temperatura do ar mais 10K;
– **Velocidade de rotação:** dados em RPM, obtidos a partir de uma potência de 2860W, com a sobreposição de um ruído normalmente distribuído;
– **Torque:** dados em Nm sem valores negativos, normalmente distribuídos em 40Nm e $\sigma = 10Nm$;
– **Desgaste da ferramenta:** estes dados são abordados em minutos, em que as variantes de qualidade do produto L, M e H agregam 5, 3 e 2 minutos, respectivamente, de desgaste na ferramenta utilizada no processo.

Com os dados supracitados, é inferida a possibilidade de cinco falhas independentes, sendo elas: falha por desgaste de ferramenta; falha de dissipação de calor; falha de energia; falha de sobretensão; e falhas aleatórias. No entanto, para os critérios de análise dos dados presentes neste projeto será avaliado apenas se a máquina irá, ou não, apresentar falha. Com exceção do ID do produto, os demais atributos foram utilizados para o diagnóstico de falha neste projeto, em que estão presentes as *tags air_temperature_k*, *process_temperature_k*, *rotational_speed_rpm*, *torque_nm* e *tool_wear_min*, deixados em inglês para representar mais fielmente o banco de dados, e também presentes como variáveis de memória criadas no PLC para fornecer os dados.

O *dataset* contém dez mil amostras, no qual pequena parte apresentaram realmente falhas. Desta forma, quatrocentos e noventa amostras foram utilizadas, onde quatrocentos e oito foram utilizadas para treinamento do KNN e oitenta e duas para teste no PLC, sendo estas distribuídas igualmente entre amostras que apresentam falhas ou não. O código-fonte 1 apresenta um exemplo em formato *JavaScript Object Notation* (JSON) para treinar o KNN implementado.

O *target* com valor 1 (um) significa falha e 0 (zero) significa que está tudo normal com a máquina industrial.

Código-fonte 1 – Exemplo dos dados para treinamento do KNN em formato JSON.

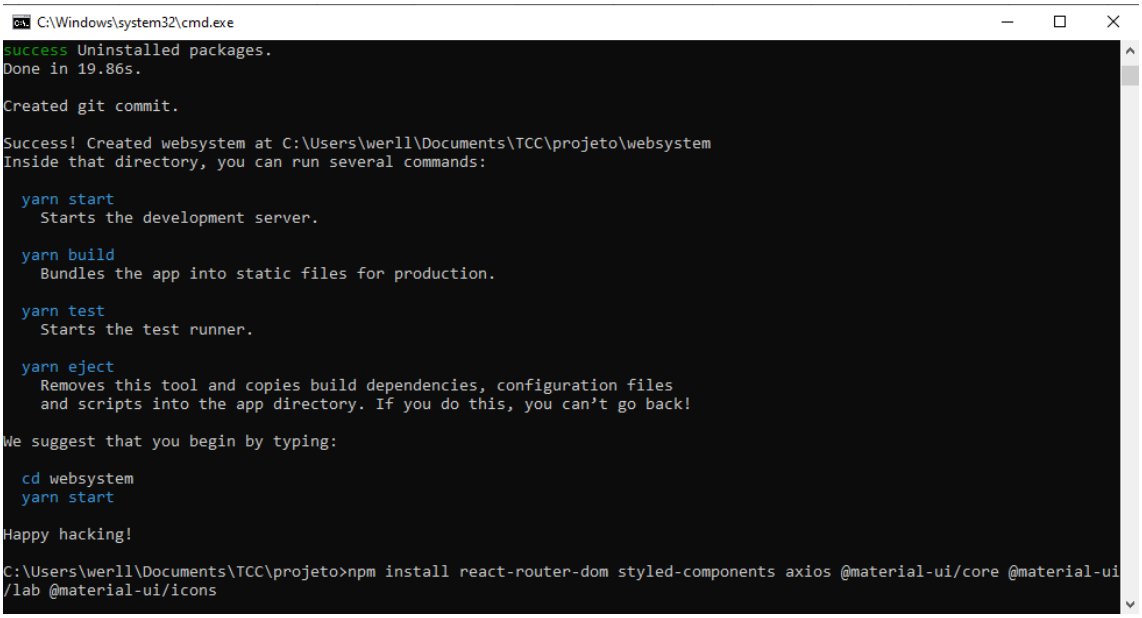
```
const trainingData = [
  {
    id: 1,
    air_temperature_k: 2989,
    tool_wear_min: 143,
    rotational_speed_rpm: 2861,
    process_temperature_k: 3091,
    torque_nm: 46,
    target: 1,
    failure_type: 'Power Failure',
  },
  {
    id: 2,
    air_temperature_k: 2989,
    tool_wear_min: 191,
    rotational_speed_rpm: 1410,
    process_temperature_k: 309,
    torque_nm: 657,
    target: 1,
    failure_type: 'Power Failure',
  },
  {
    id: 3,
    air_temperature_k: 2988,
    tool_wear_min: 208,
    rotational_speed_rpm: 1455,
    process_temperature_k: 3089,
    torque_nm: 413,
    target: 1,
    failure_type: 'Tool Wear Failure',
  },
  ...
  ];
```

Fonte: Próprio autor (2022).

3.3.5 Supervisório

Para o supervisório foi utilizado ReactJS, biblioteca baseada em JavaScript utilizada com Node.Js. O desenvolvimento se inicia com a criação de um novo projeto seguindo os passos presentes na documentação oficial da biblioteca[5], que pode ser verificada no site. Como pode ser constatado na Figura 24, o projeto recebeu o nome de "websystem" e junto a ele foram instaladas as seguintes bibliotecas: *react-router-dom*, para efetuar o roteamento entre as páginas do supervisório; *axios*, para efetuar as requisições *HTTP*; *styled-components*, para auxiliar na estilização do sistema; as bibliotecas de componentes da *@material-ui*, responsáveis por proporcionar componentes prontos que facilitarão no desenvolvimento; *recharts*, para utilizar gráficos; *react-beautiful-dnd*, responsável por habilitar manuseio dinâmico entre componentes de projeto; e *date-fns*, para manipular datas. A tela inicial do ReactJS antes de começar a ser moldada e configurada pode ser observada na Figura 25.

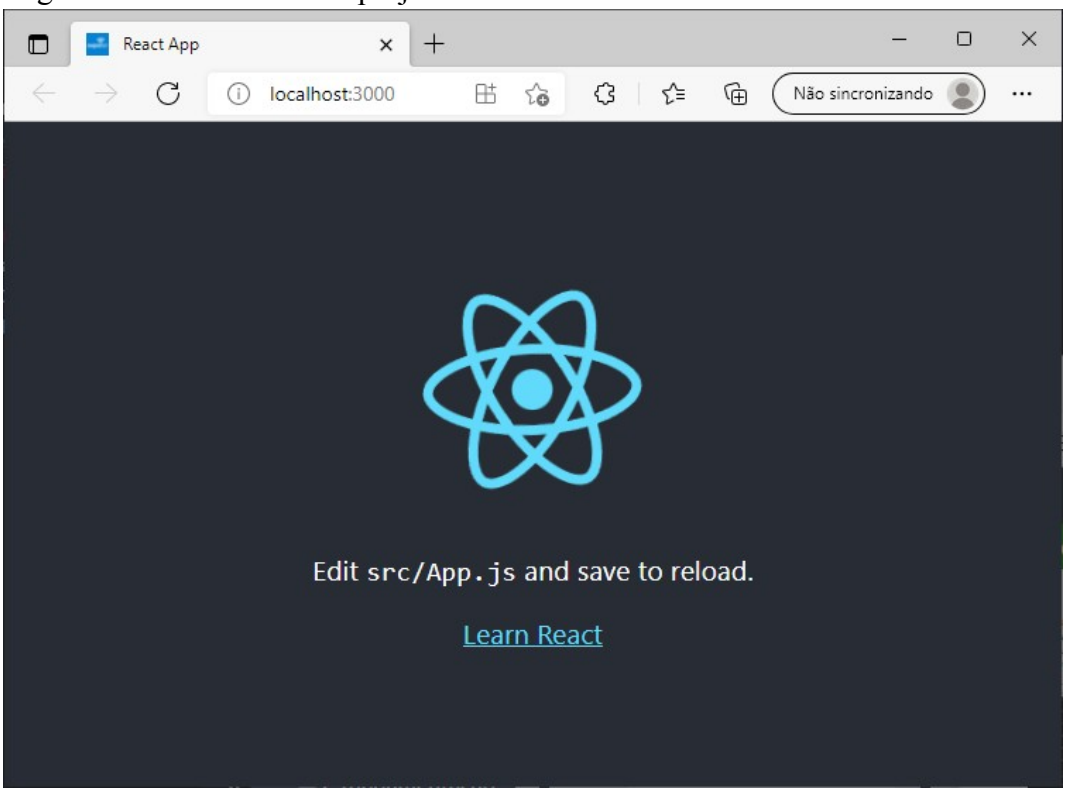
Figura 24 – Criando projeto com ReactJS e instalando bibliotecas essenciais.



Fonte: Próprio autor (2022).

[5] https://pt-br.reactjs.org/

Figura 25 – Tela inicial do projeto em ReactJS.

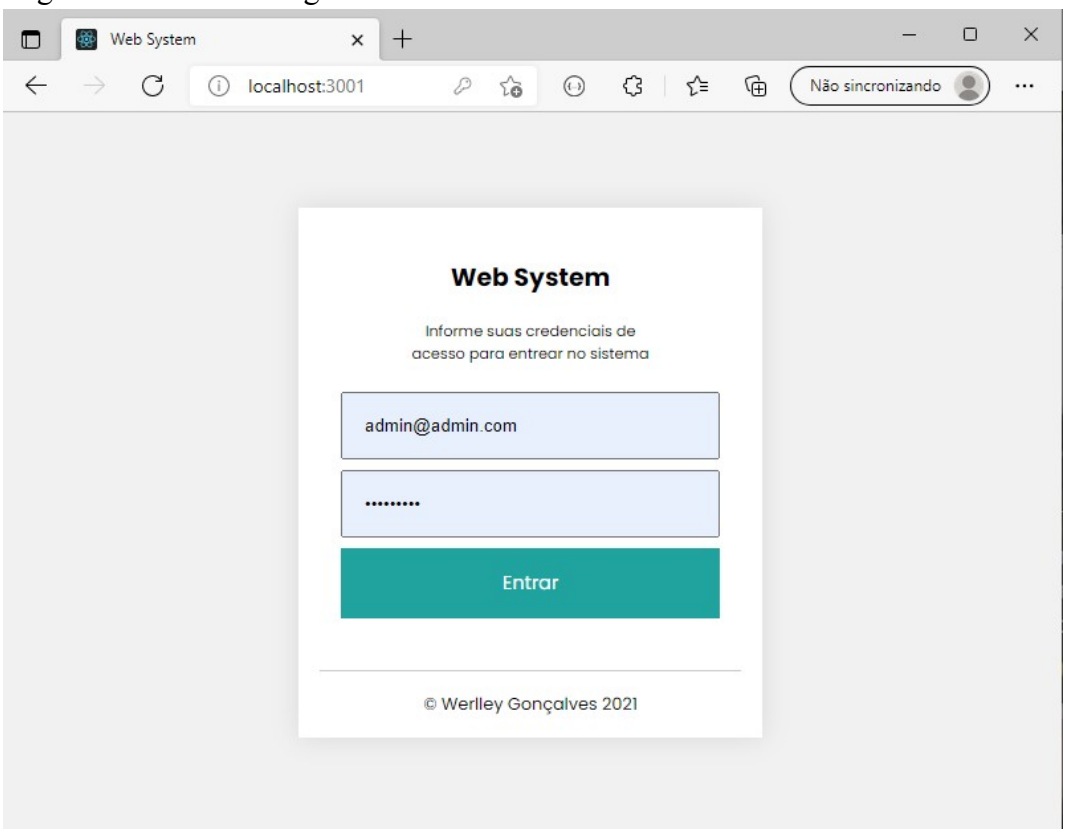


Fonte: Próprio autor (2022).

Para o sistema de supervisão foi desenvolvido 4 páginas distintas, elas: *Login*, para ter acesso às funcionalidades do sistema; Tempo Real, para ter informações dos processos de forma periódica e rápida; *Dashboard*, para acompanhar o histórico de estado das *tags* analisadas; e Configurações, possibilitando alterar as credenciais de acesso. A página de *Login* contará com informações básicas para permitir acesso ao sistema, sendo elas e-mail e senha, como pode ser observado na Figura 26. A quesito de projeto, as primeiras credenciais de acesso serão definidas diretamente no banco de dados, mas poderão ser modificadas posteriormente utilizando a página Configurações presente no ambiente privado do sistema.

A página de Tempo Real irá contar com a funcionalidade de mapear as variáveis, *tags*, requisições da API, criar novos projetos e editá-los, podendo vincular as variáveis que desejar, colocando uma imagem que exemplifique bem o processo no plano de fundo e distribuir as variáveis como bem entender de forma com que fique visualmente mais atrativo e compreensível. A funcionalidade atribuída ao botão “sincronizar” terá o papel de realizar requisições periódicas à API e atualizar o estado das variáveis conforme o enviado pelo PLC com Node-RED. A página supracitada, como observada na Figura 27, contará com *designer* simples e intuitivo, fácil de usar e principalmente interpretar.

Figura 26 – Tela de Login do sistema.

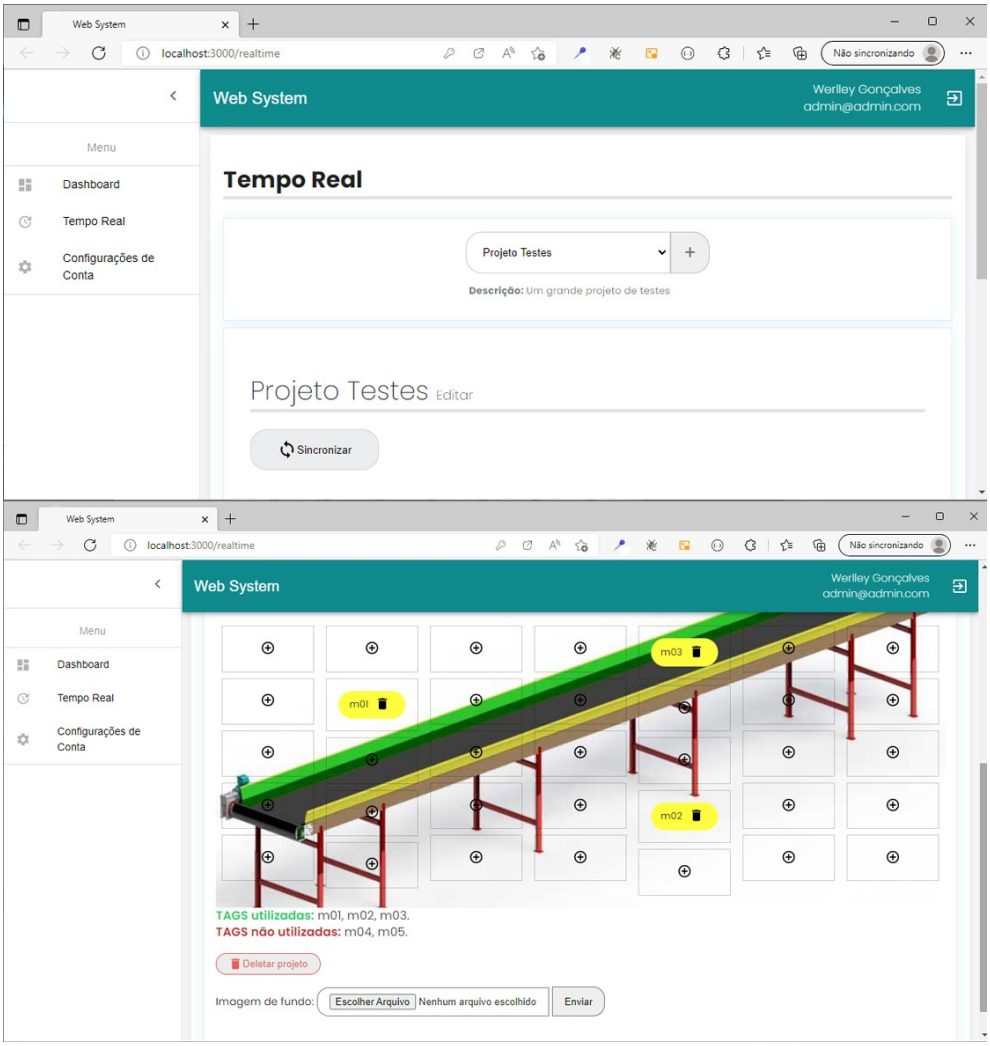


Fonte: Próprio autor (2022).

Com a finalidade de apresentar *feedback* visual e histórico das *tags* utilizadas nos projetos, a página *Dashboard* trará um gráfico interativo mostrando todos os valores da variável selecionada no intervalo de tempo que o usuário desejar, podendo selecionar dados entre cinco, dez, quinze, vinte, trinta e quarenta e cinco minutos após a última amostra ter sido coletada. Além de visualizar, também será possível realizar *download* dos dados em formato .xlsx para demais fins. Nesta página, apresentada na Figura 28, fica visível o quão prático e fácil é o manuseio e interpretação das informações nela presentes.

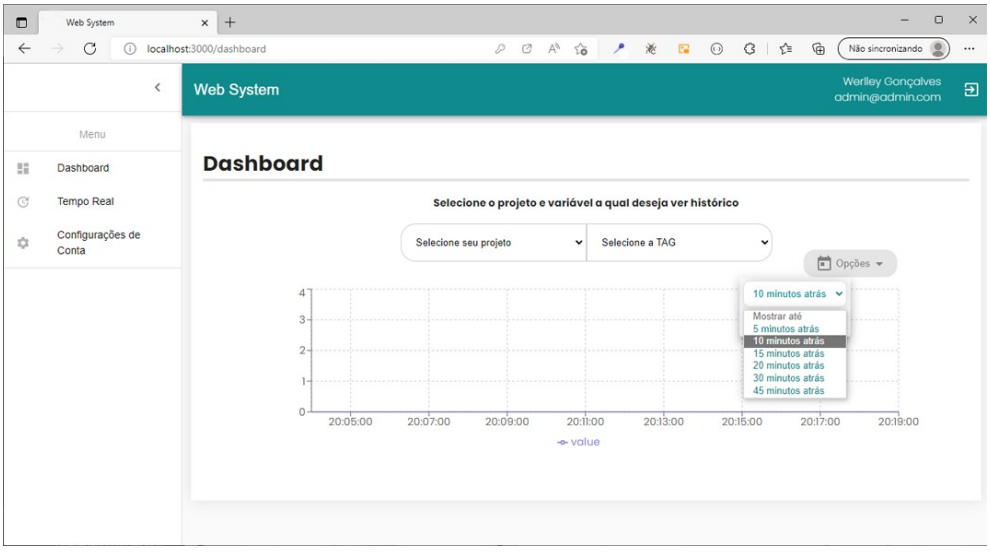
Já a página de Configurações traz as funcionalidades básicas necessárias para gerenciar as informações de acesso, como nome, e-mail e senha, visíveis na Figura 29. Neste trabalho será abordado a utilização de apenas um usuário, mas a ampliação para mais acessos também é possível.

Figura 27 – Tela da página de Tempo Real.



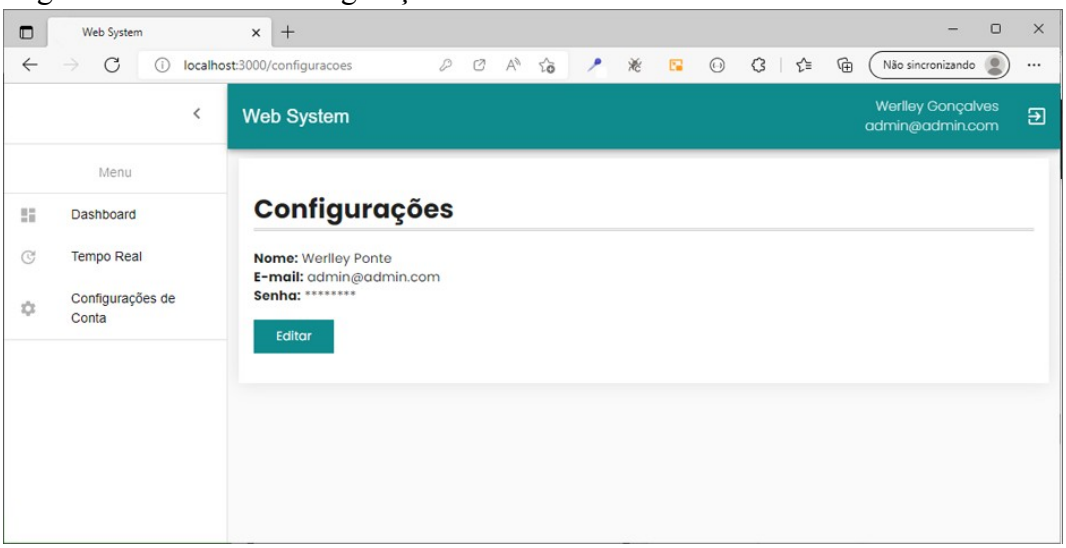
Fonte: Próprio autor (2022).

Figura 28 – Tela da página de *Dashboard*.



Fonte: Próprio autor (2022).

Figura 29 – Tela de Configurações.



Fonte: Próprio autor (2022).

3.3.6 Hospedando no serviço EC2 da AWS

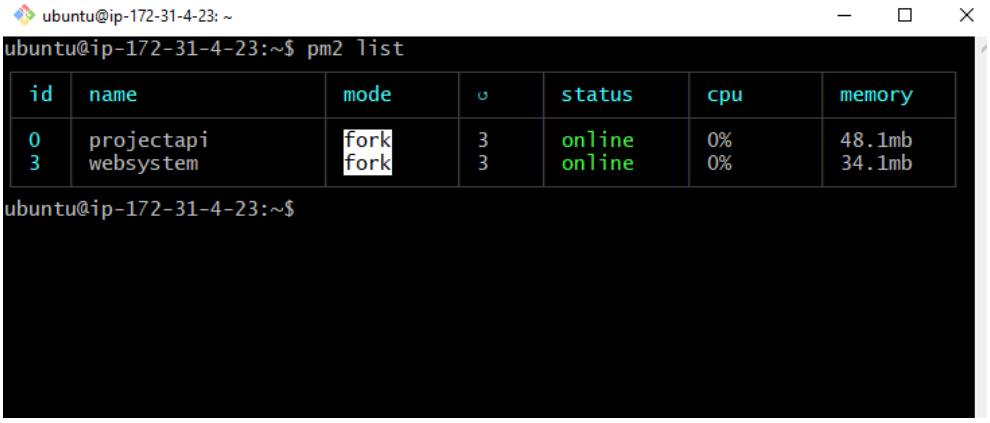
Antes de hospedar na AWS os dois projetos, supervisório e API, foram armazenados na plataforma GitHub, o que torna mais simples o manuseio do código-fonte em qualquer serviço que queira hospedar.

A princípio uma conta na AWS foi criada através do site oficial[6]. Com acesso ao console, foi executada uma instância (máquina) no serviço Amazon EC2 com as configurações: Ubuntu Server 20.04 LTS (HVM) - SSD, 1 vCPU, 1 GiB de memória e 8 GiB de armazenamento. Sendo estas as configurações a nível básico da AWS, que permite a utilização gratuita dos serviços pelo período de um ano.

Através de comunicação SSH à instância, foi instalado os pacotes: npm v8.1.2, yarn v1.22.27, Node.Js v16.13.1, MySQL 8.0.28-0ubuntu0.20.04.3 (Ubuntu) e PM2 v5.1.2. Assim como durante o desenvolvimento, além do Node.Js os gerenciadores de pacotes npm e yarn também são necessários para os projetos funcionarem no servidor *web*, o MySQL é o sistema de gerenciamento de banco de dados utilizado e PM2 é um gerenciador de processos em execução para o Node.Js, ferramenta que permitirá executar múltiplos projetos em paralelo que poderão ser acessados de maneira independente conforme as portas a qual estão vinculadas. O projeto da API e supervisório foram clonados do GitHub para a instância, logo depois o comando *pm2 start* foi executado de forma com que os dois projetos funcionem simultaneamente, como pode ser observado na Figura 30 após o comando *pm2 list*, apresentando os projetos com *status online*.

[6] https://aws.amazon.com/pt/

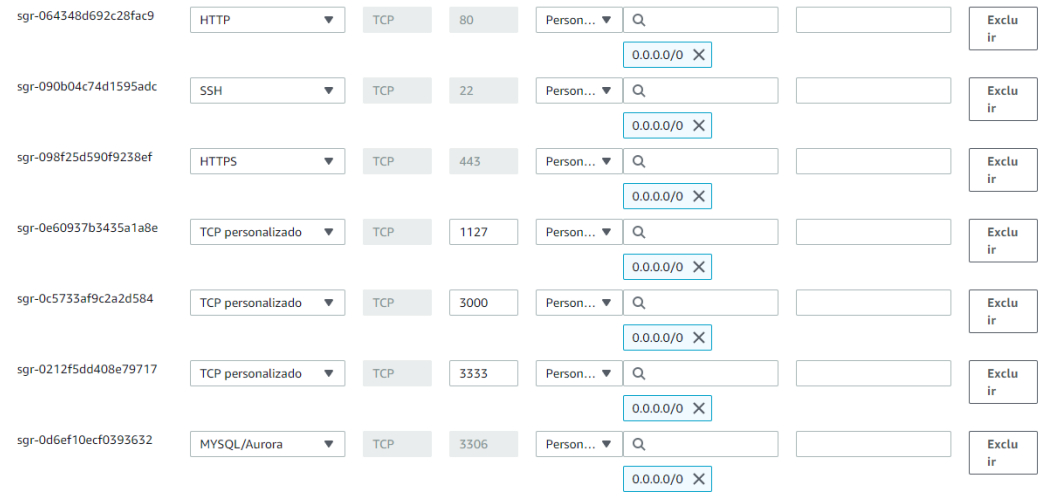
Figura 30 – Comando *pm2 list* apresentando API e supervisório funcionando simultaneamente.



Fonte: Próprio autor (2022).

Para finalizar a hospedagem, o acesso às portas referente aos projetos são liberadas por meio do "grupo de segurança" vinculado à instância acrescentando as regras de entrada da Figura 31, permitindo acesso via *HTTP*, *HTTPS* e MySQL, por meio das portas 80, 443 e 3306, respectivamente, assim como as portas 1127 e 3333 referentes ao acesso direto do supervisório e da API.

Figura 31 – Regras de entrada aplicadas no grupo de segurança da instância utilizada.



Fonte: Próprio autor (2022).

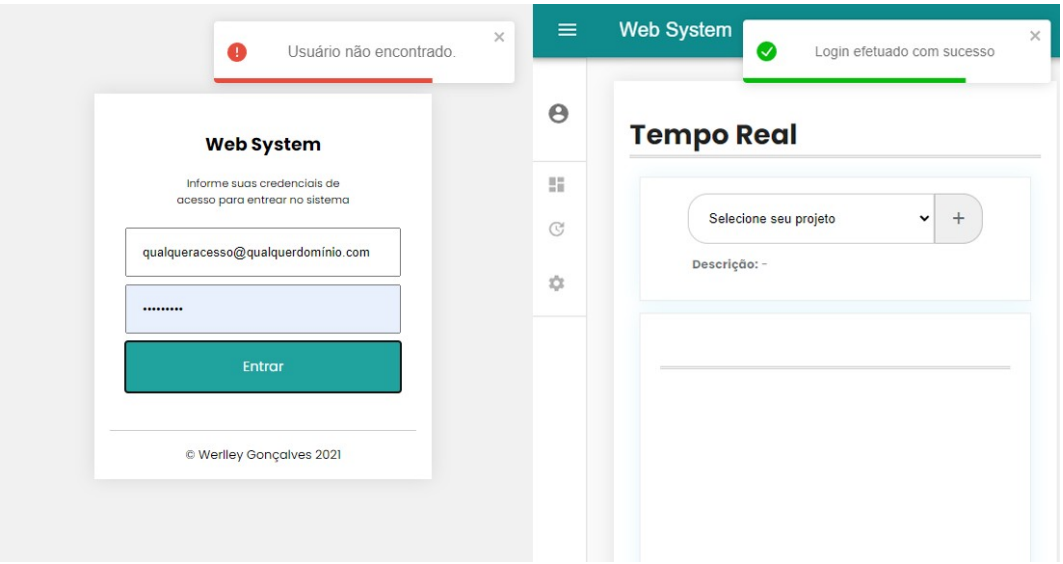
4 RESULTADOS

Neste capítulo será abordado a utilização do ecosistema ciberfísico proposto com a leitura de dados provenientes de um PLC S71200 da Siemens do Laboratório de Controle de Sistemas Dinâmicos da UFC campus Sobral. Com as configurações do item 3.3.2, do PLC foram emitidos dados selecionados do *dataset* apresentado no item 3.3.4.1 para o *gateway* (Ver item 3.3.1), responsável por consultar a API (Ver item 3.3.3) e enviar os dados obtidos para a aplicação hospedada na AWS (Ver item 3.3.6), que podem ser consultados e interpretados por meio do supervisório (Ver item 3.3.5), apresentando-os de forma gráfica e intuitiva, além de obter alerta de falhas (Ver item 3.3.4).

4.1 Utilizando Web System

No sistema de supervisão WEB desenvolvido, chamado de Web System, é necessário inicialmente confirmar as credenciais de acesso, medida de segurança para não deixar informações sensíveis acessíveis a quem não tem permissão. Caso as credenciais estejam incorretas, o usuário será notificado em tela, e caso estejam corretas a notificação também irá surgir e o acesso ao sistema é liberado, como observado na Figura 32.

Figura 32 – Bloqueio e liberação de acesso à plataforma.

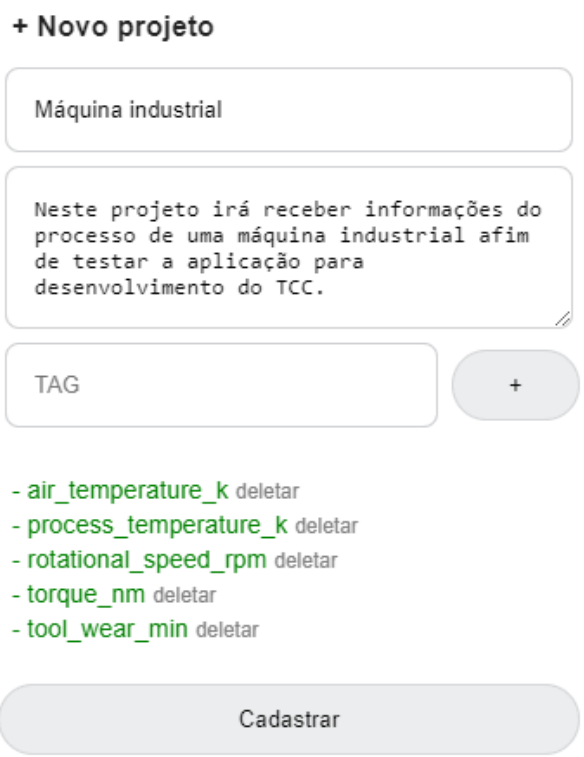


Fonte: Próprio autor (2022).

Após efetuado o *login*, na página "Tempo Real" é clicado no ícone de adição (+), que irá abrir um pequeno formulário para adicionar novo projeto, o qual é inserido as informações da Figura 33 abaixo. Clicando em "Cadastrar", uma janela ensinando de forma didática a

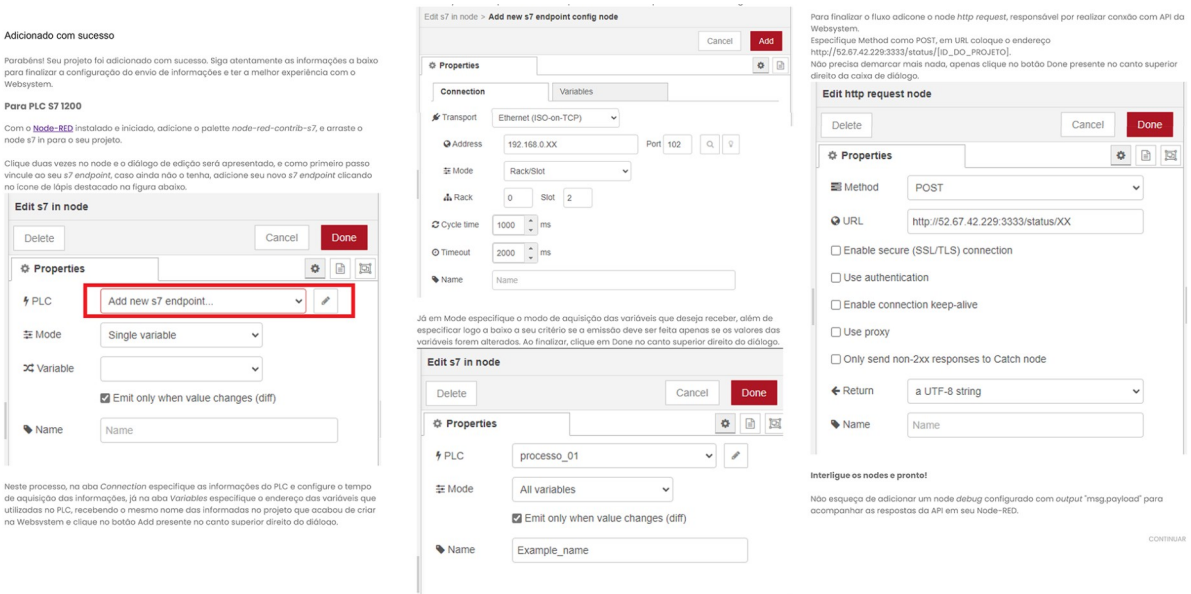
configurar o *gateway* Node-RED para receber as informações do dispositivo e enviá-las ao sistema é mostrado em tela, como na Figura 34.

Figura 33 – Adicionando novo projeto no Web System.



Fonte: Próprio autor (2022).

Figura 34 – Caixa de diálogo apresentada ao adicionar novo projeto contendo instruções de configuração do *gateway* com Node-RED.



Fonte: Próprio autor (2022).

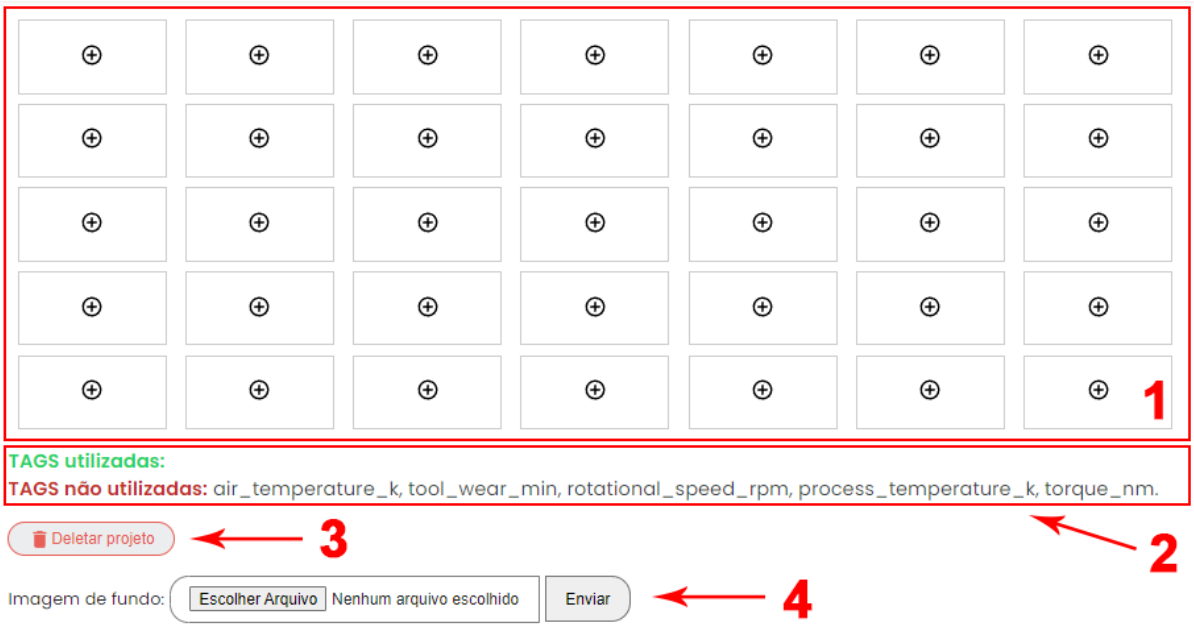
Logo após a criação, o projeto estará disponível para ser selecionado no campo indicado pela flecha de índice 1 na Figura 35. Após a seleção, o nome do projeto foi confirmado na flecha de índice 2, liberando, também, o botão de “Editar” indicado pela flecha 3, permitindo a edição do modo de visualização dos dados recebidos pela API.

Figura 35 – Selecionando o projeto após sua criação.

Fonte: Próprio autor (2022).

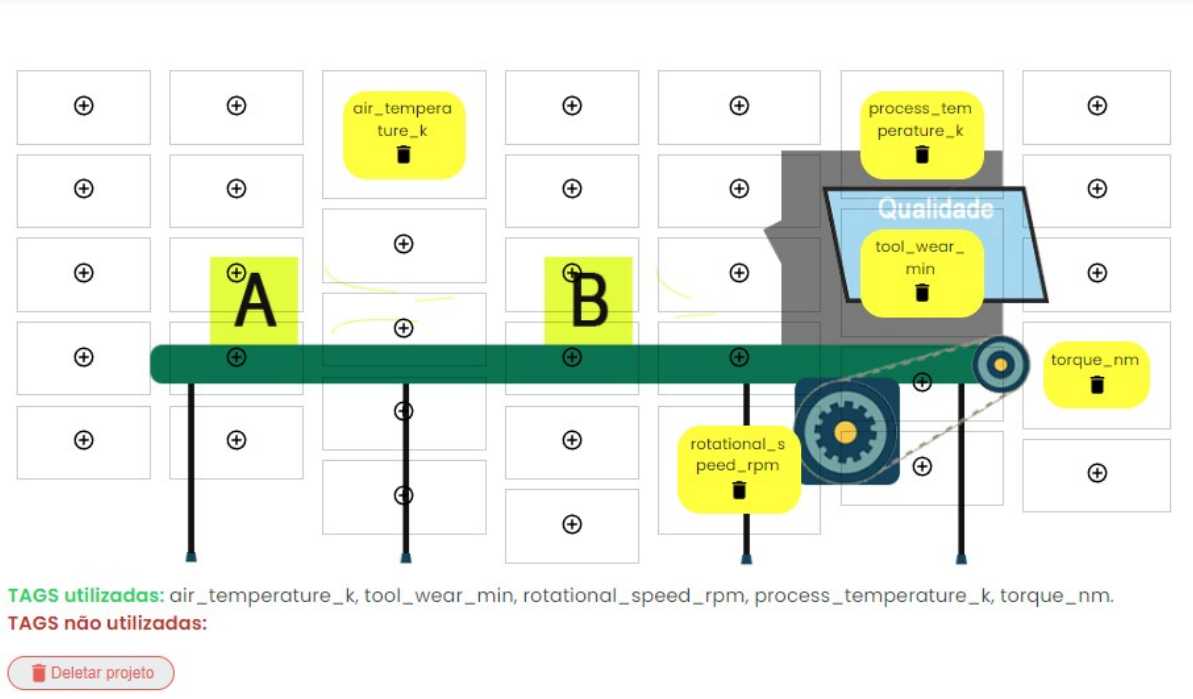
Clicando em editar, novos campos são habilitados em tela como é possível observar na Figura 36. Na seleção de índice 1 é verificado o campo onde se encontra a representação visual dos dados recebidos, nele haverá uma imagem de fundo representando o sistema, assim como a distribuição das variáveis supervisionadas. Na seleção indicada pela flecha de índice 2 está presente todas as *tags* informadas na criação do projeto, dividida em duas categorias, utilizadas e não utilizadas, o que será alterado automaticamente conforme as informações escolhidas para serem apresentadas no campo 1. A flecha de índice 3 indica o botão para deletar o projeto, caso haja necessidade, e no campo de índice 4 é apresentado o elemento responsável por receber a imagem que comporá o plano de fundo do ambiente de supervisão. Após todas as configurações, e utilizando uma imagem representando uma planta industrial de um processo com esteira criada pelo autor, o resultado do ambiente de supervisão pode ser observado na Figura 37.

Figura 36 – Configurando visualização do novo projeto.



Fonte: Próprio autor (2022).

Figura 37 – Resultado após configurar o novo projeto.

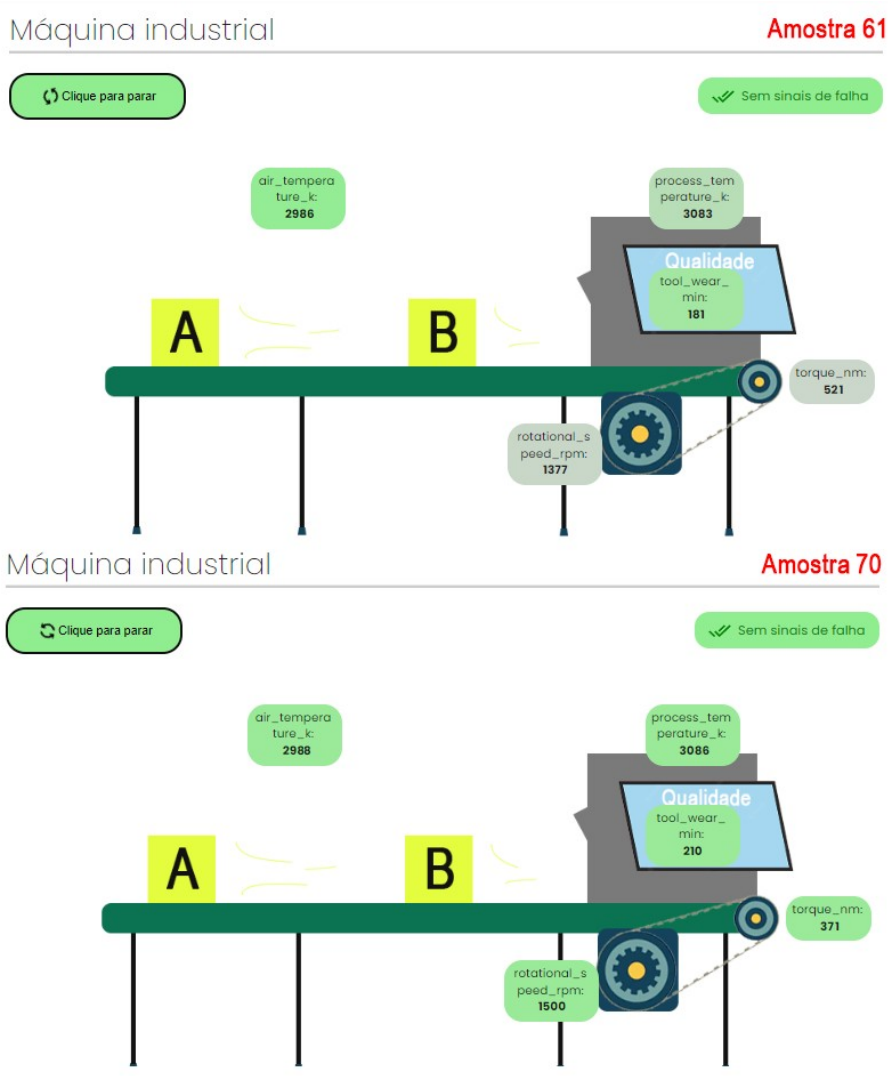


Fonte: Próprio autor (2022).

Nas etapas supracitadas tudo ocorreu de forma fácil, prática e intuitiva. As *tags* selecionáveis e arrastáveis apresentaram flexibilidade interessante, se encaixando nos campos certos da imagem de fundo, proporcionando uma visualização bastante intuitiva e atrativa das informações que serão recebidas do PLC.

Para demonstrar o funcionamento do sistema, a seguir é apresentado alguns resultados referente a aquisição dos dados consultados à API desenvolvida. O resultado de algumas amostras podem ser visualizadas na Figura 38, na qual a informação é apresentada em tela intuitivamente e fácil compreensão, mantendo *designer* minimalista, discreto e apresentando objetivamente ao usuário o *status* do processo em análise.

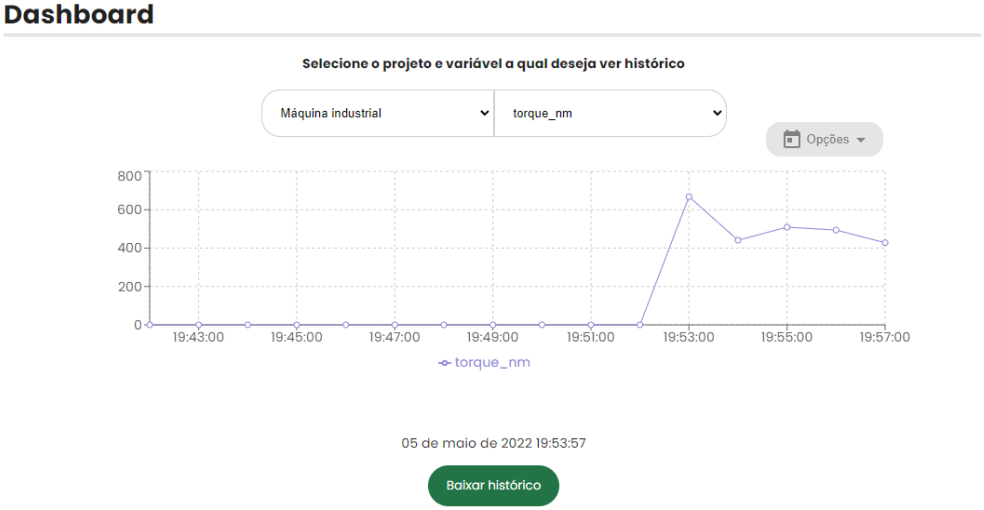
Figura 38 – Exemplos de amostras visualizadas em tempo real com Web System.



Fonte: Próprio autor (2022).

A *dashboard* foi a parte do sistema que forneceu graficamente o histórico de valores das *tags* a partir do projeto selecionado. Na Figura 39 pode-se observar como foi estruturado os dados adquiridos da *tag torque_nm*, apresentando informações em até 15 minutos após a última aquisição de dados realizada, onde a data da última é apresentada no canto inferior da página. As informações completas do histórico da *tag* puderam ser baixadas em formato de planilha (*.xlsx*) para análise externa, caso seja necessário. Na Figura 40 é ilustrado o exemplo de como ficou o histórico de dados da *tag torque_nm*, disponibilizando as 82 amostras e o tempo em que foram obtidas.

Figura 39 – Histórico de valores da *tag* torque_nm apresentados na *Dashboard*.



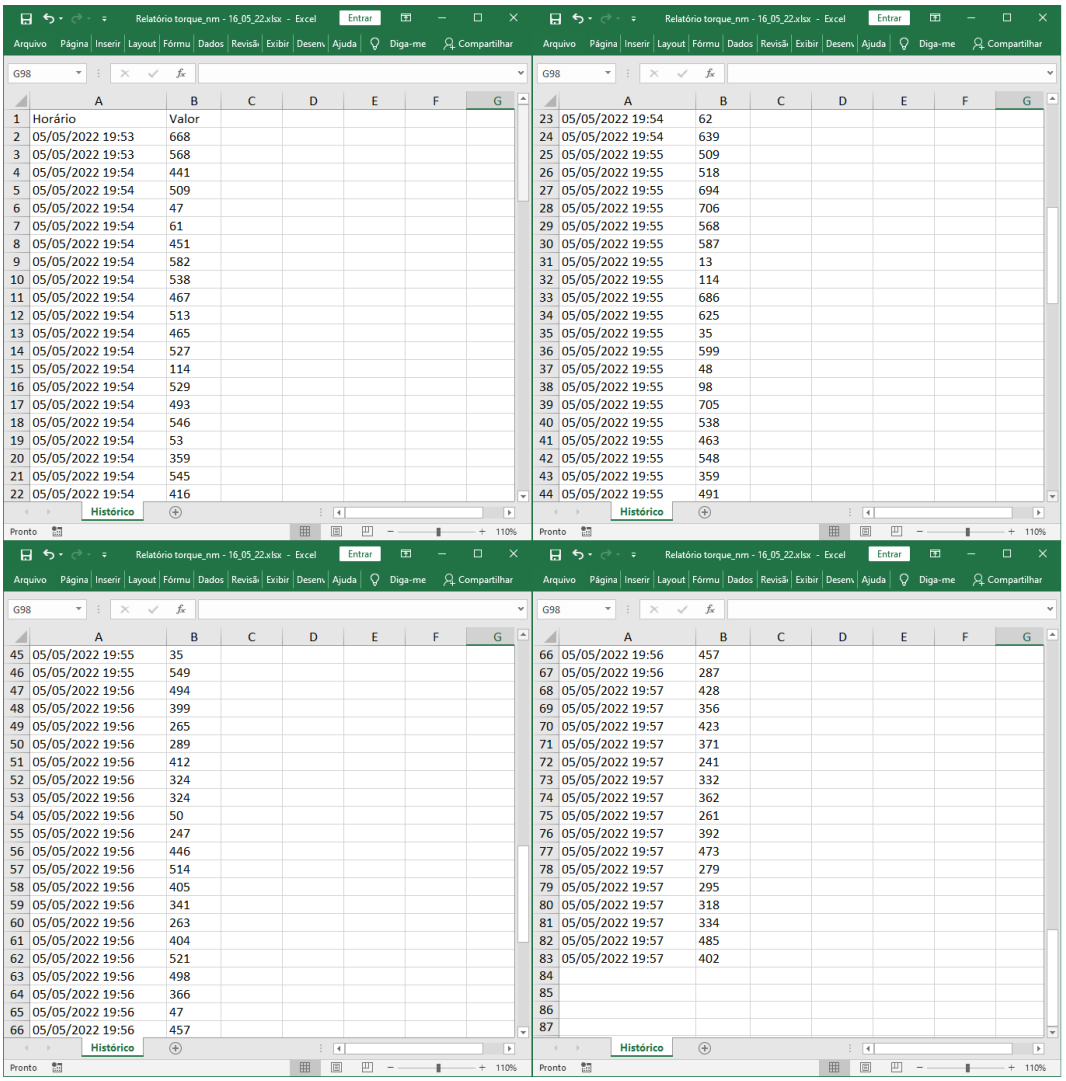
Fonte: Próprio autor (2022).

4.2 Interação entre PLC, Node-RED e API

Para o teste de comunicação do sistema, foi utilizado o PLC com endereço de IP 192.168.0.12, o qual se comunica com o *gateway* de IP 192.168.0.9 de forma física utilizando cabo de rede RJ45, conforme a Figura 41.

A configuração do Node-RED para receber os dados do novo projeto foi de forma simples e fácil, seguindo os passos apresentados na Figura 34 em 4.1, dispensando o entendimento prévio sobre protocolos de comunicação ou linguagens de baixo nível. Nesta etapa, ao receber as informações adquiridas pela biblioteca mencionada em 2.5.1, ainda no Node-RED é utilizado o *node HTTP request* para consultar a rota *status* da API utilizando método *POST*, submetendo o pacote de dados da amostra com 316 Bytes.

Figura 40 – Planilha em .xlsx com histórico de valores da *tag* torque_nm baixada utilizando o sistema.



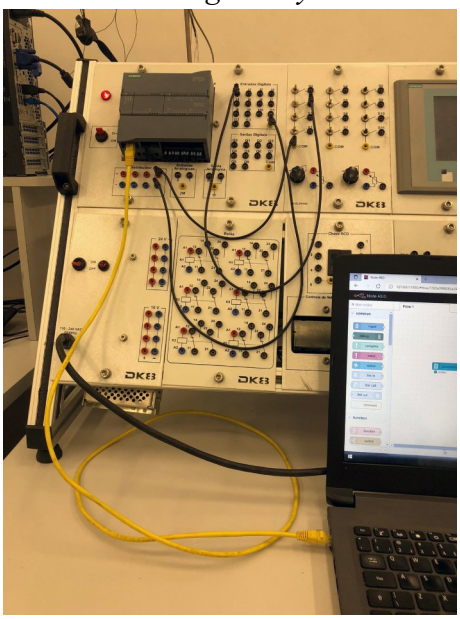
Horário	Valor
05/05/2022 19:53	668
05/05/2022 19:53	568
05/05/2022 19:54	441
05/05/2022 19:54	509
05/05/2022 19:54	47
05/05/2022 19:54	61
05/05/2022 19:54	451
05/05/2022 19:54	582
05/05/2022 19:54	538
05/05/2022 19:54	467
05/05/2022 19:54	513
05/05/2022 19:54	465
05/05/2022 19:54	527
05/05/2022 19:54	114
05/05/2022 19:54	529
05/05/2022 19:54	493
05/05/2022 19:54	546
05/05/2022 19:54	53
05/05/2022 19:54	359
05/05/2022 19:54	545
05/05/2022 19:54	416
05/05/2022 19:54	62
05/05/2022 19:54	639
05/05/2022 19:55	509
05/05/2022 19:55	518
05/05/2022 19:55	694
05/05/2022 19:55	706
05/05/2022 19:55	568
05/05/2022 19:55	587
05/05/2022 19:55	13
05/05/2022 19:55	114
05/05/2022 19:55	686
05/05/2022 19:55	625
05/05/2022 19:55	35
05/05/2022 19:55	599
05/05/2022 19:55	48
05/05/2022 19:55	98
05/05/2022 19:55	705
05/05/2022 19:55	538
05/05/2022 19:55	463
05/05/2022 19:55	548
05/05/2022 19:55	359
05/05/2022 19:55	491
05/05/2022 19:55	35
05/05/2022 19:55	549
05/05/2022 19:56	494
05/05/2022 19:56	399
05/05/2022 19:56	265
05/05/2022 19:56	289
05/05/2022 19:56	412
05/05/2022 19:56	324
05/05/2022 19:56	324
05/05/2022 19:56	50
05/05/2022 19:56	247
05/05/2022 19:56	446
05/05/2022 19:56	514
05/05/2022 19:56	405
05/05/2022 19:56	341
05/05/2022 19:56	263
05/05/2022 19:56	404
05/05/2022 19:56	521
05/05/2022 19:56	498
05/05/2022 19:56	366
05/05/2022 19:56	47
05/05/2022 19:56	457
05/05/2022 19:56	457
05/05/2022 19:56	287
05/05/2022 19:57	428
05/05/2022 19:57	356
05/05/2022 19:57	423
05/05/2022 19:57	371
05/05/2022 19:57	241
05/05/2022 19:57	332
05/05/2022 19:57	362
05/05/2022 19:57	261
05/05/2022 19:57	392
05/05/2022 19:57	473
05/05/2022 19:57	279
05/05/2022 19:57	295
05/05/2022 19:57	318
05/05/2022 19:57	334
05/05/2022 19:57	485
05/05/2022 19:57	402

Fonte: Próprio autor (2022).

O tempo de transmissão das informações se dá respeitando o intervalo de aquisição especificado nas configurações do *node s7 endpoint*, 1000ms, já o tempo de resposta da API foi em média 192ms, valor obtido analisando a resposta das requisições de método POST usando as ferramentas de desenvolvedor do navegador Microsoft Edge, resultando num ciclo médio de 1192ms para que uma amostra seja armazenada no banco de dados em nuvem, recebendo variações dependendo da conexão com a internet no dado momento.

Como apresentado em 3.3.4.1, para simular um ambiente industrial foi utilizado conjunto de dados provenientes do *dataset* retirado do trabalho Matzka (2020). Foi selecionado 82 amostras para teste, dentre elas 41 apresentando falhas e 41 não apresentando, emitidas pelo PLC e enviadas ao *gateway*.

Figura 41 – Comunicação física com cabo RJ45 entre PLC e *gateway*.

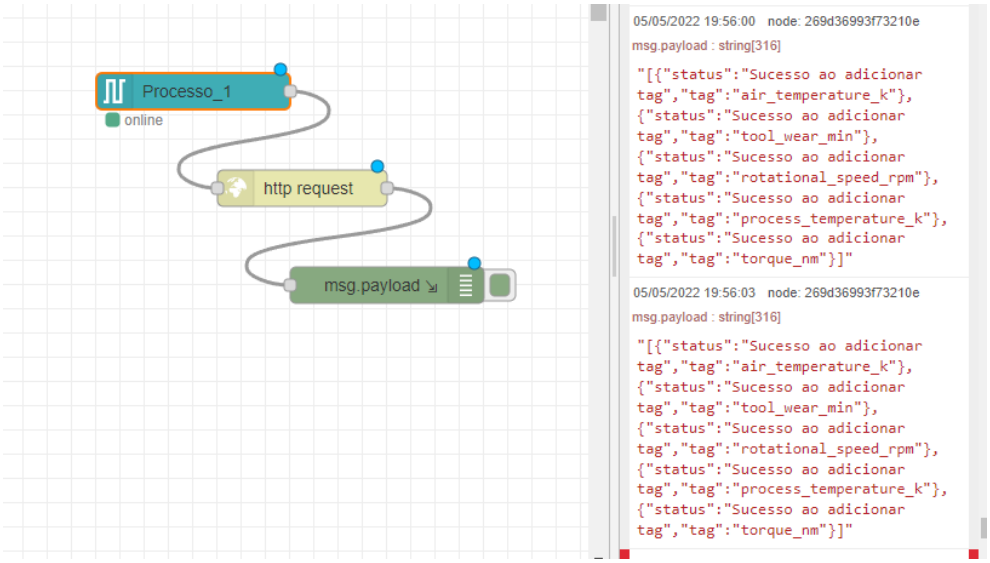


Fonte: Próprio autor (2022).

Todas as amostras emitidas pelo PLC foram recebidas no *gateway*, e na Figura 42 é possível observar o exemplo de duas amostras dos dados recebidos do dispositivo e enviados à API.

Observa-se que no próprio Node-RED é obtido *feedback* sobre o envio dos dados para a API em formato JSON, trazendo o nome da variável submetida e se ela realmente existe dentro do projeto, como é o caso da imagem onde aparece no *status* “Sucesso ao adicionar tag”, caso contrário apresentaria erro ao armazenar seus valores.

Figura 42 – Exemplo de dados recebidos pelo Node-RED e enviados à API.

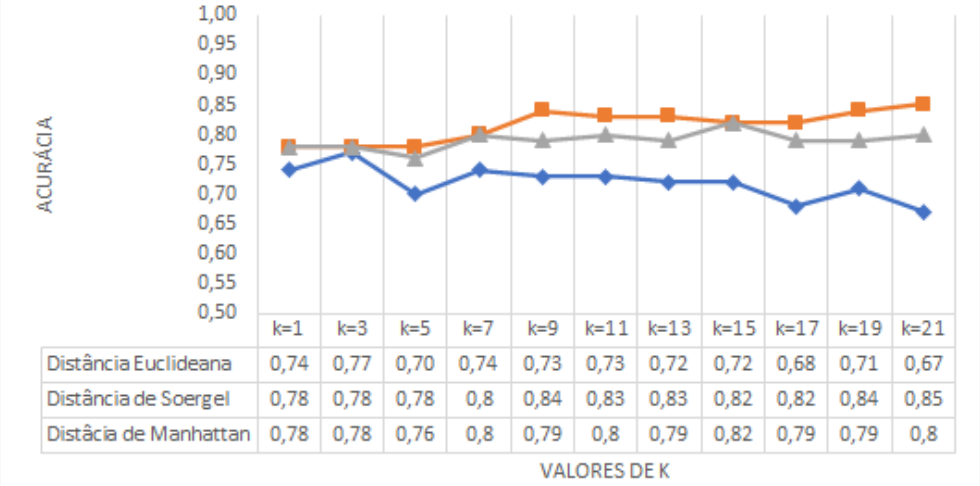


Fonte: Próprio autor (2022).

4.3 Resultado da análise de dados

Para verificar a eficiência do classificador abordado em 3.3.4, é utilizada a Equação (2.4), e o resultado pode ser observado na Figura 43, que apresenta a comparação entre a acurácia e o valor de K para diferentes cálculos de distância, utilizando os dados de teste.

Figura 43 – Comparação dos valores de acurácia obtidos com diferentes distâncias.



	k=1	k=3	k=5	k=7	k=9	k=11	k=13	k=15	k=17	k=19	k=21
Distância Euclideana	0,74	0,77	0,70	0,74	0,73	0,73	0,72	0,72	0,68	0,71	0,67
Distância de Soergel	0,78	0,78	0,78	0,8	0,84	0,83	0,83	0,82	0,82	0,84	0,85
Distâcia de Manhattan	0,78	0,78	0,76	0,8	0,79	0,8	0,79	0,82	0,79	0,79	0,8

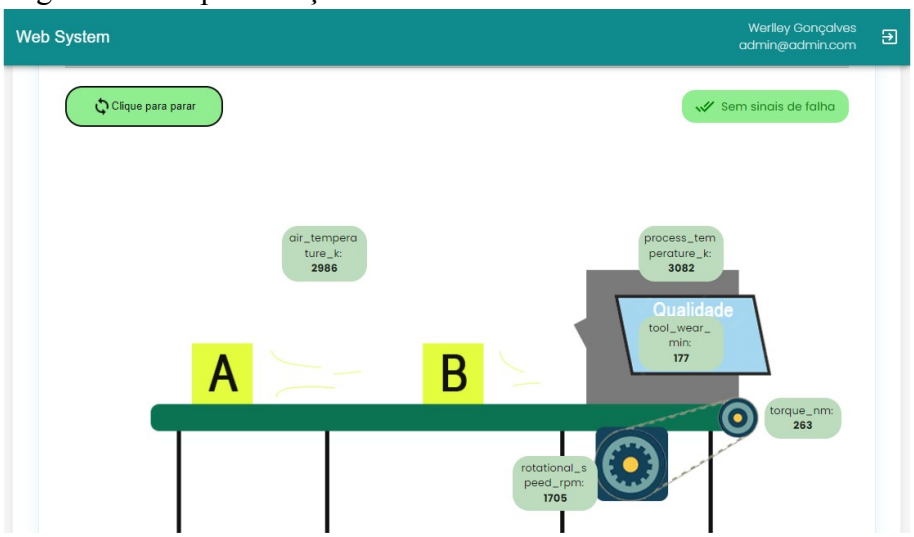
Fonte: Próprio autor (2022).

Ao modificar hiperparâmetros do classificador como o valor de K e o método do cálculo de distância, é possível observar diferentes resultados onde a Distância de Soergel com valor de K=21 apresentaram o melhor desempenho, fornecendo acurácia de 85%. Estas configurações são fixadas no classificador para dar início ao próximo passo.

Nesta etapa o método de classificação analisa cada conjunto de amostras recebido da API no Websystem e, conforme ilustrado nas Figuras 44 e 45, apresenta em tela de maneira visual a situação onde a falha não ocorre e situação de possível falha aparente no processo para que as devidas medidas possam ser tomadas pelo usuário, respectivamente.

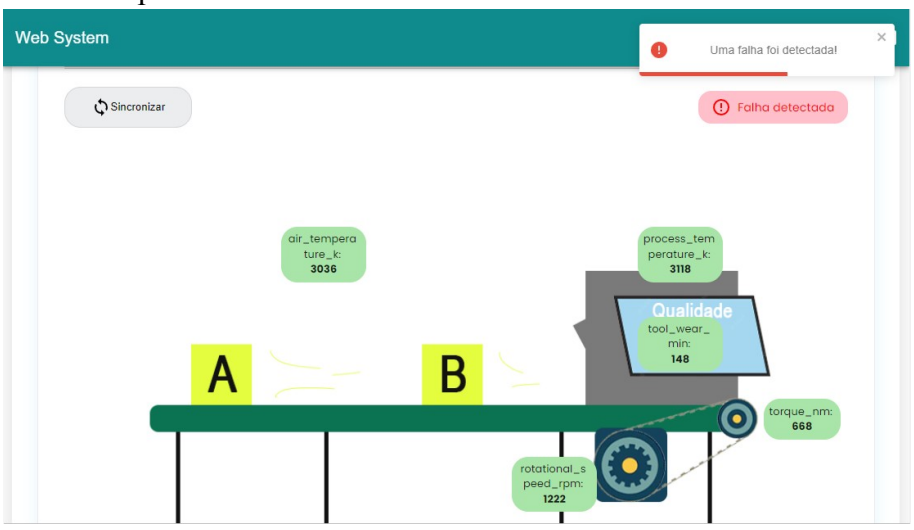
O resultado das 82 amostras coletadas é apresentado na Tabela 6. Seguindo em ordem de três colunas: a primeira especifica a amostra; na segunda, a classificação do KNN; e na terceira, a real classificação presente no *dataset*. Ambas agrupadas em três sessões verticais para agrupar as 82 amostras da melhor forma. Mais detalhes sobre as amostras podem ser conferidos no Anexo B.

Figura 44 – Representação visual do estado normal do sistema.



Fonte: Próprio autor (2022).

Figura 45 – Representação visual do estado com presença de falha detectada pelo sistema.



Fonte: Próprio autor (2022).

Tabela 6 – Comparativo entre o resultado da classificação das amostras com KNN e a real classificação apresentada no *dataset*.

Amostra	Previsto	Real	Amostra	Previsto	Real	Amostra	Previsto	Real
1	Falha	Falha	29	Falha	Falha	57	Sem falha	Sem falha
2	Falha	Falha	30	Falha	Falha	58	Sem falha	Sem falha
3	Falha	Falha	31	Falha	Falha	59	Sem falha	Sem falha
4	Falha	Falha	32	Falha	Falha	60	Sem falha	Sem falha
5	Falha	Falha	33	Falha	Falha	61	Falha	Sem falha
6	Falha	Falha	34	Falha	Falha	62	Falha	Sem falha
7	Falha	Falha	35	Falha	Falha	63	Sem falha	Sem falha
8	Falha	Falha	36	Falha	Falha	64	Sem falha	Sem falha
9	Falha	Falha	37	Falha	Falha	65	Sem falha	Sem falha
10	Falha	Falha	38	Falha	Falha	66	Sem falha	Sem falha
11	Falha	Falha	39	Falha	Falha	67	Sem falha	Sem falha
12	Sem falha	Falha	40	Falha	Falha	68	Sem falha	Sem falha
13	Falha	Falha	41	Falha	Falha	69	Sem falha	Sem falha
14	Sem falha	Falha	42	Sem falha	Sem falha	70	Falha	Sem falha
15	Falha	Falha	43	Sem falha	Sem falha	71	Falha	Sem falha
16	Sem falha	Falha	44	Sem falha	Sem falha	72	Sem falha	Sem falha
17	Falha	Falha	45	Falha	Sem falha	73	Sem falha	Sem falha
18	Falha	Falha	46	Sem falha	Sem falha	74	Sem falha	Sem falha
19	Falha	Falha	47	Sem falha	Sem falha	75	Sem falha	Sem falha
20	Falha	Falha	48	Sem falha	Sem falha	76	Sem falha	Sem falha
21	Sem falha	Falha	49	Sem falha	Sem falha	77	Sem falha	Sem falha
22	Sem falha	Falha	50	Sem falha	Sem falha	78	Sem falha	Sem falha
23	Falha	Falha	51	Sem falha	Sem falha	79	Sem falha	Sem falha
24	Falha	Falha	52	Sem falha	Sem falha	80	Sem falha	Sem falha
25	Falha	Falha	53	Sem falha	Sem falha	81	Falha	Sem falha
26	Falha	Falha	54	Sem falha	Sem falha	82	Sem falha	Sem falha
27	Falha	Falha	55	Sem falha	Sem falha			
28	Falha	Falha	56	Falha	Sem falha			

Fonte: Próprio autor (2022).

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentado alguns conceitos como Indústria 4.0, IoT, sistemas legados, Node.js, Node-RED e KNN, focando em entender como a comunicação com Node-RED e ReactJS pode ser útil para atender a pequenas e médias automações, de maneira a proporcionar um ambiente adequado para entrar no conceito de Indústria 4.0, sendo uma solução viável para sistemas legados, como foi apresentado no estudo de caso proposto.

Apesar do dispositivo utilizado, PLC S7-1200 1214c ac/dc/rly com *firmware* v2.2, não possuir suporte para comunicação OPC, podendo assim ser considerado um sistema legado, a comunicação com o *gateway* Node-RED foi realizada de forma satisfatória usando o node *node-red-contrib-s7*, permitindo a leitura de variáveis do PLC de forma fácil, prática e eficiente, sem a necessidade de nenhum conhecimento prévio sobre protocolos de comunicação ou até mesmo a utilização de *softwares* de terceiros.

Diferente de outros sistemas supervisórios presentes no mercado, que apresentam certa complexidade mediante a tamanha quantidade de ferramentas e funcionalidades por vezes não usadas, o que pode acabar confundindo o usuário, a utilização do Web System desenvolvido se mostrou fluído no uso, atendendo ao seu objetivo principal, em que seu *designer* minimalista tornou os passos para criar um novo projeto bastante fácil, com fluxo simples e direto, além de apresentar o roteiro passo-a-passo de como estar configurando o Node-RED para os dados serem enviados.

Os ajustes realizados no Node-RED para criar o caminho entre os dados do PLC à API foram mínimos. Por se tratar de uma ferramenta *Low Code*, não foi necessário nenhum conhecimento prévio sobre linguagens de programação, bastando apenas arrastar, conectar alguns blocos, *nodes*, e aplicar as configurações sugeridas pelo Web System.

Em relação à análise de dados, apesar do classificador utilizado apresentar acurácia de 85%, os resultados se mostraram bastante satisfatórios. Com seus indicadores visuais, o acompanhamento das informações do processo acontece de forma simples e de fácil compreensão, explorando bem as tonalidades de verde para dados normais e tonalidades de vermelho para indicar falhas, assim como alertas em tela.

De modo geral, o ecossistema proposto para este projeto atendeu todos os objetivos esperados. O Node-RED se mostrou uma ferramenta robusta para comunicação entre dispositivos, mas também simples de manusear. O sistema de supervisão e classificação de falhas desenvolvido em Node.Js e ReactJS, ambas de código aberto, se demonstraram ferramentas simples, mas com potencial para grandes implementações envolvendo baixo custo de investimento, resumindo-se a praticamente capital intelectual. As características minimalistas e de fácil compreensão dos processos e configurações propostos tornam o ecossistema adequado para pequenas, médias e até grandes automações, dispensando grandes investimentos com licenças, compras de módulos adicionais e até treinamentos para se adequar ao uso da plataforma. Além disso, a solução também permite a adequação de sistemas legados à indústria 4.0.

Como sugestão de futuros trabalhos, podem ser destacadas:

- ReactJS apesar de ser uma biblioteca *JavaScript* bem atual, há tecnologias mais novas como Next.Js, o que seria uma ótima opção a ser explorada em projetos futuros;
- Neste projeto foi utilizado requisições *HTTP*, o que atendeu perfeitamente a problemática estudada, mas para melhorar a eficiência desta comunicação uma excelente proposta seria agregar a tecnologia *Web Socket*, responsável por criar uma comunicação bidirecional em um único soquete TCP;
- Aumentar o valor de acurácia da aplicação e utilizar outras métricas de avaliação dos resultados, como: sensibilidade, precisão e matriz de confusão;
- Outra proposta para trabalhos futuros seria a implementação de novos métodos para classificação de falhas. Além do KNN há vários outros métodos que podem ser abordados, como a árvore de decisão e as redes neurais, por exemplo.

REFERÊNCIAS

ANDRADE, A. P. **Sistemas supervisórios: saiba tudo sobre o seu funcionamento e os benefícios gerados para indústria**. 2018. Disponível em: https://www.logiquesistemas.com.br/blog/sistemas-supervisorios/. Acesso em: 18 out. 2021.

CALABREZ, G. T. M. **Implementação de Uma Arquitetura IoT Com a Ferramenta Node-red**. Bacharel em Ciência da Computação – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, 2019.

CASTRO, L. N. de; FERRARI, D. G. **Introdução a Mineração de Dados: Conceitos básicos, algoritmos e aplicações**. [*S. l.*]: Editora Saraiva, 2016. v. 1.

CITTOLIN, G. F. **node-red-contrib-s7**. 2022. Disponível em: https://flows.nodered.org/node/node-red-contrib-s7. Acesso em: 23 mai. 2022.

CONTRIBUTORS, M. **HTTP**. 2022. Disponível em: https://developer.mozilla.org/en-US/docs/Web/HTTP. Acesso em: 24 mai. 2022.

CRUZ, F. B. da; MALUF, M. N.; CICHACZEWSKI, E. Iot e computação na nuvem: O aproveitamento de sistemas legados para industria 4.0. Caderno Progressus, 2021.

DUA, D.; GRAFF, C. **UCI Machine Learning Repository** . 2017. Disponível em: http://archive.ics.uci.edu/ml.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. de.**Inteligência Artificial: Uma abordagem de aprendizado de máquina**. [*S. l.*]: LTC - Livros Técnicos e Científicos Editora Ltda, 2011. v. 1.

FEATHERS, M. C. ***Working Effectively with Legacy Code***. [*S. l.*]: Pearson, 2004. v. 1.

FILHO, J. O. F. M. **Diagnóstico de Falhas em Máquinas Elétricas Rotativas Utilizando Técnicas de Reconhecimento de Padrões** . Bacharel em Engenharia Elétrica – Curso de Engenharia Elétrica, Universidade Federal do Ceará, Sobral, Ceará, 2021.

FOUNDATION, O.***Node-RED: Low-code programming for event-driven applications***. 2021. Disponível em: https://nodered.org/. Acesso em: 08 nov. 2021.

HURTADO, A. L. U.**Aceleración de algoritmos de clasificación basados en disimilitudes, utilizando arquitecturas computacionales con múltiples núcleos**. Doctor en Ingeniería - Industria y Organizaciones – Facultad de Ingeniería y Arquitectura, Universidad Nacional de Colombia, Manizales, Colombia, 2022.

INDÚSTRIA, P. da.**Indústria 4.0: Entenda seus conceitos e fundamentos**. 2020. Disponível em: http://www.portaldaindustria.com.br/industria-de-a-z/industria-4-0/. Acesso em: 24 out. 2021.

INSTITUTE, T. W. ***What is the Internet Of Things (IOT)?*** 2020. Disponível em: https://www.twi-global.com/technical-knowledge/faqs/what-is-the-internet-of-things-iot. Acesso em: 25 out. 2021.

JÚNIOR, A. V. de O. **Desenvolvimento de sistema SCADA para aplicações industriais e IoT**. Bacharel em Engenharia de Computação – Escola de Engenharia Elétrica, Mecânica e de Computação, Universidade Federal de Goiás, Goiânia, Goiás, 2019.

LENON. **Node.js – O que é, como funciona e quais as vantagens** . 2018. Disponível em: https://www.opus-software.com.br/node-js/. Acesso em: 08 jan. 2022.

MATZKA, S. Explainable artificial intelligence for predictive maintenance applications. In: **2020 Third International Conference on Artificial Intelligence for Industries (AI4I)** [*S. l.*: *s. n.*], 2020. p. 69–74.

MELO, D. **O que é Node.js? [Guia para iniciantes]** . 2021. Disponível em: https: //tecnoblog.net/responde/o-que-e-node-js-guia-para-iniciantes/. Acesso em: 08 jan. 2022.

NITULESCU, I.-V.; KORODI, A. *Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions*. MDPI IOT Journal, 2020.

ORACLE. **O Que é Internet of Things (IoT)?** 2014. Disponível em: https://www.oracle.com/ br/internet-of-things/what-is-iot/. Acesso em: 25 out. 2021.

PEREIRA, A. IndÚstria 4.0: Conceitos e perspectivas para o brasil. Revista da Universidade Vale do Rio Verde, 2018.

PINTO, M. C. **Sistema De Manutenção Preditiva De Falhas Em Válvulas Em Um Processo Industrial Utilizando Inteligência Artificial**. Engenheiro em Controle e Automação – Curso de Engenharia de Controle e Automação, Universidade Federal de Uberlândia, Uberlândia, Minas Gerais, 2021.

RODRIGUES, G. V. S. **Sistemas supervisórios: saiba tudo sobre o seu funcionamento e os benefícios gerados para indústria** . 2021. Disponível em: https://certi.org.br/blog/ industria-4-0-no-brasil/. Acesso em: 18 out. 2021.

SCHROEDER, G. N. **Metodologia de modelagem e arquitetura de referência do Digital Twin em sistemas ciber físicos industriais usando AutomationML**. Tese (Doutorado em Engenharia Elétrica) – Escola de Engenharia, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2018.

SIEMENS. **Documentação de aprendizado/treinamento Siemens Automation Cooperates with Education (SCE) | a partir da versão V14 SP1**. [*S. l.*]: Digital Industries, FA, 2019. v. 1.

SIEMENS. Data sheet 6es7214-1bg40-0xb0. © Copyright Siemens, 2021.

SIEMENS. **Um guia prático sobre a Indústria 4.0** . 2021. Disponível em: https: //new.siemens.com/br/pt/empresa/stories/industria/industria-4-0.html. Acesso em: 24 out. 2021.

SILVA, A. S. da. **Desenvolvimento de Sistema Para Controle e Monitoramento Remoto de um Forno Industrial de uma Empresa do Polo de Duas Rodas do Distrito Industrial**. Engenheiro em Eletrônica – Curso de Engenharia Eletrônica da Escola Superior de Tecnologia, Universidade do Estado do Amazonas, Manaus, Amazonas, 2021.

SILVA, W. S. da. **SISTEMA SCADA PARA SUPERVISÃO DE TEMPERATURA E UMIDADE**. Tese (Bacharel em Engenharia de Computação) – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Pato Branco, 2016.

TEBET, I. **Sistemas Legados: como e por que introduzir técnicas modernas** . 2021. Disponível em: https://www.objective.com.br/insights/sistemas-legados/. Acesso em: 02 nov. 2021.

VICENTE, L.; MENDES, M. J. Projeto de sistema scada para o novo paradigma da indústria 4.0. 12º Congresso Nacional de Mecânica Experimental (CNME 2020), 2021.

ANEXO A – CÓDIGO PARA DIAGNÓSTICO DE FALHAS UTILIZANDO KNN

```
// biblioteca para obter Distancia Euclidiana
const distance = require (' ml - distance ') . distance ;
// biblioteca para obter a Moda
const mode = require (' simple - statistics ') . mode ;

function Failure ( valores , kValue , trainingData ) {
  let values = valores ;
  const k = kValue ;
  const data = trainingData ;
  const classe = data . map (( info ) = >info . target ) ;
  let obj = data . map (( info ) = >{
    let aux = [];
    aux . push ( info . air_temperature_k ) ;
    aux . push ( info . process_temperature_k ) ;
    aux . push ( info . rotational_speed_rpm ) ;
    aux . push ( info . torque_nm ) ;
    aux . push ( info . tool_wear_min ) ;
    return aux ;
  }) ;
  // valores maximos de cada coluna
  const max0 = Math . max . apply (
    null ,
    obj . map (( a ,i ) = >a [0])
  ) ;
  const max1 = Math . max . apply (
    null ,
    obj . map (( a ,i ) = >a [1])
  ) ;
  const max2 = Math . max . apply (
    null ,
    obj . map (( a ,i ) = >a [2])
  ) ;
  const max3 = Math . max . apply (
    null ,
    obj . map (( a ,i ) = >a [3])
  ) ;
  const max4 = Math . max . apply (
    null ,
    obj . map (( a ,i ) = >a [4])
  ) ;
  // normalizando entrada
  values = [
    values [0] / max0 ,
    values [1] / max1 ,
```

```
45        values [2]   /   max2 ,
46        values [3]   /   max3 ,
47        values [4]   /   max4 ,
48     ];
49     //   normalizando   vetor
50     obj   = obj . map (( a ,i )   = >{
51        a [0]  = a [0]  /   max0 ;
52        a [1]  = a [1]  /   max1 ;
53        a [2]  = a [2]  /   max2 ;
54        a [3]  = a [3]  /   max3 ;
55        a [4]  = a [4]  /   max4 ;
56
57        return   a ;
58     }) ;
59     //   obtendo   distancias
60     // Euclidiana
61     //   const   dist   = obj . map (( a ,i )   = >distance . euclidean (a ,     values ) ) ;
62     // Manhattan
63     //   const   dist   = obj . map (( a ,i )   = >distance . manhattan (a ,   values ) ) ;
64     // Soergel
65     const   dist   = obj . map (( a ,i )   = >distance . soergel (a ,     values )) ;
66
67     //   Ordenando  as  distancias     do  da  menor  para   maior
68     var   distwithlabel      = dist . map (( a , i )   = >[a ,   classe [ i ]]) ;
69     distwithlabel      = distwithlabel . sort (( a ,      b )  = >{
70        if   ( a [0]  < b [0])   return   -1;
71        if   ( a [0]  > b [0])   return   1;
72        return   0;
73     }) ;
74
75     //   retornando   apenas  a  c l a s s i f i c a   o  dos  K  primeiro   elementos   que
76     //   mais  se  repete
77     var   arrayClasse    = distwithlabel . map (( a )    = >a [1]) ;
78     return   mode ( arrayClasse . slice (0 ,   k ) ) ;
79  }
80  module . exports   = Failure ;
```

ANEXO B – TABELA DE AMOSTRAS UTILIZADAS NO PROJETO

amostra	air_temperature_k	tool_wear_min	rotational_speed _rpm	process _temperature_k	torque_nm	target
1	3036	148	1222	3118	668	1
2	3033	187	1337	3116	568	1
3	3034	192	1353	3117	441	1
4	3034	200	1366	3118	509	1
5	3034	210	1341	3118	47	1
6	3034	215	1306	3118	61	1
7	3033	219	1301	3116	451	1
8	3032	2	1336	3115	582	1
9	3032	8	1351	3114	538	1
10	3032	29	1363	3112	467	1
11	3034	51	1295	3117	513	1
12	3035	62	1361	3116	465	1
13	3037	66	1332	3119	527	1
14	3037	71	2663	312	114	1
15	3036	86	1309	312	529	1
16	3035	93	1344	3119	493	1
17	3036	112	1371	3122	546	1
18	3034	208	1401	3118	53	1
19	3034	215	1521	312	359	1
20	3035	30	1366	312	545	1
21	3034	34	1377	3119	416	1
22	3033	42	1315	3118	62	1
23	3034	59	1298	3119	639	1
24	3034	69	1316	3118	509	1
25	3037	90	1363	3121	518	1
26	3034	52	1337	3122	694	1
27	3035	94	1284	3124	706	1
28	3037	194	1359	3127	568	1
29	3038	213	1256	3131	587	1
30	3038	5	2497	3131	13	1
31	3036	26	2659	3128	114	1
32	304	161	1271	3132	686	1
33	304	200	1363	3129	625	1
34	3044	205	1509	3137	35	1
35	3038	205	1365	313	599	1
36	3039	215	1422	3132	48	1
37	3034	65	2706	3126	98	1
38	3028	234	1262	3123	705	1
39	3028	246	1411	3124	538	1
40	3027	251	1477	3124	463	1
41	3026	253	1454	3123	548	1
42	2981	132	1525	3078	359	0
43	2981	135	1379	3078	491	0
44	2982	137	1575	3078	35	0
45	2982	139	1399	3079	549	0

amostra	air_temperature_k	tool_wear_min	rotational_speed _rpm	process _temperature_k	torque_nm	target
46	2982	141	1429	3079	494	0
47	2983	143	1435	3079	399	0
48	2983	145	1766	3079	265	0
49	2983	147	1703	3079	289	0
50	2983	149	1507	3079	412	0
51	2983	151	1609	3079	324	0
52	2984	154	1605	308	324	0
53	2984	156	1491	3081	50	0
54	2984	158	1891	3081	247	0
55	2985	163	1563	3082	446	0
56	2986	168	1390	3082	514	0
57	2985	170	1444	3082	405	0
58	2986	174	1589	3083	341	0
59	2986	177	1705	3082	263	0
60	2986	179	1422	3083	404	0
61	2986	181	1377	3083	521	0
62	2985	186	1334	3083	498	0
63	2986	188	1559	3084	366	0
64	2987	190	1353	3085	47	0
65	2987	192	1507	3084	457	0
66	2987	195	1694	3084	287	0
67	2987	197	1432	3084	428	0
68	2987	202	1572	3085	356	0
69	2988	207	1507	3086	423	0
70	2988	210	1500	3086	371	0
71	2989	213	1771	3086	241	0
72	2988	0	1577	3085	332	0
73	2988	3	1527	3085	362	0
74	2989	5	1827	3084	261	0
75	2988	8	1484	3084	392	0
76	2988	10	1401	3084	473	0
77	2988	12	1634	3083	279	0
78	2988	14	1604	3084	295	0
79	2989	17	1632	3084	318	0
80	299	22	1645	3086	334	0
81	299	25	1408	3087	485	0
82	299	30	1500	3087	402	0