**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**

**LEANDRO BOMER DOS ANJOS**

# INTEGRAÇÃO DE SISTEMAS DE AUTOMAÇÃO INDUSTRIAL ATRAVÉS DO PROTOCOLO DE COMUNICAÇÃO MODBUS, NO CONTEXTO DA INDÚSTRIA 4.0

**CURITIBA**

**2022**

**LEANDRO BOMER DOS ANJOS**

# INTEGRAÇÃO DE SISTEMAS DE AUTOMAÇÃO INDUSTRIAL ATRAVÉS DO PROTOCOLO DE COMUNICAÇÃO MODBUS, NO CONTEXTO DA INDÚSTRIA 4.0

## Integration of industrial automation systems through the Modbus communication protocol, in the context of Industry 4.0

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Sistemas de Informação do Curso de Bacharelado em Sistemas da Informação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. João Alberto Fabro

**CURITIBA**

**2022**

**LEANDRO BOMER DOS ANJOS**

# INTEGRAÇÃO DE SISTEMAS DE AUTOMAÇÃO INDUSTRIAL ATRAVÉS DO PROTOCOLO DE COMUNICAÇÃO MODBUS, NO CONTEXTO DA INDÚSTRIA 4.0

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Sistemas de Informação do Curso de Bacharelado em Sistemas da Informação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 14/dezembro/2021

---

João Alberto Fabro
Doutor
Universidade Tecnológica Federal do Paraná

---

Marcio Aparecido Batista
Doutor
Universidade Tecnológica Federal do Paraná

---

Fabiano Scriptore de Carvalho
Doutor
Universidade Tecnológica Federal do Paraná

**CURITIBA**
**2022**

## RESUMO

Com a crescente competitividade no setor industrial, no ano de 2011, a Alemanha apresenta um novo conceito de indústria, com objetivos de proporcionar aumento de produtividade e qualidade, além de redução de custos de produção. Tal conceito ficou conhecido como Indústria 4.0 e rapidamente se popularizou pelo mundo. Este conceito está fortemente ligado a integração de tecnologias, em especial aquelas relacionadas às tecnologias de informação. Um dos principais pontos de atenção ao implementar este conceito em sistemas que já operam está relacionado à dificuldade na integração de diferentes dispositivos. Este problema se acentua em casos de equipamentos legados, que em muitos casos possuem algumas décadas de uso. Neste contexto, este trabalho se propõe a verificar a aplicabilidade do uso do protocolo Modbus, um protocolo de comunicação muito popular na indústria desde meados de 1980, para auxiliar a integração entre sistemas legados e arquiteturas compatíveis ao conceito da Indústria 4.0. O objetivo é desenvolver um software capaz de integrar máquinas e equipamentos que só possuem a possibilidade de se comunicarem com o protocolo Modbus, para que informações provenientes destes possam ser acessadas, tornando-os compatíveis com os conceitos da Indústria 4.0. Para verificar a aplicabilidade da solução proposta foi desenvolvido um protótipo de DashBoard de monitoração de uma linha de produção, que foi e interfaceado com um dispositivo físico que pode ser acionado pelo operador da linha de produção, através de botoeiras e sensores. Deste modo, a partir deste protótipo foi possível observar que a solução proposta atende as expectativas preliminares, mostrando ser factível utilizar o protocolo Modbus para interfacear sistemas de automação em um contexto de Indústria 4.0.



**Palavras-chave:** indústria 4.0; protocolo de comunicação; modbus; integração de sistemas.

**ABSTRACT**

With growing competitiveness in the industry sector, in 2011, Germany presented a new concept of Industry, with objectives of increasing productivity and quality and reducing production costs. This concept became known as Industry 4.0 and quickly became popular around the world. This concept is strongly linked to the integration of technologies, especially those related to information technologies. One of the main points of attention when implementing this concept in systems that already operate is related to the difficulty in integrating different devices, this problem is accentuated in cases of legacy equipment, which in many cases have a few decades of use. In this context, this work aims to verify the applicability of the use of the Modbus protocol, a communication protocol very popular in the industry since the mid-1980s, to assist the integration between legacy systems with architectures compatible with the Industry 4.0 concept. The objective is to develop software capable of integrating machines and equipment that only have the possibility of communicating with the Modbus protocol, so that information from them can be accessed, making them compatible with the concepts of Industry 4.0. A DashBoard prototype for monitoring a production line was developed, and interfaced with a physical device that can be activated by the production line operator, through buttons and sensors. Thus, from this prototype it was possible to observe that the proposed solution meets the preliminary expectations, showing that it is feasible to use the Modbus protocol to interface automation systems in an Industry 4.0 context.

**Keywords:** industry 4.0; communication protocol; modbus; systems integration.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTAGEM DE CÓDIGOS FONTE

## LISTA DE ABREVIATURAS E SIGLAS

### Abreviaturas

| | |
|---|---|
| art. | Artigo |
| cap. | Capítulo |
| sec. | Seção |

### Siglas

| | |
|---|---|
| CLP | Controlador Lógico Programável |
| CPS | *Cyber Phisical Systems* - Sistemas Ciber-Físicos |
| IoT | *Internet of Things* - Internet das "Coisas" |
| IoS | *Internet of Services* - Internet dos "Serviços" |
| OPC UA | *Open Platform Communications – Unified Architecture* - Comunicações de Plataforma Aberta – Arquitetura Unificada |
| RAMI 4.0 | *Reference Architectural Model for Industrie 4.0* - Modelo de Referência para Arquitetura da Indústria 4.0 |
| REST | *Representational State Transfer* - Transferência de Estado Representacional |
| RFID | *Radio Frequency Identification* - Identificação por Rádio-Frequência |
| SSE | *Server-Sent Events* - Eventos Enviados pelo Servidor |
| URL | *Uniform Resource Locator* - Localizador Uniforme de Recursos |
| UTFPR | Universidade Tecnológica Federal do Paraná |

**SUMÁRIO**

# 1 INTRODUÇÃO

A história da produção industrial é marcada por revoluções que impactaram tanto a atividade produtiva como a sociedade. Pode se dizer que a gênese do que conhecemos por indústria é anterior ao século XVIII, período onde o modelo produtivo predominante, hoje descrito como manufatura artesanal, era caracterizado por oficinas de artesãos, as guildas, que fabricavam produtos apenas a partir do uso de ferramentas manuais, o que resultava em produtos construídos em pequena escala, com pouca padronização, alto custo de produção e consequentemente alto preço. Outra característica marcante deste período era a alta qualificação dos artesãos, tamanha que normalmente os mesmos tinham domínio sobre todo o ciclo produtivo (SACOMANO *et al.*, 2018; QUINTINO *et al.*, 2019).

Em meados de 1760, o avanço tecnológico, com destaque para o tear a vapor de James Watt, impulsionou o desenvolvimento de uma nova era, que ficaria conhecida como a primeira revolução industrial. Neste período, a produção se torna mais concentrada e devido a crescente mecanização a mão de obra se torna menos qualificada, assim como, as condições de trabalho são deterioradas, com jornadas extenuantes de até 16 horas, e sem qualquer preocupação com a saúde e segurança dos trabalhadores (SACOMANO *et al.*, 2018; QUINTINO *et al.*, 2019).

Já no século XIX, impulsionado por um expressivo aumento na produção de aço e pela popularização do uso da energia elétrica, o setor industrial passou por uma nova revolução. A segunda revolução industrial teve seu modo produtivo modelado a partir de estudos de Adam Smith, Frederick Taylor e Henry Ford, e era marcado pela intensa racionalização, divisão e especialização do trabalho, características que podiam ser vistas principalmente nas até então inéditas linhas de produções. Nesta fase a indústria já apresentava produção em massa e produtos com maior qualidade e alto grau de padronização (SACOMANO *et al.*, 2018; QUINTINO *et al.*, 2019).

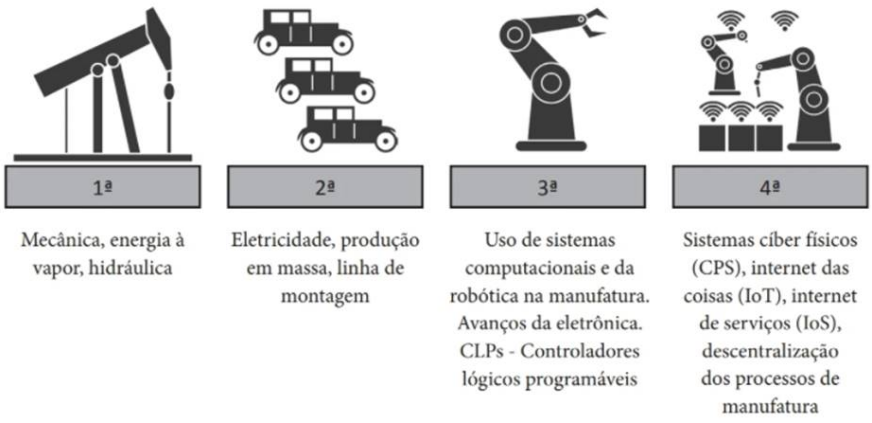
Na década de 1960, em um cenário de escassez de recursos, provocado principalmente pela derrota na segunda guerra mundial, o Japão se obriga a desenvolver um novo modelo produtivo para a indústria, especialmente caracterizado pela necessidade de otimização de recursos materiais e humanos. Assim, a terceira revolução industrial é caracterizada pelo substancial incremento dos padrões de qualidade, redução dos estoques e da inversão da direção do vetor do fluxo produtivo (que deixa de “empurrar” os produtos ao mercado, independentemente da demanda, para um modelo “puxado” pela demanda) e pela da redução do número de postos de trabalho devido ao uso de automação. Além de um novo modelo produtivo dominante, que ficou conhecido com Sistema Toyota de Produção ou *Lean Manufactering*, os avanços tecnológicos novamente foram impulsionadores determinantes dessa revolução, se destacando principalmente a microeletrônica, a automação industrial e as tecnologias da informação (SACOMANO *et al.*, 2018; QUINTINO *et al.*, 2019; PINTO, 2013).

Como é possível observar, as revoluções industriais são impulsionadas por fatores socioeconômicos, e a partir do uso de tecnologias e conceitos inovadores é possível remodelar

o setor produtivo. No caso da quarta revolução industrial, que também é conhecida como Indústria 4.0, não é diferente, porém, segundo Schwab (2019), o atual processo revolucionário difere principalmente pela sua multidisciplinaridade que, a partir da fusão de diversas áreas do conhecimento, tem um potencial transformador muito maior que as revoluções anteriores. Desta forma, a Indústria 4.0 se desenvolveu principalmente devido a um contexto de crescente competitividade a níveis globais, especialmente pelo vigor de economias emergentes como China e Índia. Assim, em 2011, durante a feira anual de Hanôver na Alemanha, foi apresentado um novo paradigma para indústria, denominado na ocasião como Indústria 4.0 (TESSARINI; SALTORATO, 2018; SILVA; KOVALESKI; PAGANI, 2019; QUINTINO *et al.*, 2019; SACOMANO *et al.*, 2018). Tal modelo tem se espalhado pelo mundo e mesmo que apresente adaptações às características de cada país, compartilha dos mesmos alicerces do modelo precursor. Dentre os países que possuem fortes programas de aceleração do desenvolvimento industrial, destacam-se, Estados Unidos, China e Coréia do Sul (TESSARINI; SALTORATO, 2018). A Figura 1 apresenta um resumo das principais características das revoluções industriais.

Ao mesmo tempo que a Indústria 4.0 aparenta possuir um grande potencial de prover ganhos substanciais a diversos aspectos da cadeia de valor, possui potencial igualmente alto de impactar outros aspectos da sociedade (econômicos, políticos e sociais) (TESSARINI; SALTORATO, 2018; SCHWAB, 2019).

**Figura 1 – Resumo das revoluções industriais**




**Fonte: (SACOMANO *et al.*, 2018).**

Um dos aspectos que merece especial atenção, principalmente no âmbito de países em desenvolvimento como o Brasil, seria o aprofundamento da defasagem e dependência tecnológica em relação às grandes potências industriais, afinal, tais tecnologias estão sendo desenvolvidas e padronizadas por nações que já possuem setores industriais dominantes em uma

perspectiva global (ARIAS *et al.*, 2020). Segundo Schwab (2019) 17% da população mundial ainda sequer aproveita plenamente das contribuições da segunda revolução industrial, devido à falta de acesso a eletricidade, similarmente ao que ocorre com 4 bilhões de pessoas que, devido a falta de acesso a Internet, ainda experienciam um mundo pré terceira revolução industrial. Logo, é razoável considerar a relevância do desenvolvimento de tecnologias próprias que consigam se aproveitar de conceitos básicos desse novo paradigma, ao mesmo tempo em que mantém certa autonomia.

Segundo Sacomano *et al.* (2018) a integração de sistemas é um dos principais desafios durante a implantação de projetos baseados no modelo de Indústria 4.0, principalmente pela falta de compatibilidade entre os diversos tipos de dispositivos presentes na indústria. Esta dificuldade torna-se ainda mais acentuada em países como o Brasil, onde a idade média das máquinas presentes na indústria é de 17 anos (SACOMANO *et al.*, 2018). Por fim, ao falar em integração de sistemas, consequentemente surge a necessidade de discutir os meios para alcançar essa desejada integração, e um dos fatores determinantes para isso são os protocolos de comunicação industrial.

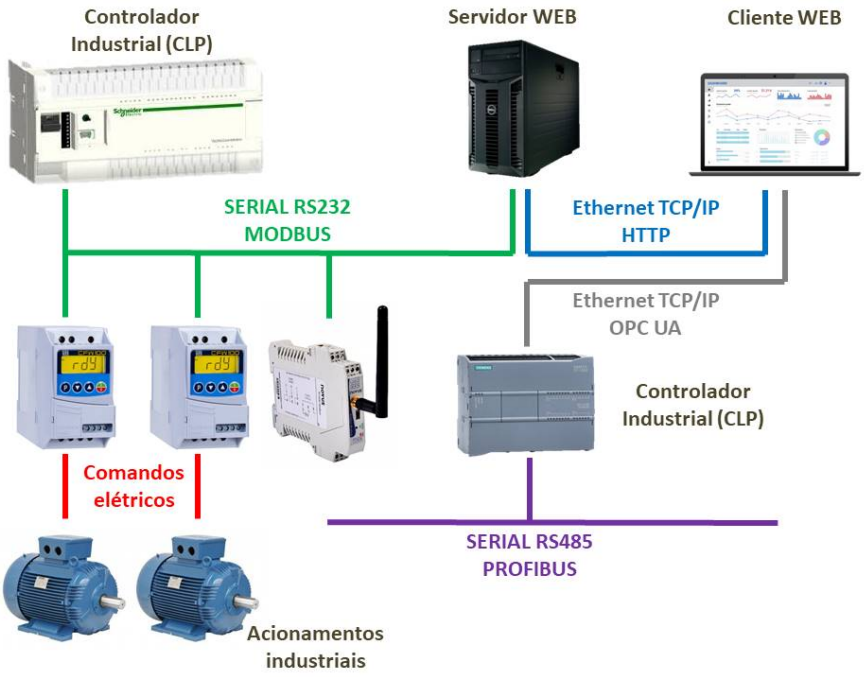
No contexto da Indústria 4.0, o protocolo recomendado para a interface entre os dispositivos físicos e os sistemas informatizados é o OPC UA (LIN *et al.*, 2017), que é implementado sobre uma interface Ethernet TCP/IP. O problema é que tal protocolo é relativamente novo, tendo sido lançado oficialmente em 2008 (MELO, 2020). Assim, muitos dos dispositivos presentes na indústria brasileira ainda não são compatíveis com este protocolo.

Tal problema poderia ser facilmente resolvido realizando substituição dos dispositivos antigos presentes na “Camada Física” por versões compatíveis com os atuais padrões de comunicação, o problema é que esta solução normalmente implicaria em consideráveis investimentos.

Portanto, é necessário desenvolver alternativas para os casos em que seja necessário integrar sistemas desatualizados ou que simplesmente não sejam compatíveis com o protocolo recomendado. No que diz respeito a protocolos de comunicação na camada de aplicação, o Modbus pode representar uma boa alternativa, devido ao seu uso consagrado na indústria desde os anos 1980. Este protocolo também possui algumas características interessantes, como ser um protocolo aberto, possuir versões a partir de meio físicos seriais e Ethernet e ser compatível com uma gama muito grande de dispositivos (AZEVEDO; SOUZA, 2014; ARRUDA; CAMPEDELLI; PEREIRA, 2011).

Assim, a partir do uso deste protocolo (Modbus) poderiam ser integrados diversos tipos de dispositivos, inclusive aqueles incompatíveis com protocolos de comunicação atualmente utilizados, conforme exemplo da Figura 2, onde é possível observar os dispositivos de automação legados (CLP, Inversores de Frequência e Conversor de Sinal) comunicando através do protocolo Modbus.

**Figura 2 – Exemplo de rede industrial**




**Fonte: Autoria própria (2022).**

## 1.1 Problema

Dentro de todo este contexto, a pergunta que se pretende responder com este trabalho é: É viável realizar a integração entre sistemas de automação industrial no contexto da Indústria 4.0 a partir da utilização do protocolo de comunicação MODBUS?

## 1.2 Justificativa

A falta de uma forma confiável e eficiente de troca de informações entre os níveis de "chão de fábrica" e estratégicos de uma indústria pode impactar negativamente nos resultados do negócio como um todo, em especial, em um ambiente globalizado de intensa competitividade no qual as indústrias estão inseridas. Assim, a integração de sistemas proposta no paradigma de indústria 4.0 pode representar uma promissora forma de mitigar as deficiências na troca de informações entre os diferentes níveis gerenciais na indústria.

Neste contexto, uma possível dificuldade ao integrar sistemas estaria relacionada à baixa compatibilidade entre dispositivos legados presentes na indústria brasileira com protocolos usualmente utilizados na integração de sistemas na Indústria 4.0. Assim, é necessário desenvolver alternativas e o uso protocolo Modbus pode representar um avanço nesses sentido.

## 1.3 Objetivo

Este trabalho busca avaliar a viabilidade de realizar a integração de dispositivos de automação legados que usem protocolo Modbus à um ambiente compatível com os conceitos da Indústria 4.0.

### 1.3.1 Objetivos específicos

- Pesquisar literatura relacionada aos temas de interesse do trabalho;
- Analisar a capacidade do protocolo MODBUS de prover comunicação entre dispositivos de automação legados (como CLPs sem suporte aos protocolos da Indústria 4.0) e o sistema informatizado;
- Construir e testar um protótipo que, a partir de dispositivos comuns na indústria simule um cenário industrial habitual;
- Projetar e implementar uma aplicação que realize a interface entre um dispositivo de automação legado que se comunica através protocolo Modbus e um componente com comunicação compatível com os protocolos da Indústria 4.0, usando a arquitetura REST, a partir desse momento essa aplicação será tratada pelo nome Modbus2REST API;
- Projetar e implementar um interface de usuário baseada na arquitetura REST, interface esta que irá acessar os recursos de comunicação disponibilizados pela aplicação Modbus2REST API;

## 1.4 Visão geral

De forma sucinta, este trabalho se propõe à apresentar uma solução para integração entre sistemas de informação e dispositivos industriais (em especial os dispositivos de automação legados), em um contexto de Indústria 4.0, a partir do uso do protocolo Modbus. A escolha desse protocolo se deve, principalmente a duas características, ser um protocolo aberto e ser um dos protocolos industriais mais populares na indústria, sendo regularmente disponibilizado em equipamentos deste segmento. É muito comum em ambientes industriais, a existência de dispositivos de automação legados, com suporte ao protocolo Modbus, que atualmente não podem ser integrados a sistemas de monitoramento e controle com as características propostas pela Indústria 4.0, e este trabalho visa preencher esta lacuna.

Para verificar a viabilidade da solução proposta para resolver problemas concretos foi desenvolvido um protótipo, que tem o intuito de simular uma aplicação real da indústria. A ideia geral desse protótipo é realizar a interface entre um sistema Web, baseado na arquitetura REST,

e um controlador industrial. O protótipo foi concebido a partir de um situação comum na indústria, a supervisão de linha de montagem. Assim, foi considerado o seguinte cenário.

### 1.4.1 Cenário

Foi considerada uma linha de montagem com três posto de trabalho, com um trabalhador (operador) em cada posto, que realizam as tarefas específicas para montagem do produto, que pode ser um eletro-doméstico, um componente automotivo ou qualquer outro. O produto em questão parte de seu estado bruto, e precisa passar linearmente por cada uma das estações.

No momento anterior a implantação do sistema, a linha de montagem apresenta alguns problemas. O controle da produção é realizado de forma manual, ou seja, após o término dos turnos de trabalho os dados de produção (hora de início, hora de fim e quantidade produzida) são registrados manualmente, o que pode implicar em inconsistências. Além disso, no decorrer da produção ocorrem situações onde o operador necessita de auxílio, por exemplo, quando faltam insumos no posto de trabalho. Nesses casos, o operador precisa entrar em contato com quem pode lhe auxiliar (supervisor), situação que pode causar atrasos na produção.

Deste modo, o sistema se propõe a melhorar o desempenho desta suposta linha de montagem a partir de dois aspectos, automatização do controle da produção e implantação de um meio de criação e reconhecimento de chamados (ocorrências) da produção.

# 2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos fundamentais para o desenvolvimento e melhor compreensão do trabalho. A primeira sub-seção apresenta os principais conceitos da Indústria 4.0, a segunda sub-seção apresenta conceitos relacionados as redes industriais com foco no protocolo Modbus, enquanto que a terceira sub-seção apresenta conceitos relacionados à serviços Web. Finalmente, na quarta sub-seção, trabalhos relacionados ao aqui proposto são apresentados, com ênfase na comparação das características entre eles e o presente trabalho.

## 2.1 Indústria 4.0

O termo Indústria 4.0 tem seu embrião no ano de 2011, na feira anual de Hannover na Alemanha e está relacionado a um projeto do governo alemão, a Plataforma Industrial 4.0 (Plattform Industrie 4.0) que tinha como principal objetivo fomentar o desenvolvimento tecnológico do setor industrial e a partir da integração entre os equipamentos de automação e destes com os humanos, otimizando os processos industriais (SACOMANO *et al.*, 2018). A partir de 2013, a Plataforma Industrial 4.0 passou a ser divulgada em empresas, associações e no meio acadêmico, até que em 2015 é relançada, agora como um programa do governo alemão (SACOMANO *et al.*, 2018). Assim, em um contexto de revolução digital, caracterizada pela popularização da Internet e de dispositivos computadorizados (PCs, smartphones e tablets), surge o conceito de Indústria 4.0, que assim como o a Plataforma Industrial 4.0, se baseia na integração entre máquinas e pessoas a partir do uso de sistemas informatizados para aumentar a produtividade, a flexibilidade e qualidade dos processos industriais(SACOMANO *et al.*, 2018).

Arias *et al.* (2020) afirma que ainda não existe uma definição consolidada a respeito do termo Indústria 4.0, porém, existem elementos que comumente estão presentas nas tentativas de definição, dentre os quais se destacam o uso de sistemas CPS (Ciber Physical Systems), IoT, análise de dados, virtualização e automatização dos processos industriais.

Sacomano *et al.* (2018) define Indústria 4.0 como:

> "um termo coletivo que engloba tecnologias e conceitos de cadeia de valor de uma organização. Dentro da estrutura modular das fábricas inteligentes (smart factories) da Indústria 4.0, sistemas ciber físicos (cyber-physical systems – CPS) monitoram processos físicos, criando uma cópia virtual do mundo físico, podendo tomar decisões descentralizadas. Por meio da internet das coisas (internet of things – IoT), CPS comunicam-se e cooperam uns com os outros e com humanos em tempo real. Via internet de serviços (IoS), serviços internos e externos à organização são oferecidos e utilizados pelos participantes da cadeia de valor"(SACOMANO *et al.*, 2018).

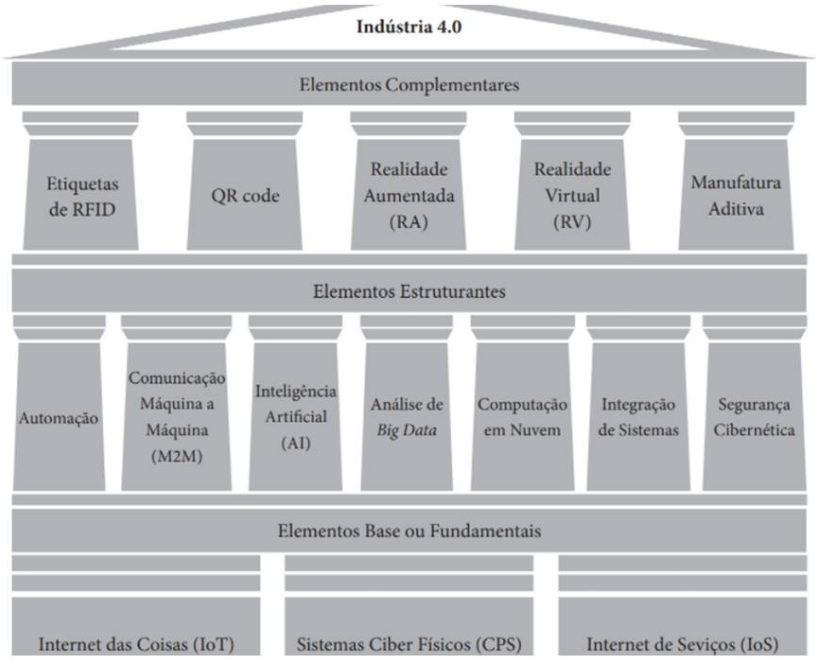
Outro ponto importante sobre a Indústria 4.0 é que além das práticas direcionadas a manufatura de bens, sua proposta também busca fomentar um ecossistema de prestação serviços

relacionados ao uso de sistemas digitais com uso extenso de práticas de mineração e análise de grandes quantidades de dados (SACOMANO *et al*., 2018).

Segundo Schwab (2019), a Indústria 4.0 também é caracterizada por uma expressiva agregação entre tecnologias de diversas áreas do conhecimento e segundo este autor, isto seria o principal fator que torna a Indústria 4.0, ou quarta revolução industrial, única e com potencial transformador sem precedentes. A partir da maior sofisticação e capacidade de integração de tecnologias populares desde a revolução digital como computação, Internet, robótica e automação é possível produzir relevante progresso em diversas áreas do conhecimento como biologia, nanotecnologia e geração de energia limpa.

O paradigma da Indústria 4.0 propõe que os processos industriais se utilizem de um arcabouço tecnológico com propósito de aumentar a produtividade e a flexibilidade, assim como, reduzir custos de produção. Segundo Sacomano *et al.* (2018) a Indústria 4.0 possui três elementos base, os Sistemas Ciber-Físicos (CPS), a internet das coisas (IoT) e a internet dos serviços (IoS), que seriam os elementos fundamentais nos quais o modelo se apoia. Da mesma forma possui elementos estruturantes, que seriam os elementos/conceitos que permitem o desenvolvimento de processos nos moldes da Indústria 4.0, que poderiam ser classificados desta forma: Automação, comunicação entre máquinas, inteligência artificial, Big Data, computação em nuvem, integração de sistemas e segurança da informação. Por fim existiriam os elementos complementares, os quais ampliariam as possibilidades das aplicações da Indústria 4.0, que são as tecnologias como RFID (*Radio-Frequency Identification*), QRCode (*Quick Response Codes*) e Realidade Aumentada (SACOMANO *et al*., 2018). A Figura 3 apresenta um resumo desta classificação.

**Figura 3 – Elementos formadores da Indústria 4.0**



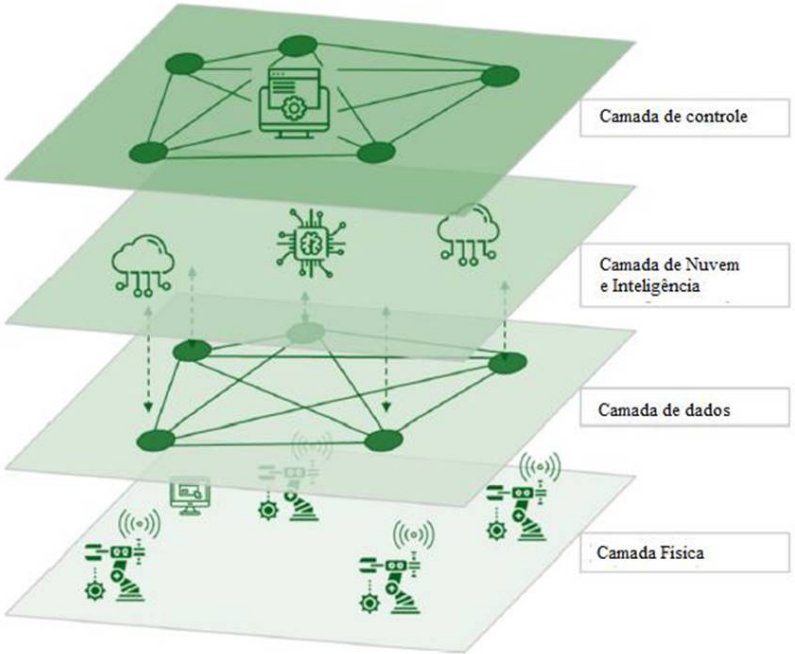

**Fonte: (SACOMANO *et al*., 2018).**

Por outro lado, Quintino *et al.* (2019) define nove pilares para Indústria 4.0, Sistemas Ciber-Físicos, IoT, Realidade Aumentada, Big Data, Robótica Autônoma, Impressão 3D, Simulação, Integração de Sistemas e Computação em Nuvem.

Independentemente da forma de classificar os elementos presentes no modelo é possível verificar que o princípio da Indústria 4.0 está fortemente relacionado à integração entres os dispositivos digitais de vários níveis da cadeia produtiva, assim como, a integração destes com os seres humanos, sendo assim possível verificar o papel central que a integração de sistemas possui no paradigma da Indústria 4.0.

Segundo Silva e Webber (2020) a arquitetura de sistema para fábricas inteligentes é formada por quatro camadas tangíveis (Figura 4): Camada Física, Camada de Dados, Camada de Nuvem e Inteligência e Camada de Controle. A “Camada Física” realiza a interface entre o mundo físico o virtual, sendo que nesta camada estão presentes, por exemplo, os controladores industriais. A “Camada de Dados” é a camada onde ocorre efetivamente a distribuição dos dados e informações entre o dispositivos físicos e entre estes e os sistemas de informação, sendo esta a denominada camada da IoT. A “Camada de Nuvem e Inteligência” tem o papel de realizar a análise dos dados e prover as bases científicas para as tomadas de decisões. Por fim, é na “Camada de Controle” onde as informações são analisadas pelas pessoas e as decisões são tomadas.

**Figura 4 – Camadas de fábricas inteligentes**




**Fonte: (SILVA; WEBBER, 2020).**

### 2.1.1 Principais elementos da Indústria 4.0

De forma similar ao que ocorre com a definição do conceito da Indústria 4.0 a classificação dos seus elementos formadores também é proposta por formas distintas de acordo com cada autor, conforme visto no capítulo anterior. Apesar desta dificuldade é possível verificar que os elementos a seguir citados tem sua relevância destacada independente do autor (ARIAS *et al.*, 2020; SACOMANO *et al.*, 2018; SCHWAB, 2019; ALMEIDA, 2019).

### 2.1.2 Sistemas Ciber-físicos (CPS - *Cyber-Physical Systems*)

Como o nome sugere, são sistemas que realizam a interface entre o mundo físico e o mundo digital. São dispositivos formados por unidades de controle conectadas ao meio físico através de sensores e atuadores. Tais dispositivos também apresentam interfaces de comunicação, através das quais poderá ser realizada a troca de dados e informações com outros elementos da cadeia de valor (MELO, 2020). Através da camada virtual é possível realizar funções como, gêmeo virtual, interfaces de gerenciamento e controle (*dashboards*), controle automático descentralizado e captura e armazenamento de dados (SACOMANO *et al.*, 2018).

### 2.1.3 Internet das coisas (IoT - *Internet of things*)

Conceito que diz respeito a conexão dos mais diversos tipos de dispositivos (por exemplo, sensores, atuadores, eletrodomésticos e automóveis) a redes de computadores, desta forma permitindo acesso remoto aos mesmos (MELO, 2020; SACOMANO *et al.*, 2018; ALMEIDA, 2019). Além da troca de dados entre pessoas e dispositivo, este conceito também prevê a troca de dados entre dispositivos. Um ponto que necessita de atenção no desenvolvimento de aplicações deste tipo é a compatibilidade entre protocolos, algo que nem sempre é trivial (SACOMANO *et al.*, 2018).

### 2.1.4 Internet dos serviços (IoS - *Internet of services*)

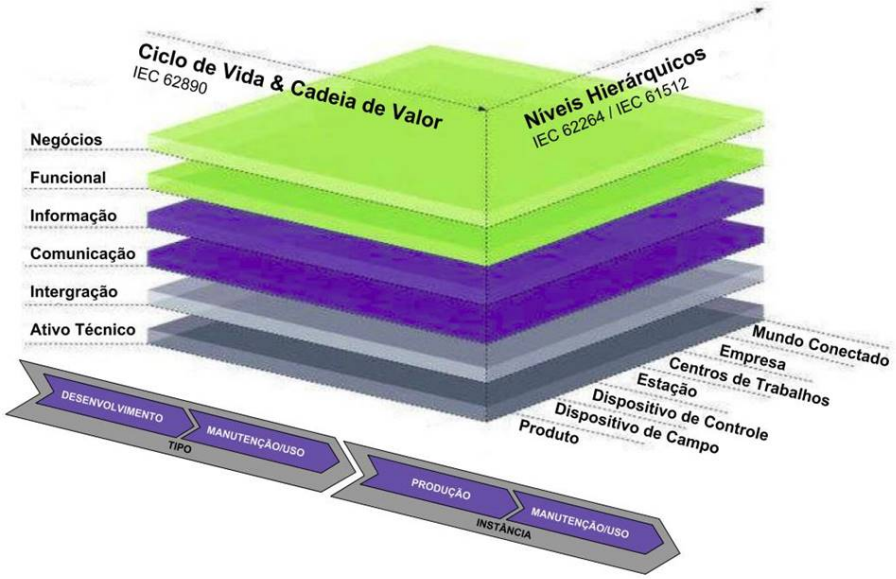
Seria a noção de uma rede de negócios conectada através da Internet que tem a capacidade de realizar a interface direta entre provedor do serviço, usuário e fornecedores (SACOMANO *et al.*, 2018; MELO, 2020). Essa relação mais próxima entre os componentes da rede de negócios visa agregar valor a cadeia de suprimentos, onde através da Internet o consumidor pode pesquisar, negociar e pagar um serviço ao mesmo tempo em que o o provedor de serviços procede de forma similar com seus fornecedores, assim, produzindo uma grande rede de negócios (MELO, 2020).

Para Sacomano *et al.* (2018) tais elementos formam a base da Indústria 4.0. Mas além desses elementos, Inteligência Artificial, Realidade Aumentada, Big Data, Robótica Autônoma, Impressão 3D, Simulação, Integração de Sistemas e Computação em Nuvem também são regularmente associados a Indústria 4.0.

### 2.1.5 RAMI 4.0

A partir da boa aceitação do conceito de Indústria 4.0 pelo mercado surge a necessidade de padronização, afinal, um dos pilares deste novo paradigma de indústria é a integração de sistemas, para que isto seja possível, dispositivos de diferentes fabricantes precisam seguir os mesmos padrões. Além disso, foi necessário estabelecer um modelo de referência para adequação de fábricas aos conceitos da Indústria 4.0 (MELO, 2020). Para isso, um grupo de instituições e empresas alemãs lançaram o RAMI 4.0 (*Reference Architectural Model for Industrie 4.0*), um modelo tridimensional em que cada eixo representa um aspecto no contexto da Indústria 4.0. Esses eixos são, eixo de níveis hierárquicos, eixo de ciclo de vida e cadeia de valor e eixo das camadas (Figura 5) (MELO, 2020).

**Figura 5 – Modelo tridimensional RAMI 4.0**




**Fonte: (MELO, 2020).**

Este modelo está sendo introduzido pela organização de normatização e normalização industrial, suportando assim, diversos membros da cadeia de valor, como fabricante de controladores, fabricantes de máquinas e consumidores. Uma das características desse modelo é a recomendação pelo uso de tecnologias de controle e comunicação no estado da arte (MELO, 2020).

Com relação a protocolos de comunicação, a RAMI 4.0 recomenda apenas o uso do protocolo OPC UA (*Open Platform Communications – Unified Architecture*) na camada de comunicação, dado que tal protocolo consegue atender todos os requisitos necessário para aplicações da Indústria 4.0 (MELO, 2020).

### 2.1.6 Desafios para indústria brasileira

Apesar de ser originalmente um projeto alemão, a Indústria 4.0 já é um elemento chave para definir o grau de competitividade das indústrias em um contexto de economia globalizada. Desta forma, discussões em torno deste conceito são relevantes tanto em países que já possuem um setor industrial desenvolvido quanto em países em desenvolvimento no aspecto industrial. Segundo Pereira e Simonetto (2018) o Brasil se enquadra no segundo grupo, por possuir um setor industrial com características entre a segunda e a terceira revolução industrial, sendo a indústria automotiva aquela que possui o maior grau de desenvolvimento.

Segundo a Agência Brasileira de Desenvolvimento Industrial, para que o Brasil consiga caminhar em direção a quarta revolução industrial, é necessário desenvolvimento em cinco eixos. Criação de um programa brasileiro de manufatura avançada; Buscar acordo bilateral com a Alemanha, entre o programa de manufatura avançada criado e o alemão Industrie 4.0; Criação de uma rede de *Testbeds*[1] de manufatura avançada no Brasil; Alinhamento e criação de linhas de fomento; Engajamento de pequenas e médias empresas (PEREIRA; SIMONETTO, 2018).

## 2.2 Redes Industriais

As redes de comunicação industrial tem o objetivo de prover a troca de dados entre dispositivos de campo, controladores lógicos, sistemas supervisórios, sistemas de gerenciamento de produção e sistemas de controle. As principais características de uma rede de comunicação industrial são, topologia, meio de transmissão, número máximo de dispositivos, distância máxima, método de comunicação, taxa de transmissão e tempo de ciclo (AZEVEDO; SOUZA, 2014).

### 2.2.1 Modbus

A Modicon Company publicou o protocolo Modbus em 1979 para ser utilizado como interface de comunicação entre diferentes tipos de dispositivos de automação (ARRUDA; CAMPEDELLI; PEREIRA, 2011). É bastante popular na indústria, principalmente pela sua fácil operação e manutenção e por ser tratar de um protocolo aberto (AZEVEDO; SOUZA, 2014). Devido

[1] *Testbeds* são ambientes de teste e demonstração de tecnologias, que buscam simular a realidade de ambientes de produção.

a tais características este protocolo se mantém entre os mais populares da indústria, mesmo em dispositivos novos (TAMBOLI *et al.*, 2015).

O Modbus a princípio foi desenvolvido sobre comunicação serial, que podia ser implementado em duas versões, RTU e ASII. Em sua versão serial, o Modbus é do tipo Mestre-Escravo, acomoda até 246 dispositivos, com distância e taxa de transmissão máximas de 350 metros e 57,6kbps, respectivamente. Com relação ao funcionamento, a comunicação sempre é iniciada pelo mestre da rede que envia uma solicitação para um escravo que processa esta solicitação e responde ao mestre. A comunicação é realizada através de um quadro de dados que possui as seguintes informações, *Slave ID*, *Function code*, *Data* e *Error Code* (AZEVEDO; SOUZA, 2014).

A partir de novas exigências do mercado, a rede Modbus também ganhou uma versão baseada na rede Ethernet TCP/ IP, o Modbus TCP. Nesta versão, o protocolo não possui limitação de dispositivos na rede e pode alcançar taxas de transmissão de até 100 Mbps (AZEVEDO; SOUZA, 2014).

## 2.3 Serviços Web

No âmbito dos sistema distribuídos, um serviço Web seria a execução de uma tarefa, por parte de um servidor, em resposta a uma requisição de um cliente através da Web. Esta tecnologia permite a interação entre diferentes aplicações desenvolvidas em diferentes linguagens de programação, plataformas e sistemas operacionais. Tais serviços podem interagir sem a supervisão humana, assim, permitindo o desenvolvimento de aplicações complexas, a partir da interação de vários serviços, geralmente através do protocolo HTTP (COULOURIS *et al.*, 2013; RIBEIRO; FRANCISCO, 2016).

Segundo Ribeiro e Francisco (2016), as arquiteturas de serviços Web mais populares na literatura especializada são, SOAP (Simple Object Access Protocol) e REST (*Representational State Transfer*).

### 2.3.1 REST

REST é um estilo de arquitetura com ênfase na manipulação de recursos de dados, em vez de interfaces e desta forma, simplificando a interação entre aplicação cliente e servidor, onde os clientes, usando URLs e as operações HTTP GET, PUT, DELETE e POST realizam requisições ao servidor. Geralmente as representações REST são formatadas em XML (*Extensible Markup Language*) ou JSON (*JavaScript Object Notation*), porém, podem adotar outras notações de representação e até mesmo mas que uma notação (RIBEIRO; FRANCISCO, 2016).

Uma das características marcantes de serviços REST, é que a partir de uma URL inicial, são disponibilizadas informações dinâmicas sobre as demais interações disponíveis no serviço

em questão, esta tecnologia é conhecida como HATEOAS (*Hypermedia As The Engine Of Application State*) (RIBEIRO; FRANCISCO, 2016).

O estilo de arquitetura REST possui as seguintes características, arquitetura cliente-servidor, *stateless*, cache, interface uniforme, sistema de camadas e código sob demanda (RIBEIRO; FRANCISCO, 2016).

## 2.4 Trabalhos Relacionados

A comunicação industrial através do protocolo Modbus é um tema relativamente popular em trabalhos relacionados a automação industrial. No caso do trabalho de Schmitt *et al.* (2013), este protocolo é utilizado para implementar uma rede de comunicação entre um robô industrial e uma fonte de soldagem, permitindo desta forma o desenvolvimento de uma célula de soldagem automatizada a arco voltaico.

Na mesma linha, o trabalho de Arruda, Campedelli e Pereira (2011) utiliza o protocolo Modbus, padrão RS485, para desenvolver uma rede para controle de um veículo autônomo, que possibilitou aos autores concluírem que o protocolo apresenta as características necessárias para este tipo de tarefa.

Já com relação a trabalhos relacionados a plataformas computacionais para Indústria 4.0, Silva e Webber (2020) buscam realizar uma análise comparativa entre as principais plataformas de desenvolvimento para Indústria 4.0 disponíveis no mercado. Para isso utilizam o Método Analítico Hierárquico, assim realizando uma análise comparativa multicritério. Foram analisadas três plataformas, Amazon Web Services (AWS), IBM Watson e Microsoft Azure, o que permitiu aos autores verificar que todas as plataformas apresentaram um bom desempenho e cumprem o que se espera das mesmas. Também foi possível verificar que cada plataforma tem pontos de destaque, e que em um comparativo geral a plataforma da Microsoft se sobressai.

O trabalho de GODOY (2016) busca propor uma arquitetura de automação industrial baseada em comunicação Modbus TCP e um servidor WEB, servidor este, desenvolvido através da plataforma ScadaBR. Para tal, foi utilizado um estudo de caso de controle de vazão através de PIDplus. Ao final do artigo os autores concluíram que a arquitetura proposta apresentou resultados adequados a tarefa proposta, logo que, a partir de tal arquitetura é possível qualificar a análise e tomadas de decisões, através da disponibilização em tempo real dos dados adquiridos do processo e assim podendo possibilitar um melhor desempenho do processo produtivo.

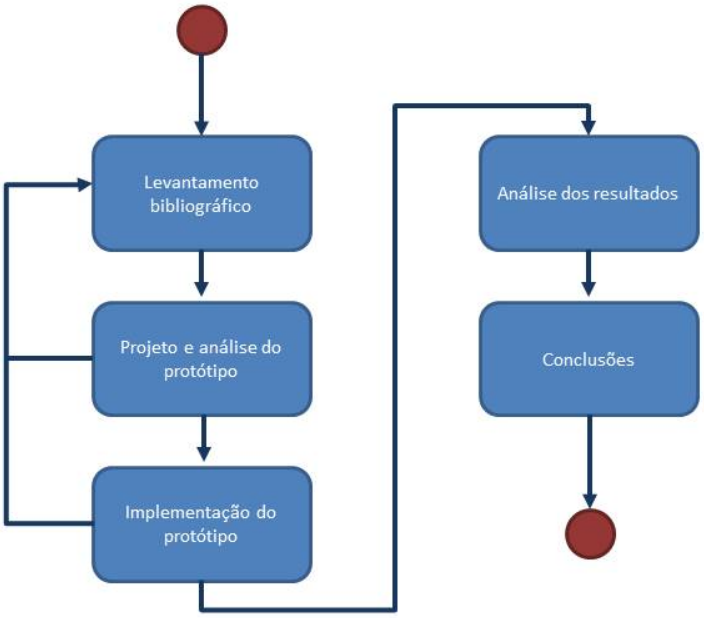
# 3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados métodos e recursos necessários para o desenvolvimento deste trabalho. Este trabalho busca analisar a aplicabilidade do uso do protocolo Modbus para realizar integração de sistemas em um contexto da Indústria 4.0. Tal análise será realizada através do desenvolvimento de sistema WEB, desenvolvido em linguagem Java utilizando o *framework* Spring-Boot [1].

## 3.1 Etapas da pesquisa

Assim, o fluxo do desenvolvimento deste trabalho é representado pelo fluxograma a seguir (Figura 6).

**Figura 6 – Fluxograma de desenvolvimento da pesquisa**




**Fonte: Autoria própria (2022).**

[1] Atualmente Spring Framework é o framework Java mais popular, seus recursos permitem o desenvolvimento eficiente e simples para aplicações WEB. Os principais conceitos que o Spring Framework são, Inversão de Controle (IoC), Injeção de Dependência (DI), Programação Orientada a Aspectos (AOP), API de Persistência Java (JPA). Por sua vez, o Spring Boot foi projetado para simplificar o desenvolvimento de aplicativos Spring, fornecendo uma experiência inicial mais rápida e mais acessível para todo o desenvolvimento Spring (JOVANOVIĆ *et al.*, 2017).

### 3.1.1 Levantamento bibliográfico:

Nesta etapa do trabalho foram levantadas referências bibliográficas a respeito do tema da pesquisa, a partir das quais o trabalho foi embasado. Os principais termos pesquisados foram, "Indústria 4.0", "Modbus", "Redes Industriais", "IoT", "Automação industrial"e "Integração de Sistemas". Tais referências são artigos, livros, dissertações, teses, *white papers* e especificações técnicas. Vale ressaltar que não foram aplicados quaisquer métodos de revisão sistemática durante o levantamento bibliográfico. Foram usadas as seguintes bases:

- **Portal de Periódicos CAPES/MEC**
  - *https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/*
- **Web biblioteca UTFPR**
  - *https://webapp.utfpr.edu.br/bibservices/minhaBiblioteca*
- **IEEE Xplore**
  - *https://ieeexplore-ieee-org.ez48.periodicos.capes.gov.br/Xplore/guesthome.jsp*
- **ACM Digital Library**
  - *https://dl-acm-org.ez48.periodicos.capes.gov.br/*
- **Google Acadêmico**
  - *https://scholar.google.com.br/scholar?q=*

### 3.1.2 Análise e projeto visando o desenvolvimento de um protótipo:

Após o devido embasamento teórico foi desenvolvido tanto a análise quanto o projeto do sistema.

### 3.1.3 Implementação do protótipo:

O sistema foi implementado de acordo com a especificação previamente realizada.

### 3.1.4 Análise dos resultados:

Foi realizado uma análise qualitativa visando constatar se o sistema proposto consegue apresentar um desempenho adequado ao cenário proposto.

### 3.1.5 Conclusões:

Finalização do trabalho, onde foi discutidos os resultados e propostos possíveis caminhos para trabalhos futuros.

## 3.2 Recursos de hardware e software

### 3.2.1 Recursos de hardware

Para desenvolvimento do sistema será utilizado um notebook, com as seguintes características:

- Sistema operacional: Windows 10 Pro
- Processador: Intel(R) Core(TM) i5-5200U CPU @2.20 GHz;
- Memória RAM: 12 GB;

Com relação ao CLP, será utilizado o seguinte equipamento:

- Modelo: CPU 1214C DC/DC/DC;
- Código: 6ES7214-1AG40-0XB0;
- Fabricante: Siemens;

Este equipamento possui uma porta Ethernet através da qual será realizada a comunicação Modbus TCP. Vale ressaltar que apesar de possuir uma porta de comunicação Ethernet, esta versão de equipamento não é compatível a protocolos de comunicação usuais em no contexto da Indústria 4.0.

### 3.2.2 Recursos de Software

Serão utilizados os seguintes aplicativos:

- IDE Eclipse 2021-06 (4.20.0) - Responsável pelo desenvolvimento do programa JAVA/ Spring Boot;
- TIA Portal V16 - Responsável por realizar as configurações do programa do CLP;

# 4 DESENVOLVIMENTO

Neste capítulo serão apresentadas as atividades realizadas para alcançar os objetivos propostos pelo trabalho em questão. Conforme dito no capítulo anterior a proposta central deste trabalho é desenvolver um sistema baseado em um cenário real da indústria, a partir do qual seria verificada a viabilidade de utilizar o protocolo Modbus para prover comunicação com equipamentos industriais em um contexto de Indústria 4.0.

Este capítulo possui duas seções, a primeira apresenta a análise e projeto do sistema, enquanto que a segunda seção descreve os detalhes da implementação do sistema desenvolvido.

## 4.1 Projeto de Sistema

Nesta seção são apresentados a análise e o projeto do sistema, dentre os quais estão presentes, análise de requisitos, casos de uso, diagrama de classes e diagrama de componentes.

### 4.1.1 Requisitos do sistema

**Requisitos funcionais**

1. O software deverá mostrar os chamados presentes na linha de montagem;
2. O software deverá permitir que os chamados sejam reconhecidos pelo supervisor;
3. O software deverá registar os chamados realizados durante o turno de trabalho, sendo que deverão ser registradas as seguintes informações:

   Hora do chamado, hora do reconhecimento do chamado, tipo do chamado, status do chamado (se foi ou não reconhecido), posto onde ocorreu o chamado e usuário que realizou o reconhecimento do chamado;
4. O software deverá mostrar o desempenho da produção em tempo real, ou seja, deverá ser calculado e mostrado o quanto foi produzido em relação ao quanto deveria ter sido produzido a cada instante de tempo.
5. O software deverá apresentar a meta de produção;
6. O software deverá realizar o monitoramento de um sinal analógico;

**Requisitos não funcionais**

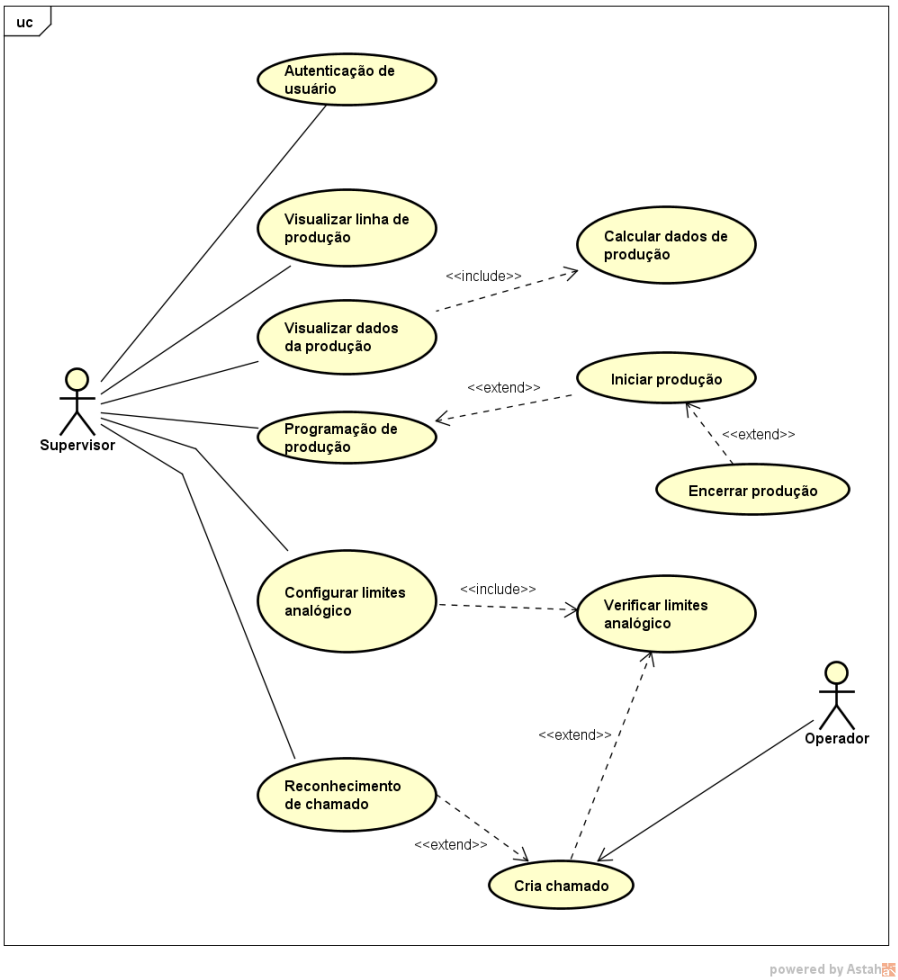
1. O software deverá realizar aquisição e tratamento de dados do controlador industrial.

2. O software deverá se comunicar com o controlador industrial através do protocolo Modbus TCP.
3. O software deverá ser um aplicação distribuída, utilizando a arquitetura REST.
4. O software deverá ser implementado na linguagem Java no que tange ao servidor REST.
5. O software deverá ser implementado na linguagem Javascript no que tange a interface com o usuário.
6. O software deverá utilizar banco de dados relacional.

### 4.1.2 Casos de uso

A partir deste levantamento dos requisitos foi possível elaborar o diagrama de casos de uso, apresentado na Figura 7.

**Figura 7 – Diagrama de casos de uso**




**Fonte: Autoria própria (2022).**

O detalhamento dos casos de uso é apresentado entre nas tabelas de 1 a 11.

**Tabela 1 – Casos de uso [Autenticação de usuário]**

| | |
|---|---|
| **Nome do caso de uso** | Autenticação de usuário |
| **Descrição** | Realiza a autenticação dos usuários cadastrados no sistema |
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Sistema na tela inicial aguardando login |
| **Pós-condições** | Usuário tem acesso às funcionalidades do sistema de acordo com seu nível de acesso |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| | **{Aguarda dados de acesso}** |
| | Sistema aguarda que os dados de acesso (usuário/ senha) sejam preenchidos |
| Fornece dados de acesso do usuário | |
| | **{Valida dados de acesso}** |
| | Verifica se os dados informados correspondem aos dados armazenados. (A1) |
| | **{Libera acesso ao sistema}** |
| | Libera acesso de usuário ao sistema de acordo com seu nível de acesso. |
| | **{Abre tela inicial do sistema}** |
| | Abre a tela inicia do sistema |
| **Fluxo Alternativo** | |
| A1: em Valida dados de acesso | Caso os dados informados não correspondam a um usuário cadastrado, informa erro ao usuário e retorna ao fluxo básico em **{Aguarda dados de acesso}**. |

**Fonte: Autoria própria (2022)**

**Tabela 2 – Casos de uso [Visualizar linha]**

| | |
|---|---|
| **Nome do caso de uso** | Mostrar linha |
| **Descrição** | Mostra condição dos postos de trabalho |
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Supervisor logado no sistema |
| **Pós-condições** | |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| | **{Mostra condição da linha}** |

**(continua)**

**(continuação)**

| | |
|---|---|
| | O sistema mostra uma visão geral da linha de montagem, onde será possível observar a existência de chamados pendentes nos postos de trabalho. Para tal, a cor do posto e trabalho indicará a condição do mesmo, as cores, azul, amarelo e vermelho, representaram respectivamente, chamados relacionados à logística, alerta ao líder da linha e parada de emergência. Além das condições dos postos de trabalho também será mostrado o desempenho da produção |

**Fonte: Autoria própria (2022)**

**Tabela 3 – Casos de uso [Visualizar dados de produção]**

| **Nome do caso de uso** | Mostrar dados de produção |
|---|---|
| **Descrição** | Mostra dados de produção referente a meta e quantidade produzida |
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Supervisor logado no sistema |
| **Pós-condições** | |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| | **{Mostra dados de produção}** |
| | O sistema mostra os valores relacionados a meta de produção, que serão, quantidade planejada para o dia, quantidade produzida no dia e a diferença entre esses dois dados, também será mostrado a quantidade desejada em tempo real, dado calculado pelo próprio sistema, também será mostrada a quantidade produzida até o momento e a diferença entre desejado e produzido. Além dos valores relacionados a contagem de itens produzidos, também será mostrado o valor atual do sinal analógico. |

**Fonte: Autoria própria (2022)**

**Tabela 4 – Casos de uso [Calcular dados de produção]**

| **Nome do caso de uso** | Calcular dados de produção |
|---|---|
| **Descrição** | Calcula dados de produção referente a meta e quantidade produzida |

**(continua)**

**(continuação)**

| | |
|---|---|
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Supervisor logado no sistema |
| **Pós-condições** | |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| | **{Calcular condição da linha}** |
| | Realiza os cálculos referente à meta de produção e diferença entre produzido e planejado, tanto no que diz respeito ao dia quanto ao momentâneo. |

**Fonte: Autoria própria (2022)**

**Tabela 5 – Casos de uso [Programar produção]**

| | |
|---|---|
| **Nome do caso de uso** | Programar produção |
| **Descrição** | Define meta e horário de fim de produção |
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Supervisor logado no sistema |
| **Pós-condições** | Meta e horário de fim de produção configurados e libera comando de início de produção |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| Seleciona campo referente ao dado que deseja ser configurado | |
| | **{Libera edição do campo}** |
| | O sistema libera a edição do campo desejado (**produção planejada** ou **horário de fim da produção**). Obs. Ambos os dados devem ser configurados. |
| Confirma valores editados | |
| | **{Registra valores}** |
| | O sistema salva os valores configurados |
| | **{Libera inicio de produção}** |
| | O sistema libera o comando de início de produção |

**Fonte: Autoria própria (2022)**

**Tabela 6 – Casos de uso [Iniciar produção]**

| | |
|---|---|
| **Nome do caso de uso** | Iniciar produção |
| **Descrição** | Inicia contagem de peças e cálculo dos dados de produção |
| **Ator envolvido** | Supervisor |

**(continua)**

**(continuação)**

| | |
|---|---|
| **Pré-condições** | Meta e horário de fim de produção registrados |
| **Pós-condições** | Contagem de peças e cálculo de produção iniciados |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| Confirma inicio de produção | |
| | **{Inicia produção}** |
| | O sistema inicia a contagem de peças inclusive calculando a produção planejada a cada instante de tempo. (A1) |
| | **{Horário de fim alcançado}** |
| | Ao ser alcançado o horário de fim da produção a contagem de peças é interompida. |
| | **{Registra dados de produção}** |
| | Os dados de produção, quantidade planejada, quantidade produzida e horário de início e fim, são registrados permanentemente no sistema. |
| **Fluxo Alternativo** | |
| A1: em **{Inicia produção}** | Caso deseje, o supervisor pode forçar o encerramento da produção antes que o horário programado seja alcançado. Retornando ao fluxo básico em **{Registra dados de produção}** |

**Fonte: Autoria própria (2022)**

**Tabela 7 – Casos de uso [Encerrar produção]**

| | |
|---|---|
| **Nome do caso de uso** | Encerrar produção |
| **Descrição** | Encerra contagem de peças e cálculo dos dados de produção |
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Produção iniciada |
| **Pós-condições** | Produção finalizada |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| Usa comando Encerrar produção | |
| | **{Encerrar produção produção}** |
| | O sistema encerra a contagem de peças. |
| | **{Registra dados de produção}** |
| | Os dados de produção, quantidade planejada, quantidade produzida e horário de início e fim, são registrados permanentemente no sistema. |

**Fonte: Autoria própria (2022)**

**Tabela 8 – Casos de uso [Configurar limites analógico]**

| **Nome do caso de uso** | Configurar limites analógico |
|---|---|
| **Descrição** | Define valor de alarme máximo e mínimo do sinal analógico |
| **Ator envolvido** | Supervisor |
| **Pré-condições** | Supervisor logado no sistema na tela inicial |
| **Pós-condições** | Valores máximo e mínimo do alarme do sinal analógico configurados, assim, permitindo o monitoramento e no caso onde os valores programados sejam excedidos, será criado um chamado. |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| Abre tela de configuração dos limites do sinal analógico | |
| Seleciona campo referente ao dado que deseja ser configurado | |
| | **{Libera edição do campo}** |
| | O sistema libera a edição do campo desejado (**máximo** ou **mínimo**). Obs. Ambos os dados devem ser configurados. |
| Confirma valores editados | |
| | **{Registra valores}** |
| | O sistema salva os valores configurados |
| | **{Inicia monitoramento do sinal analógico}** |
| | O sistema inicia o monitoramento do sinal analógico, caso os limites de máximo e mínimo sejam extrapolados, será gerado um novo chamado. |

**Fonte: Autoria própria (2022)**

**Tabela 9 – Casos de uso [Verificar limites analógico]**

| **Nome do caso de uso** | Verificar limites analógico |
|---|---|
| **Descrição** | Verifica se o valor do sinal analógico extrapola os limites de máximo e mínimo previamente configurados |
| **Ator envolvido** | |
| **Pré-condições** | Limites máximo e mínimo configurados |
| **Pós-condições** | |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| | **{Verifica limites}** |

**(continua)**

**(continuação)**

| | |
|---|---|
| | O sistema verifica regularmente se os limites de máximo e mínimo previamente configurados foram ultrapassados pelo sinal analógico. (A1) |
| **Fluxo Alternativo** | |
| A1: em **{Verifica limites}** | Caso seja verificado que algum dos limites foi ultrapassado, será criado um novo chamado indicando hora e tipo de alarme (máximo ou mínimo). Retorna fluxo básico em **{Verifica limite}**. |

**Fonte: Autoria própria (2022)**

**Tabela 10 – Casos de uso [Criar chamados]**

| **Nome do caso de uso** | Criar chamado |
|---|---|
| **Descrição** | Cria chamados que deverão ser reconhecidos pelo supervisor |
| **Ator envolvido** | Operador |
| **Pré-condições** | |
| **Pós-condições** | Novo chamado criado aguardando reconhecimento |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| Clica em um botão físico de acordo com sua necessidade, botão vermelho indica a ocorrência de emergência na estação de trabalho, botão amarelo indica que o operador necessita de algum suporte por parte do supervisor e botão azul indica a necessidade de reposição de peça na estação de trabalho. | |
| | **{Cria chamado}** |
| | O sistema registra o chamado, informando data da ocorrência e tipo de chamado (de acordo com o botão que foi clicado) |
| **Fluxo Alternativo** | |
| | |

**Fonte: Autoria própria (2022)**

**Tabela 11 – Casos de uso [Reconhecer chamados]**

| **Nome do caso de uso** | Reconhecer chamado |
|---|---|
| **Descrição** | Reconhecer chamados previamente criados |
| **Ator envolvido** | Supervisor |

**(continua)**

**(continuação)**

| | |
|---|---|
| **Pré-condições** | Supervisor logado no sistema, chamado criado e aguardando reconhecimento |
| **Pós-condições** | Chamado reconhecido |
| **Fluxo Básico** | |
| **Usuário** | **Sistema** |
| Abre tela com o registro de chamados. | |
| Seleciona o chamado que deseja reconhecer. | |
| | **{Mostra detalhes do chamado}** |
| | O sistema mostra detalhes do chamado. |
| Confirma reconhecimento. | |
| | **{Reconhece chamado}** |
| | O sistema registra o reconhecimento do chamado. |

**Fonte: Autoria própria (2022)**

Conforme apresentado no diagrama e no detalhamentos dos casos de uso, o sistema irá envolver dois atores, supervisor e operador.

Enquanto a interação do operador ocorre unicamente através da botoeira o supervisor irá interagir ativamente com o sistema, assim, a primeira ação do mesmo será realizar sua autenticação de usuário, deste modo, liberando as demais funcionalidades do sistema, que são, visualização do estado da linha, visualização dos dados da produção, programação da produção, configuração dos limites analógicos e reconhecimento de chamados.

Como pode ser observado, os casos de uso podem ser agrupados em dois grupos, visualização e reconhecimento de ocorrências (chamados) nos postos de trabalho e programação e visualização de estado da produção.

No que diz respeito a programação e acompanhamento da produção, a ideia é permitir que o supervisor possa estabelecer uma meta para um determinado período de produção, por exemplo, um turno de trabalho. Desta forma, ao iniciar a produção será possível verificar se a quantidade de itens produzidos esta condizente com a meta previamente estabelecida.

Digamos que o supervisor deseje produzir sessenta itens entre as treze e as quatorze horas, ou seja, um item por minuto. Neste caso, as treze horas e trinta minutos será mostrado pelo sistema que deveriam ter sido produzidos trinta itens. Caso realmente tenham sido produzido tal quantidade de itens o sistema mostrará que a diferença entre o planejado e o produzido é igual a zero, caso contrário, será mostrado ao supervisor a diferença entre o planejado e produzido.

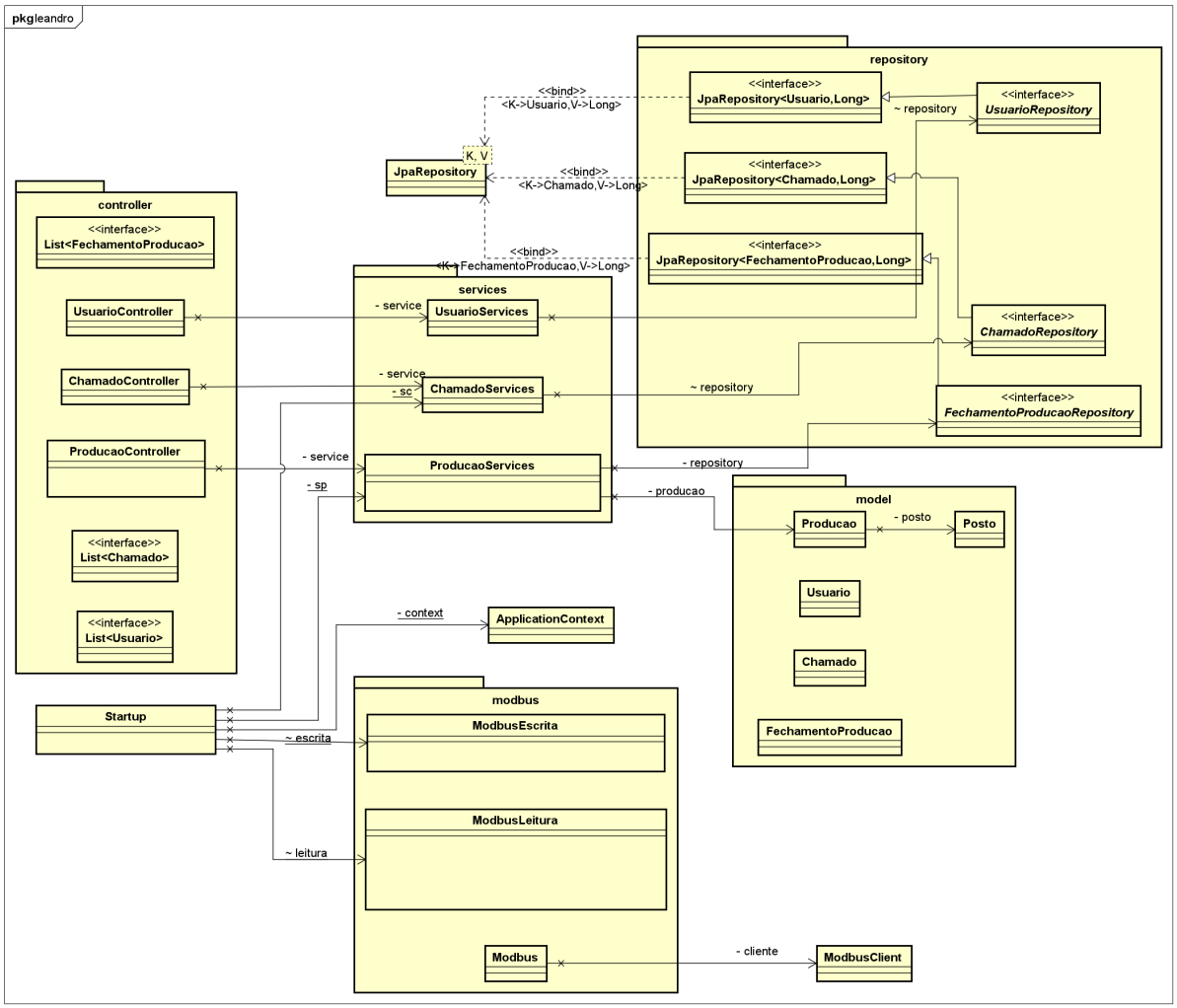
Portanto, o supervisor irá programar o horário de fim de produção e a quantidade de itens que deverão ser produzidos até este horário. Assim, ao iniciar a produção, o sistema irá apresentar a quantidade de itens produzidos em relação aos itens desejados naquele momento. A contagem de itens produzidos será realizada através de um sensor.

Além de programar a meta de produção e horário de término, o supervisor também poderá definir limites de máximo e mínimo de um sinal analógico. Esse sinal analógico, têm o intuito de demonstrar como seria o controle de um sinal analógico através do sistema. Este sinal analógico poderia ser, um medidor de temperatura, de nível ou pressão.

### 4.1.3 Diagrama de Classes

A partir do diagrama de classes (Figura 8) é possível obter uma perspectiva estrutural do sistema proposto. Para facilitar a visualização algumas informações foram omitidas, tais como, atributos e assinatura dos métodos das classes, ainda assim, é possível ter uma visão macro do relacionamento das classes que compõe o sistema.

**Figura 8 – Diagrama de Classes**




**Fonte: Autoria própria (2022).**

Conforme mostrado no diagrama, o sistema é composto por cinco pacotes, *(controller, services, repository, model e modbus)*, todos esses contidos em um pacote raiz. O pacote *controller* contém as classes responsáveis por controlar e responder as requisições REST reali-

zadas pela *Interface do usuário*. O pacote *service* contém as classes responsáveis por definir as regras de negócio do sistema. O pacote *repository* contém as interfaces com o banco de dados. O pacote *model* contém as classes responsáveis pela modelagem dos dados do banco de dados. Por fim, o pacote *modbus* contém as classes encarregadas pelo controle e processamento da comunicação com o Controlador Lógico Programável através do protocolo Modbus.

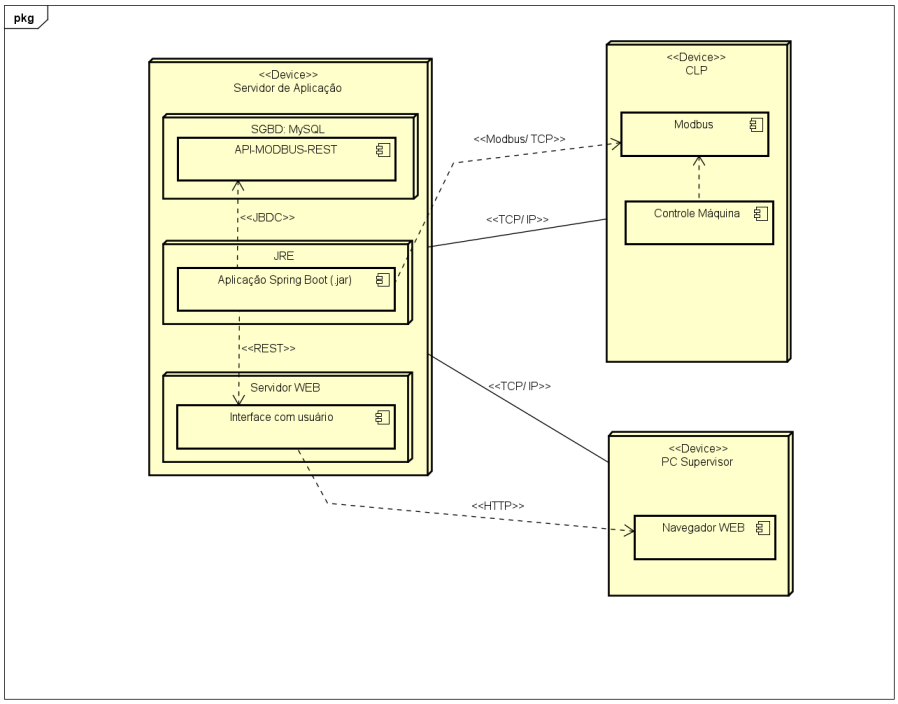
Ainda podemos sintetizar a interação entre as classes da seguinte forma, as classes *controller* instanciam suas respectivas classes *service*, que por sua vez implementam a suas respectivas interfaces *repository*, essas interfaces, através do *Spring Data JPA*, implementam em tempo de execução as classes do tipo *model*.

Finalmente, a classe *Startup*, que está presente no pacote raiz da aplicação, possui a finalidade de instanciar e executar a aplicação *Spring Boot*, assim como, instanciar e controlar as classes de leitura e escrita Modbus.

### 4.1.4 Diagrama de Implantação

Ainda sobe uma perspectiva estrutural do sistema, porém, neste caso relacionado a arquitetura física do mesmo, o diagrama de implementação (Figura 9) apresenta a proposta da arquitetura física do protótipo, que supõe pelo menos três dispositivos, um servidor da aplicação, o controlador industrial (neste caso um CLP) e um PC para acesso ao sistema.

**Figura 9 – Diagrama de Implantação**




**Fonte: Autoria própria (2022).**

Estarão presentes no **Servidor de Aplicação**, o servidor REST (Modbus2REST API), o sistema gerenciador de banco de dados e o servidor Web com a aplicação da interface com

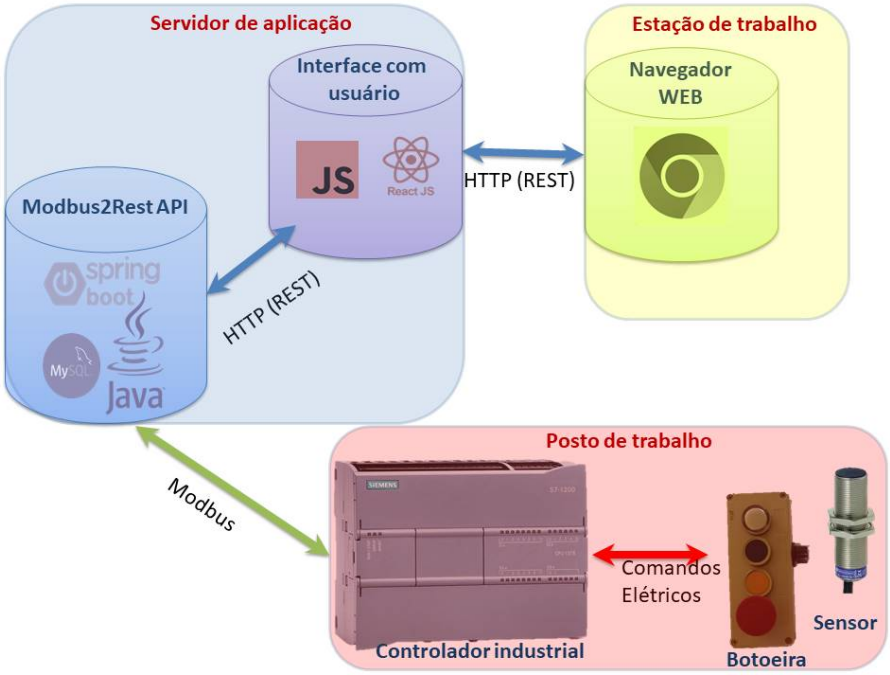
o usuário. Como é possível observar no diagrama, o sistema gerenciador de banco de dados (SGBD), o servidor REST e a aplicação de interface com o usuário estarão armazenadas em um único dispositivo, porém, caso fosse necessário, tais componentes poderiam ser alocados em dispositivos distintos, graças a opção por tecnologias compatíveis à arquiteturas distribuída.

O Controlador Industrial (CLP), representa o dispositivo industrial que será responsável por processar os sinais físicos (botões digitais e sinal analógico) e converte-los em mensagens Modbus, assim, estabelecendo a troca de dados entre CLP e servidor

Por fim, o PC Supervisor tem a finalidade de prover a interface do usuário, neste caso, o supervisor da linha de trabalho. Este equipamento realizará o acesso ao sistema através de um navegador WEB, via requisições HTTP à aplicação de Interface com Usuário.

Assim, diante dos elementos apresentados e com auxilio da Figura 10, é possível compreender o sistema sob uma perspectiva holística.

**Figura 10 – Visão geral (perspectiva estrutural)**



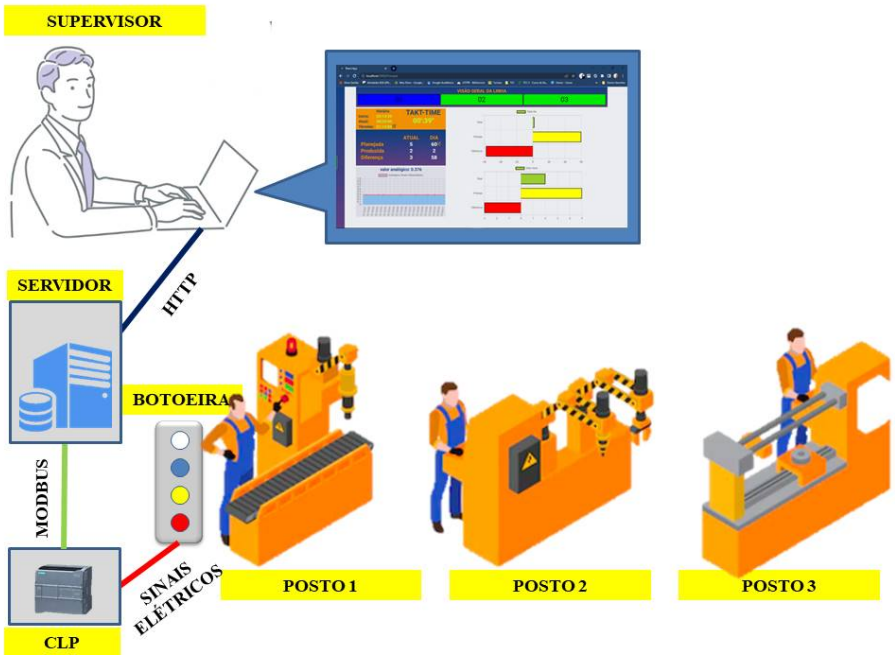

**Fonte: Autoria própria (2022).**

Do mesmo modo, é pertinente compreender o sistema sob a perspectiva dos usuários. Este sistema possui dois tipos de interface (Figura 11), uma para os operadores e uma para o supervisor da linha de montagem.

A interface para o operador, está relacionada aos elementos físicos do sistema, os quais, são controlados a partir do controlador industrial ou simplesmente CLP (Controlador Lógico Programável). O propósito do CLP é tratar os sinais elétricos e convertê-los em mensagens Modbus.

Uma característica relevante do CLP é a variedade de tipos sinais que ele pode tratar, sendo possível processar tanto sinais digitais, quanto sinais analógicos. Os sinais digitais são aqueles que representam uma quantidade finita de estados, normalmente tais sinais são discre-

**Figura 11 – Visão geral (perspectiva dos usuários)**




**Fonte: Autoria própria (2022).**

tos, ou seja, representam dois estados (ligados e desligado). Por sua vez, os sinais analógicos são aqueles que a variação no tempo representa a variação de uma grandeza física, por exemplo, pressão, temperatura, nível e etc. No trabalho em questão, com o objetivo de representar esta esta variedade, foram utilizados sinais analógicos e sinais digitais.

O dispositivo físico de interface para operador é baseado numa botoeira industrial (Figura 12-a), que possui três botões, parada de linha (vermelho), alerta para o supervisor da linha (amarelo) e alerta para o setor logístico (azul), além de um Led branco que indica a existência de um chamado ativo no respectivo posto de trabalho.

- **Botão emergência:** Ao pressionar este botão, a linha é parada;
- **Alerta para o líder da linha:** Este botão indica ao líder que o posto de trabalho necessita de algum tipo de suporte, através da abertura de um chamado;
- **Alerta para o setor logístico:** Este botão indica ao setor de logística que o posto de trabalho necessita de reposição de insumos, através da abertura de um chamado;

Além dos botões, a botoeira também contém um potenciômetro (Figura 12-c), pelo qual é possível modular um sinal na entrada analógico (0 à 10 Vcc) do CLP, que poderá ser observado através da interface do supervisor. Ademais, ao final da linha de montagem existe um sensor indutivo (Figura 12-b) com o objetivo de realizar a detecção dos produtos acabados, assim, permitindo a contagem automática dos mesmos. O sensor indutivo é acionado através da proximidade de materiais condutores de energia elétrica de natureza metálica, portanto, consideramos que a produto da linha de montagem é metálico.

**Figura 12 – Dispositivos físicos**

**(a)** **(b)** **(c)**

**Fonte: Autoria própria (2022).**

Assim, os botões, o LED e o sensor indutivo, são os sinais discretos que o CLP trata, enquanto que o potenciômetro é o sinal analógico tratado pelo CLP.

Por sua vez, a interface para o supervisor da linha de montagem tratasse da aplicação acessada via navegador WEB. Esta aplicação permite que seja realizado um planejamento prévio e acompanhamento do andamento da produção. Assim, podem ser previamente definidos, o momento de fim de turno (hora de fim) e quantidade que deverá ser produzida (total desejado), enquanto as informações de hora de início, total produzido e a diferença entre produzido e planejado podem ser acompanhado durante o andamento da produção. A partir dessa interface também é possível verificar e reconhecer chamados criados pelos operadores, além de indicar a existência de um chamado não reconhecido a partir de uma representação gráfica da linha de montagem. Por exemplo, caso o operador do posto de trabalho número um pressione o botão azul (alerta de falta de insumo), este posto será retratado na cor azul na representação gráfica da linha de montagem, o mesmo vale para os demais botões e postos de trabalho.

## 4.2 Implementação do Sistema

Esta seção apresenta os detalhes de maior relevância da implementação do sistema, com detalhes da aplicação Modbus2REST API (Servidor REST) e da Interface com usuário, banco de dados e programa do CLP.

### 4.2.1 Implementação da aplicação Modbus2REST API (Servidor REST)

Dentre os itens que compõe o sistema pode se dizer que o servidor REST é o componente de maior relevância às intenções deste trabalho, afinal, será neste componente que as mensagens Modbus serão convertidos para uma tecnologia mais usual no que diz respeito a sistemas de informação atuais, que no caso deste trabalho serão, mensagens HTTP sobe uma arquitetura REST.

Neste trabalho optou-se pela utilização do *framework Spring Boot* para desenvolvimento do servidor REST, esta escolha se deve tanto pela sua ampla utilização pela comunidade de desenvolvedores quanto pela agilidade que o mesmo prove ao processo de desenvolvimento de aplicações Java WEB.

### 4.2.2 Inicialização do projeto do Servidor REST

Para inciar a aplicação foi utilizada a plataforma *Spring Initializr* (VMWARE, 2022). A partir dessa plataforma é possível gerar um projeto MAVEN, que por sua vez, possibilita a construção do projeto *Spring* a partir de uma estrutura pré-configurada. Dentre as configurações possíveis se destacam, a versão do *Spring*, a versão do java, o tipo de empacotamento e as dependências do projeto. A seguir as configurações do projeto podem ser visualizadas.

Primeiramente foram definidas as configurações referentes ao *Spring Boot*.

**Listagem 1 – Configurações iniciais - Configurações Spring Boot**

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
       www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven
       .apache.org/xsd/maven-4.0.0.xsd">
4    <modelVersion>4.0.0</modelVersion>
5    <parent>
6      <groupId>org.springframework.boot</groupId>
7      <artifactId>spring-boot-starter-parent</artifactId>
8      <version>2.6.3</version>
9      <relativePath/> <!-- lookup parent from repository -->
10   </parent>
```

**Fonte: Autoria própria (2022).**

A seguir foram definidas as informações do projeto como, nome, versão, artifactID e groupId.

**Listagem 2 – Configurações iniciais - Informações do projeto,**

```
<groupId>com.leandro</groupId>
<artifactId>API_MODBUS_REST_REV1</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>API_MODBUS_REST_REV1</name>
```

**Fonte: Autoria própria (2022).**

Mais adiante foi definida a versão do Java, que neste caso foi o JAVA-11.

**Listagem 3 – Configurações iniciais - Versão do Java**

```
<properties>
  <java.version>11</java.version>
</properties>
```

**Fonte: Autoria própria (2022).**

Por fim, foram definidas as dependências do projeto, dentre as quais podem ser citadas, **spring-boot-starter-data-rest**, **spring-boot-starter-web**, **spring-boot-starter-data-jpa** e **mysql-connector-java**.

A dependência **spring-boot-starter-data-rest** está relacionada as bibliotecas responsáveis por disponibilizar os recursos da arquitetura REST ao projeto em questão.

**Listagem 4 – Dependências (REST)**

```
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
  </dependency>
```

**Fonte: Autoria própria (2022).**

Já a dependência **spring-boot-starter-web** está relacionada as bibliotecas de construção de aplicações web, incluindo RESTful.

**Listagem 5 – Dependências (WEB)**

```
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
```

**Fonte: Autoria própria (2022).**

A dependência **spring-boot-starter-data-jpa** diz respeito as bibliotecas que permitem acessar e persistir dados entre objeto Java e um banco de dados relacional.

**Listagem 6 – Dependências (JPA)**

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

**Fonte: Autoria própria (2022).**

Por fim, a dependência **mysql-connector-java** está relacionada ao driver JDBC(*Java Database Connectivity*) para MySQL (MYSQL, 2022).

**Listagem 7 – Dependências (JDBC MySQL)**

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
```

**Fonte: Autoria própria (2022).**

Além das bibliotecas instaladas a partir das definições das dependências do projeto também foi necessária a instalação da biblioteca **EasyModbusJava.jar** (ROSSMANN-ENGINEERING, 2022), esta biblioteca é responsável por efetuar a integração entre plataforma do protocolo Modbus com a biblioteca Java.

### 4.2.3 Detalhes da implementação do Servidor REST

Com as definições inciais do projeto devidamente realizada, foi possível implementar a aplicação propriamente dita. Conforme apresentado na figura 8, o projeto é formado por cinco pacotes específicos inclusos em um pacote raiz, que além de conter os demais pacotes também possui a classe **Startup.java**.

Dentre as classes presentes no projeto, a classe **Startup.java** se destaca por possuir algumas funcionalidades relevantes ao propósito do trabalho em questão, afinal, é nesta classe que o método **main** do Java é implementado e como pode ser observado no trecho de código a seguir (Listagem 8), duas *threads* são iniciadas neste método, **t1**, onde a aplicação Spring é iniciada e **t2**, onde é executada a interface Modbus.

Conforme trecho de código a seguir (Listagem 9), ao iniciar a aplicação Spring na *thread* **t1**, o contexto da aplicação é salvo no atributo **context**, que por sua vez, é utilizado para obter os *beans*[1] das classes **ChamadoService** e **ProducaoService**, *beans* que terão seus propósitos discutidos mais adiante.

[1] Basicamente, um *bean* é uma classe Java escrita de acordo com uma convenção

**Listagem 8 – Classe Startup (método main)**

```
public static void main(String[] args) {
  escrita = new ModbusEscrita(0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
  leitura = new ModbusLeitura(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0);
  new Thread(t1).start();
    new Thread(t2).start();
}
```

**Fonte: Autoria própria (2022).**

**Listagem 9 – Classe Startup (thread t1)**

```
private static Runnable t1 = new Runnable() {
      public void run() {
        context = SpringApplication.run(Startup.class, "");
      springStarted = true;
      System.out.println(springStarted);
      sc = context.getBean(ChamadoServices.class);
      sp = context.getBean(ProducaoServices.class);
      }
  };
```

**Fonte: Autoria própria (2022).**

Enquanto isso, a *thread* **t2** (Listagem 10) é responsável por instanciar a classe **Modbus**, a qual é incumbida de realizar a interface entre o servidor REST e o controlador industrial. Para alcançar seu objetivo esta *thread* executa os métodos **interfaceDadosLeitura**, **interfaceDadosEscrita** e **criarChamado**, ciclicamente a cada 800 ms.

O método **interfaceDadosLeitura** (Listagem 11) realiza a leitura dos dados do controlador industrial (parte inteira e fracionária da leitura do sensor analógico, estado do posto de trabalho e estado do sensor de contagem de peças) e registra tais dados no **bean** da classe **ProducaoService**. Desta forma, disponibilizando tais dados para operações do servidor REST.

**Listagem 10 – Classe Startup (thread t2)**

```
private static Runnable t2 = new Runnable() {
      public void run() {
        while(true) {
          try {
          Modbus modbus = new Modbus();
          System.out.println("conectado classe principal");
          while(true) {

            holdingRegistersLeitura = interfaceDadosLeitura(
                holdingRegistersLeitura,
                modbus);
            holdingRegistersEscrita = interfaceDadosEscrita(
                holdingRegistersEscrita,
                modbus);
                criaChamado();

                Thread.sleep(800);
          }
        } catch (InterruptedException e) {
          System.out.println("nao conectado classe principal");
          e.printStackTrace();
        }
        }
      }
};
```

**Fonte: Autoria própria (2022).**

**Listagem 11 – Classe Startup (interfaceLeitura)**

```
private static int[] interfaceDadosLeitura(
              int[] holdingRegistersLeitura,
              Modbus modbus) {

  if (holdingRegistersLeitura != null)
    leitura.SetAll(holdingRegistersLeitura);

  try {
    holdingRegistersLeitura = modbus.readHoldingRegister();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }

    sp.setModbusDataExchance(
        leitura.getSensorAnalogicoParteInteira(),
        leitura.getSensorAnalogicoParteDecimal(),
        leitura.getStatusPosto(),
        leitura.getStatusSensor01());

    return holdingRegistersLeitura;
}
```

**Fonte: Autoria própria (2022).**

Por sua vez, o método **interfaceDadosEscrita** (Listagem 12) realiza a escrita dos dados no controlador, neste caso, somente a variável **limparChamado** está sendo trocada entre controlador e a servidor REST.

**Listagem 12 – Classe Startup (interfaceEscrita)**

```
1  private static int[] interfaceDadosEscrita(
2                 int[] holdingRegistersEscrita,
3                 Modbus modbus) {
4    holdingRegistersEscrita[7] = escrita.getComandoLimparChamados();
5    if (leitura.getStatusPosto() == 0) escrita.
     setComandoLimparChamados(0);
6    try {
7      modbus.whriteHoldingRegister(20, holdingRegistersEscrita);
8    } catch (IOException e) {
9      // TODO Auto-generated catch block
10     e.printStackTrace();
11   }
12
13   return holdingRegistersEscrita;
14 }
```

**Fonte: Autoria própria (2022).**

Por fim, o método **criarChamado** (Listagem 13) é responsável por criar novos chamados, para isso, verifica se houve mudança no estado do posto de trabalho ou se a leitura do sensor analógico ultrapassou os limites superior ou inferior previamente definidos.

Com relação a variável responsável por representar o estado do posto de trabalho, são possíveis valores de zero a três, onde, valor igual a zero, indica que não existe chamado pendente, valor igual a um, indica chamado de emergência presente, valor igual a dois, indica chamado para o líder da linha e finalmente quando o valor desta variável for igual a três indica a existência de chamado do tipo logística.

É importante observar que todas as variáveis trocadas entre servidor REST e o controlador industrial são do tipo inteiro, isto se deve a própria estrutura do protocolo **Modbus**, no caso dos métodos **interfaceDadosLeitura** e **interfaceDadosEscrita** são retornados arrays de inteiros que representam as variáveis trocadas entre os componentes. Desta forma, o valor do sinal analógico é obtido através de duas variáveis, uma para representar a parte inteira e outra para representar a parte fracionária deste sinal. Para aprofundar o entendimento do processo de troca de dados com o controlador industrial é necessário analisar o pacote **Modbus**, afinal, neste pacote que são instanciadas as classes da biblioteca **EasyModbusJava.jar**.

O pacote **Modbus** possui três classes, **Modbus**, **ModbusEscrita** e **ModbusLeitura**. As classes **ModbusEscrita** e **ModbusLeitura** possuem o objetivo de agrupar as variáveis de leitura e escrita, respectivamente, ou seja, tais classes não implementam funcionalidades especiais, na verdade, são utilizadas para facilitar a manipulação dos dados trocados entre a aplicação e o controlador industrial.

**Listagem 13 – Classe Startup (criarChamados)**

```
1  private static void criaChamado() {
2    if(leitura.getStatusPosto() >=1 && leitura.getStatusPosto() <=3)
     {
3        if(!detectorBordaStatusPosto) {
4          sc.create(new Chamado(1, leitura.getStatusPosto()));
5          detectorBordaStatusPosto = true;
6        }
7      } else {
8      detectorBordaStatusPosto = false;
9    }
10
11   if(sp.getProducao().getAnalogico() < sp.getProducao().getNivelMin
     ()) {
12     if(!detectorBordaAlarmeAnalogicoMin) {
13         sc.create(new Chamado(0, 4));
14         detectorBordaAlarmeAnalogicoMin = true;
15       }
16   } else {
17     detectorBordaAlarmeAnalogicoMin = false;
18   }
19
20   if(sp.getProducao().getAnalogico() > sp.getProducao().getNivelMax
     ()) {
21     if(!detectorBordaAlarmeAnalogicoMax) {
22         sc.create(new Chamado(0, 5));
23         detectorBordaAlarmeAnalogicoMax = true;
24       }
25   } else {
26     detectorBordaAlarmeAnalogicoMax = false;
27   }
28 }
```

**Fonte: Autoria própria (2022).**

Por sua vez, a classe **Modbus** é responsável por instanciar a classe **ModbusClient** e assim possibilitar a conexão via protocolo Modbus. Após a classe **ModbusClient** estar devidamente instanciada é realizada a conexão Modbus propriamente dita, para isso, foi configurado (Listagem 14) um *timeout* de 1500 ms, endereço IP e porta do servidor **modbus**.

**Listagem 14 – Classe Modbus (construtor)**

```
1  public class Modbus {
2
3    private ModbusClient cliente;
4    private boolean conectado = false;
5
6    public Modbus(){
7      cliente = new ModbusClient();
8      cliente.setConnectionTimeout(1500);
9      while(!conectado) {
10       try {
11         cliente.Connect("192.168.0.100", 502);
12         System.out.println("modbus conectado");
13         conectado = true;
14       } catch (IOException e) {
15         // TODO Auto-generated catch block
16         e.printStackTrace();
17       }
18     }
19   }
```

**Fonte: Autoria própria (2022).**

Além das operações realizadas no construtor dois métodos são realizados pela classe **Modbus**, **readHoldingRegister** e **writeHoldingRegister**.

O método **readHoldingRegister** (Listagem 15) executa o método **ReadHoldingRegisters** do objeto **ModbusClient** instanciado no construtor, esse método possui dois parâmetros, endereço inicial e quantidade de registros. É pertinente discutir o primeiro parâmetro desse método, logo que o mesmo pode gerar confusão. Este endereço está relacionado ao index no *array* de registros previamente reservados no controlador industrial para comunicação Modbus. No caso deste projeto, foram reservados vinte registros a partir do index zero.

**Listagem 15 – Classe Modbus (readHoldingRegister)**

```
1   public int[] readHoldingRegister() throws UnknownHostException,
     IOException {
2     try {
3       return cliente.ReadHoldingRegisters(0, 20);
4     } catch (ModbusException | IOException e) {
5       // TODO Auto-generated catch block
6       e.printStackTrace();
7       cliente.Disconnect();
8       cliente.Connect("192.168.0.100", 502);
9     }
10    return null;
11  }
```

**Fonte: Autoria própria (2022).**

Similarmente, o método **writeHoldingRegister** (Listagem 16) executa o método **WriteMultipleRegisters** também do objeto **ModbusClient**, esse método, do mesmo modo que o método anterior, possui dois parâmetros, desta vez porém, um representa o endereço inicial de escrita enquanto que o outro parâmetro recebe um *array* com os valores dos registros que deverão ser carregados no controlador.

**Listagem 16 – Classe Modbus (writeHoldingRegister)**

```
1  public void whriteHoldingRegister(int address, int[] value) throws
    UnknownHostException, IOException {
2    try {
3      cliente.WriteMultipleRegisters(address, value);
4    } catch (ModbusException | IOException e) {
5      // TODO Auto-generated catch block
6      e.printStackTrace();
7      cliente.Disconnect();
8      cliente.Connect("192.168.0.100", 502);
9    }
10 }
```

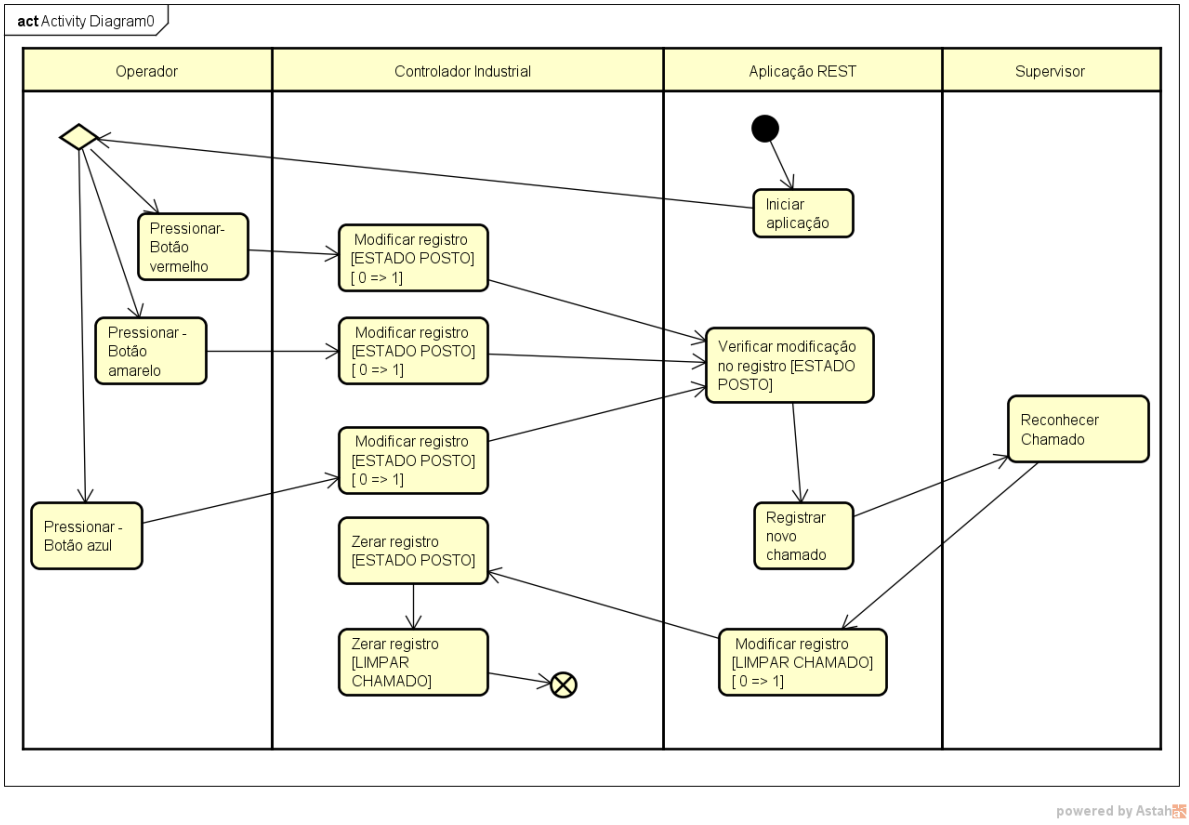
**Fonte: Autoria própria (2022).**

Discutindo a respeito dos registros manipulados, inicialmente foram reservados vinte registros de leitura e vinte de escrita, porém, no decorrer da implementação do servidor foi verificado que somente seriam necessários quatro registros de leitura e uma de escrita. Assim, os registros lidos são, parte inteira do sensor analógico, parte decimal do sensor analógico, estado do posto de trabalho e estado do sensor digital responsável pela contagem de peças. Enquanto que o registro que é escrito no controlador corresponde a função de limpeza de chamado.

Deste modo, quando o operador do posto de trabalho pressiona um botão, o controlador modifica a valor do registro que representa o estado do posto de trabalho, por sua vez, esse registro é lido pela servidor REST e disponibilizado para interface de operação. E finalmente quando o supervisor realizar o reconhecimento do chamado o registro de limpar chamado será escrito no controlador que consequentemente irá zerar o registro de estado do posto, indicando que não existe chamado pendente. A Figura 13 ilustra, através de um diagrama de atividades, o fluxo de controle dos chamados.

Além da interface **Modbus-Rest** também é pertinente discutir algumas características relevantes da implementação da servidor REST, dentre as quais, se destaca a manipulação dos dados de produção que são apresentados ao supervisor. Conforme proposto inicialmente, além do processamento de chamados o sistema também irá proporcionar ao supervisor da linha de montagem uma interface de visualização dos indicadores de produção.

**Figura 13 – Diagrama de Atividades - Chamado**




**Fonte: Autoria própria (2022).**

As informações apresentadas ao supervisor são, horário de início e fim de produção, produção planejada, produção realizada, diferença entre planejado e realizada, *takt-time*[2], sinal analógico e estado do posto de trabalho. De acordo com o discutido no final da sub-seção 4.1.2, no que diz respeito ao planejamento e controle da produção, os dados são apresentados ao supervisor tanto em uma perspectiva diária quanto em um perspectiva instantânea, desta forma, existem três atributos para cada uma dessas perspectivas, total planejado, total produzido e diferença entre planejado e produzido. Por sua vez, o atributo *takt-time* representa o tempo disponível para produção de cada item, por exemplo, digamos que a meta seja produzir dez itens em um hora, neste caso o *takt-time* inicial seria de seis minutos e caso ocorra atraso ou antecipação na produção esse valor é atualizado.

A manipulação dos dados de produção ocorre nas classes, **ProducaoModel**, **ProducaoService** e **ProducaoController**. Naturalmente, a classe **ProducaoModel** é responsável pela modelagem dos dados de produção, portanto, esta classe basicamente contém os atributos e respectivos *getter* e *setters* relacionados à tais dados. A classe **ProducaoService** contém as regras de negócio relacionadas aos dados de produção. Por fim, a classe **ProducaoControl-**

[2] Takt-time é o ritmo de produção necessário para atender a um determinado nível considerado de demanda, dadas as restrições de capacidade da linha ou célula. Concretamente, o takt-time é o ritmo de produção alocado para a produção de uma peça ou produto em uma linha ou célula (ALVAREZ; JR, 2001).

**Ier** (Listagem 17), responsável por controlar as requisições REST possui uma particularidade em relação ao que normalmente verifica-se em aplicações deste tipo. Esta particularidade está relacionada ao uso da tecnologia SSE (*Server-Sent Events*), a qual, permite ao *client* que no trabalho em questão é aplicação de interface com usuário receber atualizações automáticas do *server* (servidor REST).

O uso desta tecnologia se fez necessário para realização da atualização das informações na interface de visualização utilizada pelo supervisor. O funcionamento deste método, pode ser resumido da seguinte forma, quando o *client* realiza a requisição na URL (*Uniform Resource Locator*) encarregada por transmitir as informações da produção é aberta uma conexão SSE que envia uma mensagens com as informações de produção a cada dois segundos.

**Listagem 17 – Classe ProducaoController (SSE)**

```
@GetMapping(value="/producao/emitter")
public SseEmitter eventEmitter() {
    SseEmitter emitter = new SseEmitter(-1L);
    //create a single thread for sending messages asynchronously
    ExecutorService executor = Executors.newSingleThreadExecutor();
    executor.execute(() -> {
        try {
          while (true) {
            if(true) {
              emitter.send(service.getProducao());
            }
              Thread.sleep(2000);
            }
        } catch(Exception e) {
             emitter.completeWithError(e);
        } finally {
             emitter.complete();
        }
    });
    executor.shutdown();
    return emitter;
}
```

**Fonte: Autoria própria (2022).**

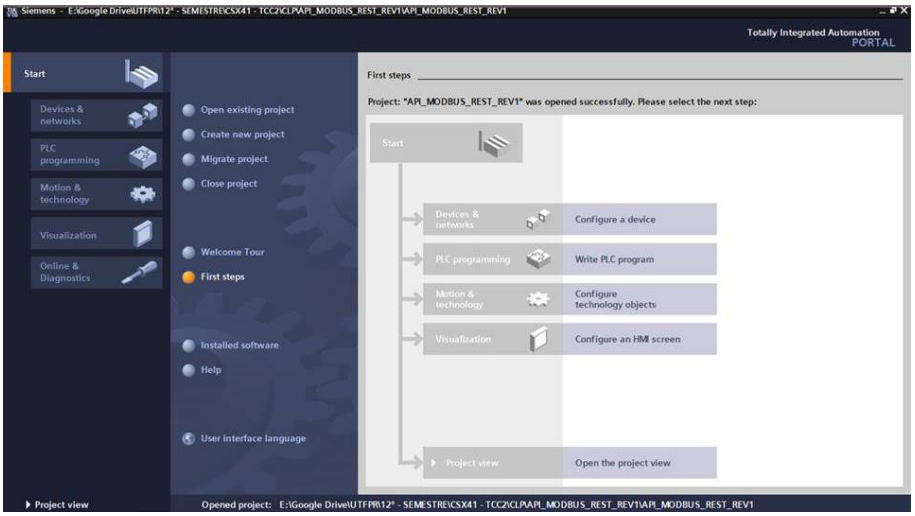
### 4.2.4 Implementação do programa do CLP

Como já foi mencionado algumas vezes, faz parte do objetivo central deste trabalho desenvolver uma aplicação capaz de promover a interface entre dispositivos industriais com comunicação Modbus e sistemas de informação compatíveis aos conceitos da Indústria 4.0, assim, é fundamental analisar o funcionamento deste componente. No protótipo desenvolvido para este trabalho o dispositivo industrial utilizado trata-se de um CLP, porém, é importante ressaltar

que a proposta deste trabalho é desenvolver um sistema compatível a qualquer dispositivo que possa se comunicar através de Modbus.

O CLP utilizado trata-se de item da linha S7-1200 fabricado pela Siemens AG. Este dispositivo é principalmente utilizado em sistemas automatizados industriais de baixo e médio porte, ainda assim, também é possível encontrá-lo em aplicações não industriais. Os programas deste tipo de equipamento são desenvolvidas através da plataforma TIA Portal e para o desenvolvimento do protótipo em questão foi utilizada a versão v16 desta plataforma (Figura 14).

**Figura 14 – TIA Portal**




**Fonte: Autoria própria (2022).**

Discutindo os detalhes do programa do CLP, que foi desenvolvido em linguagem *Ladder* que é comumente utilizada ao programar equipamentos deste tipo, foram criadas duas FB (*Functions Blocks*)[3], **ControleAnalogico** para o controle do sinal analógico e **StatusPosto** para o controle de estado posto de trabalho, além do uso do OB1 (*Organization Block 1*)[4]. Além das funções, também foram definidos quatro DB's (*Data Block*)[5], **ControleAnalogico_DB**, **DB_ModbusConexao**, **DB_ModbusDados** e **StatusPosto_DB**, responsáveis por, armazenar os dados da instância da FB **ControleAnalogico**, armazenar os dados da conexão Modbus, armazenar os dados trocados entre os componentes e armazenar os dados da instância da FB **StatusPosto**, respectivamente.

A respeito da comunicação Modbus implementada no controlador, foi utilizado um bloco padrão **MB_SERVER** (Figura 15) que gera um servidor Modbus no dispositivo. Este bloco é instanciado na OB1 e contém três parâmetros de entrada e quatro de saída, dos quais foram utilizados apenas dois parâmetros de entrada. O **MB_HOLD_REG** que recebe a àrea de me-

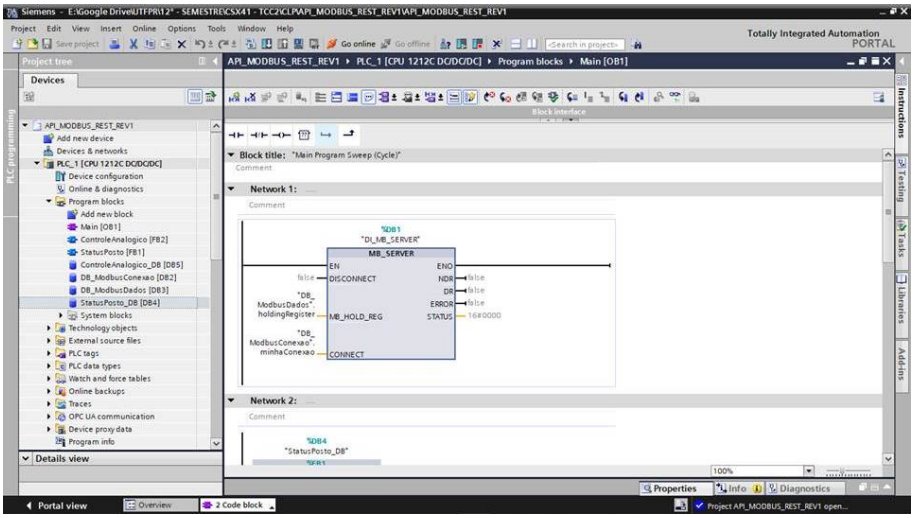
[3] FB's são um tipo de função que armazena dados em suas instancias (SIEMENS, 2021).

[4] OB's são as funções que realizam interfaces do programa do usuário com o sistema operacional do CLP, no caso da OB1 é executada a cada ciclo de varredura do CLP (SIEMENS, 2021).

[5] DB's como o nome sugere, são blocos que armazenam dados (SIEMENS, 2021).

mória que contém os dados trocados entre a aplicação REST e o CLP, por sua vez, o parâmetro **CONNECT** recebe a àrea de memória que contém os dados da conexão Modbus.

**Figura 15 – Detalhe bloco de comunicação Modbus**




**Fonte: Autoria própria (2022).**

A àrea recebida pelo parâmetro **MB_HOLD_REG** é a estrutura contida pela **DB_ModbusDados**, esta estrutura contém dois *arrays*, leitura e escrita, com vinte inteiros cada.

### 4.2.5 Implementação da Aplicação de Interface com o usuário

Visto que o objetivo deste trabalho é principalmente relacionado à interface entre o servidor REST e o controlador industrial, não se considerou relevante abordar maiores detalhes da aplicação de interface com o usuário, ainda assim, é interessante apresentar uma visão geral deste componente.

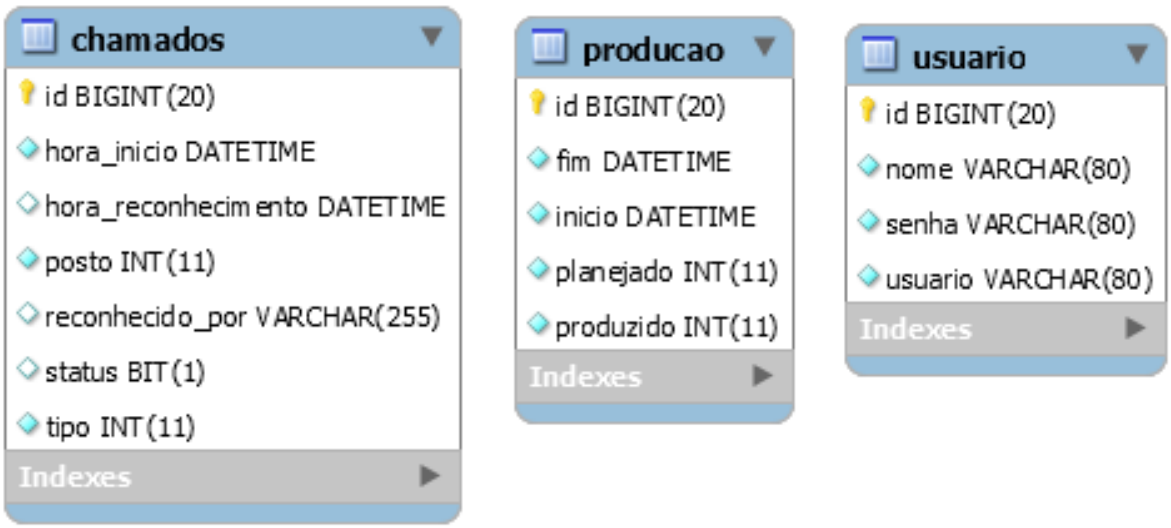
O intuito deste componente é fornecer uma interface amigável ao supervisor da linha onde o mesmo pode monitorar o estado dos postos de trabalho e indicadores de produção, programar meta de produção e reconhecer chamados. Para alcançar seus objetivos este componente foi desenvolvido em Javascript a partir da biblioteca React. Assim, resultando em uma aplicação plenamente compatível à arquitetura REST.

Um detalhe que vale ser mencionado é que além das bibliotecas usuais em aplicações React também foi necessário o uso da biblioteca **react-chartjs-2**, a qual, facilitou a construção dos gráficos presentes na aplicação.

### 4.2.6 Implementação do Banco de Dados

O banco de dados implementado para o sistema é extremamente simples, possuindo apenas três tabelas (chamados, produção e usuário) que não possuem qualquer relacionamento entre si (Figura 16).

**Figura 16 – Diagrama de Entidade Relacionamento**




**Fonte: Autoria própria (2022).**

# 5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir das implementações descritas no capítulo anterior. Em síntese, a implementação dos componentes do sistema produziu dois artefatos acessíveis aos usuários (operador e supervisor), uma aplicação acessível via navegador *WEB* e interface física que dentre outras coisas, possui a botoeira utilizada pelo operador para realizar os chamados.

## 5.1 Interface física

### 5.1.1 Visão geral

De acordo com o que foi inicialmente proposto, foi construído um protótipo para representar um posto de trabalho de uma linha de montagem em um contexto industrial. Para tal, foram utilizados os seguintes materiais (Figura 17):

**Figura 17 – Interface física**

**(a)** **(b)**


**Fonte: Autoria própria (2022).**

1. Botoeira de quatro furos, JNG;
    a) Botão de pulso azul, XA2EA61, Schneider;
    b) Botão de pulso amarelo, XB5AA51, Schneider;
    c) Botão de emergência tipo cogumelo, XA2ES542, Schneider;
    d) Sinalizador LED de painel 24Vdc, SLDS24, Steck;
    e) Potenciômetro, 1k Ohms;
2. Cabo Ethernet 2 m, CAT 5;

3. Controlador lógico programável, 6ES7 212-1AE40-0XB0, Siemens;
4. Fonte 24Vdc-10A, NDR-240-24, Mean Well;
5. Sensor indutivo M18, XS2 M18MB250, Telemecanique;
6. Plug de tomada;

### 5.1.2 Operação

A operação da interface física se dá através dos botões (onde o operador notifica as ocorrências da linha a partir da criação de chamados), do potenciômetro (que controla o sinal analógico) e do sensor indutivo (que realiza a contagem de peças produzidas).

Assim, para criar um novo chamado basta que o operador pressione o botão que corresponde ao tipo de ocorrência que deseja informar. A partir do momento que o chamado é criado, o sinalizador LED ficará piscando confirmando que o chamado foi devidamente criado, só será desligado quando o supervisor reconhecer o chamado.

Relembrando,

- Botão vermelho, indica parada de emergência;
- Botão amarelo, indica que o operador necessita de suporte da supervisão;
- Botão azul, indica a necessidade de reposição de peça;

Além dos botões e do sinalizador LED, também está presente na botoeira um potenciômetro que tem o objetivo de demonstrar o controle de um sinal analógico a partir do sistema desenvolvido.

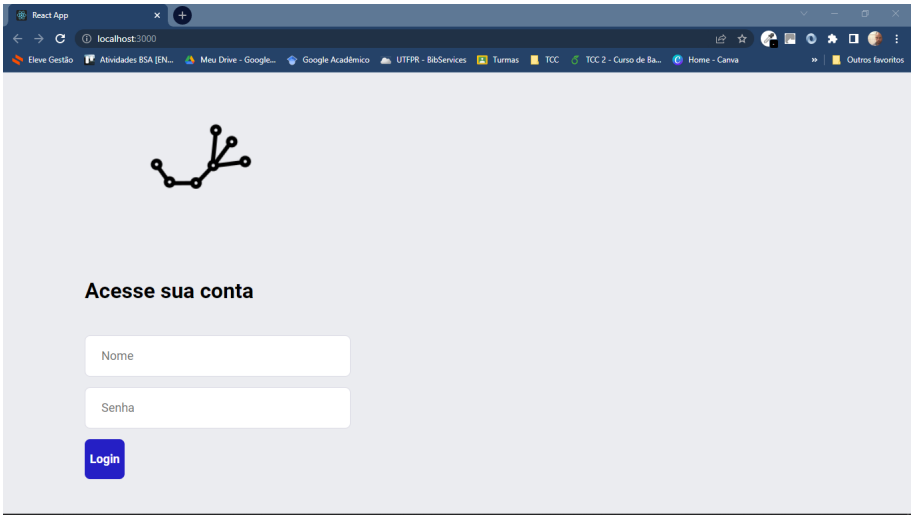
Por fim, o sensor indutivo realiza a contagem de peças a partir da detecção das mesmas, ou seja, este sensor deve ser instalado ao final da linha de montagem e desta forma, a cada peça que o sensor detectar o contador é incrementado em uma unidade.

## 5.2 Interface com o usuário

### 5.2.1 Visão geral

É a partir da aplicação de interface com o usuário que o supervisor é capaz de interagir com o sistema. A primeira interação entre o supervisor e o sistema é a autenticação de usuário (Figura 18).

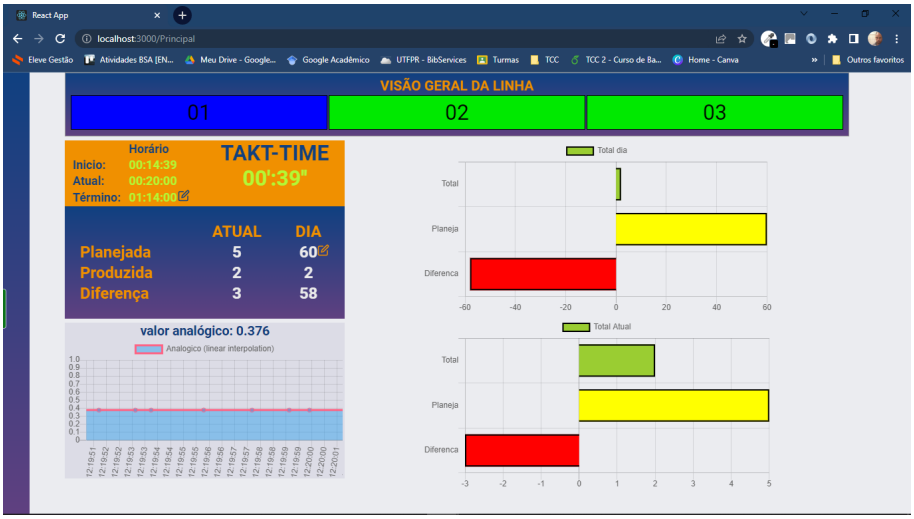
**Figura 18 – Tela de login**




**Fonte: Autoria própria (2022).**

Após realizar a autenticação, o usuário será direcionado a tela principal da aplicação (Figura 19). Esta tela funciona como *Dashboard* do sistema, onde é possível visualizar o estado dos postos da linha de montagem, os dados de produção e o gráfico de tendência do sinal analógico. Além dessas informações esta tela também contém um menu de navegação pelo qual o supervisor pode abrir as demais telas do sistema (Figura 20).

**Figura 19 – Tela Principal**

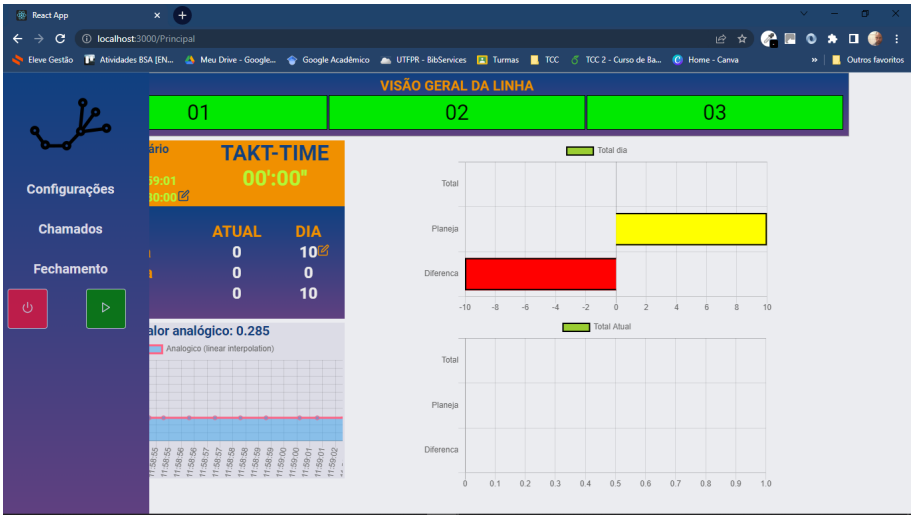



**Fonte: Autoria própria (2022).**

No que diz respeito aos dados de produção, são apresentados, os dados de contagem da produção (planejado, produzido e diferença) tanto momentâneos (atual) quanto diários (dia), os dados de horário da produção (início, atual e fim) e o *Takt-time* da produção.

O *takt-time* é recalculado a cada novo ciclo produtivo, ou seja, o takt-time é igual ao tempo total de produção restante dividido pelo total de peças que faltam para que a meta programada seja alcançada.

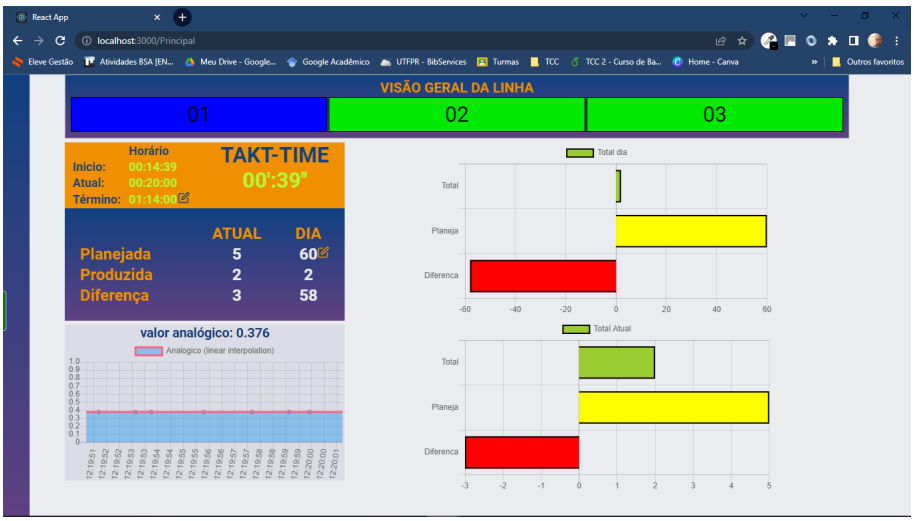
**Figura 20 – Tela de Principal - Detalhe menu**




**Fonte: Autoria própria (2022).**

Com relação a visualização do estado da linha de montagem, são representados três postos de trabalho, porém, a partir do protótipo desenvolvido somente é possível demonstrar o funcionamento de um posto de trabalho, isto devido à limitações de *I/O's* do controlador utilizado. O estado do posto de trabalho é representado pelas mesmas cores dos botões de criação de chamado, vermelho representa parada de emergência, amarelo solicitação de auxílio e azul indica necessidade de reposição de peça, por sua vez, a não existência de nenhum chamado é representada pela cor verde (Figura 21).

**Figura 21 – Tela de Principal - Detalhes produção**




**Fonte: Autoria própria (2022).**

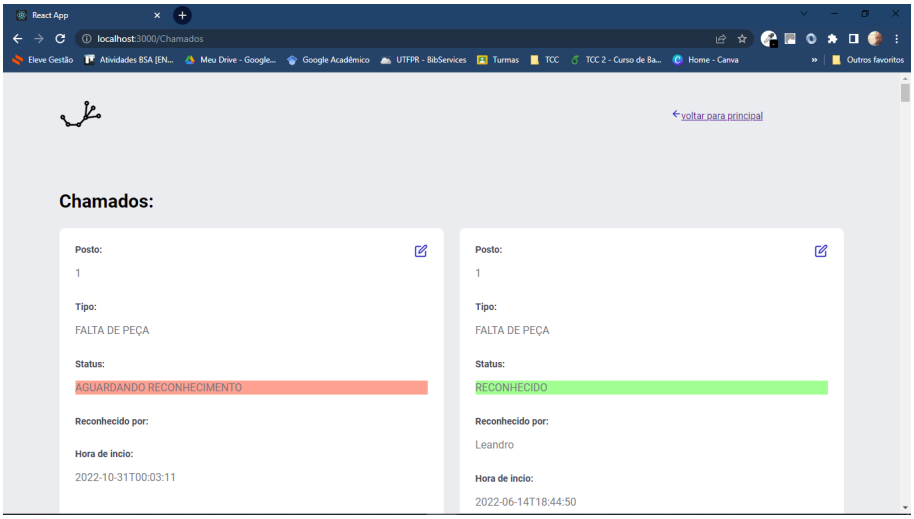
Ainda na tela principal, ao clicar na opção **Configuração** o usuário sera direcionado para a tela de configuração (Figura 22) do alarme do sinal analógico, onde poderá configurar os limites máximos e mínimos deste sinal.

**Figura 22 – Tela de Configurações**


**Fonte: Autoria própria (2022).**

Por sua vez, ao clicar na opção **Chamados** o usuário é direcionado a tela dos chamados (figura 23), onde são apresentadas todas os chamados até então criados, tanto os chamados criados pelo operador, quanto aqueles criados automaticamente, a partir da extrapolação de limite previamente configurado para o sinal analógico. É também a partir desta tela que o supervisor poderá abrir um chamado específico e assim realizar o reconhecimento do mesmo.

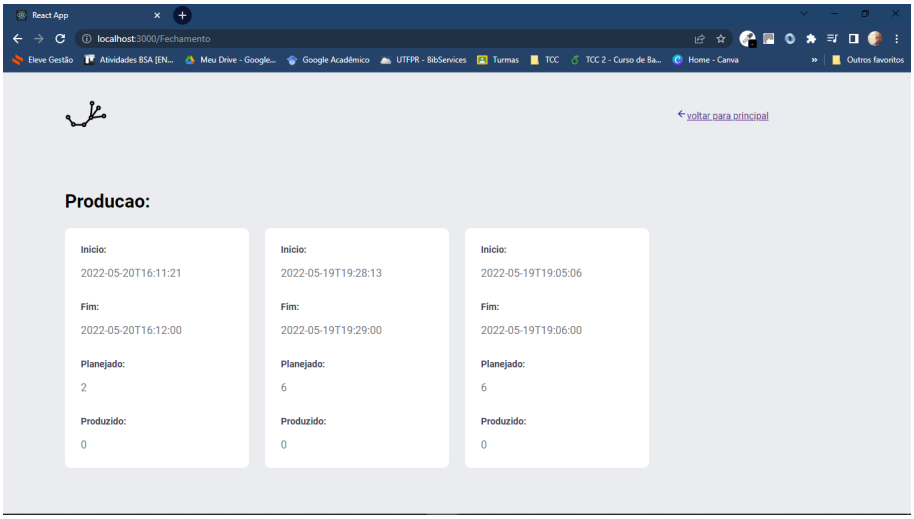
**Figura 23 – Tela de Chamados**




**Fonte: Autoria própria (2022).**

Por fim, a opção **Fechamento** direciona o usuário para uma tela onde é possível visualizar os dados das produções realizadas previamente (figura 24).

**Figura 24 – Tela de Fechamento de Produção**

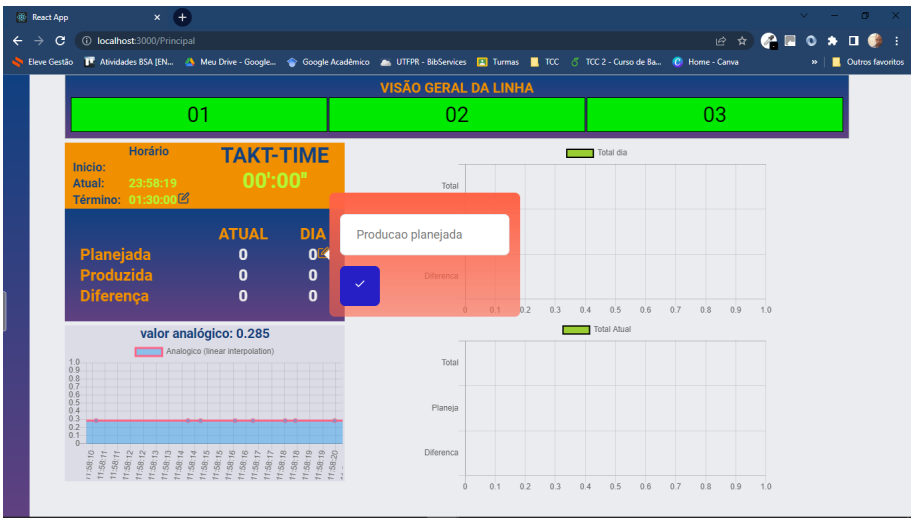



**Fonte: Autoria própria (2022).**

### 5.2.2 Operação

Para realizar o controle de produção, primeiramente o supervisor deverá preencher **horário de término** (Figura 26) e **Quantidade planejada** (Figura 25) para o dia. Em seguida, deverá iniciar o controle da produção, para isso deverá clicar no botão de início, presente no menu principal. A partir deste momento a produção será controlada até que o horário de término seja alcançado. Caso o supervisor deseje encerrar o controle da produção antes do término configurado, basta clicar no botão de encerramento.

**Figura 25 – Tela de Principal - Detalhes configuração de produção**

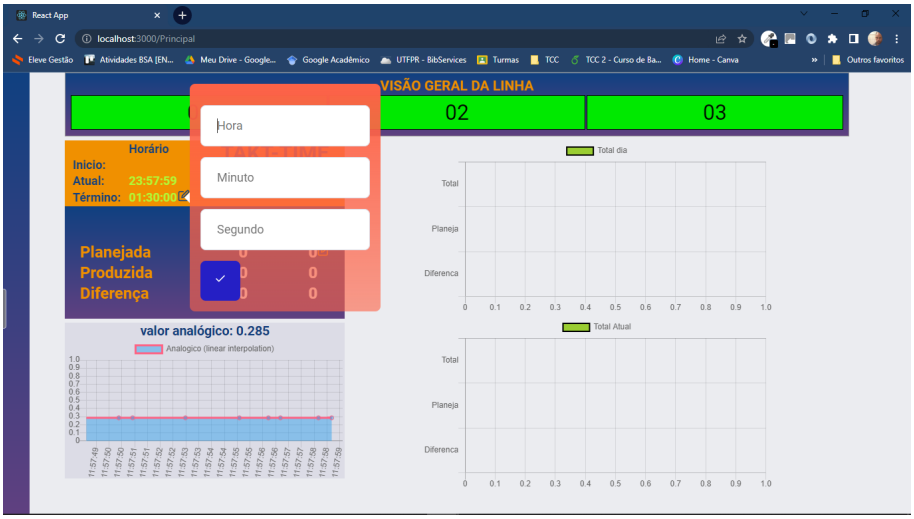



**Fonte: Autoria própria (2022).**

Durante o controle da produção, o supervisor poderá acompanhar a evolução dos indicadores diários e momentâneos da produção, tanto através da tabela de dados produtivos, quanto nos gráficos de produção.

Ao final da produção os indicadores serão armazenados e poderão ser visualizados na tela de **Fechamento** de produção.

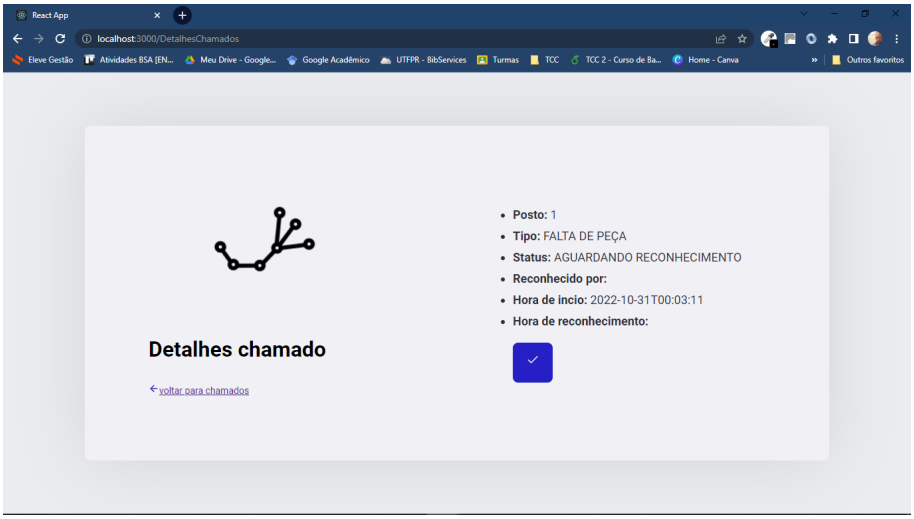
**Figura 26 – Tela de Principal - Detalhes configuração de produção**




**Fonte: Autoria própria (2022).**

Após um chamado ser criado, através dos botões pelo operador ou devido ao alarme do sinal analógico, o mesmo será salvo na lista de chamados, com a indicação de pendência de reconhecimento. Desta forma, o supervisor deverá selecionar o chamado pendente, assim, abrindo uma tela (Figura 27) com os detalhes do chamado onde o mesmo poderá ser reconhecido, através do botão **reconhecimento** de chamado.

**Figura 27 – Tela de Detalhes chamado**




**Fonte: Autoria própria (2022).**

Assim, a partir da operação do protótipo foi possível examinar a capacidade da solução em atender as premissas do trabalho em questão. Deste modo, foi verificado que todas as funções planejas para o sistema puderam ser executadas conforme esperado, ou seja, a arquitetura aplicada foi apropriada para realizar a criação e controle de chamados (ocorrências) e

planejamento e controle de produção. Além disso, de forma mais ampla, também foi possível observar a capacidade da solução de realizar a interface entre sistemas industriais (compatíveis ao protocolo Modbus) e sistemas computadorizados.

# 6 CONCLUSÃO

Como discutido, um dos elementos centrais do conceito de Indústria 4.0 é a integração de sistemas, porém, sua relevância é proporcional ao desafios que a mesma impõe, em especial quando se deseja adequar fábricas ou processos compostos por maquinário com defasagem tecnológica, como é o caso da indústria brasileira.

Portanto, faz-se necessário o desenvolvimento de soluções que consigam promover a compatibilização desses equipamentos legados às arquiteturas e premissas do conceito da Indústria 4.0. Desta forma, o objetivo central deste trabalho era avaliar a viabilidade do uso do protocolo Modbus para realizar a interface entre dispositivos industriais e sistemas informatizados no contexto da Indústria 4.0.

Para tal, foi desenvolvido um sistema que busca representar características comuns de aplicações industriais, tais como, integração entre dispositivos eletroeletrônicos industriais e sistemas informatizados, controle de sinal analógico e aquisição e armazenamento de dados de produção. O desenvolvimento desse sistema implicou na implementação de uma aplicação Java, de um banco de dados MySQL, de uma aplicação React (JavaScript) e de um programa de CLP via TIA Portal, além da instalação física dos componentes industriais.

Dentre os componentes presentes no sistema, é a aplicação Java que possui a maior relevância para que o objetivo do trabalho fosse alcançado, uma vez que a mesma, a partir do uso da biblioteca **EasyModbus** (ROSSMANN-ENGINEERING, 2022), estabelece a comunicação **Modbus** com o controlador industrial e a partir dos dados obtidos disponibiliza *endpoints* com base em uma arquitetura REST. E é exatamente este processo de conversão de protocolos de comunicação a contribuição pretendida pelo trabalho em questão.

Este trabalho se propôs a apresentar uma sugestão de solução para um problema comum da indústria e a partir da implementação de um protótipo comprovar a viabilidade do uso deste solução em um cenário real. E baseado nos resultados obtidos é possível concluir que tal solução se mostrou promissora e pode servir de inspiração para aplicações reais. Dentre as possibilidades possíveis podem ser citados:

- O acompanhamento da produção em tempo real, por variados níveis gerenciais da organização, fator que pode contribuir para otimização do tempo de resposta a mudanças e a problemas;
- Como visto no protótipo, a capacidade de indicação, acompanhamento e resolução de ocorrências (chamados/ alarmes) de forma remota;
- A aquisição, processamento e armazena de dados do processo que podem servir como base para soluções de *Business intelligence* e *Big Data*;

Outra característica relevante desta solução é a sua relação de custo beneficio, visto que é construída inteiramente a partir linguagens, bibliotecas, softwares e protocolos livres.

Contudo, é evidente que ainda existem questões em aberto que podem ser respondidas em trabalhos futuros, dentre as quais podem ser citadas, uma avaliação mais elaborada do desempenho e limitações do protocolo Modbus no contexto da Industria 4.0, avaliação das questões de segurança da informação para uma solução deste tipo ou ainda comparar esta solução com outras propostas similares.

# REFERÊNCIAS


ALMEIDA, P. S. D. **Indústria 4.0: Princípios básicos, aplicabilidade e implantação**. [*S.l.*]: Saraiva Educação SA, 2019.

ALVAREZ, R. d. R.; JR, J. A. V. A. Takt-time: conceitos e contextualização dentro do sistema toyota de produção. **Gestão & Produção**, SciELO Brasil, v. 8, p. 1–18, 2001.

ARIAS, A. P. *et al.* **A nova agenda da grande indústria: uma análise da indústria 4.0 com base em documentos e materiais de divulgação do projeto alemão Plattform Industrie 4.0**. 2020. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2020.

ARRUDA, T. A.; CAMPEDELLI, É. R.; PEREIRA, G. Desenvolvimento de uma rede modbus para o controle de um veículo autônomo. *In*: . [*S.l.*]: X SBAI – Simpósio Brasileiro de Automação Inteligente, 2011.

AZEVEDO, J. A. P.; SOUZA, A. B. d. Comparativo entre redes de automação industrial e suas características. **Eng. de Sistemas Eletroeletrônicos,Automação e Controle Industria**, 2014.

COULOURIS, G. *et al.* **Sistemas Distribuídos-: Conceitos e Projeto**. [*S.l.*]: Bookman Editora, 2013.

GODOY, E. P. Automação e controle de processos na nuvem: Proposta e estudo de caso. *In*: . [*S.l.*]: XXI Congresso Brasileiro de Automática - CBA2016, 2016.

JOVANOVIĆ, Ž. *et al.* Java spring boot rest web service integration with java artificial intelligence weka framework. *In*: **International Scientific Conference "UNITECH 2017**. [*S.l.*: *s.n.*], 2017. p. 323–327.

LIN, S.-W. *et al.* **Architecture Alignment and Interoperability - An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper**. [*S.l.*], 2017.

MELO, P. F. S. d. **Dispositivo de controle para a indústria 4.0 baseado no RAMI 4.0 e OPC UA**. 2020. Dissertação (Mestrado) — Universidade Estadual Paulista (UNESP), 2020.

MYSQL. 2022. Url https://www.mysql.com/.

PEREIRA, A.; SIMONETTO, E. de O. Indústria 4.0: conceitos e perspectivas para o brasil. **Revista da Universidade Vale do Rio Verde**, v. 16, n. 1, 2018.

PINTO, G. A. **A organização do trabalho no século XX: taylorismo, fordismo e toyotismo**. [*S.l.*]: Expressão Popular, 2013.

QUINTINO, L. F. *et al.* **Industria 4.0**. [*S.l.*]: SAGAH, 2019. ISBN 9788595028531.

RIBEIRO, M.; FRANCISCO, R. Web services rest conceitos, análise e implementação. 2016.

ROSSMANN-ENGINEERING. **Easy Modbus**. 2022. Url https://start.spring.io/.

SACOMANO, J. B. *et al.* **Indústria 4.0**. [*S.l.*]: Editora Blucher, 2018. ISBN 9788521213710.

SCHMITT, Á. *et al.* Implementação de uma rede industrial para células de soldagem robotizadas utilizando o protocolo modbus. **Revista Ilha Digital**, v. 4, p. 61–66, 2013.

SCHWAB, K. **A quarta revolução industrial**. [*S.l.*]: Edipro, 2019.

SIEMENS, A. **SIMATIC S7-1200 Programmable controller**. [*S.l.*], 2021. V4.5.

SILVA, M. H. da; WEBBER, C. G. Análise comparativa entre plataformas para o desenvolvimento da indústria 4.0. **Scientia cum Industria**, v. 8, n. 2, p. 115–122, 2020.

SILVA, V. L.; KOVALESKI, J.; PAGANI, R. N. Competências bases para o trabalho humano na indústria 4.0. **Revista Foco**, v. 12, n. 2, p. 112–129, 2019.

TAMBOLI, S. *et al.* Implementation of modbus rtu and modbus tcp communication using siemens s7-1200 plc for batch process. *In*: IEEE. **2015 international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)**. [*S.l.*], 2015. p. 258–263.

TESSARINI, G.; SALTORATO, P. Impactos da indústria 4.0 na organização do trabalho: uma revisão sistemática da literatura. **Revista Produção Online**, v. 18, n. 2, p. 743–769, 2018.

VMWARE, I. **Spring Initializr**. 2022. Url https://start.spring.io/.