UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE MOREIRA DE SOUZA

AQUISIÇÃO, TRANSFERÊNCIA E ARMAZENAMENTO DE DADOS PARA INDÚSTRIA 4.0

RUSSAS

2022

FELIPE MOREIRA DE SOUZA

AQUISIÇÃO, TRANSFERÊNCIA E ARMAZENAMENTO DE DADOS PARA INDÚSTRIA 4.0

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

Orientador: Prof. Ms. Filipe Maciel Roberto

RUSSAS

2022

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

S238a Souza, Felipe Moreira de.
Aquisição, Transferência e Armazenamento de Dados para Indústria 4.0 / Felipe Moreira de Souza. – 2022.
70 f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2022.
Orientação: Prof. Me. Filipe Maciel Roberto.

1. Indústria 4.0. 2. Internet das Coisas. 3. Sistemas Ciber-Físicos. 4. Computação em Nuvem. I. Título.
CDD 005

FELIPE MOREIRA DE SOUZA

AQUISIÇÃO, TRANSFERÊNCIA E ARMAZENAMENTO DE DADOS PARA INDÚSTRIA 4.0

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

Aprovada em:

BANCA EXAMINADORA

Prof. Ms. Filipe Maciel Roberto (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

Profa. Dra. Rosineide Fernando da Paz
Universidade Federal do Ceará (UFC)

RESUMO

A Indústria 4.0, também conhecida como a Quarta revolução Industrial, tem como base os paradigmas de Internet das Coisas e Sistemas Ciber-Físicos. Esses paradigmas integrados, são responsáveis por realizar o processo de controle e automação da indústria. Para que isso ocorra, alguns procedimentos devem ser adotados. Primeiramente, a coleta de dados dos dispositivos da indústria deve ser feita. Os dados coletados precisam ser preparados e enviados de forma segura e confiável para armazenamento e análise em um ambiente externo, a fim de possibilitar a tomada de decisões com base nesses dados pelos gestores da indústria. Esse fato ocorre devido à crescente quantidade de dados gerados por esses dispositivos, onde o gasto com *hardware* para realizar o seu processamento se torna algo inviável. Como solução, outro paradigma é adotado: A Computação em Nuvem. A nuvem se baseia no oferecimento de serviços computacionais de armazenamento, *softwares*, análise e entre outros serviços por meio da internet. Este trabalho apresenta o levantamento e a comparação das tecnologias de coleta, formatação, transferência e armazenamento de dados. Nesse meio termo, também é realizado um estudo sobre segurança para transferência de dados. Com essas tecnologias mapeadas, foi possível implementar a solução de um cenário para exemplificar um subconjunto delas e assim realizar uma implantação eficiente da Indústria 4.0. Para o desenvolvimento e testes dessa solução, uma instância local do MongoDB foi utilizada para armazenar os dados de um moinho para fabricação de cimento, onde cada conjunto de dados é identificado por meio de sua marca de tempo e possui 152 *tags* que representavam essas informações. Após isso, foi utilizado um *software* para simular a geração de valores associados aos dados do moinho. Os dados foram adquiridos por um serviço coletor em Python, utilizando o protocolo *Open Platform Communications* (OPC) e portanto, transferidos por meio do sistema de comunicação Apache Kafka para a nuvem da *Amazon Web Services* (AWS), de onde foram pré-processados e analisados por duas aplicações Python e os seus resultados escritos em uma instância do MongoDB em nuvem. Por último, de acordo com os estudos levantados e os resultados obtidos, foi possível notar a eficiência das tecnologias selecionadas para a implementação do cenário, a fim de servir como fonte de contribuição para os engenheiros e gestores da indústria.

Palavras-chave: Indústria 4.0, Sistemas Ciber-Físicos, Internet das Coisas, Computação em Nuvem.

ABSTRACT

Industry 4.0, also known as the Fourth Industrial Revolution, is based on the Internet of Things and Cyber-Physical Systems paradigms. These integrated paradigms are responsible for carrying out the control and automation process of the industry. For this, all must be adopted. The collection of data from the devices of the industry must be done. The collected data needs to be prepared and sent securely and reliably for storage and analysis in an external environment, in order to enable decision-making based on this data by industry managers. This fact occurs due to the growing amount of data generated by these devices, where the expense with hardware to perform its processing becomes something unfeasible. As a solution, another paradigm is adopted: Cloud Computing. The cloud is based on the provision of computing services of storage, software, analysis and other services through the internet. This work presents the survey and comparison of collection, formatting, transfer and data storage. In the meantime, a study on safety is also carried out for data transfer. With these technologies mapped, it was possible to implement the solution of a scenario to exemplify a subset of them and thus carry out a deployment industry 4.0 efficient. For the development and testing of this solution, a local instance of MongoDB was used to store data from a cement mill, where each dataset is identified by its timestamp and has 152 tags that represented this information. After that, a software was used to simulate the generation of values associated with mill data. The data were acquired by a Python collector service, using the Open Platform Communications (OPC) protocol and therefore, transferred through the Apache Kafka communication system to the Amazon Web Services (AWS) cloud, where they were pre-processed. and analyzed by two Python applications and their results written to a MongoDB instance in the cloud. The scenario raised consists of using data from a mill to manufacture cement, where each set of data is identified through its timestamp and has 152 tags that represent these information. Finally, according to the studies surveyed and the results obtained, it was possible to notice the efficiency of the technologies selected for the implementation of the scenario, in order to serve as a source of contribution for industry engineers and managers.

Keywords: Industry 4.0, Cyber-Physical Systems, Internet of Things, Cloud Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OPC	*Open Platform Communications*
AWS	*Amazon Web Services*
ICS	*Industrial Control System*
SCADA	*Supervisory Control and Data Acquisition*
DCS	*Distributed Control System*
PLC	*Programmable Logic Controller*
RTU	*Remote Terminal Unit*
HMI	*Human-Machine Interface*
MTU	*Master Terminal Unit*
IED	*Intelligent Electronic Device*
MES	*Manufacturing Execution System*
ERP	*Enterprise Resource Planning*
CPS	*Cyber Physical-System*
IoT	*Internet of Things*
RFID	*Radio Frequency Identification*
NFC	*Near Field Communication*
URLLC	*Ultra-Reliable Low-Latency Communication*
TSN	*Time Sensitive Networking*
OPC UA	*Open Platform Communications Unified Architecture*
IaaS	*Infrastructure as a Service*
PaaS	*Platform as a Service*
SaaS	*Software as a Service*
API	*Application Programming Interface*
REST	*Representational State Transfer*
HTTP	*Hypertext Transfer Protocol*
MQTT	*Message Queuing Telemetry Transport*
CSV	*Comma-Separated Values*
XML	*Extensible Markup Language*
JSON	*JavaScript Object Notation*
RPC	*Remote Procedure Call*

TLS	*Transport Layer Security*
SQL	*Structured Query Language*
NOSQL	*Not Only Structured Query Language*
EC2	*Elastic Compute Cloud*
ECS	*Elastic Compute Service*
RSA	Rivest-Shamir-Adleman
SSH	*Secure Shell*

SUMÁRIO

1 INTRODUÇÃO

A Quarta Revolução Industrial, também conhecida como indústria 4.0, surgiu em meados de 2011. Ela se baseia no conceito de Internet das Coisas e Sistemas Ciber-Físicos (MÜLLER *et al.*, 2021). A Internet das coisas é uma rede global de dispositivos de detecção e atuação que interconecta qualquer produto ou objeto físico visível no globo através de máquinas baseadas em protocolos de comunicação padrão, que reúnem e enviam os dados para análise, com o propósito de um mundo mais inteligente, confortável, eficiente e eficaz (HAJIHEYDARI *et al.*, 2019). Os Sistemas Cyber Físicos são compostos de componentes físicos e de *software* profundamente ligados por meio de sensores (MÜLLER *et al.*, 2021).

A implementação da indústria 4.0 é feita em um fluxo que parte desde a aquisição de dados até a tomada de decisões. Os dados são coletados e transmitidos para um armazenamento em nuvem, onde são analisados (OLIVEIRA; AFONSO, 2019). Com os dados analisados, a indústria toma decisões para otimizar a produção, proporcionando maior flexibilidade e agilidade na cadeia de valores da indústria (SOUZA *et al.*, 2020).

Para que esses dados fiquem disponíveis ao sistema de análise no cenário descrito acima, devem ser pensadas tecnologias e processos de aquisição, transferência e armazenamento para um melhor desempenho possível dessa análise (SHAFIQ *et al.*, 2019). Para implementar as soluções exigidas por algum cenário deste projeto, existem hoje diversas tecnologias e processos que abrangem as mais diversas áreas da computação, onde especificamente na indústria 4.0, podem ser aplicados nos seguintes estágios:

- Os dados são adquiridos na indústria por meio de agentes como sensores, indentificadores de radiofrequência (radares) e câmeras por meio de rede com e sem fio para a medição do estado do produto e condições ambientais (SHAFIQ *et al.*, 2019).
- Esses dados são serializados, criptografados, empacotados e transmitidos em rede para a nuvem através de protocolos de comunicação específicos de cada implementação (COMAN; FLORESCU, 2018).
- Os dados transmitidos para a nuvem são descriptografados, armazenados e podem ser acessados remotamente em tempo real através da internet (COMAN; FLORESCU, 2018).
- A nuvem representa a centralização de armazenamento e processamento. Assim, permite o uso comum de infraestrutura, serviços, software ou dados (TRINKS; FELDEN, 2018).
- Os dados na nuvem precisam ser filtrados, uma vez que nem todos os dados brutos são úteis para uma análise eficiente. Os dados são organizados e encapsulados para as informações

úteis serem extraídas e analisadas sintaticamente (SHAFIQ *et al.*, 2019).

– Para as análises exploratórias e identificação rápida de padrões na indústria, um sistema de análise visual é implementado, contendo uma saída gráfica para a visualização de informações aprimoradas e armazenadas na arquitetura (SHAFIQ *et al.*, 2019).

O objetivo deste trabalho é apresentar um conjunto de tecnologias e procedimentos disponíveis para aquisição, transferência e armazenamento de dados para a indústria 4.0. Como objetivos específicos, tem-se:

– Realizar uma revisão bibliográfica sobre tecnologias de coleta, formatação, transmissão e armazenamento de dados;
– Realizar um estudo complementar sobre segurança da informação, a fim de aplicá-la na transfêrencia dos dados;
– Fazer um comparativo de tecnologias para definir um conjunto à ser utilizado em algum cenário;
– Identificar um cenário de aplicação para exemplificação das tecnologias mapeadas;
– Implementar uma solução de coleta, formatação, comunicação confiável e armazenamento de dados para pré-processamento e análise, de forma a evoluir um sistema síncrono, comunicando-se por meio de banco de dados para um assíncrono com reatividade, onde a comunicação é realizada por meio de sistema de mensagens.

Este trabalho é relevante pelo fato de agregar conhecimentos dispersos, tanto para a comunidade acadêmica quanto para os engenheiros e gestores da indústria. Permitirá o mapeamento do estado da arte da área de estudo e disponibilizará como deve ser projetado um sistema desde a coleta até o armazenamento de dados. Essa coleta de dados é importante pelo fato de que o mercado de sistemas e indústrias baseadas em dados irá crescer em grandes escalas (SAHAL *et al.*, 2020). Assim, a economia mundial vem a ser modelada na tomada de decisões baseadas em dados coletados (MOURA *et al.*, 2019).

A pesquisa inicia com uma introdução ao problema (capítulo 1). Em seguida, a fundamentação teórica é apresentada (capítulo 2). Após isso, alguns trabalhos relacionados à essa pesquisa são abordados (capítulo 3). Feito isso, uma proposta de solução para coleta, formatação, transmissão segura e armazenamento de dados é apresentada de forma genérica e após isso, o desenvolvimento dessa solução (capítulo 4). Um cenário e sua infraestrutura são apresentados para os testes da solução desenvolvida, assim como também, os seus resultados são relatados (capítulo 5). Por último, as conclusões relacionadas à solução desenvolvida são

descritas e algumas frentes de trabalhos futuros são mencionadas como sugestões interessantes para o trabalho (capítulo 6).

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os principais conceitos utilizados para a implementação da Indústria 4.0. É dividido em 4 partes: a primeira parte descreve o Sistema de Controle Industrial, onde os dados da indústria são coletados, monitorados e transmitidos para um ambiente externo. A segunda parte aborda sobre a Indústria 4.0, onde ressalta a ideia de automação e paradigmas habilitadores conhecidos como Sistemas Ciber-Físicos e Internet das Coisas. A terceira parte descreve a Computação em Nuvem, que disponibiliza os serviços de armazenamento e acesso rápido a dados em um ambiente fora da indústria. A quarta parte descreve sobre os padrões de projeto para fins de estruturação de código. A última parte descreve as tecnologias habilitadoras da indústria 4.0, assim, enfatizando as etapas de coleta, formatação, transferência segura e armazenamento. É necessária a integração de todos esses paradigmas e tecnologias para uma implementação eficiente e correta da indústria 4.0.

2.1 Sistema de Controle Industrial

Segundo a abordagem da Trend Micro (2021), o Sistema de Controle Industrial, do inglês: *Industrial Control System* (ICS), é um termo descritivo sobre diferentes tipos de sistemas de controle e instrumentação associadas, como dispositivos, redes e sistemas de controle para automatizar processos industriais. Atualmente, os dispositivos e protocolos usados em um ICS se aplicam em quase todos os setores industriais e infraestruturas críticas, como as indústrias de manufatura, transporte, energia e tratamento de água. Os dois tipos mais comuns de ICSs são o Sistema de controle Supervisório e Aquisição de Dados e o Sistema de Controle Distribuído, respectivamente também chamados de *Supervisory Control and Data Acquisition* (SCADA) e *Distributed Control System* (DCS). Ambos são descritos nas partes 3 e 4 desta seção. A implementação de um ICS pode muitas vezes ser um híbrido entre SCADA e DCS, onde os atributos de ambos os sistemas são incorporados.

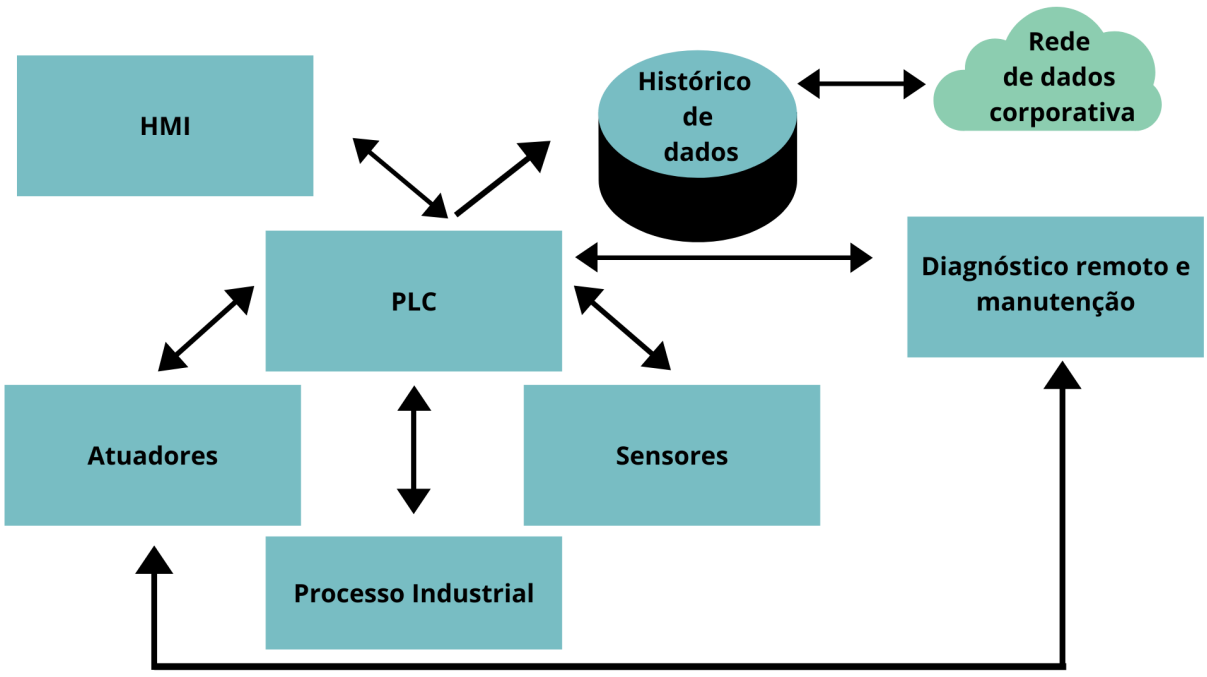
2.1.1 Componentes do ambiente ICS

Seguindo a abordagem da Trend Micro (2021), os componentes do Sistema de Controle Industrial são:

– *Programmable Logic Controller* (PLC): Trata-se de um componente de controle de processos executados por sensores e atuadores;

- *Remote Terminal Unit* (RTU): São dispositivos de campo controlados através de microprocessadores;
- Loop de controle: Consistem em Controladores Lógico Programáveis e atuadores que interpretam sinais de sensores e interruptores;
- *Human-Machine Interface* (HMI): Interface que permite a interação entre o humano e o hardware do controlador. Essa interface exibe informações de status e histórico de dados coletados. Além disso, pode configurar pontos de ajuste.
- Diagnóstico remoto e manutenção: Termo utilizado para identificar e prevenir falhas que podem acontecer nos processos;
- Servidor de controle: Responsável por hospedar o software de controle de supervisão para DCS ou SCADA. Se comunica com os dispositivos de controle de nível inferior;
- *Master Terminal Unit* (MTU): Também conhecido como servidor SCADA, o MTU é usado para emitir comandos para os RTUs em campo;
- *Intelligent Electronic Device* (IED): Dispositivos que coletam dados e se comunicam com outros dispositivos. Eles permitem a automatização do controle de nível local;
- Historiador de dados: Banco de Dados que contém todas as informações do processo. Esses dados são exportados para alguma corporação, a fim de serem utilizados para análise de processos e planejamentos empresariais.

Figura 1 – Fluxo do ICS



Fonte: Elaborado pelo autor (2021).

2.1.2 Componentes de comunicação e protocolos

A comunicação estabelecida dentro do ICS necessita de alguns componentes e protocolos de comunicação para ser realizada. As seguintes definições desses componentes e tecnologias, foram construídas baseseadas na abordagem da Trend Micro (2021) e no guia de Stouffer *et al.* (2006):

- Componentes:
  - Redes *Fieldbus*: Conecta sensores com um PLC ou controlador. Esse tipo de tecnologia elimina a necessidade de fiação entre esses dispositivos. Os sensores, portanto, se comunicam com o controlador por meio de um protocolo específico;
  - Rede de controle: Responsável por conectar o nível de controle de supervisão ao nível inferior de módulos de controle;
  - Roteadores de comunicação: Dispositivo que transfere mensagens entre redes. Como exemplo, podem conectar o MTU e os RTUs à redes de longa distância em um SCADA;
  - *Firewall*: Recurso que realiza a proteção de dispositivos em rede com controle de pacotes de comunicação;
  - Modens: Dispositivos utilizados para realizar a conversão de dados digitais em sinais para transmissão em linha telefônica. No SCADA, ele permite a comunicação serial de longa distância entre MTUs e dispositivos de campo;
  - Ponto de acesso remoto: Engloba dispositivos, áreas e locais distintos de uma rede de controle para configuração de acesso a dados.
- Protocolos:
  - *Profibus*: Estabelece comunicações do RTU para MTU e vice-versa. Além disso, também podem stabelecer conexões de MTU para MTU. O Profibus é usado tanto para operar sensores e atuadores por meio de um controlador central quanto para monitorar equipamentos de medição por meio de um sistema de controle de processo;
  - *EtherCAT*: Um protocolo que incorpora a tecnologia *Ethernet* em ambientes industriais;
  - *Modbus*: Usa comunicação serial com PLCs. Uma das implementações desse protocolo é a *Modbus*-TCP que utiliza a pilha de protocolos da internet TCP/IP para transmissão de dados;
  - Plataforma de Comunicação Aberta: Trata-se de um conjunto de padrões da *Microsoft*

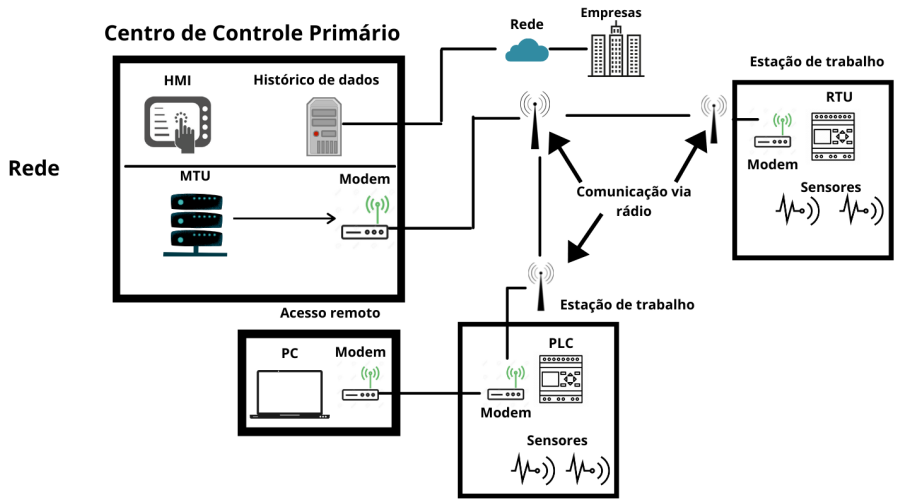
para comunicações industriais.

Algumas plataformas que fazem uso de alguns desses protocolos são abordadas na proposta deste trabalho.

2.1.3 Sistema de Controle Supervisório e Aquisição de Dados

De acordo com o guia de Stouffer *et al.* (2006), o SCADA é um tipo de sistema usado para controlar ativos dispersos em longa distância, onde a aquisição centralizada de dados é tão importante quanto o controle. Esses sistemas podem ser usados em redes de transmissão e distribuição de utilidades elétricas, além de transporte ferroviário, por exemplo. O SCADA é projetado para coletar informações de campo, transferir para uma instalação de computador central e exibir as informações para o operador de forma gráfica ou textual, assim, permitindo que o operador monitore ou controle um sistema inteiro em tempo real. Praticamente, realizam o funcionamento dos componentes do ICS. Assim, esses sistemas consistem em *hardware* e *software*. A arquitetura do SCADA é representada na Figura 2.

Figura 2 – Arquitetura do SCADA



Fonte: Elaborado pelo autor (2021).

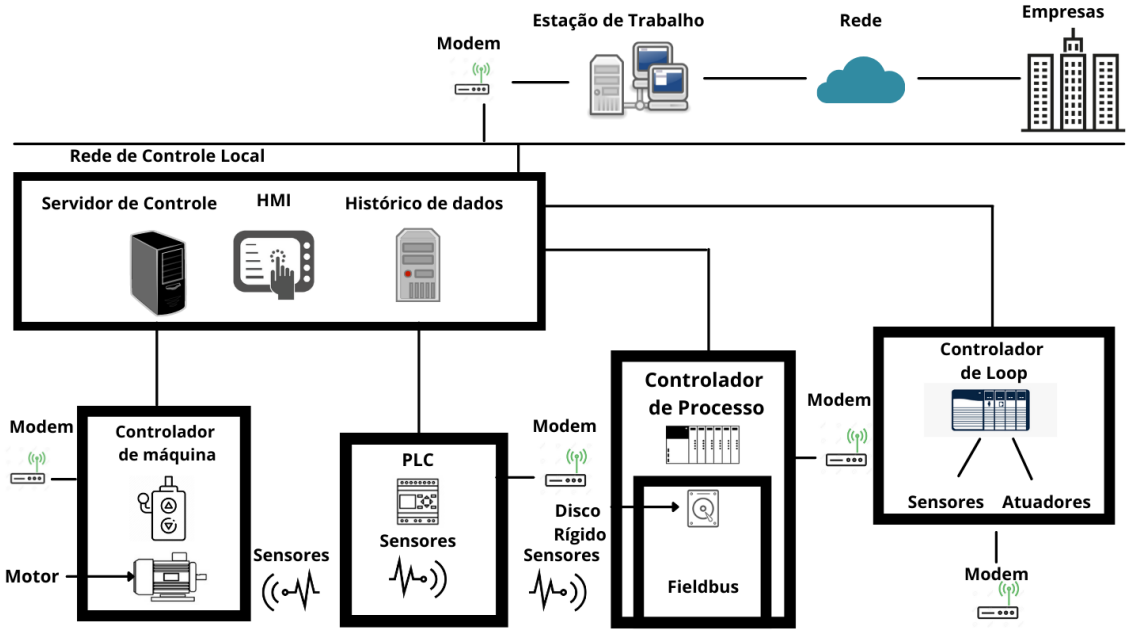
Na Figura 2, os componentes do Centro de Controle Primário se conectam entre si através de uma rede local. As informações de campo presentes nas estações de trabalho, são coletadas e registradas no centro de controle, onde o MTU, por sua vez, envia comandos

para os locais de campo. Esse tráfego de informações é possibilitado através de protocolos de comunicação que utilizam técnicas de telemetria, como rádio, linha telefônica ou satélite. Essas informações são exibidas no HMI, onde ações podem ser geradas com base nesses dados. O Historiador de dados pode exportar as informações para uma rede externa com destino à alguma corporação. Além disso, análises de tendências e relatórios podem ser gerados no centro de controle. As estações de trabalho podem ser acessadas remotamente para diagnósticos e reparos.

2.1.4 Sistema de Controle Distribuído

De acordo com o guia de Stouffer *et al.* (2006), o DCS é utilizado para controle de sistemas de produção dentro da mesma localização geográfica. Tem aplicação para indústrias de refinaria de petróleo e gás, além de tratamento de água. O DCS usa um circuito de controle supervisionado centralizado para mediar controladores que compartilham as tarefas de um processo de produção. Trata-se de um tipo de sistema modularizado, onde uma única falha no sistema não tem um impacto tão significante. O DCS faz interface com uma rede corporativa, fornecendo assim, uma visão de negócios. No DCS, um servidor de controle se comunica com PLCs e outros dispositivos através de uma rede de controle. O supervisor envia pontos de ajuste e solicita dados dos dispositivos distribuídos em campo. Esses dispositivos controlam os atuadores do processo com base nos comandos do servidor e *feedbacks* dos sensores do processo. A arquitetura do DCS é representada na Figura 3.

Figura 3 – Arquitetura do DCS



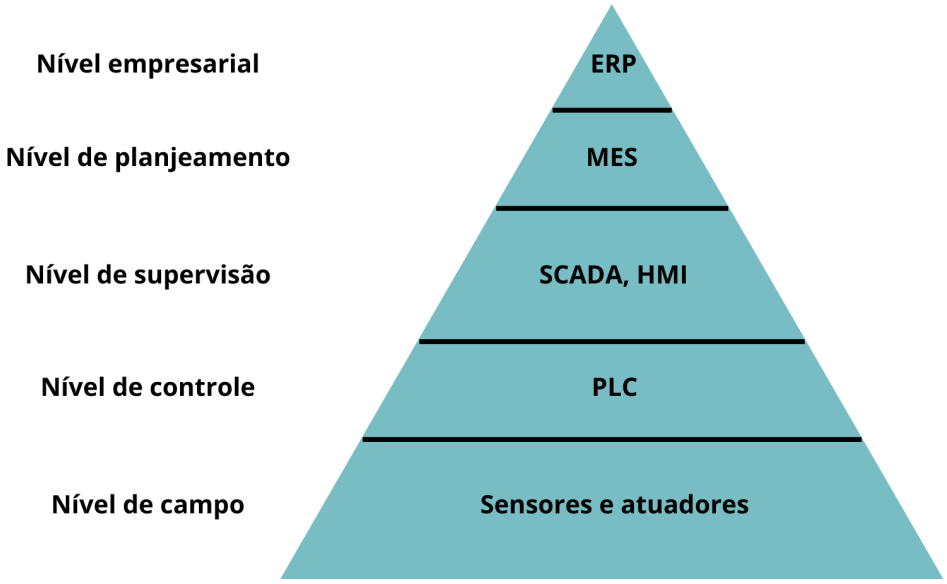
Fonte: Elaborado pelo autor (2021).

A Figura 3 apresenta uma interação entre componentes similar ao SCADA, porém, se comunicando à curta distância.

2.2 Indústria 4.0

Segundo a abordagem de Sauter *et al.* (2011), o objetivo predominante da automação sempre foi tornar os processos mais eficientes, no sentido de que a automação retira os humanos do trabalho que pode ser delegado para as máquinas de forma mais rápida, segura, com maior precisão e a um custo reduzido. Um constituinte substancial dos sistemas de automação modernos é a informação. O processamento dessa informação é uma tarefa essencial na automação. Dependendo da subárea de uma determinada automação, o significado, conteúdo e as propriedades da informação variam. Por exemplo, um sistema de controle exige informações do sensor e do atuador com requisitos rigorosos em tempo real, enquanto o planejamento da produção precisa lidar com grandes quantidades de dados sobre pedidos de clientes ou listas de materiais. A utilidade e necessidade de uma automação de forma hierárquica pode ser melhor representada na Figura 4.

Figura 4 – Pirâmide de Autômação



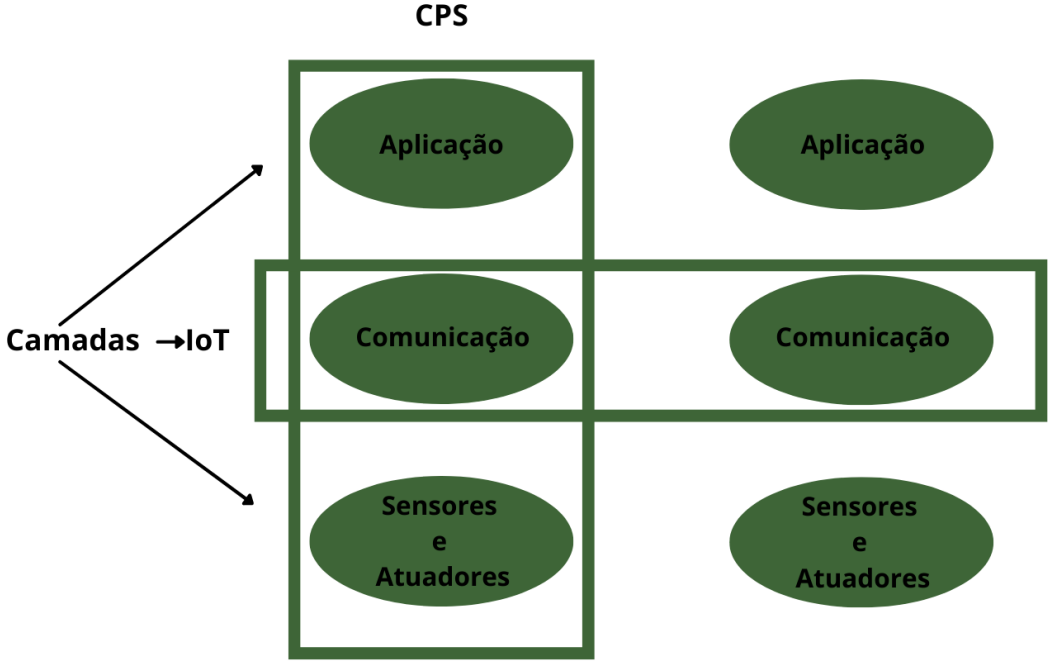
Fonte: Elaborado pelo autor (2021).

Com base em Kumar (2019), os níveis representados na figura acima são caracterizados da seguinte forma:

– Nível de campo: Simboliza a área de produção através do trabalho físico realizado entre sensores, atuadores e alguns dispositivos como motores elétricos e interruptores;
– Nível de controle: Os dispositivos do nível de campo são manipulados através dos PLCs;
– Nível de supervisão: O SCADA monitora e controla os diversos sistemas a partir de um único lugar;
– Nível de planejamento: Um sistema que monitora todo o processo, desde a matéria-prima até o produto pronto na indústria. Esse sistema é conhecido como *Manufacturing Execution System* (MES), do português: Sistema de Execução da Manufatura;
– Nível empresarial: Utiliza um sistema de gestão integrado da empresa, conhecido como *Enterprise Resource Planning* (ERP), do português: Planejamento de Recursos Empresariais. Esse sistema permite que a empresa seja capaz de monitorar todos os níveis do negócio, desde a fabricação, vendas, compras e finanças.

De acordo com Wollschlaeger *et al.* (2017), paradigmas habilitadores como Internet das Coisas e Sistemas Ciber-Físicos, possibilitam tendências industriais recentes, como: alto grau de interconexão e automação cognitiva, além de coleta e processamento de dados com base em nuvem. A definição da Indústria 4.0 foi justamente composta pelas ideias de Internet das Coisas e Sistemas Ciber-Físicos, com a criação de produtos e serviços inteligentes através da Internet. Basicamente, a transformação digital é o núcleo da Indústria 4.0. A Figura 5 representa a integração entre Internet das Coisas e Sistemas Ciber-Físicos.

Figura 5 – Integração entre Internet das Coisas e Sistemas Ciber-Físicos



Fonte: Elaborado pelo autor (2021).

Conforme definido acima, os Sistemas Ciber-Físicos apresentam uma arquitetura vertical, onde na camada de sensores e atuadores, os dados são coletados em tempo real e comandos são executados. A camada de comunicação é usada para troca de dados entre a camada de aplicação e a camada de sensores e atuadores. Por último, a camada de aplicação é usada para analisar dados e tomar decisões (KUMAR; ARORA, 2020).

A Internet das Coisas, por seu lado, representa a camada de comunicação em um contexto horizontal, que interconecta várias arquiteturas de Sistemas Ciber-Físicos em redes heterogêneas (KUMAR; ARORA, 2020).

2.2.1 Sistema Ciber-Físico

De acordo com a abordagem de Lin *et al.* (2017), o Sistema Ciber-Físico ou *Cyber Physical-System* (CPS) em inglês, é a integração entre componentes cibernéticos e físicos através de tecnologias de computação e comunicação. O CPS tem como objetivo: realizar o monitoramento e controle de componentes físicos de forma segura, eficiente e inteligente, aproveitando componentes cibernéticos. Os componentes cibernéticos de um sistema são os sensores modernos aliados à computação com tecnologias de comunicação, que monitoram e controlam os componentes físicos do mundo real. Algumas áreas de aplicação do CPS tem uso para rede inteligente e transporte inteligente. De uma forma mais detalhada, os sensores são implantados para medir e monitorar o status dos componentes físicos, atuadores são implantados para garantir as operações desejáveis no físico e as redes de comunicação são usadas para entregar dados medidos e comentários de *feedbacks* entre sensores, atuadores e centros de controle.

2.2.2 Internet das coisas

De acordo com Lin *et al.* (2017), Internet das coisas, ou *Internet of Things* (IoT) em inglês, é a infraestrutura de rede para conectar um grande número de dispositivos, controlando e monitorando esses dispositivos usando as tecnologias do espaço cibernético. Com isso, o objetivo da IoT é realizar a interconexão de várias redes para que a coleta de dados, compartilhamento, análise e gerenciamento de recursos possam acontecer através de redes heterogêneas. Portanto, em sistemas baseados na cooperação de trabalho entre sensores e atuadores para atingir um objetivo específico, a rede IoT é como o veículo que realiza esse trabalho cooperativo de forma distribuída (PENA *et al.*, 2017).

2.2.2.1 Tecnologias IoT

Algumas tecnologias habilitadoras de IoT estão listadas a seguir.

- *Radio Frequency Identification* (RFID): Essa tecnologia fornece uma opção de forma simples e de baixo consumo de energia. É responsável por implantar receptores de rádio bidirecionais para rastrear etiquetas associadas à objetos. Em IoT, essas etiquetas possibilitam a comunicação entre objetos, a fim de relatar status (NEC, 2020);
- *Near Field Communication* (NFC): Outra solução simples e com um baixo consumo de energia. Essa tecnologia consiste em protocolos de comunicação para dispositivos eletrônicos. Os chips NFCs funcionam à curto alcance (NEC, 2020);
- 5G: Esse tipo de rede é projetado para serviços de banda larga móvel em smarpthones ou tablets, porém, também foi adaptado para a comunicação da indústria através da IoT. Para isso, é necessário haver comunicação em grandes quantidades de dispositivos (como sensores) conectados. Também, é necessário estabelecer um padrão de comunicação de baixa latência para os sistemas de controle. Esse padrão é conhecido como *Ultra-Reliable Low-Latency Communication* (URLLC). Esses recursos, portanto, habilitam ao 5G, uma comunicação sem fio e sensível ao tempo, o que é algo necessário para a automação industrial (FARKAS *et al.*, 2019);
- *Time Sensitive Networking* (TSN): Se trata de um conjunto de padrões que garantem baixa latência em uma rede Ethernet (VAILLANT, 2020). O Ethernet, por sua vez, funciona na indústria local, partindo do princípio de envio e recepção de pacotes de dados (VENTURELLI, 2020). O TSN oferece suporte para tecnologias como *Open Platform Communications Unified Architecture* (OPC UA), EtherCAT ou Profinet (VAILLANT, 2020).

2.3 Computação em Nuvem

Velasquez *et al.* (2018) aborda a Computação em Nuvem (do inglês: Cloud Computing) como um termo relacionado ao armazenamento de grandes quantidades de dados, que por exemplo, são produzidos por máquinas e sensores em um processo de produção. De forma escalável, a nuvem permite que a capacidade de armazenamento e processamento seja contratada de acordo com a demanda, ocasionando assim, a redução de custos com servidores, por exemplo.

2.3.1 Arquitetura da Nuvem

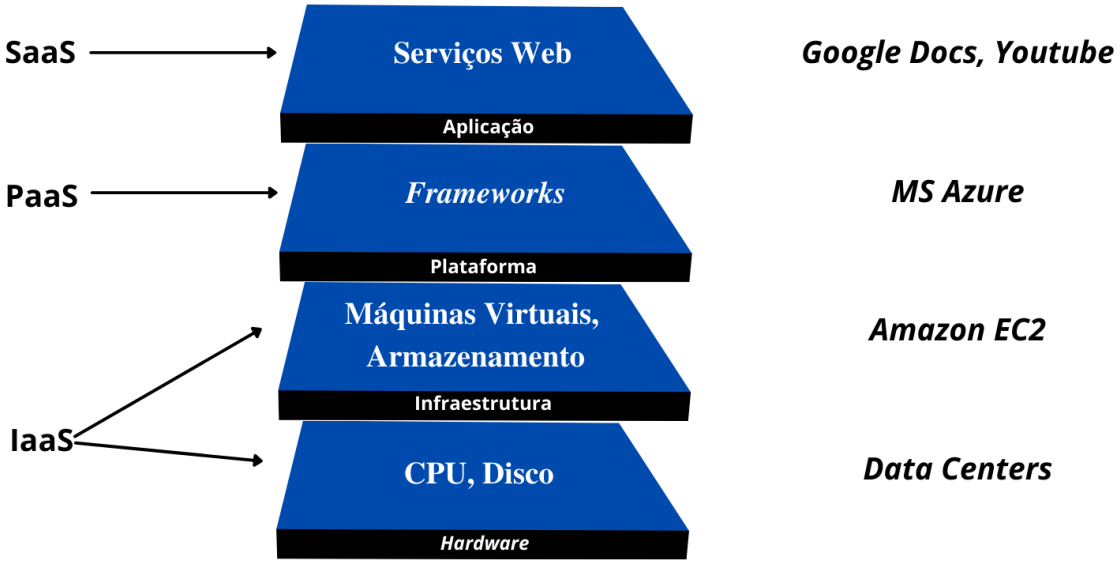
Baseado na abordagem de Steen (2020), a nuvem é organizada em quatro camadas.

- *Hardware*: Composta por processadores, roteadores e sistema de energia e refrigeração;
- Infraestrutura: Responsável por técnicas de virtualização para armazenamento e recursos computacionais;
- Plataforma: Fornece meios para desenvolver e implantar aplicativos que precisam ser executados na Nuvem;
- Aplicação: Nessa camada é onde ocorre a execução dos aplicativos reais.

O trabalho de Mukundha e Vidyamadhuri (2017), além do conteúdo de Steen (2020), abordam nessas camadas, três tipos de serviços disponibilizados pelos provedores de nuvem:

- *Infrastructure as a Service* (IaaS): São serviços para as camadas de hardware e infraestrutura. Essa estrutura disponibiliza máquinas virtuais, evitando custos com hardware;
- *Platform as a Service* (PaaS): Esse modelo oferece plataformas de programação, servidores Web, bancos de dados, entre outros. Os recursos podem ser usados sem a necessidade de gastos em infraestrutura. Esse recurso proporciona uma maximização de produtividade e minimização do tempo de desenvolvimento de aplicativos;
- *Software as a Service* (SaaS): Esse modelo funciona para a camada de aplicação. Os usuários instalam diretamente os aplicativos por assinatura na nuvem e acessam diretamente o software de seus clientes em nuvem. Não é necessário gerenciar infraestrutura. Esse tipo de serviço é muito importante por proporcionar consistência de dados em empresas.

Figura 6 – Organização da Nuvem

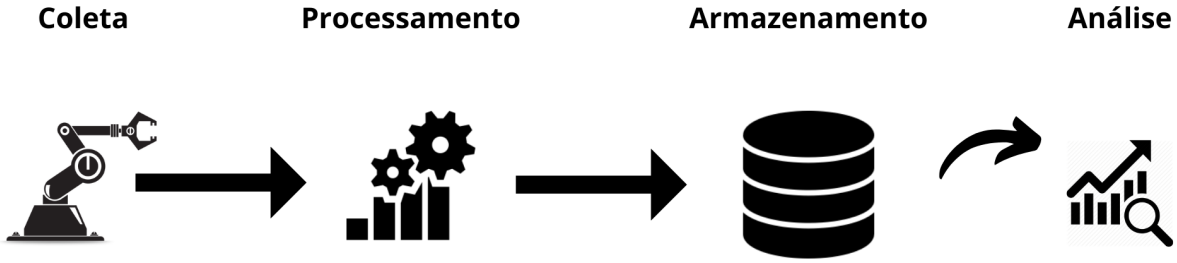


Fonte: Elaborado pelo autor (2021).

Como mostrado na Figura 6, os exemplos de alguns provedores de serviços em nuvem referentes à cada camada estão citados à direita.

Segundo a AWS (2021), é possível ter uma noção de como a nuvem disponibiliza os serviços de análise. Através do serviço *AWS IoT Analytics*, todas as etapas necessárias para analisar dados de dispositivos IoT são automatizadas. Esse serviço coleta dados, processa esses dados e os armazena em um *data store* para serem analisados. A Figura 7 representa esse fluxo de atividades:

Figura 7 – Serviço de nuvem



Fonte: Elaborado pelo autor (2021).

A computação em nuvem veio a ser um meio de terceirizar a computação local, se tornando uma opção séria para muitas empresas. De fato, esse campo emergiu do processamento paralelo com um forte foco no compartilhamento mundial de recursos. Os serviços da nuvem funcionam como um armazenamento completamente encapsulado em uma única unidade, com uma interface disponibilizada aos clientes. Provedores de nuvem como a Amazon e o Google, gerenciam vários *data centers* estabelecidos em diferentes locais no mundo inteiro. Eles podem oferecer à um usuário final, a capacidade de construir um sistema distribuído de área ampla, consistindo assim, em um sistema com uma coleção de máquinas virtuais espalhadas pela Internet (STEEN, 2020).

2.3.2 Conteinerização

Segundo Veritas (2021), a conteinerização é uma solução de virtualização que surgiu mediante à alguns problemas que ocorriam em ambientes virtuais. Por exemplo:

- Incosistência do ambiente;
- Dependência do sistema operacional para execução de aplicativos;
- Dificuldades na replicação de aplicativos.

Portanto, a conteinerização veio como uma solução mais regular, permitindo um uso mais eficiente do sistema operacional. Essa solução tem como principal característica, a execução

de aplicativos em espaços isolados de usuário. Esses espaços são conhecidos como contêineres. Esses podem executar em alguma infraestrutura como a nuvem, sem a necessidade de refatorar o código caso o mesmo seja transferido para um outro ambiente. Uma das tecnologias mais populares na área da conteinerização é o Docker. Essa é uma tecnologia de código aberto, baseada no sistema operacional Linux e que realiza a criação de contêineres através de imagens de aplicações.

2.4 Padrões de Projeto

Os padrões de projeto são definições de como um problema pode ser solucionado. Essas definições se tratam de modelos utilizados e aprovados anteriormente, a fim de se alcançar um aumento de produtividade no desenvolvimento de *software* e contribuindo para a organização e manutenção de projetos (LUTTI, 2018). Segundo a abordagem da Source Making (2021), os padrões de projeto podem ser de três tipos:

- Criação: Se caracterizam em identificar como deve ser feita a instanciação de classes ou a definição de objetos. Exemplos: *Factory Method* para criar uma instância de várias classes derivadas e *Singleton*, onde pode haver somente uma instância de uma classe;
- Estruturais: Definem a composição de classes e objetos. Exemplos: *Composite* para estruturação de objetos simples e compostos em árvores e o *Facade*, onde uma única classe representa um subsistema por inteiro;
- Comportamentais: Consistem em descrever a comunicação entre os objetos. Exemplos: *Command* para encapsular uma solicitação de comando como um objeto *Observer* para notificação de mudanças em algumas classes e o *Strategy*, onde dentro de uma classe, ocorre o encapsulamento de algum algoritmo.

Assim, os padrões de projeto se tornam relevantes por permitirem acelerar o processo de desenvolvimento, além do fato de tornar os códigos mais legíveis.

2.5 Tecnologias habilitadoras

São diversas as tecnologias com diferentes características disponíveis no mercado. Essas tecnologias adaptadas em diferentes cenários são identificadas e caracterizadas nas subseções seguintes.

2.5.1 Aquisição e preparo de dados

Desde que os sistemas de coleta extraem os dados da indústria, esses dados devem ser preparados e disponibilizados para um ambiente externo através de alguns recursos desses sistemas.

2.5.1.1 Coleta

Dois dos sistemas de coleta de dados mais usuais, são:

- *PI System*: Pertencente a companhia OSIsoft, o PI System coleta dados de algumas fontes (como dispositivos da indústria) através de interfaces que utilizam algum protocolo de dispositivo especifico, como Modbus para PLCs. Os dados então, são armazenados no PI server (OSISOFT, 2018);
- *KEPServerEx*: É a plataforma de gerenciamento da empresa Kepware. O KepServerEX extrai informações das fontes de dados e através de uma *Application Programming Interface* (API) usando OPC UA disponibiliza o acesso à esses dados para um servidor (KEPWARE, 2021).

Os dados coletados para os servidores desses sistemas, precisam ser transferidos para um ambiente de análise eficiente. No PI System, através de uma API com uma arquitetura conhecida como *Representational State Transfer* (REST), onde essa utiliza o *Hypertext Transfer Protocol* (HTTP), os dados no PI server são disponibilizados para os serviços de nuvem (OSISOFT, 2021). No KepServerEx, Os dados no servidor são transmitidos para a nuvem através de API REST e protocolo de transmissão *Message Queuing Telemetry Transport* (MQTT). O OPC também pode ser utilizado para essa transmissão (KEPWARE, 2017).

2.5.1.2 Formatos de dados

Os dados contidos nos servidores dos sistemas de aquisição de dados devem ser preparados para transmissão. Para isso, existem diversos formatos possíveis. Segundo Manoochehri (2014), os mais utilizados para a transmissão por meio da rede são os seguintes:

- *Comma-Separated Values* (CSV): Trata-se de um formato onde os dados são bem organizados em linhas únicas. Os dados que são bem representados nesse formato, geralmente são provenientes de servidores e sensores da Web.
- *Extensible Markup Language* (XML): Consiste em um formato de dados de uso generali-

zado para o compartlhamento de grandes massas de informações.

- *JavaScript Object Notation* (JSON): Uma especificação do JavaScript para possibilitar o intercâmbio de dados.
- *Binário*: Dados serializados com um mecanismo extensível do Google, conhecido como *protobuf*. Esses dados serializados são estruturados em uma extensão chamada *.proto*. Assim, um arquivo é criado e compilado para um código fonte gerado em uma linguagem desejada. O código fonte então, realiza a gravação e leitura de dados (SOUZA, 2018).

2.5.1.3 Vantagens e desvantagens

Os formatos identificados para envio de dados são os mais usuais do mercado, porém, é importante considerar suas vantagens e desvantagens como uma forma de classificá-los para cada caso de uso. Portanto, as seguintes informações referentes a esses possíveis formatos de dados identificados foram construídas com base no livro de Manoochehri (2014):

- Vantagens: O CSV se destaca pela facilidade para analisar os dados e por ter um bom suporte de uso, se encaixando desde planilhas até bancos de dados. O XML, por sua vez, contém uma coleção de dados evidentes, que podem ser convertidos em uma variedade de formatos de acordo com as necessidades do trabalho. O JSON é apropriado para transferência de dados, se tornando assim, bastante viável para os programadores e administradores de banco de dados. De acordo com a abordagem realizada em Grossmann (2020), o formato binário é a melhor opção quando o assunto é velocidade de transmissão, uma vez que ocupam menos espaço na rede, evitando assim, uma sobrecarga de dados na transmissão. Além disso, ele fornece abstração de dados independentemente da linguagem ou plataforma de serialização.
- Desvantagens: O CSV não é adequado para dados com grandes volumes, ou seja, se torna complicado modelar estruturas de dados complexas com o CSV. Em alguns casos, o XML não é uma boa opção para os desenvolvedores de software, uma vez que a sobrecarga de dados dessa estrutura se torna alta ao ocuparem mais espaço e, assim, não se tornando conveniente para os programadores. Portanto, esse formato não é adequado para uma análise rápida de dados. O JSON não é simples quanto o CSV e não tem a extensibilidade do XML (pelo fato de não ser apropriado para ser convertido em outros formatos de dados). Com base nas informações em Grossmann (2020), o dado binário não é legível quando comparado aos outros formatos.

2.5.2 Transferência segura de dados

Para os dados formatados existem modelos de comunicação disponíveis que possibilitam a transmissão desses dados para um sistema de processamento e análise.

- REST: Trata-se de uma arquitetura para serviço Web que utiliza endereços de rede para identificação e um protocolo específico para controle de cache, autenticação e conteúdo (KLEPPMANN, 2017);
- *Remote Procedure Call* (RPC): Trata-se de uma chamada de procedimento utilizada por um emissor para estabelecer uma comunicação com um servidor em uma máquina diferente e que pode pertencer à uma rede diferente (GHOSH, 2014);
- Modelos orientados a mensagens: Modelo utilizado para combater a transparência de acesso. Existem situações em que não se pode presumir que o receptor está em execução. Portanto, são utilizadas mensagens ao invés de realizar uma chamada de procedimento para estabelecer uma comunicação (STEEN, 2020).

Além do que foi descrito acima, os modelos de comunicação divergem em mais alguns aspectos. Para isso, é necessário considerar as definições de comunicação síncrona e assíncrona. Na comunicação síncrona, um remetente envia uma solicitação e é bloqueado até que essa seja aceita. Na comunicação assíncrona, um remetente pode continuar executando imediatamente após o envio da mensagem (STEEN, 2020). Os modelos orientados a mensagens são classificados em dois tipos: transiente e persistente. Na comunicação transiente, uma mensagem é armazenada somente enquanto o emissor e o receptor estiverem em atividade. Na comunicação persistente, um emissor envia uma mensagem que pode ser armazenada em algum *middleware* de comunicação e entregue ao receptor quando for preciso. Em outras palavras, o receptor não precisa estar ativo no momento do envio da mensagem (STEEN, 2020).

Existem diversas tecnologias e protocolos associados a esses modelos. Segundo Steen (2020) e Barmin (2021), além da abordagem da Altexsoft (2020), seguem algumas definições abaixo:

- Ambas as tecnologias HTTP 1 e HTTP 2 são protocolos de comunicação;
- gRPC é uma versão mais recente do modelo RPC;
- ZeroMQ se trata de uma biblioteca que fornece abstração na comunicação;
- Sistemas de mensagens são recursos que tornam o processo de troca de dados simples e confiável;
- RabbitMQ e Apache Kafka são considerados sistemas de mensagens;

– RabbitMQ tem suporte ao protocolo MQTT (CLOUDAMQP, 2021).

Com base nas mesmas fontes utilizadas para o levantamento das definições acima, um comparativo entre essas tecnologias e protocolos é revelado na Tabela 1.

Tabela 1 – Comparação entre tecnologias HTTP (1), HTTP 2 (2), ZeroMQ (3), RabbitMQ (4) e Apache Kafka (5)

	Arquitetura	**Operação**	**Prós**	**Contras**
1	REST	Síncrona. Via serviços Web.	Suporta vários formatos (como XML ou JSON) para a troca de dados.	Busca excessiva e insuficiente.
2	gRPC	Síncrona. Via chamadas de procedimento.	Otimiza a carga de dados na rede.	É um modelo com baixa descoberta, além de ser fracamente acoplado.
3	Transiente	Assíncrona. Fila de mensagens.	Tem alta variedade de padrões de mensagens, além de ser rápido.	Uma mensagem é descartada, caso não tenha algum processo ativo para recebê-la.
4	Persistente	Assíncrona. Fila de mensagens.	Tem grande suporte de protocolos. É altamente escalável.	É um modelo não transacional.
5	Persistente	Assíncrona. Fila de mensagens.	Tolerante a falhas e confiável. Adequado para processamento de dados em tempo real.	Não há monitoramento completo. Problemas com número crescente de mensagens.

Fonte: Elaborado pelo Autor (2021).

Como visto na tabela comparativa acima, o protocolo HTTP 1 é muito bem indicado para ser utilizado em interfaces públicas, uma vez que suporta a transmissão de vários tipos de dados (como XML ou JSON). O protocolo HTTP 2 é apropriado em uma transferência leve na rede. A biblioteca ZeroMQ é uma boa opção para trabalhar com vários padrões de mensagens. Os sistemas de mensagens RabbitMQ e Apache Kafka são boas opções de comunicação para um sistema de análise de dados (BARMIN, 2021). Porém, como visto na tabela, ambos diferem

em alguns aspectos. O RabbitMQ, por ter uma uma alta escalabilidade, é uma otima opção para um cenário onde seja necessário lidar com uma quantidade de dados crescentes, porém, pelo o fato de ser um modelo não transacional, não é tão rápido quanto o Apache Kafka. Por último, o Apache Kafka é relevante para uma comunicação rápida de dados, porém, quando o número de mensagens aumenta, ele se comporta de uma maneira menos eficiente.

2.5.2.1 Segurança

De acordo com a abordagem de Steen (2020), para garantir que o tráfego dos dados não seja violado, é necessário habilitar um padrão de segurança para as tecnologias de transmissão de dados. Isso é possível com a utilização do *Transport Layer Security* (TLS). Esse protocolo funciona com a utilização de chaves públicas, chaves privadas e uma chave de sessão para garantir a segurança de cada conexão. Uma conexão TLS é iniciada usando uma sequência conhecida como *handshake*. Nesse procedimento, um servidor envia informações de versão do TLS, configurações de criptografia, dados específicos da sessão e outras informações necessárias para a comunicação. Do outro lado, o usuário utiliza essas informações para autenticar o servidor e continuar com a conexão segura. Assim, ele cria uma chave de sessão e a criptografa com a chave pública do servidor. Essa chave de sessão é enviada para o servidor e esse a descriptografa com a chave privada. Estabelecida essa conexão, todas as mensagens enviadas entre o cliente e o servidor são criptografadas utilizando a chave de sessão.

2.5.3 Armazenamento de dados

Os dados podem ser armazenados em diversas tecnologias de bancos de dados relacionais ou não relacionais. Ambos estão descritos com base na abordagem de Khasawneh *et al.* (2020):

- Bancos de dados relacionais: Fazem uso da *"Structured Query Language (SQL)"* para consultar os dados armazenados. Esses dados possuem um esquema fixo predefinido e são organizados em tabelas. São bancos com escalabilidade vertical.
- Bancos de dados não relacionais: São atribuídos ao termo *"Not Only Structured Query Language (NOSQL)"*. Esses sistemas não exigem um esquema predefinido de dados, além de permitir o armazenamento de diversos tipos de dados que podem ser estruturados, semi-estruturados ou não estruturados. São bancos com escalabilidade horizontal.

Com base nas informações da Microsoft (2020) e Ksiazek (2019), além da abordagem

de Drake e Ostezer (2014), é construído um quadro comparativo entre os bancos de dados relacionais e diversos tipos de bancos de dados não relacionais, conforme segue a Tabela 2.

Tabela 2: Comparativo entre Bancos de dados relacionais (1), Armazenamento de valor chave (2), Banco de dados coluna (3), Armazenamento de documentos (4), Banco de dados de grafos (5) e Banco de dados de séries temporais (6).

	Carga de trabalho	**Tipo de dados**	**Uso**	**Exemplos**
1	Relações impostas por meio de restrições e consultas otimizadas por índices.	Os dados são normalizados. Exigem alta integridade.	Gerenciamento de pedido e estoque.	PostgreSQL e MySQL
2	Dados acessados por chave única. Não há a necessidade de uniões.	Um valor único por chave. Não existe imposição de esquema.	Armazenamento em cache. Gerência de sessões.	Riak e Redis
3	Gravações rápidas e uma alta taxa de transferência com uma baixa latência escalonável.	Os dados são armazenados em uma ou mais famílias de colunas.	Dados de sensor, análise da Web e recomendações.	Cassandra e HBase
4	Inserções são comuns. Não há incompatibilidade de impedância.	Cada documento usa um esquema. Os dados são semi-estruturados.	Gerência de estoque e catálogo de produtos.	CouchDB e MongoDB
5	Relações dinâmicas entre itens de dados. Relações de primeira classe entre objetos.	Nós com relações expostas na linguagem de consulta.	Detecção de fraude, Gráficos sociais e planejamento de organizações.	OrientDB e Neo4j
6	Anexos sequenciais em ordem cronológica. Dados lidos em paralelo.	Dados com carimbo de data e hora usado como chave primária.	Sensor e dados IoT. Telemetria e monitoramento.	InfluxDB e Timescale

Fonte: Elaborado pelo Autor (2021).

Como visto acima, os bancos de dados relacionais e os vários modelos de bancos de dados não relacionais podem ser utilizados para diversas áreas, dependendo se determinado

banco de dados possui as especificações necessárias para se tornar apropriado em certo cenário.

Segundo Drake e Ostezer (2014), o modelo relacional tem sido muito útil para gerenciar dados. No entanto, o modelo relacional se tornou limitado em relação à algumas necessidades que foram surgindo, entre elas: o aumento do tráfego de informações, velocidade de processamento de dados e gerenciamento de dados não estruturados. Como alternativa, o crescimento do modelo NOSQL se tornou iminente.

3 TRABALHOS RELACIONADOS

Para uma compreensão mais sólida desta pesquisa, alguns trabalhos sobre os conceitos habilitadores da Indústria 4.0 são abordados.

O trabalho de Moura *et al.* (2019) apresenta uma estrátegia para a Indústria 4.0, onde os historiadores de dados são integrados à nuvem. Primeiramente, os historiadores de dados são abordados, ressaltando sua importância para o controle industrial. Também, são descritas as suas limitações. Após isso, os conceitos de nuvem em relação a bancos de dados e serviços são abordados. Definido os conceitos, a estratégia então é apresentada, enfatizando as etapas de coleta de dados de séries temporais, armazenamento desses dados em historiadores e réplicas desses dados enviadas para a nuvem. Por último, conceitos importantes sobre a propriedade dos dados são discutidos, ressaltando-os como um requisito importante para a transformação digital.

O trabalho de Atzori *et al.* (2010) aborda uma pesquisa direcionada às diferentes visões do paradigma da Internet das Coisas, onde tecnologias habilitadoras são relatadas e revisadas. Além disso, define as aplicações beneficiadas pela implantação da IoT. Esse trabalho, portanto, foca exclusivamente no entendimento desse paradigma como um todo, ou seja, realiza abordagens sobre objetos inteligentes, tecnologias de internet e a semântica da IoT.

No trabalho de Imen Graja *et al.* (2020) são examinadas abordagens existentes para modelar o CPS, além da descrição dos campos relacionados a esse paradigma. Com base nesse estudo, propriedades relacionadas a esse paradigma são classificadas e tem sua importância discutida em diferentes domínios de aplicação. Esse trabalho foca em estruturar o CPS, abordando seus dispositivos inteligentes interconectados, além de estudar linguagens e formalismos identificados na literatura. No término da pesquisa, conclui-se que são necessários métodos de modelagem genéricos para permitir a especificação e verificação do comportamento do CPS.

O trabalho de Shafiq *et al.* (2019) aborda os componentes críticos da Indústria 4.0. Sua importância e desafios conforme identificados na literatura são apresentados. Além disso, uma estrutura de caso de teste para a implementação da Indústria 4.0 é proposta, onde as vibrações do corpo de uma máquina são monitoradas. O sistema cobre camadas que vão desde a aquisição, preparo, transmissão, processamento e suporte à decisão. Foi observado que um dos principais desafios que as empresas encontram na implantação da indústria 4.0, é a sincronização das máquinas convencionais com o ambiente virtual. Por fim, na seção de resultados, as análises são expostas graficamente, onde o comportamento atual da máquina é comparada com o registro

de informações referentes ao comportamento anterior da mesma.

3.1 Semelhanças e diferenças

É possível observar nos trabalhos apresentados acima, vários procedimentos, onde alguns tratam de coleta, preparo e monitoramento de dados com processamento local e externo, outros tratam da integração do meio virtual com o meio físico, além de abordar sobre protocolos de comunicação para isso e também é abordado o fluxo de todas essas etapas. Entretanto, esses trabalhos não focam em todas essas etapas de forma abrangente, onde alguns procedimentos são brevemente descritos, mas não explorados.

Este trabalho se assemelha ao de Shafiq *et al.* (2019), pois nele há um pipeline de etapas elaboradas, desde a coleta ao armazenamento de dados para a análise e tomada de decisões na Indústria 4.0. Esta pesquisa não se destina apenas à abordagem breve desses procedimentos, mas também, ao levantamento das tecnologias utilizadas em cada etapa.

Conforme mostra a Tabela 3, é possível observar o que cada trabalho explora e com isso comparar com o que é abordado neste trabalho.

Tabela 3: Comparativo entre os trabalhos

Trabalhos por abordagem	**Coleta de dados**	**Preparo de dados**	**Comunicação confiável**	**Armazenamento para análise**
(MOURA *et al.*, 2019)	✓			✓
(GRAJA *et al.*, 2020)	✓		✓	
(ATZORI *et al.*, 2010)	✓		✓	
(SHAFIQ *et al.*, 2019)	✓	✓		✓
Este trabalho	✓	✓	✓	✓

Fonte: Elaborado pelo Autor (2021).

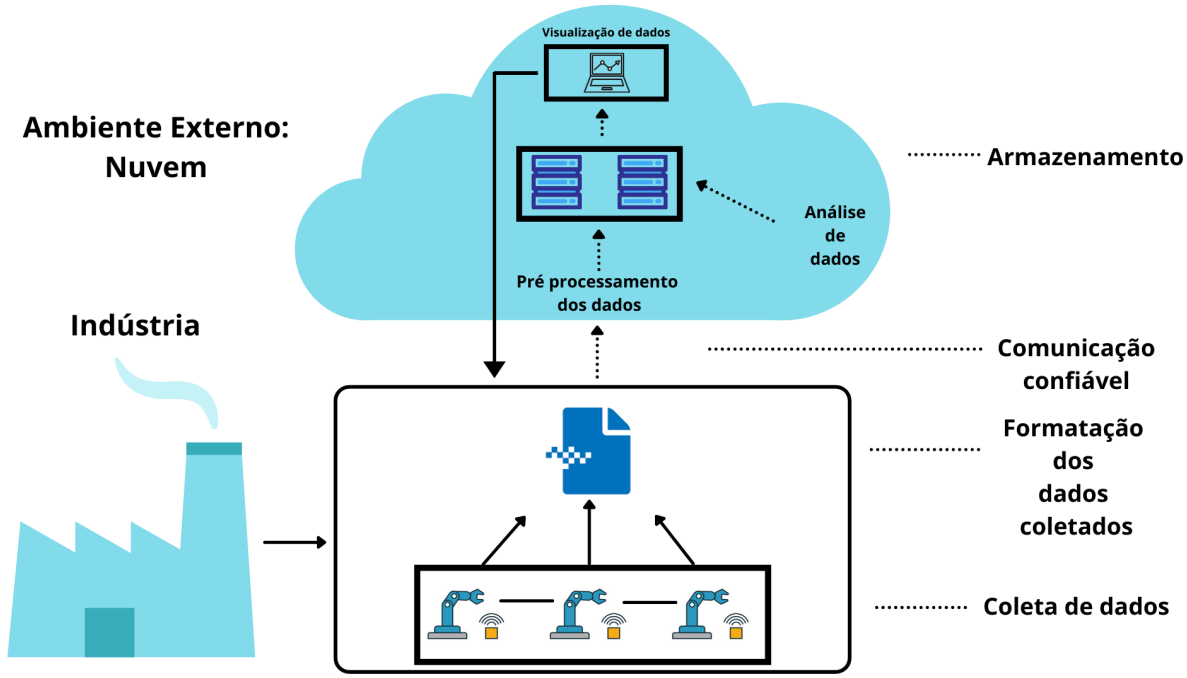
4 PROPOSTA

Este capítulo apresenta uma ideia de como habilitar a coleta, formatação, comunicação confiável e armazenamento para a implantação da indústria 4.0. Está estruturado da seguinte forma: Na primeira seção é construído um *pipeline* do fluxo de etapas deste projeto; na segunda seção é abordado um pipeline representantivo do fluxo das possíveis tecnologias integradas e na última seção é mostrado o desenvolvimento da ideia deste trabalho.

4.1 Fluxo do projeto

O fluxo da proposta deste trabalho é representado na Figura 8.

Figura 8: Fluxo da proposta



Fonte: Elaborado pelo autor (2021).

Como mostra a Figura 8, um conjunto de etapas de solução referentes à proposta deste trabalho é apresentado. As seguintes informações descrevem esse conjunto de etapas:

1. Coleta de dados: Define como os dados serão coletados através de dispositivos da indústria local;
2. Formatação dos dados coletados: Etapa responsável pela preparação dos dados para serem transmitidos da forma mais viável possível;
3. Comunicação Confiável: Identifica o padrão de transmissão, ressaltando tecnologias existentes com o objetivo de proporcionar uma transmissão segura para um ambiente

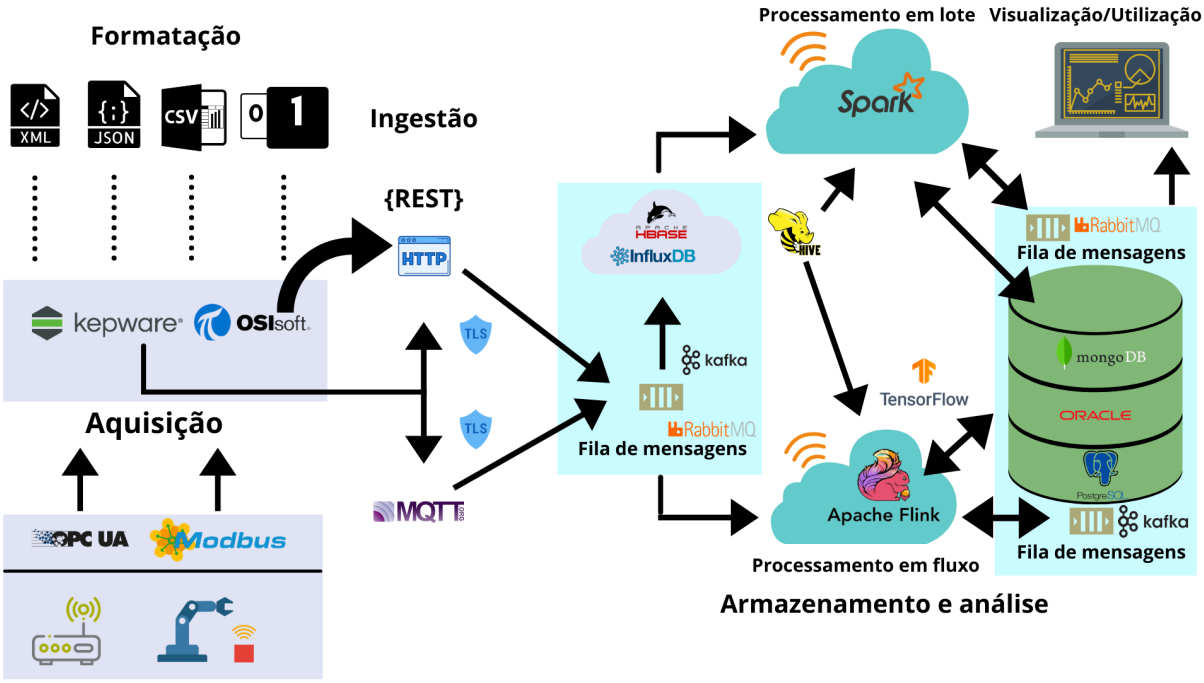
externo;

4. Armazenamento: Armazena os dados na nuvem, disponibilizando-os para pré-processamento e análise. Os resultados dessas ações são visualizados nesse ambiente externo.

4.2 Fluxo das tecnologias habilitadoras

Nesta seção, as tecnologias identificadas e comparadas na Fundamentação Teórica deste trabalho, serão representadas em um *pipeline* de forma integrada. O fluxo desse *pipeline* foi baseado na Figura *"Big data architecture using open source technologies"* de Gupta (2020). O fluxo citado pode ser observado na Figura 9.

Figura 9: Fluxo de tecnologias



Fonte: Elaborado pelo autor (2021).

Na aquisição, as companias Kepware ou OSIsoft fornecem softwares que coletam dados das máquinas industriais. Para isso, utilizam protocolos como OPC UA ou Modbus. Esses dados podem ser formatados em XML, JSON, CSV ou em formato Binário. Com os dados formatados, os sistemas de aquisição realizam a transferência desses dados. A OSIsoft utiliza REST HTTP e o Kepware utiliza tanto o REST HTTP quanto o MQTT. O TLS é acoplado nessa transferência. Esses dados são transmitidos em fila de mensagens, onde por meio do Apache Kafka (para mensagens em fluxo) ou RabbitMQ (para mensagens em lote), são disponibilizados para bancos, como o HBase ou o InfluxDB. Ambos na nuvem. Os dados na nuvem são lidos para

análise através da linguagem de consulta Hive. Esses dados podem ser analisados utilizando duas ferramentas: Apache Flink ou Spark. Além disso, uma biblioteca de aprendizado de máquina, conhecida como Tensor Flow pode ser habilitada na análise. Em seguida, os dados analisados são disponibilizados através de fila de mensagens do Apache Kafka ou RabbitMQ para o banco de dados MongoDB ou para um banco de dados relacional, como PostgreSQL ou Oracle. Esses dados também podem ser escritos diretamente nesses bancos. Por último, os dados nesses bancos são disponibilizados em um painel de visualização para serem utilizados em alguma tomada de decisão.

4.3 Implementação

Para a implementação do cenário descrito na proposta deste trabalho, um serviço de coleta e um de análise que antes se comunicavam por meio de operações em banco de dados e de forma síncrona, foram adaptados para se comunicarem de forma assíncrona por meio do sistema de mensagens Apache Kafka e utilizando para isso, a biblioteca kafka-python (POWERS, 2020). Também adaptado ao Apache Kafka, um serviço de pré-processamento de dados foi criado. Os serviços de pré-processamento e análise foram aplicados à padrões de projeto, além de programados de forma reativa. Esses três serviços estão na linguagem de programação Python na versão 3.8.3[1] e além disso, foi utilizada uma ferramenta chamada MatrikonOPC Explorer para simular a geração de valores referentes à dados de sensores, realizando assim, testes no servidor OPC (MATRIKON, 2022). Para o armazenamento dos dados, o banco de dados MongoDB foi utilizado. Também houve a utilização do serviço de nuvem da AWS para hospedar e executar aplicações, além de permitir adicionar um certificado de segurança aos dados. Os três serviços Python serão descritos a seguir.

4.3.1 Coletor OPC

Esse serviço, como descrito acima, coleta os dados do simulador via protocolo OPC, serializa esse conjunto de dados para o formato JSON e o envia para o Apache Kafka, no tópico *"raw_values"* (do português: dados brutos). Esse, assim, é denominado pelo fato de que os dados são coletados da indústria antes de haver algum tratamento a fim de aumentar a eficiência da análise desses dados. Um exemplo de dados JSON para esse caso é mostrado na Figura 10.

[1] https://www.python.org/downloads/release/python-383/

Figura 10: Exemplo JSON


```
"{\"timestamp\": \"2022-01-17 17:00:30\", \"read\": {\"RD1_MD_VRM01_PID_MILL_FEED\": 32585.0}, \"predicted\": {}}"
```

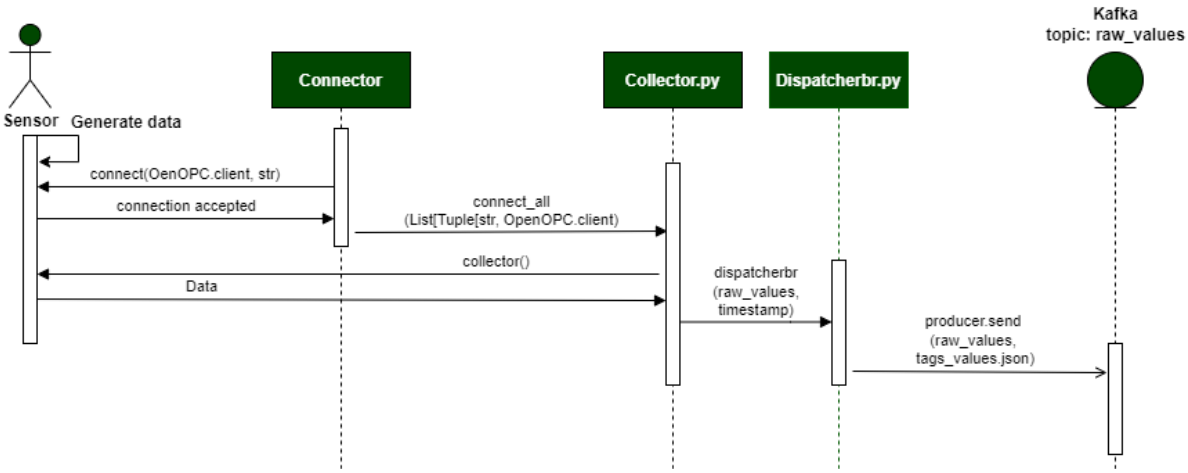

Fonte: Elaborado pelo autor (2021).

Para cada rótulo desse conjunto de dados:

- *timestamp*: Representa a marca de tempo em que cada conjunto de dados é salvo no Apache Kafka, em formato de ano, mês, dia, hora, minuto e segundo;
- *read*: Representa as informações de máquina propriamente ditas, identificando cada atributo de máquina e com valores associados à elas;
- *predicted*: Representa as predições que serão feitas posteriomente sobre esse conjunto de dados.

Um esquema que mostra o fluxo de funcionamento desse serviço, é ilustrado na Figura 11 a seguir como um diagrama de sequência.

Figura 11: Fluxo do serviço de coleta



Fonte: Elaborado pelo autor (2021).

Cada componente do fluxo acima será explicado a seguir, onde o componente que não possuir uma classe, será identificado como um arquivo com a extensão *.py*. Essa regra é válida para todos os outros serviços.

4.3.1.1 Connector

O arquivo connector é responsável por gerenciar a conexão do cliente com um servidor da indústria, utilizando o protocolo de comunicação OPC e definindo 5 tentativas de conexão ao máximo. Caso as 5 tentativas falhem, há um tratamento de erro para uma mensagem informando falha na conexão, onde essa é exibida no log de execução.

4.3.1.2 Collector

Estabelecida a conexão OPC, a coleta se torna possível. Nesse arquivo, os dados são coletados de 30 em 30 segundos de acordo com a marca de tempo atual. Assim como na conexão, ocorre no máximo 5 tentativas de coleta dos dados, onde caso o limite de tentativas de coleta para certo dado for ultrapassado, esse não é coletado, havendo também um tratamento de erro para exibir uma mensagem no log de execução e o próximo dado a ser coletado terá mais 5 tentativas. Esses dados a serem coletados estão armazenados no MongoDB. A Figura 12 mostra um exemplo de um esquema com nome e identificador de um atributo a ser coletado.

Figura 12: Nome e indentificador

```
tag_name: "RD1_MD_VRM01_PID_MILL_FEED"
tag_id: "Random.EIP.PLC01.QXR_RD1VRM01I02_MODE"
```

Fonte: Elaborado pelo autor (2021).

Quando o atributo é lido, a informação sobre o dado é adquirida. Os dados coletados são passados para o arquivo que será mencionado a seguir.

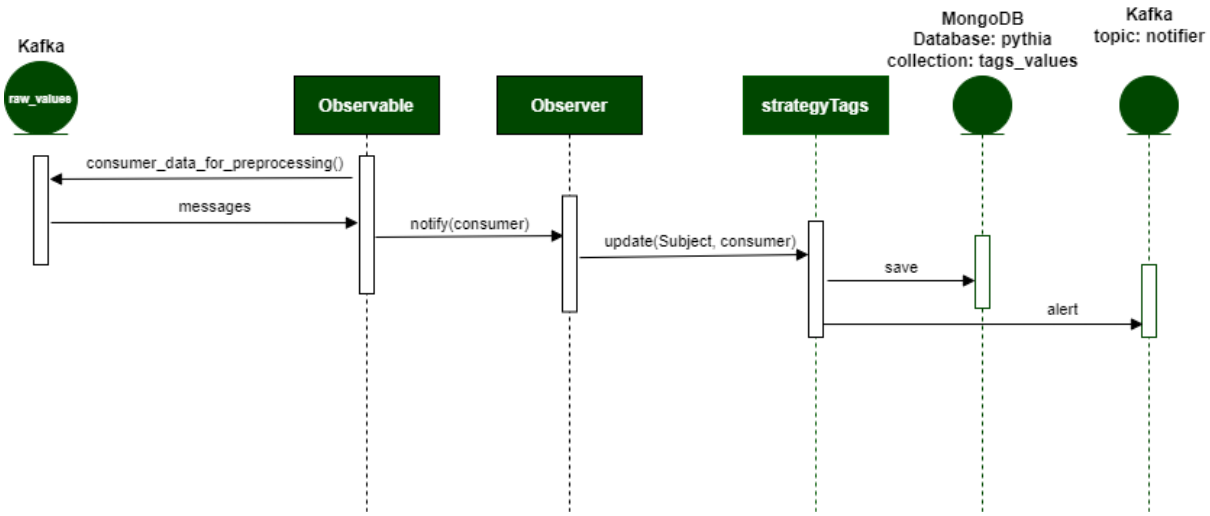
4.3.1.3 Dispatcher.py

Como descrito acima, os dados coletados são mapeados em três rótulos: *timestamp, read e predicted*. Assim, um serializador é definido e adicionado ao produtor do Apache Kafka para ser possível converter os dados em JSON no momento do envio desses para o tópico *"raw_values"* do sistema de mensagens.

4.3.2 Processador

Os dados que são publicados na fila do Apache Kafka, são consumidos para serem tratados. Essa é a principal funcionalidade desse serviço: o Pré-processamento dos dados. Esse serviço foi construído de forma reativa e foi estruturado em dois padrões de projeto: Observer e Strategy. Os componentes desse serviço serão explicados a seguir com base nos padrões de projeto aplicados. A Figura 13 revela um diagrama de sequência para o fluxo de funcionamento desse serviço.

Figura 13: Fluxo do serviço de pre-processamento



Fonte: Elaborado pelo autor (2021).

Para o entendimento desse fluxo, cada componente da figura acima será explicado a seguir.

4.3.2.1 Observer

Esse padrão consiste em duas interfaces: *Observable* e *Observer*, conhecidos respectivamente no português, como Observador e Observável. O *Observable* realiza a recepção das mensagens vinda do Apache Kafka e avisa ao *Observer*, onde esse engatilha de forma reativa, as ações necessárias para os dados que chegam do tópico. A seguir, as funcionalidades implementadas serão descritas, onde os quatro primeiros métodos remetem ao *Observable* e o último, ao *Observer*.

- *attach(Observer)*: Esse método adiciona observadores com uma funcionalidade nativa do Python chamada *append()*.
- *detach(Observer)*: Caso necessário, esse método exclui observadores com uma funcionalidade nativa do Python chamada *remove()*.
- *notify(consumer)*: Inicialmente o estado da classe que implementa o *Observable* é definida como "sem mensagem". Nesse método, ao chegar uma mensagem do Apache Kafka, o estado muda para "com mensagem" em cada observador.
- *data_pre_processing_logic()*: Se trata da aplicação da lógica do negócio. Os dados são recebidos com o consumidor do Apache Kafka. Esse consumidor realiza uma assinatura no tópico *"raw_values"* e notifica o observador da chegada de cada mensagem invocando o método *notify(consumer)* no qual é descrito acima.

– *update(Subject, consumer)*: Conforme os observadores são notificados, um conjunto de ações é acionado sobre os dados que chegam do Apache Kafka. Esse conjunto de ações é definido com o padrão *Strategy*.

4.3.2.2 Strategy

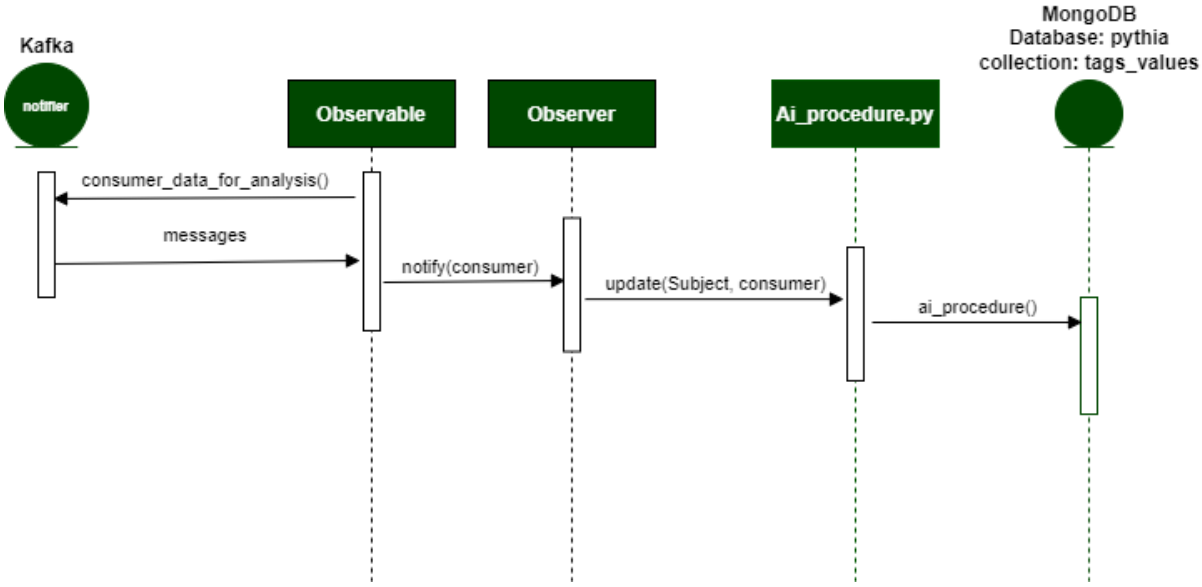
Esse padrão realiza ações específicas para o preparo dos dados de determinada fonte, de forma a tornar o código genérico, onde caso se queira trabalhar com outras fontes de dados em algum outro momento, se torna necessário apenas adicionar uma nova estratégia para determinada fonte. Assim, todos os outros arquivos do serviço ficariam inalterados de certa forma. A funcionalidade do *Strategy* aplicada ao serviço em questão, é explicada a seguir.

– *getBeforeProcess()*: Define o conjunto de ações sobre os dados. Primeiramente, dois pré-processamentos são aplicados aos dados. Esses pré-processamentos são definidos em um arquivo separado chamado *eventProcess* e instanciados na função *getbeforeProcess()*. O primeiro pré-processamento acontece caso o dado *RD1_MD_VRM01_ON_OFF* vier como 0.0 ou *False*. Nesse caso, todas os outros dados do rótulo *read* receberão como valor, a string *"NA"* de *"Not Available"*. O segundo pré-processamento realiza a média entre os dados *RD1_PV_VRM01_POSITION_ROLLER_1* e *RD1_PV_VRM01_POSITION_ROLLER_2*, retira esses do conjunto de dados e armazena o valor da média em um novo dado chamado *RD1_MD_AI_BED_HEIGHT*. Por último, os dados tratados são enviados para o MongoDB na coleção *"tags_values"* (do português: valores de tags) e para cada conjunto de dados enviado, uma mensagem informando a chegada desses é publicada no tópico *"notifier"* (do português: notificador) do Apache Kafka.

4.3.3 Inteligência Artificial Simulada

De forma similar ao Processador, esse serviço também é aplicado ao padrão de projeto *Observer*, funcionando da seguinte forma: O observável fica escutando as mensagens do tópico *"notifier"* no Apache Kafka e quando chega uma mensagem informando que existe um novo dado, o observador é notificado e um conjunto de ações específicas é acionado de forma reativa para, assim, engatilhar a predição final sobre os dados. O fluxo do diagrama de sequência desse serviço é mostrado a seguir na Figura 14.

Figura 14: Fluxo do serviço de análise



Fonte: Elaborado pelo autor (2021).

As funções contidas no arquivo *Ai_procedure.py* serão explicados a seguir.

- *fake_dataframe(timestamp)*: A estrutura da predição é construída e retornada como um Dataframe;
- *ai_procedure()*: É onde está presente todas as ações engatilhadas pela classe observadora. Essas ações são:
  - *get_last_n_read_values(number, reverse)*: Quando o *ai_procedure* é disparado, a Inteligência Artificial Simulada vai no MongoDB, consulta se existem dados de máquina armazenados e retorna esses dados lidos com marcas de tempo em ordem descendente.
  - *next_time_to_predict(dataframe)*: A marca de tempo do último conjunto de dados de máquina armazenado no banco, deve ser retornada para que um novo conjunto de dados de máquina sejam armazenado 30 segundos à frente. Assim, durante esse intervalo alguma ação é realizada.
  - *set_predictions()*: De forma assíncrona, essa instrução realiza a ação mencionada na função acima em cerca de 30 segundos antes do próximo conjunto de dados de máquina ser armazenado no banco. Essa ação é a predição simulada definida em *fake_dataframe()*
  - *set_last_time()*: Apenas define o tempo da última predição realizada.

5 TESTES E RESULTADOS

É apresentado neste capítulo, o cenário da Indústria 4.0 que foi implementado, a estrutura desse cenário, a infraestrutura necessária para os testes, a execução e os resultados.

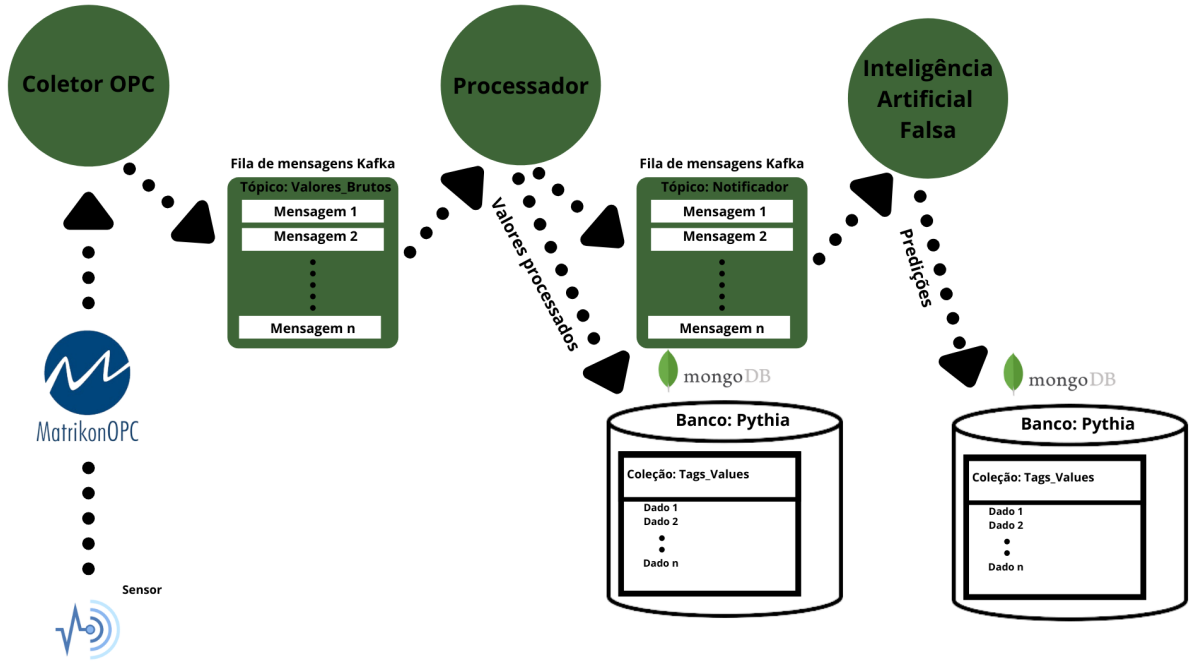
5.1 Cenário

Os dados trabalhados são de um moinho para fabricação de cimento. Esses dados são vistos como *Tags* que representam os atributos dessa máquina, como por exemplo: temperatura, umidade, nível de argila, minérios e entre outros. São 152 atributos ao todo.

5.2 Estrutura do cenário

O cenário foi implementado e configurado inicialmente em uma máquina local de sistema operacional *Windows*. Essa solução se estendeu até um ambiente em nuvem para a realização dos testes, onde a Figura 15 revela a interação entre serviços e ferramentas no cenário.

Figura 15: Interação entre os componentes



Fonte: Elaborado pelo autor (2022).

O simulador MatrikonOPC Explorer simula uma entrada de dados vinda de sensores industriais, onde esses dados são coletados via protocolo OPC pelo primeiro serviço. Após isso, o Apache Kafka que está contido em um conteiner Docker, é utilizado como principal tecnologia

de comunicação entre os serviços e ao final, os dados são visualizados no MongoDB.

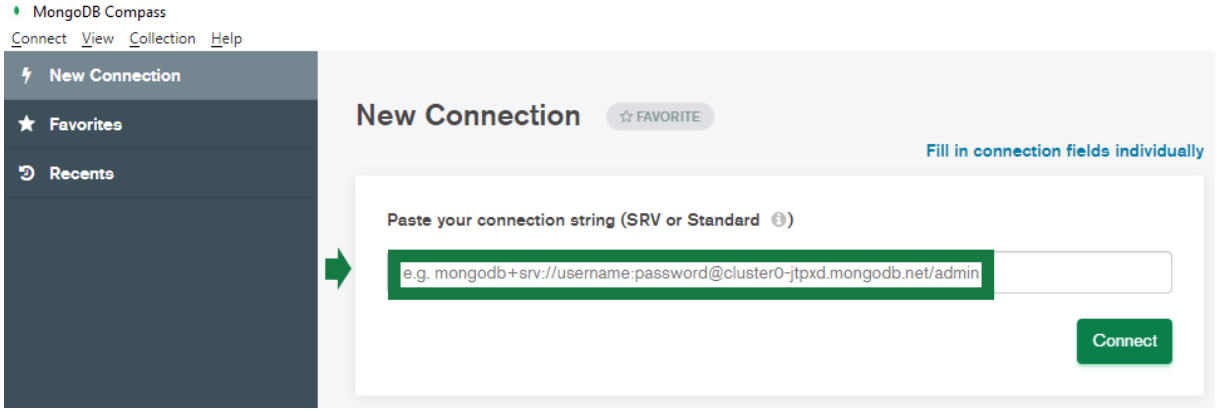
5.3 Infraestrutura necessária

Esta seção apresenta os componentes que foram necessários para executar o cenário.

5.3.1 MongoDB Local

Como mencionado na proposta deste trabalho, mais especificamente visto na Figura 12, os dados de máquina foram armazenados em um mongoDB local para que, assim, se tornasse possível coletá-los. Foi possível utilizar o MongDB localmente na versão *Community*[1], de onde uma instância desse banco foi instalada na máquina, onde assim, se tornou possível armazenar esses dados. Conforme mostra a Figura 16, a visualização dos dados locais se tornou possível por meio da ferramenta MongoDB Compass, que também vem como um pacote adicional na instalação do servidor (PAPIERNIK, 2021).

Figura 16: Conexão com o banco

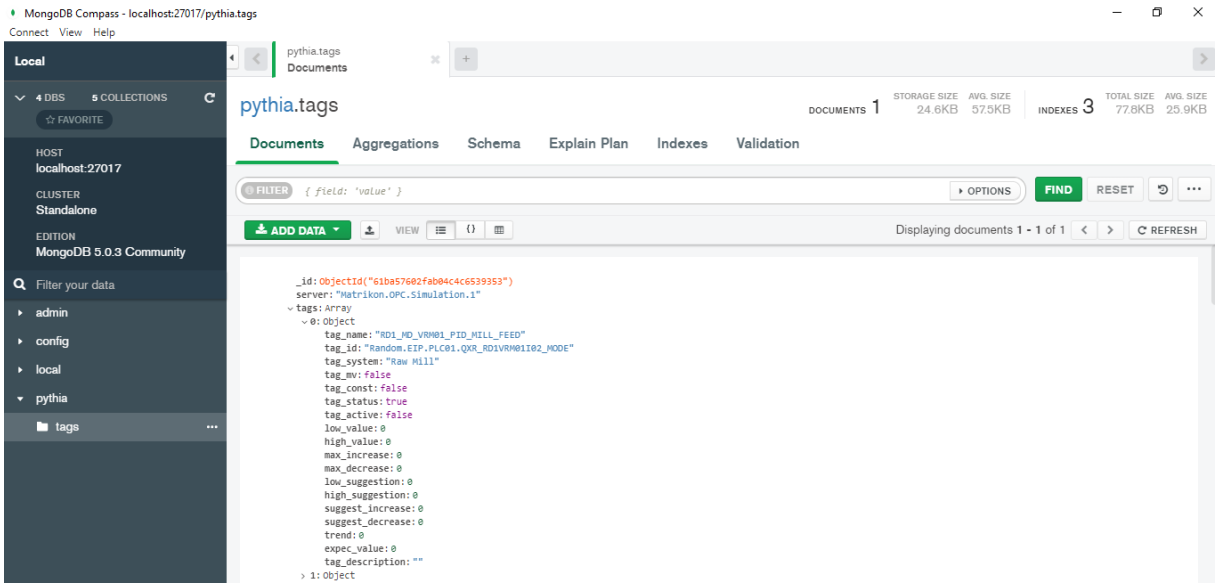


Fonte: Elaborado pelo autor (2022).

A interface acima possui a função de se conectar ao banco, onde o campo indicado vem preenchido automaticamente com as informações de conexão genérica do próprio banco ou pode ser personalizado com a inserção de uma *string* de conexão. Após a conexão e fora do MongoDB Compass, um arquivo de configuração chamado *"run_migrations.bat"* foi executado para armazenar os dados de máquina no MongoDB para a coleta conforme a Figura 17.

[1] https://www.mongodb.com/try/download/community

Figura 17: Dados armazenados para a coleta



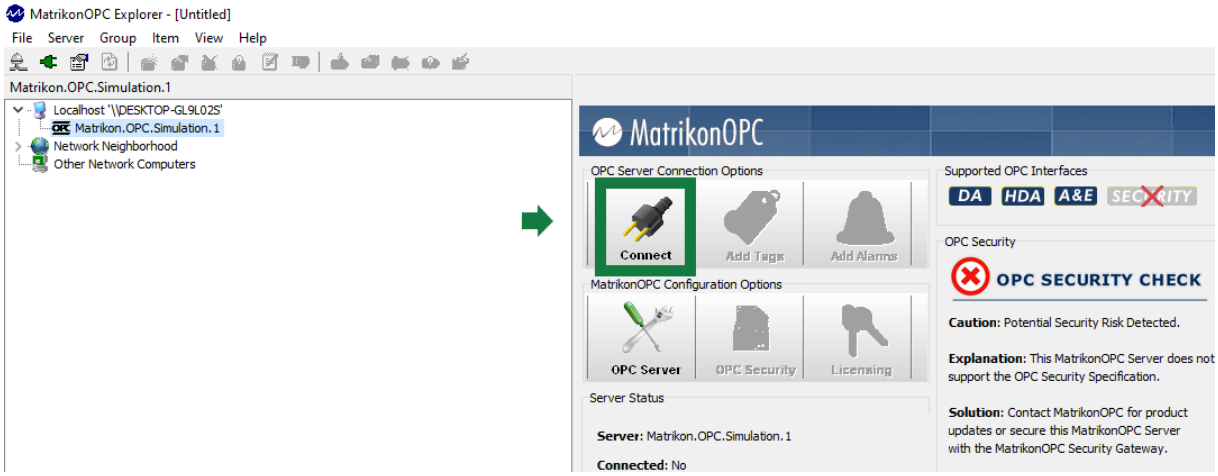
Fonte: Elaborado pelo autor (2022).

Os dados no MongoDB da Figura 17 estão armazenados em uma base de dados chamada *"pythia"*, na coleção *"tags"*. Assim, os dados estavam prontos para serem coletados.

5.3.2 MatrikonOPC Explorer

Os valores referentes aos dados de máquina foram gerados no simulador MatrikonOPC Explorer por meio da conexão desse com o OPC Server. Foi necessário realizar um registro[2] para a instalação dessa ferramenta na máquina local. A conexão com um servidor para gerar esses valores é ilustrada na Figura 18 a seguir.

Figura 18: Conexão com o servidor



Fonte: Elaborado pelo autor (2022).

[2] https://www.matrikonopc.com/downloads/176/software/index.aspx

Após a Figura 18 acima, foi realizado um conjunto de etapas.

– O simulador Matrikon.OPC.Simulation.1 foi selecionado;
– Foi acionado a botão *"Connect"*;
– Foi acessado a sequência:
  1. *File*;
  2. *Open*;
  3. caminho do arquivo;
  4. TAGS-SIMULACAO.XML.

Assim, foi possível gerar valores randômicos de máquina por meio do simulador.

A seguir, a constituição do arquivo TAGS-SIMULACAO.XML é apresentada com um exemplo de 14 *tags*.

Figura 19: Exemplo de Tags simuladas

```
<?xml version="1.0"?>
- <Session>
  - <Hostname RemoteHost="\\DESKTOP-01IVP5U" Remote="0">
    - <Server GroupCount="1" Connected="1" Name="Matrikon.OPC.Simulation.1">
      - <Group Connected="2" Name="Group0" ItemCount="14" PercentDeadband="0,00" TimeBias="0" ReqUpdateRate="1000" Active="-1">
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_H2O_RD1_ARGICAL</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_H2O_RD1_ARGILA_AT</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_H2O_RD1_ARGILA_DECAP</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_H2O_RD1_ARGILA_MT</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_H2O_RD1_CALCARIO</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_H2O_RD1_MINERIO</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1AWF01_TypeProduct</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1AWF01H01</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1WF02_TypeProduct</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1WF02H01</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1WF03_TypeProduct</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1WF03H01</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1WF04_TypeProduct</Item>
          <Item Active="-1" ReqDataType="0" AccessPath="">Random.idmzpisrv01\QXR_RD1WF04H01</Item>
        </Group>
      </Server>
    </Hostname>
  </Session>
```

Fonte: Elaborado pelo autor (2022).

O arquivo XML visto na Figura 19 acima possui uma sessão com um *host* remoto, onde um servidor é definido. Esse servidor possui um grupo. Dentro desse grupo os atributos de máquina estão presentes como itens.

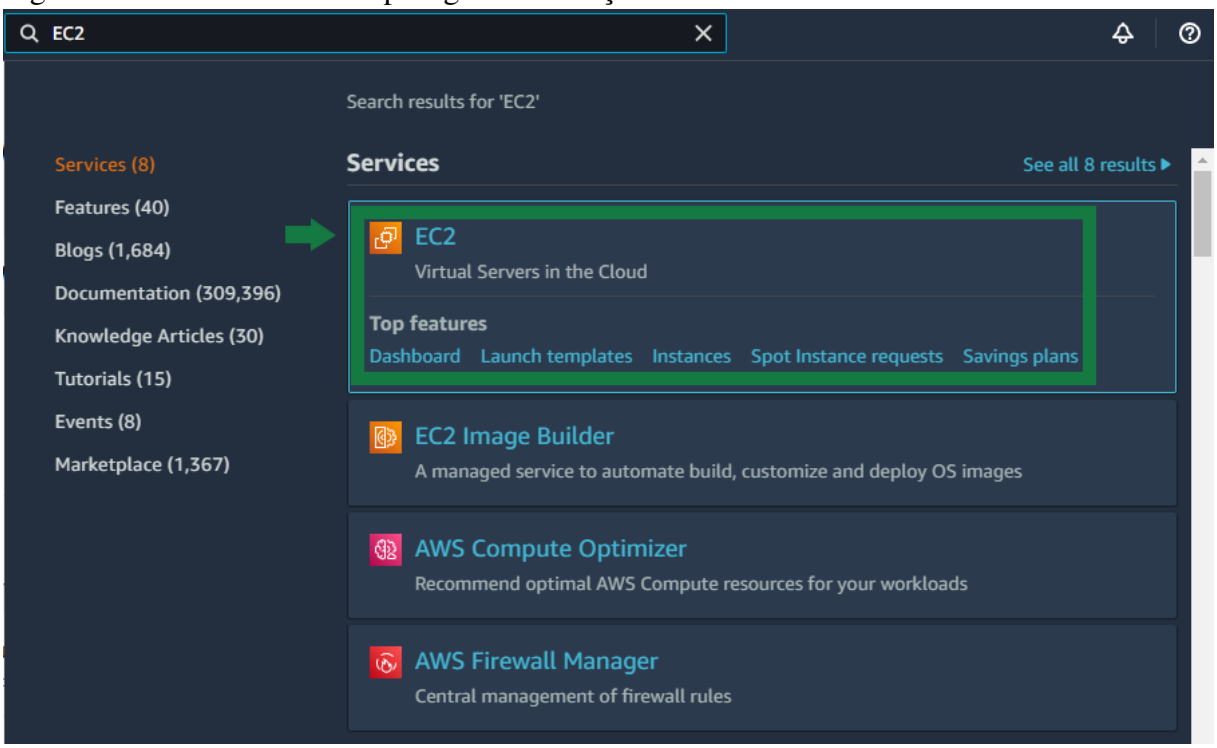
5.3.3 Ambiente externo

Foi utilizado o provedor de serviços de nuvem da AWS para hospedar tanto o sistema de mensagens quanto os serviços de pré-processamento e análise de dados. Além disso, foi possível adicionar uma camada de segurança aos dados na rede. Para a configuração de nuvem, primeiramente foi necessário um cadastro[3]. Após o cadastro, a região de São Paulo foi

[3] https://portal.aws.amazon.com/billing/signup?type=enterprise/start

selecionada para hospedar as aplicações. Feito isso, o termo *"Elastic Compute Cloud (EC2)"* foi buscado na barra de pesquisa da tela inicial, confome segue a Figura 20.

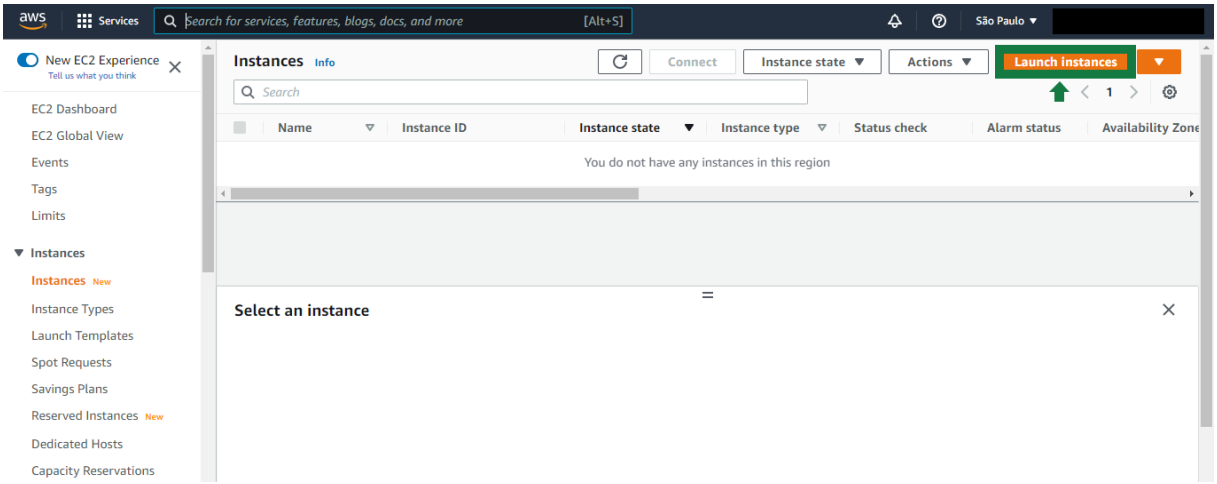
Figura 20: Plataforma de hospedagem e execução



Fonte: Elaborado pelo autor (2022).

A plataforma EC2 permite alugar computadores virtuais para hospedar e executar aplicações. Em seguida, a opção *"Launch instances"* foi acionada, conforme revela a Figura 21.

Figura 21: Criar instância



Fonte: Elaborado pelo autor (2022).

Feito isso, um conjunto de etapas foi realizada para criar uma instância do servidor.

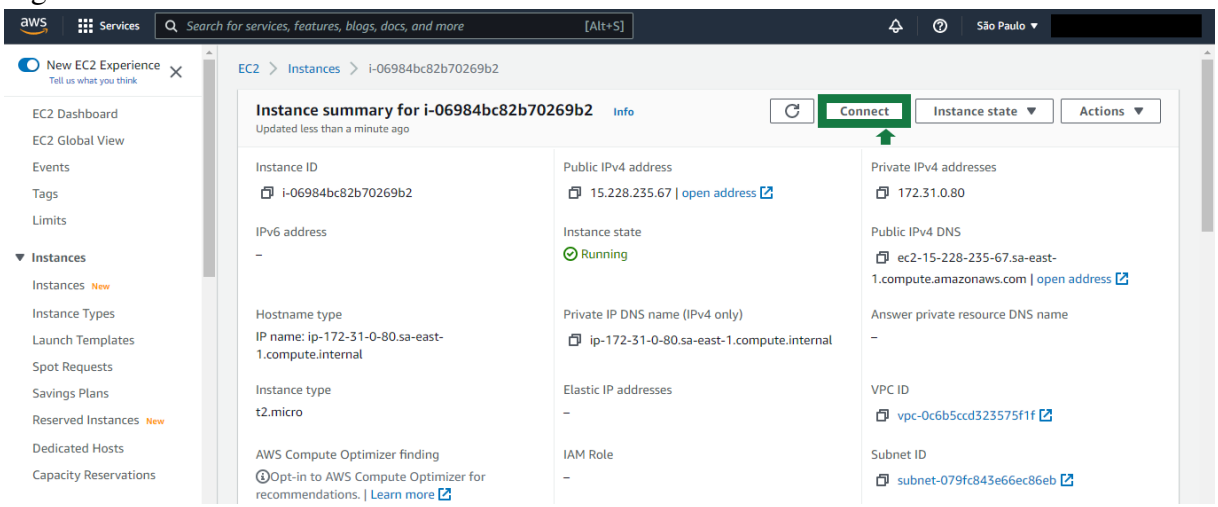
– Na nova tela, o termo *"Elastic Compute Service (ECS)"* foi buscado. Esse termo se refere

à um serviço que permite a execução de aplicativos na nuvem como contêineres;

– Na tela seguinte, várias opções de imagens de servidores para esse serviço estavam disponíveis. Portanto, foi escolhido uma imagem de servidor chamada *"Amazon ECS-Optimized Amazon Linux 2 AMI"*;
– Selecionada a imagem de servidor, foi definido uma capacidade de um 1 *gigabyte* de memória, apenas para testes.
– Por último, foi criado um par de chaves Rivest-Shamir-Adleman (RSA) para a máquina local como um arquivo *.pem*, onde por meio de terminal de comando, esse par de chaves permitiu a conexão da máquina local com a instância criada na nuvem para que, assim, o serviço pudesse ser utilizado (COBB, 2022).

Assim, foi possível visualizar a instância criada, conforme mostra a Figura 22.

Figura 22: Instância criada



Fonte: Elaborado pelo autor (2022).

Na Figura 22 acima, a opção *"Connect"* contém a *string* de conexão da instância criada. Portanto, para a conexão, foram realizadas as seguintes etapas:

– O diretório onde o arquivo *.pem* se encontra foi acessado;
– Um terminal *Powershell* do *Windows* foi aberto;
– A *string* de conexão foi inserida, utilizando o protocolo *Secure Shell* (SSH) para estabelecer segurança na comunicação com o servidor (GONÇALVES, 2021).

Assim, se tornou possível utilizar o servidor criado, conforme mostra a Figura 23, onde o acesso ao servidor da nuvem foi estabelecido.

Figura 23: Acesso ao servidor da nuvem

Fonte: Elaborado pelo autor (2022).

5.3.3.1 Pré-Processamento e análise

Ambos os serviços de pré-processamento e análise foram armazenados em um repositório de uma plataforma de hospedagem de código. Dentro do servidor da AWS, essa plataforma foi instalada com o comando *"sudo yum install git"*. Após isso, o repositório dos serviços foi clonado dentro do servidor da AWS com o comando *"git clone <Endereço do repositório>"*. Com isso, os serviços foram hospedados na nuvem, conforme mostra a Figura 24.

Figura 24: Serviços na nuvem

Fonte: Elaborado pelo autor (2022).

Na nomenclatura dos serviços acima, ambos o Processador e a Inteligência Artificial Simulada descritas na proposta deste trabalho, se referem respectivamente aos seus termos do inglês: *"Processor"* e *"Simulated AI"*.

5.3.3.2 Apache Kafka

Por questões de praticidade, o servidor do Apache Kafka foi colocado em um contêiner do Docker dentro da nuvem, através do arquivo *docker-compose.yml*, de onde está contida a configuração de imagens do Apache Kafka. O conteúdo desse arquivo pode ser encontrado no Apêndice A deste trabalho. Portanto, as seguintes etapas foram realizadas para a utilização do Apache Kafka na nuvem:

– De forma similar aos serviços, o servidor de nuvem foi acessado no terminal e ao entrar no

diretório dos serviços, um editor de texto foi instalado com o comando *"sudo yum install nano"*;

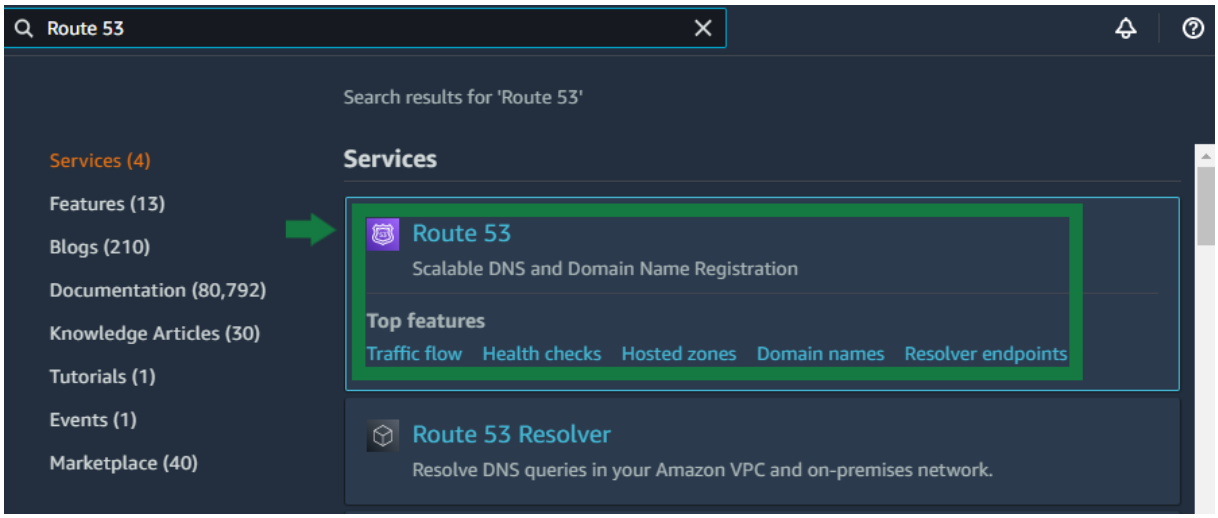
- Com o editor instalado, foi possível criar um arquivo de configuração dentro da pasta do Apache Kafka com o comando *"nano docker-compose-yml"* para que, assim, o conteúdo das imagens pudesse ser adicionado ao arquivo recém-criado;
- Com o arquivo preenchido, foi necessário habilitar o comando responsável por subir as imagens desse arquivo no contêiner;
- Para isso foi instalado o orquestrador de comandos do Docker no servidor: o Docker Compose[4];
- Com o orquestrador instalado, foi necessário alocar mais memória ao servidor de nuvem[5]. Para este trabalho, foram alocados 16 *gigabytes* de memória ao servidor.

Após todas essas etapas concluídas, se tornou possível a utilização do sistema de mensagens no servidor da nuvem.

5.3.3.3 TLS

Para criar o certificado TLS, foi necessário a obtenção de um domínio disponível. Para isso, o termo *"Route 53"* foi buscado na barra de pesquisa da tela inicial da AWS, conforme segue a Figura 25.

Figura 25: Plataforma para domínio



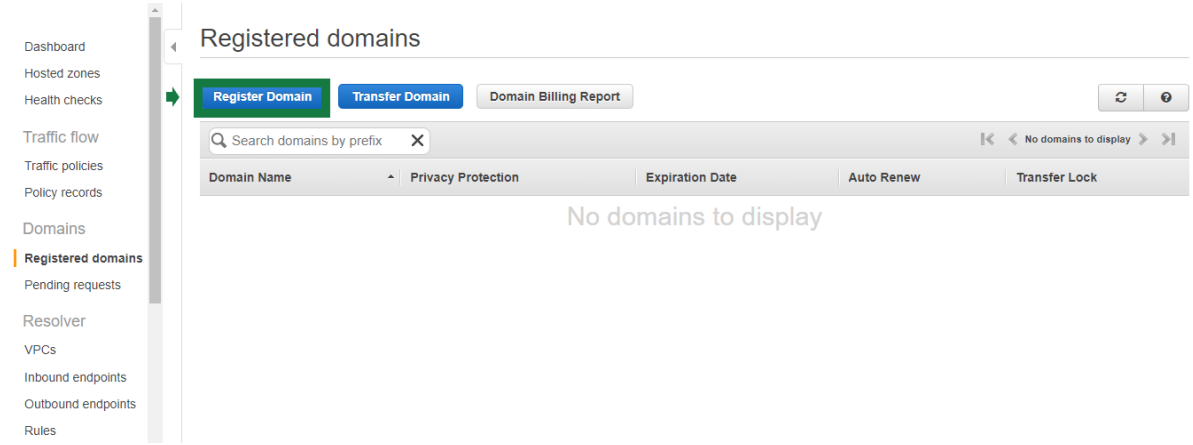
Fonte: Elaborado pelo autor (2022).

A plataforma *"Route 53"* permite registrar algum domínio, onde esse foi usado para a

[4] https://docs.docker.com/compose/install/
[5] https://pplware.sapo.pt/linux/dica-linux-como-aumentar-a-memoria-swap-do-sistema/

implantação da camada de segurança dos dados. Ao entrar no painel da plataforma, foi acionado a opção *"Register Domain"* conforme mostra a Figura 26.

Figura 26: Registrar domínio



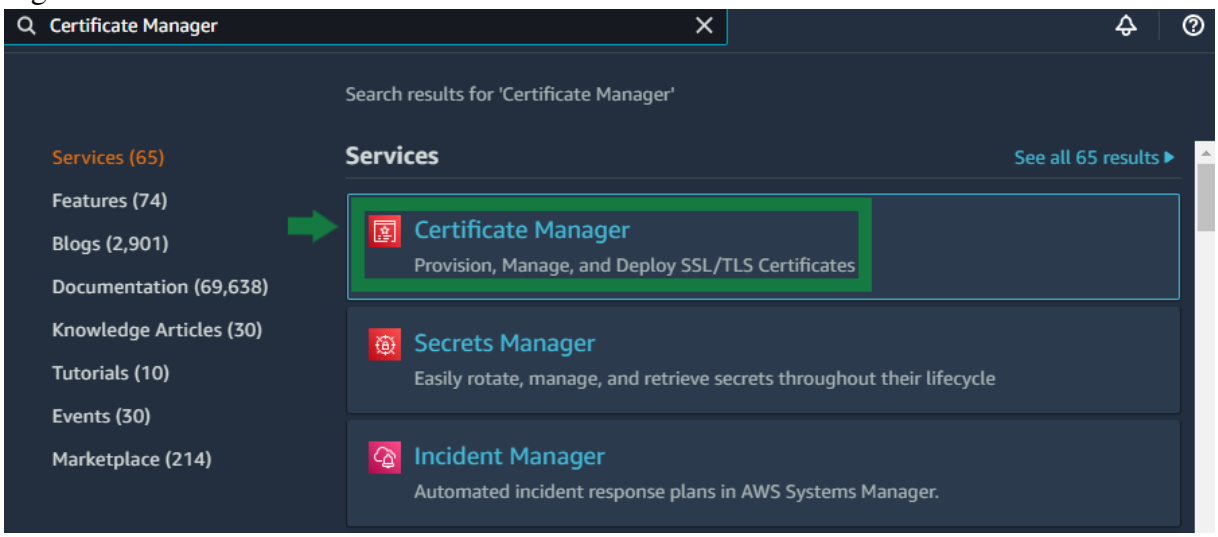
Fonte: Elaborado pelo autor (2022).

Após isso, houveram as etapas seguintes:

- Houve uma pesquisa por algum domínio disponível. No caso deste trabalho, o domínio selecionado foi o *"mypythia.net"*;
- As informações de contato foram inseridas;
- As informações foram verificadas e a obtenção foi realizada.

Feito isso, em até três dias úteis o domínio se tornou disponível para uso. Com o domínio disponibilizado, a Figura 27 apresenta a busca pelo termo *"Certificate Manager"*.

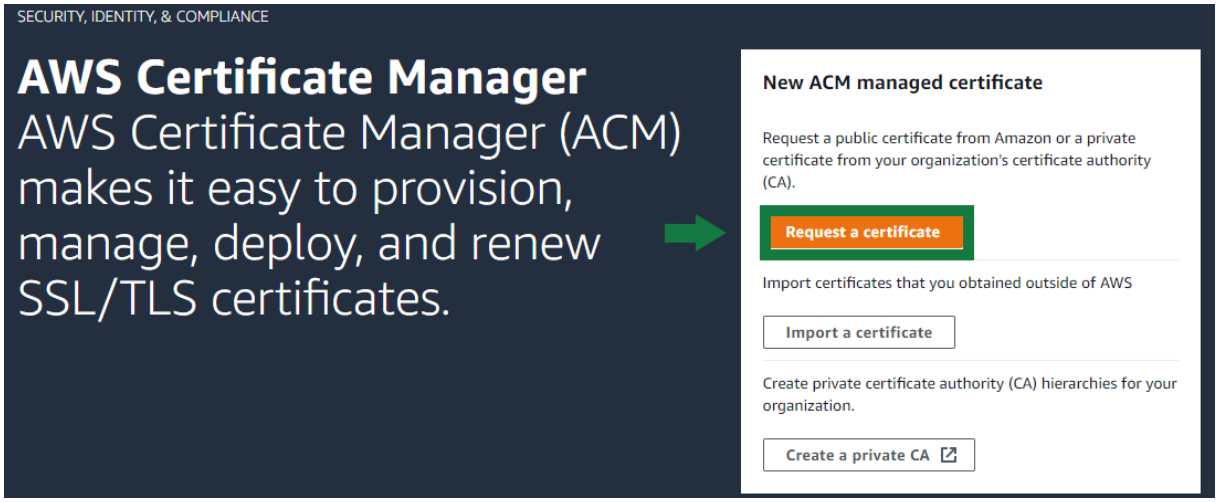
Figura 27: Gerenciador de certificados



Fonte: Elaborado pelo autor (2022).

Acionado o gerenciador, o certificado é solicitado, conforme apresenta a Figura 28.

Figura 28: Solicitar certificado



Fonte: Elaborado pelo autor (2022).

Após isso:

- O domínio registrado foi inserido com um "*." antes dele para que o certificado pudesse servir para todos os sub-domínios;
- A configuração padrão foi mantida e a solicitação foi realizada;
- O certificado adquirido foi acessado;
- A opção para a criação de registros no *"Route 53"* foi acionada e o certificado foi validado;
- Com o certificado validado, ao acessar novamente a EC2, grupos de destino e balanceadores de carga foram criados para ambos o Apache Kafka e o Kafdrop;
- Por último, houve a criação do roteamento para o trafégo de informações entre os balanceadores de carga e o domínio[6].

Dessa forma, se tornou possível a transferência dos dados de forma segura na rede.

5.3.3.4 MongoDB Atlas

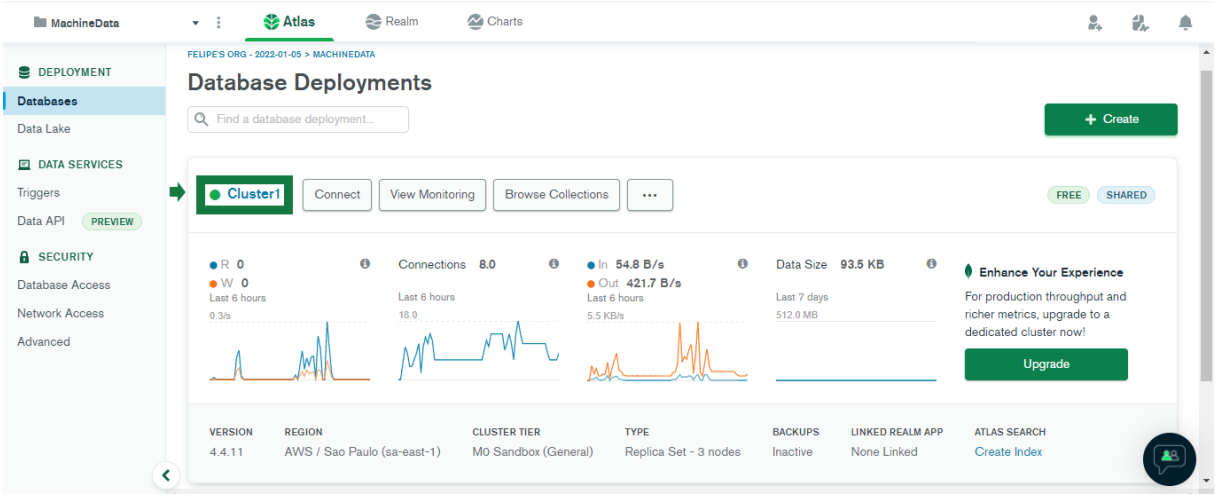
Segundo informações em MongoDB (2021), o MongoDB Atlas é um serviço que disponibiliza a implantação de um cluster de banco de dados em alguma plataforma de nuvem como a AWS, Microsoft Azure e Google Cloud, assim, permitindo a gravação e leitura de dados em baixa latência. As plataformas mencionadas disponibilizam várias nuvens distribuídas geograficamente. A configuração para a criação do Cluster passou por um conjunto de etapas:

[6] https://docs.aws.amazon.com/pt$_br$/*Route*53/*latest*/*DeveloperGuide*/*routing* – *to* – *elb* – *load* – *balancer.html*

– Primeiramente uma conta tradicional com email e senha foi criada[7];
– Foi realizado o *login*;
– Um novo projeto foi criado e nomeado;
– Com o projeto construído, deu-se início a construção de uma instância para o banco de dados:
    1. Foi criado um servidor compartilhado;
    2. O provedor de serviços de nuvem da AWS foi selecionado;
    3. A região de São Paulo, onde se desejou armazenar os dados foi selecionada;
    4. O Cluster foi criado;

Conforme mostra a Figura 29, foi possível observar a instância do MongoDB criada.

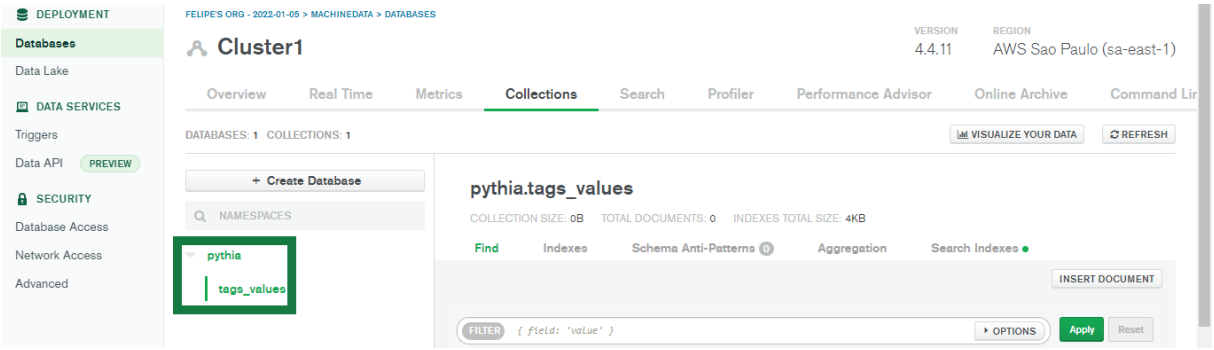
Figura 29: Tela principal do MongoDB Atlas



Fonte: Elaborado pelo autor (2022).

O Cluster é acessado, conforme mostra a Figura 30.

Figura 30: Armazenamento definido



Fonte: Elaborado pelo autor (2022).

[7] https://account.mongodb.com/account/register

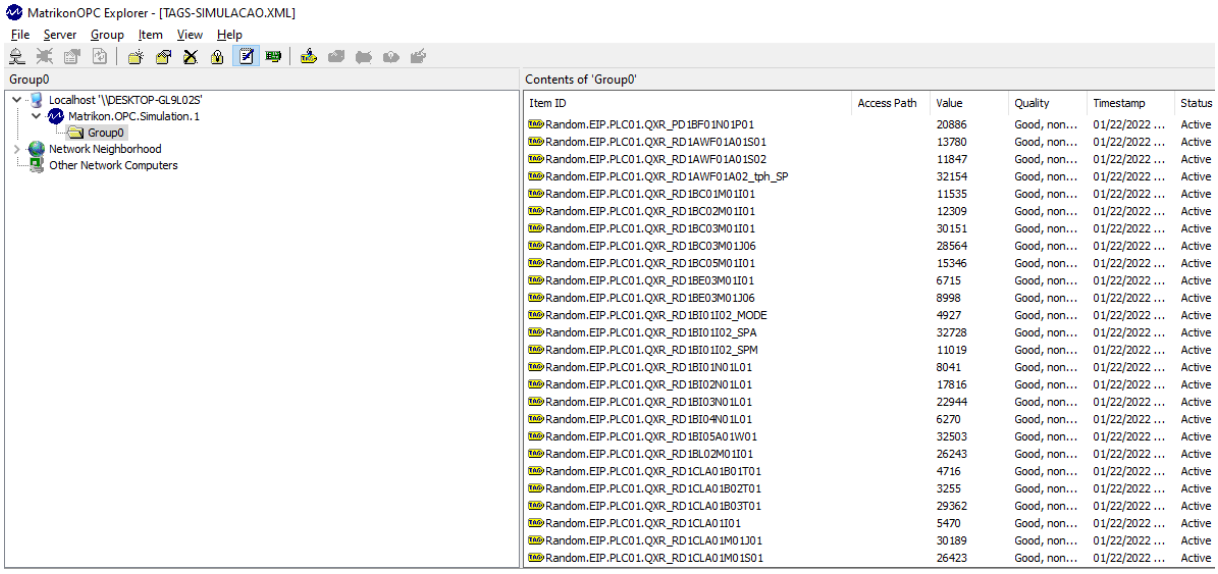
Na Figura 30, o banco de dados *"pythia"* e a coleção *"tags_values"* foram definidos para o armazenamento dos dados pré-processados e das predições.

5.4 Execução

A execução foi realizada desde uma máquina local de sistema operacional *Windows* até um ambiente em nuvem da AWS.

Primeiramente, os dados foram gerados randomicamente no simulador MatrikonOPC Explorer, conforme mostra a Figura 31.

Figura 31: Valores randômicos simulados



Fonte: Elaborado pelo autor (2022).

Após isso, dentro do servidor na nuvem, o contêiner com as imagens do Apache Kafka foram colocadas em execução com o comando *"docker-compose up -d"*, conforme mostra a Figura 32.

Figura 32: Sistema de comunicação em funcionamento

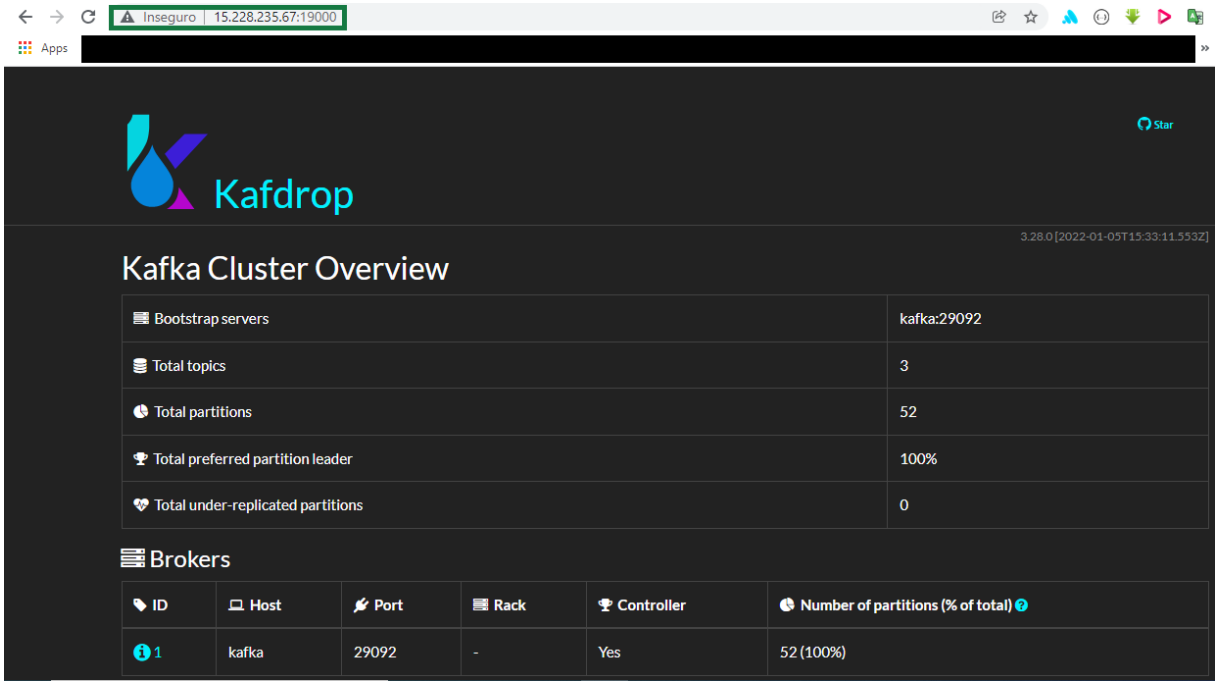
```
PS C:\Users\Felipe-NOT\Downloads> ssh -i "pythia3IoTIndustry.pem" ec2-user@ec2-15-228-235-67.sa-east-1.compute.amazonaws.com
Last login: Sat Jan 22 22:25:15 2022 from 177.37.241.185

   __|  __|  __|
   _|  (   \__ \   Amazon Linux 2 (ECS Optimized)
 ____|\___|____/

For documentation, visit http://aws.amazon.com/documentation/ecs
11 package(s) needed for security, out of 18 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-0-80 ~]$ cd Pre-processingAndAnalysis/
[ec2-user@ip-172-31-0-80 Pre-processingAndAnalysis]$ ls
docker-compose.yml  Processor  Simulated AI
[ec2-user@ip-172-31-0-80 Pre-processingAndAnalysis]$ docker-compose up -d
Creating network "pre-processingandanalysis_broker-kafka" with driver "bridge"
Creating pre-processingandanalysis_zookeeper_1 ... done
Creating pre-processingandanalysis_kafka_1     ... done
Creating pre-processingandanalysis_kafdrop_1   ... done
[ec2-user@ip-172-31-0-80 Pre-processingAndAnalysis]$
```

Fonte: Elaborado pelo autor (2022).

De acordo com a seção anterior, antes do TLS ser adicionado, ao executar o sistema

de comunicação, a interface de visualização do Apache Kafka funcionava de uma forma insegura na rede, conforme mostra a Figura 33.

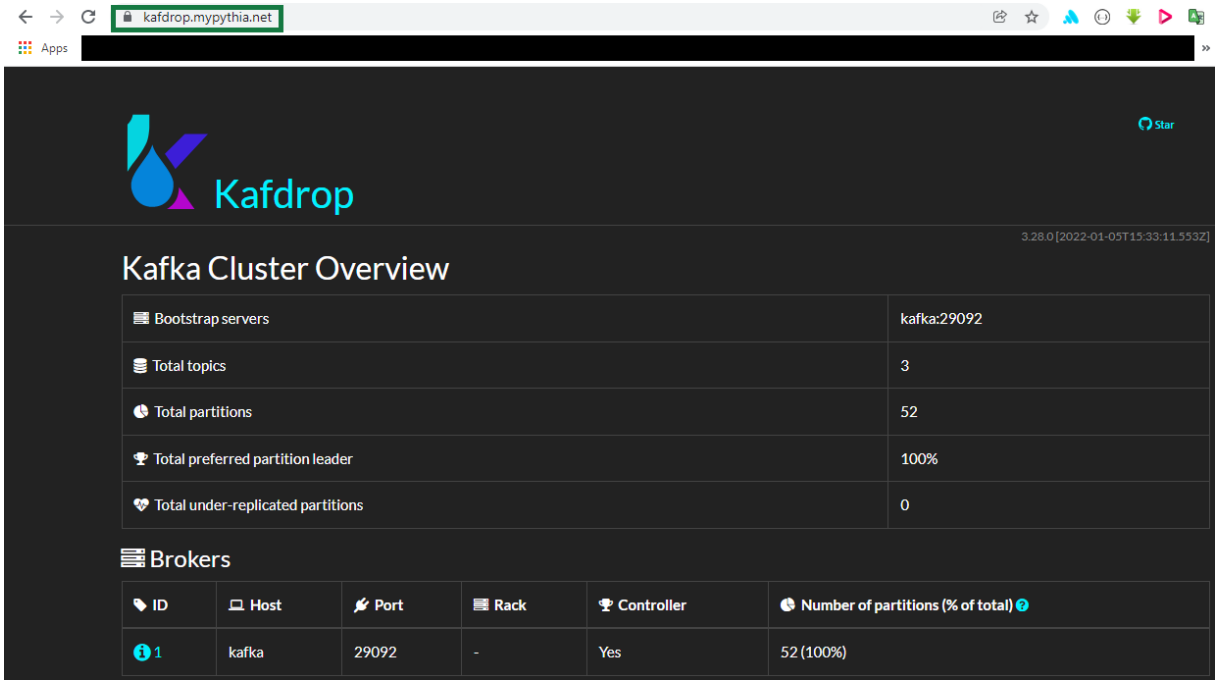
Figura 33: Rede insegura



Fonte: Elaborado pelo autor (2022).

Com o registro do domínio *myptyhia* e a inclusão de seu certificado, a comunicação se tornou segura conforme a Figura 34.

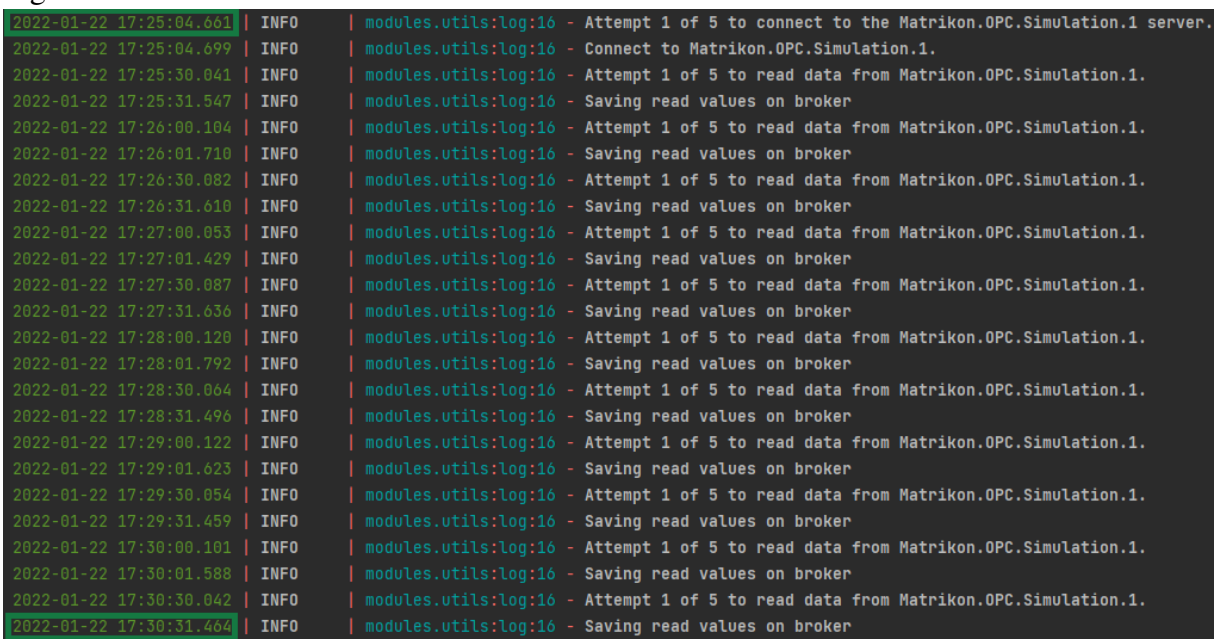
Figura 34: Rede segura



Fonte: Elaborado pelo autor (2022).

Com os dados randômicos sendo gerados e o sistema de comunicação funcionando de forma segura, os três serviços foram executados. O coletor OPC foi executado localmente através do ambiente de desenvolvimento PyCharm[8], onde a coleta foi iniciada às 17:25:04 do dia 22 de Janeiro de 2022 e concluída às 17:30:31 do mesmo dia. Assim, foram utilizadas 11 amostras para o teste, conforme mostra a Figura 35.

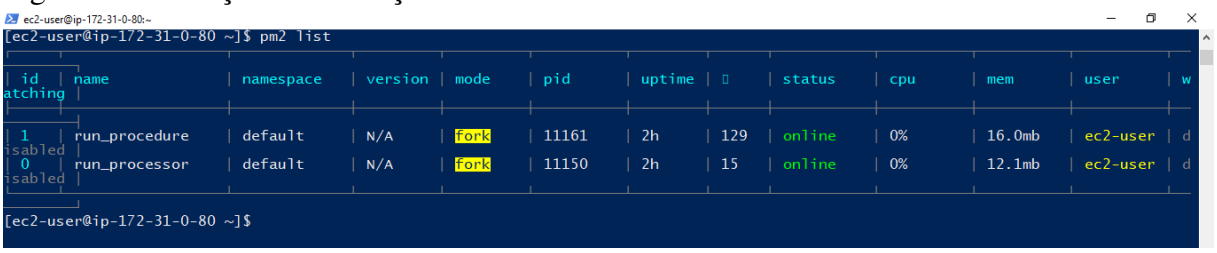
Figura 35: Coleta de dados



Fonte: Elaborado pelo autor (2022).

Com a coleta dos dados em funcionamento, os serviços de pré-processamento e análise foram executados em segundo plano no servidor em nuvem. Para isso, foram utilizados os comandos *"pm2 start run_processor.py –interpreter=python3"* e *"pm2 start run_procedure.py –interpreter=python3"*, respectivamente no diretório em nuvem de cada serviço. Conforme mostra a Figura 36, a inserção do comando *"pm2 list"* tornou possível a visualização desses serviços em funcionamento.

Figura 36: Serviços em execução na nuvem



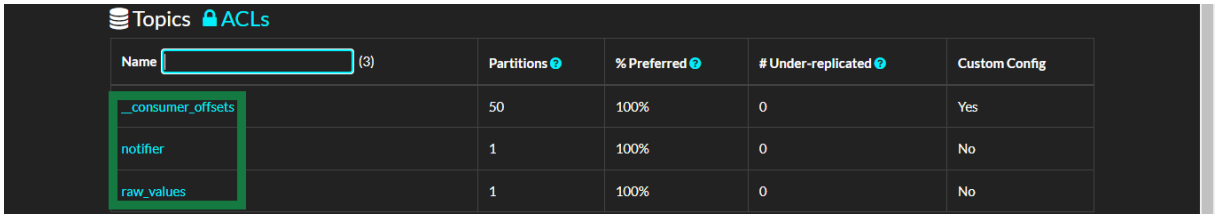
Fonte: Elaborado pelo autor (2022).

[8] https://www.jetbrains.com/pt-br/pycharm/

5.5 Resultados

Com os três serviços em funcionamento, os dados foram armazenados no sistema de comunicação e por último no MongoDB Atlas. A seguinte visualização de tópicos do Apache Kafka na Figura 37, foi possível de ser realizada:

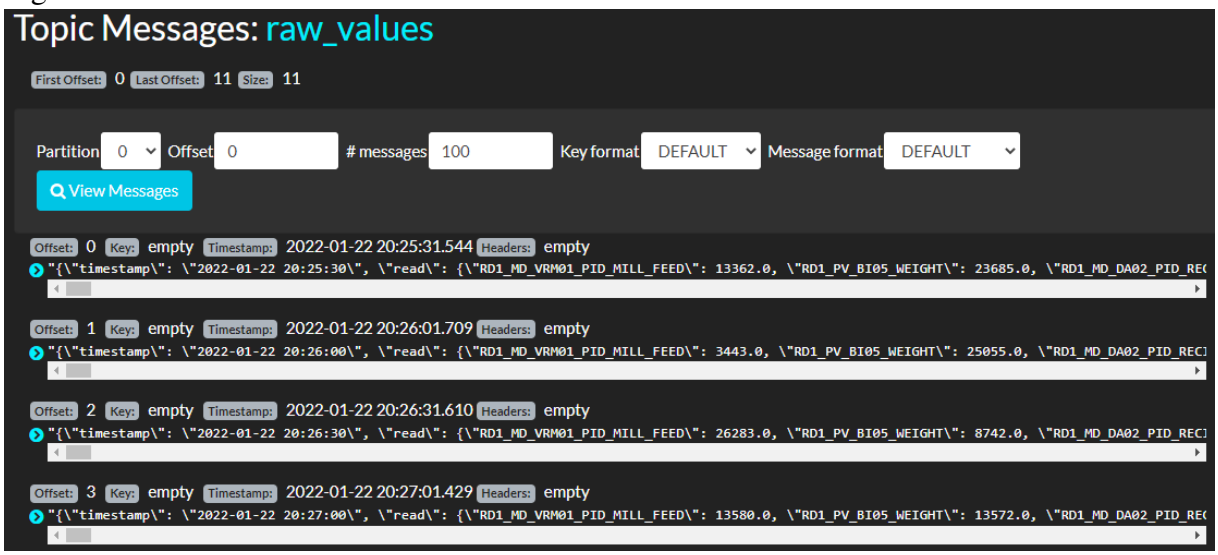
Figura 37: Tópicos



Fonte: Elaborado pelo autor (2022).

O tópico *"raw_values"* armazenou os primeiros conjuntos de valores enviados pelo coletor. Esses valores brutos foram identificados por suas devidas marcas de tempo e os dados preenchidos com esses valores não possuiam algum tratamento, conforme mostra a Figura 38.

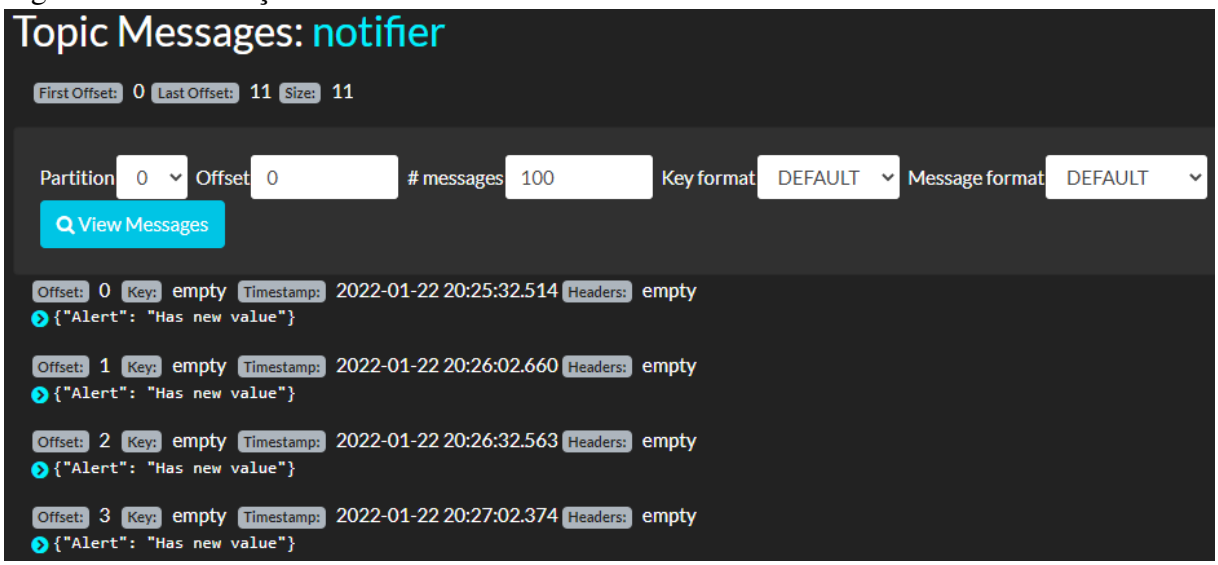
Figura 38: Valores brutos



Fonte: Elaborado pelo autor (2022).

Os valores acima foram consumidos pelo Processador que realizava de forma reativa, alguns pré-processamentos dos dados, armazenava esses dados no mongoDB e enviava ao tópico *"notifier"*, mensagens de alerta a cada conjunto de valores pré-processados que chegavam no MongoDB, conforme mostra a Figura 39.

Figura 39: Notificações

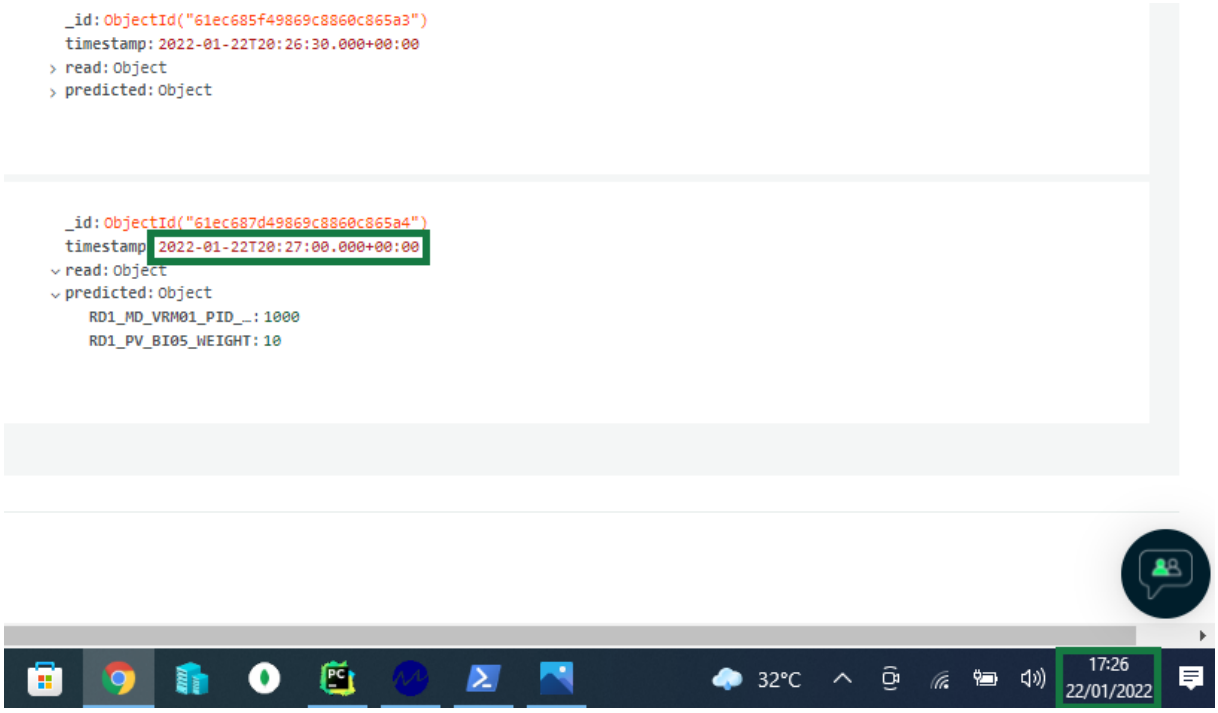


Fonte: Elaborado pelo autor (2022).

O tópico *"__consumer_offsets"* apenas revela informações de consumo dos dados.

Em reação ao alerta de mensagens do tópico acima, as predições começaram a ocorrer em tempos assíncronos de acordo com que as mensagens de alerta eram consumidas, conforme mostra a Figura 40.

Figura 40: Predição

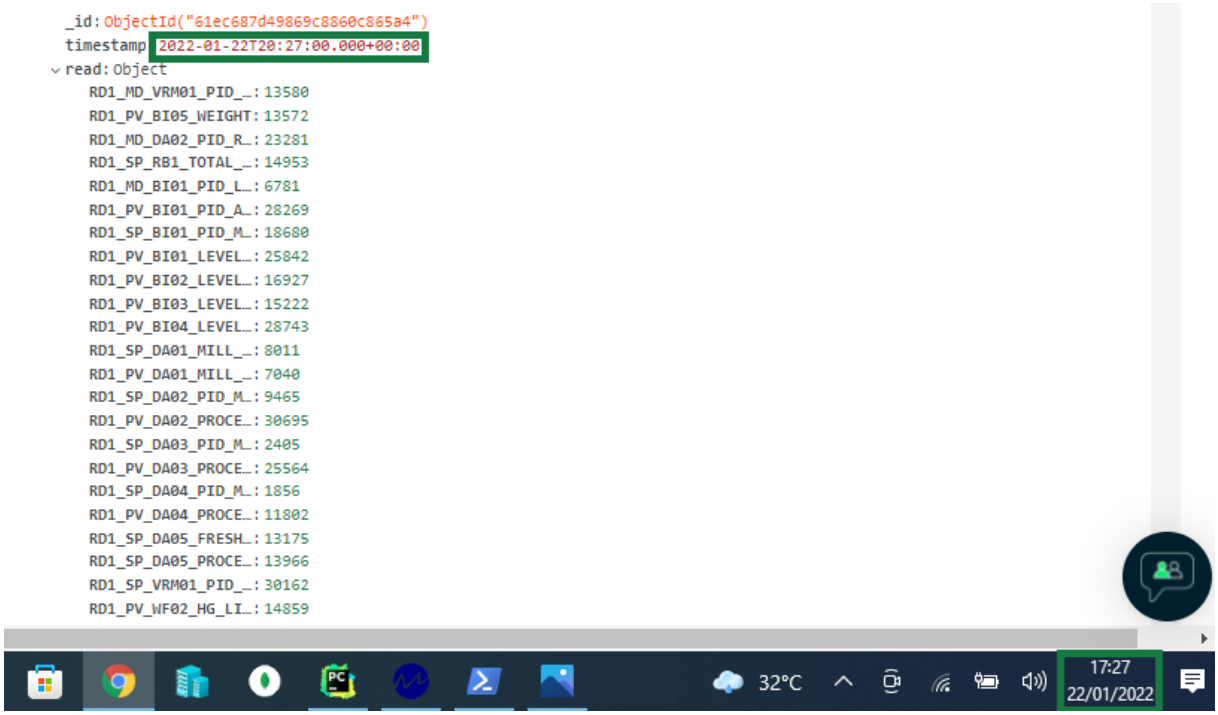


Fonte: Elaborado pelo autor (2022).

Pelo fuso horário global com 3 horas à frente de Brasília, na figura acima, a predição do conjunto de valores com o tempo em 17:27:00 foi escrita às 17:26:30 aproximadamente ou

20:26:30 no relógio global. Portanto, às 17:26:30, os valores pré-processados previstos não foram escritos, porém, esses são escritos às 17:27:00 ou 20:27:00, conforme mostra a Figura 41.

Figura 41: Valores pré-processados previstos



Fonte: Elaborado pelo autor (2022).

Por último, conforme mostra a Figura 42, foi possível observar em relação ao mesmo conjunto de dados, que a última *tag* chamada de "RD1_MD_AI_BED_HEIGHT", consistiu em um tratamento realizado para tornar esse conjunto de dados mais favorável para a análise.

Figura 42: Exemplo de tratamento aplicado

Fonte: Elaborado pelo autor (2022).

A execução desse cenário pode ser visualizada com os serviços operando em um outro período e mais detalhadamente em https://youtu.be/aByq0g3Z4rA.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Considerações

Devido ao enorme aumento dos dados gerados por aplicações e sistemas, trabalhar com dados se tornou algo de caráter produtivo para a economia mundial.

Na construção do documento, foram apresentadas diversas tecnologias, técnicas e ferramentas no que diz respeito aos diversos paradigmas da computação, como Sistemas de Controle Industrial, Internet das Coisas, Sistemas Ciber-Físicos, Computação em Nuvem e Padrões de Projeto.

A pesquisa sobre os assuntos relacionados, surgiu como um meio de apresentar uma solução eficiente para a automação industrial com grande volume de dados, onde para isso, foram relizadas as etapas de coleta, transferência e armazenamento de dados que foram disponibilizados para análise em tempo real e portanto, acelerando o processo de tomada de decisão pelos gestores da indústria.

Conforme apresentado na proposta, para o desenvolvimento do cenário levantado neste trabalho, foi necessário: adaptar um serviço coletor para um sistema de mensagens, onde os dados foram serializados em um formato adequado para o envio na rede. Após isso, construir um serviço reativo de pré-processamento de dados aplicado a padrões de projeto para receber, tratar e enviar os dados para um banco de dados não relacional, além do fato de enviar alertas de mensagens para um tópico do Apache Kafka e por último, adaptar uma Inteligência Artificial Simulada para realizar predições de forma reativa e armazená-las no banco.

Para o cenário definido, foi utilizado uma instância local do MongoDB para armazenar as *tags* a serem coletadas, um simulador para uma entrada de valores das *tags*, onde esses valores foram adquiridos pelo serviço coletor e o sistema de mensagens Apache Kafka conteinerizado com o Docker na AWS, juntamente com os serviços de pré-processamento e análise de dados, de onde esses foram executados no ambiente de nuvem. Portanto, como observado no painel do MongoDB Atlas, as previsões simuladas dos dados foram escritas no banco conforme chegavam as notificações do sistema de mensagens e aproximandamente 30 segundos antes que os valores pré-processados também fossem escritos.

Na primeira versão do sistema, a comunicação entre os serviços acontecia de forma síncrona e por meio de operações em banco de dados. Ao adicionar o sistema de mensagens aos serviços, a comunicação se tornou assíncrona e a programação reativa à entrada de dados, o

que gerou um ganho na velocidade de processamento e transmissão. Além disso, a hospedagem dos componentes do sistema na nuvem, deixou o sistema configurado de forma a tornar a comunicação segura e a reduzir futuros gastos com infraestrutura física para cobrir serviços de armazenamento.

6.2 Trabalhos futuros

Anteriormente, várias tecnologias e ferramentas foram apresentadas para melhorar o máximo possível a eficiência do cenário da Indústria 4.0 definido neste trabalho. No entanto, futuras alterações podem ser consideradas como sugestões de melhoria para o trabalho. São elas:

- Realizar um comparativo entre a execução síncrona e local com a solução assíncrona e em ambiente de nuvem que foi desenvolvida neste trabalho, a fim de visualizar em números, os ganhos de desempenho em função da quantidade de espaço e processamento gasto na execução, além da velocidade de transferência dos dados;
- A adição de um banco de dados de séries temporais como o InfluxDB;
- A codificação com mais algum outro padrão de projeto possívelmente interessante para o trabalho, como por exemplo, o padrão *Command*;
- Implantação da computação de névoa;
- Implantação da computação de borda.

REFERÊNCIAS

ALTEXSOFT. **Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC**. 2020. Disponível em: https://www.altexsoft.com/blog/soap-vs-rest-vs-graphql-vs-rpc/. Acesso em: 20 mar. 2021.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128610001568.

AWS. **AWS IoT Analytics**. 2021. Disponível em: https://aws.amazon.com/pt/iot-analytics/?c=i&sec=srv. Acesso em: 15 mar. 2021.

BARMIN, A. **NTRODUCTION TO MESSAGE BROKERS. PART 1: APACHE KAFKA VS RABBITMQ**. 2021. Disponível em: https://freshcodeit.com/blog-introduction-to-message-brokers-part-1-apache-kafka-vs-rabbitmq. Acesso em: 19 mar. 2021.

CLOUDAMQP. **RabbitMQ as a Service**. 2021. Disponível em: https://www.cloudamqp.com/. Acesso em: 20 mar. 2021.

COBB, M. **RSA algorithm (Rivest-Shamir-Adleman)**. 2022. Disponível em: https://www.techtarget.com/searchsecurity/definition/RSA. Acesso em: 18 jan. 2022.

COMAN, C. M.; FLORESCU, A. Electric grid monitoring and control architecture for industry 4.0 systems. In: **2018 International Symposium on Fundamentals of Electrical Engineering (ISFEE)**. [*S. l.*: *s. n.*], 2018. p. 1–6.

DRAKE, M.; OSTEZER. **A Comparison of NoSQL Database Management Systems and Models**. 2014. Disponível em: https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models. Acesso em: 12 mar. 2021.

FARKAS, J.; VARGA, B.; MIKLÓS, G.; SACHS, J. 5g-tsn integration for industrial automation. **ERICSSON TECHNOLOGY**, 2019. Disponível em: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-tsn-integration-for-industrial-automation.

GHOSH, S. **Distributed Systems**. 2. ed. [*S. l.*]: Chapman and Hall/CRC, 2014. ISBN 9781498760058.

GONÇALVES, A. **Como funciona o SSH**. 2021. Disponível em: https://www.hostinger.com.br/tutoriais/como-funciona-o-ssh. Acesso em: 18 jan. 2022.

GRAJA, I.; KALLEL, S.; GUERMOUCHE, N.; CHEIKHROUHOU, S.; KACEM, A. H. A comprehensive survey on modeling of cyber-physical systems. **Concurrency and Computation: Practice and Experience**, v. 32, n. 15, p. e4850, 2020. E4850 cpe.4850. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4850.

GROSSMANN, T. **How to Use Google's Protocol Buffers in Python**. 2020. Disponível em: https://www.freecodecamp.org/news/googles-protocol-buffers-in-python/. Acesso em: 13 fev. 2021.

GUPTA, S. C. **Architecture for High-Throughput Low-Latency Big Data Pipeline on Cloud**. 2020. Disponível em: https://towardsdatascience.com/scalable-efficient-big-data-analytics-machine-learning-pipeline-architecture-on-cloud-4d59efc092b5. Acesso em: 13 fev. 2021.

HAJIHEYDARI, N.; TALAFIDARYANI, M.; KHABIRI, S. Iot big data value map: How to generate value from iot data. In: **Proceedings of the 2019 the 5th International Conference on E-Society, e-Learning and e-Technologies**. New York, NY, USA: Association for Computing Machinery, 2019. (ICSLT 2019), p. 98–103. ISBN 9781450362351. Disponível em: https://doi.org/10.1145/3312714.3312728.

KEPWARE. **IoT Gateway for KEPServerEX**. 2017. Disponível em: https://www.kepware.com/en-us/products/kepserverex/advanced-plug-ins/iot-gateway/.

KEPWARE. **OPC Unified Architecture (UA) Server Interface**. 2021. Disponível em: https://www.kepware.com/en-us/products/kepserverex/features/opc-ua-server-interface/. Acesso em: 18 fev. 2021.

KHASAWNEH, T. N.; AL-SAHLEE, M. H.; SAFIA, A. A. Sql, newsql, and nosql databases: A comparative survey. In: **2020 11th International Conference on Information and Communication Systems (ICICS)**. [*S. l.*: *s. n.*], 2020. p. 013–021.

KLEPPMANN, M. **Designing Data-Intensive Applications**. Beijing: O'Reilly, 2017. ISBN 978-1-4493-7332-0. Disponível em: https://www.safaribooksonline.com/library/view/designing-data-intensive-applications/9781491903063/.

KSIAZEK, K. **An Introduction to Time Series Databases**. 2019. Disponível em: https://severalnines.com/database-blog/introduction-time-series-databases. Acesso em: 15 mar. 2021.

KUMAR, A.; ARORA, A. An anfis-based compatibility scorecard for iot integration in websites. **J. Supercomput.**, Kluwer Academic Publishers, USA, v. 76, n. 4, p. 2568–2596, apr 2020. ISSN 0920-8542. Disponível em: https://doi.org/10.1007/s11227-019-03026-x.

KUMAR, R. **What is the five layer automation pyramid?** 2019. Disponível em: https://medium.com/world-of-iot/92-what-is-the-five-layer-automation-pyramid-d0ccc1b903c3. Acesso em: 16 mar. 2021.

LIN, J.; YU, W.; ZHANG, N.; YANG, X.; ZHANG, H.; ZHAO, W. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. **IEEE Internet of Things Journal**, v. 4, n. 5, p. 1125–1142, 2017.

LUTTI. **Design Patterns – O que são e quais os benefícios?** 2018. Disponível em: https://www.opus-software.com.br/design-patterns/. Acesso em: 18 jan. 2022.

MANOOCHEHRI, M. **Data just right : introduction to large-scale data analytics / Michael Manoochehri.** Upper Saddle River, NJ: Addison-Wesley, 2014. (Addison-Wesley data and analytics series). ISBN 9780321898654.

MATRIKON. **Products: Desktop Tools**. 2022. Disponível em: https://www.matrikonopc.com/products/opc-desktop-tools/opc-explorer.aspx?utm_campaign=Referral&utm_medium=website&utm_source=OPCF&utm_content=OPCF-Product-Catalog&utm_term=global. Acesso em: 18 jan. 2022.

MICROSOFT. **Entender os modelos de armazenamento de dados**. 2020. Disponível em: https://docs.microsoft.com/pt-br/azure/architecture/guide/technology-choices/data-store-overview. Acesso em: 12 mar. 2021.

MONGODB. **Multi-Cloud Data Distribution**. 2021. Disponível em: https://www.mongodb.com/cloud/atlas/multicloud-data-distribution. Acesso em: 06 jan. 2022.

MOURA, R. L. de; WERNER, L. B.; GONZALEZ, A. Management and ownership: A data strategy in the industry 4.0 context. In: **Proceedings of the 3rd International Conference on Big Data and Internet of Things**. New York, NY, USA: Association for Computing Machinery, 2019. (BDIOT 2019), p. 23–28. ISBN 9781450372466. Disponível em: https://doi.org/10.1145/3361758.3361769.

MUKUNDHA, C.; VIDYAMADHURI, K. Cloud computing models: A survey. **Adv. Comput. Sci. Technol.**, v. 10, n. 5, p. 747–761, 2017.

MÜLLER, J. M.; BULIGA, O.; VOIGT, K.-I. The role of absorptive capacity and innovation strategy in the design of industry 4.0 business models - a comparison between smes and large enterprises. **European Management Journal**, v. 39, n. 3, p. 333–343, 2021. ISSN 0263-2373. Disponível em: https://www.sciencedirect.com/science/article/pii/S0263237320300190.

NEC. **What are IoT enabling technologies?** 2020. Disponível em: https://www.nec.co.nz/market-leadership/publications-media/what-are-iot-enabling-technologies/. Acesso em: 16 mar. 2021.

OLIVEIRA, M.; AFONSO, D. Industry focused in data collection: How industry 4.0 is handled by big data. In: **Proceedings of the 2019 2nd International Conference on Data Science and Information Technology**. New York, NY, USA: Association for Computing Machinery, 2019. (DSIT 2019), p. 12–18. ISBN 9781450371414. Disponível em: https://doi.org/10.1145/3352411.3352414.

OSISOFT. **Administração do PI System**. [*S. l.*], 2018. SP2 a. Disponível em: https://learning.osisoft.com/administracao-do-pi-system.

OSISOFT. 2021. Disponível em: https://www.osisoft.pt/pi-system/pi-cloud/osisoft-cloud-services. Acesso em: 12 fev. 2021.

PAPIERNIK, M. **How To Use MongoDB Compass**. 2021. Disponível em: https://www.digitalocean.com/community/tutorials/how-to-use-mongodb-compass. Acesso em: 18 jan. 2022.

PENA, M. D. V.; RODRIGUEZ-ANDINA, J. J.; MANIC, M. The internet of things: The role of reconfigurable platforms. **IEEE Industrial Electronics Magazine**, v. 11, n. 3, p. 6–19, 2017.

POWERS, D. **kafka-python Documentation**. [*S. l.*], 2020. v. 2.0.2. Disponível em: https://kafka-python.readthedocs.io/en/latest/.

SAHAL, R.; BRESLIN, J. G.; ALI, M. I. Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case. **Journal of Manufacturing Systems**, v. 54, p. 138–151, 2020. ISSN 0278-6125. Disponível em: https://www.sciencedirect.com/science/article/pii/S0278612519300937.

SAUTER, T.; SOUCEK, S.; KASTNER, W.; DIETRICH, D. The evolution of factory and building automation. **IEEE Industrial Electronics Magazine**, v. 5, n. 3, p. 35–48, 2011.

SHAFIQ, S. I.; SZCZERBICKI, E.; SANIN, C. Proposition of the methodology for data acquisition, analysis and visualization in support of industry 4.0. **Procedia Computer Science**, v. 159, p. 1976–1985, 2019. ISSN 1877-0509. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050919315716.

SOURCE MAKING. **Design Patterns**. 2021. Disponível em: https://sourcemaking.com/design_patterns. Acesso em: 18 jan. 2022.

SOUZA, E. F. **Protobuf — Uma alternativa ao JSON e XML**. 2018. Disponível em: https://medium.com/trainingcenter/protobuf-uma-alternativa-ao-json-e-xml-a35c66edab4d. Acesso em: 13 fev. 2021.

SOUZA, M. L. H.; COSTA, C. A. da; RAMOS, G. de O.; RIGHI, R. da R. A survey on decision-making based on system reliability in the context of industry 4.0. **Journal of Manufacturing Systems**, v. 56, p. 133–156, 2020. ISSN 0278-6125. Disponível em: https://www.sciencedirect.com/science/article/pii/S0278612520300807.

STEEN, A. S. T. Maarten van. **Distributed Systems**. 3. ed. [*S. l.*: *s. n.*], 2020. ISBN 978-1543057386.

STOUFFER, K.; FALCO, J.; KENT, K. Guide to supervisory control and data acquisition (scada) and industrial control systems security. 01 2006.

TREND MICRO. **Industrial Control System**. 2021. Disponível em: https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system. Acesso em: 8 fev. 2021.

TRINKS, S.; FELDEN, C. Edge computing architecture to support real time analytic applications : A state-of-the-art within the application area of smart factory and industry 4.0. In: **2018 IEEE International Conference on Big Data (Big Data)**. [*S. l.*: *s. n.*], 2018. p. 2930–2939.

VAILLANT, S. **5G Time Sensitive Networking**. 2020. Disponível em: https://techradar.softwareag.com/technology/5g-tsn/. Acesso em: 15 mar. 2021.

VELASQUEZ, N.; ESTEVEZ, E.; PESADO, P. Cloud computing, big data and the industry 4.0 reference architectures. **Journal of Computer Science and Technology**, v. 18, p. e29, 12 2018.

VENTURELLI, M. **Redes Ethernet Industrial: Conceito e aplicação na Automação e Controle Industrial**. 2020. Disponível em: https://www.automacaoindustrial.info/redes-ethernet-industrial-conceito-e-aplicacao-na-automacao-e-controle-industrial/. Acesso em: 15 mar. 2021.

VERITAS. **O que é conteinerização? Quais são os benefícios?** 2021. Disponível em: https://www.veritas.com/pt/br/information-center/containerization. Acesso em: 18 jan. 2022.

WOLLSCHLAEGER, M.; SAUTER, T.; JASPERNEITE, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. **IEEE Industrial Electronics Magazine**, v. 11, p. 17 – 27, 03 2017.

APÊNDICE A – CONFIGURAÇÃO PARA O APACHE KAFKA

Figura 43: Imagens do Apache Kafka

```
version: '3'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    networks:
      - broker-kafka
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
  kafka:
    image: confluentinc/cp-kafka:latest
    networks:
      - broker-kafka
    depends_on:
      - zookeeper
    ports:
      - 9092:9092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT_HOST://15.228.235.67:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
  kafdrop:
    image: obsidiandynamics/kafdrop:latest
    networks:
      - broker-kafka
    depends_on:
      - kafka
    ports:
      - 19000:9000
    environment:
      KAFKA_BROKERCONNECT: kafka:29092

networks:
  broker-kafka:
    driver: bridge
```

Conforme mostra a Figura 43, o arquivo está configurado para o ambiente de nuvem, onde um IP de nuvem é especificado no *HOST*. Nesse arquivo de configuração:

- A imagem do Zookeeper pertencente à plataforma Confluent gerencia os pacotes do Apache Kafka;
- A imagem do Apache Kafka corresponde ao sistema de comunicação persistente que escuta na porta padrão 9092. Essa imagem é também pertencente à plataforma Confluent;
- A imagem do Kafdrop pertencente ao provedor de serviços Obsidian Dynamics é o gerenciador de mensagens do Apache Kafka que fornece uma interface de visualização das mensagens que chegam nos tópicos. Este gerenciador de mensagens escuta na porta padrão 19000.