UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E COMPUTAÇÃO

MESTRADO ACADÊMICO EM ENGENHARIA ELÉTRICA E COMPUTAÇÃO

RAUL DE VASCONCELOS BOTÊLHO

ARQUITETURA IIOT BASEADA EM APACHE KAFKA PARA UM SISTEMA DE ALERTAS EM MÁQUINAS INDUSTRIAIS

SOBRAL

2025

RAUL DE VASCONCELOS BOTÊLHO

ARQUITETURA IIOT BASEADA EM APACHE KAFKA PARA UM SISTEMA DE ALERTAS EM MÁQUINAS INDUSTRIAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica e Computação do Programa de Pós-Graduação em Engenharia Elétrica e Computação do da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica e Computação. Área de Concentração: Sistemas Embarcados e suas Aplicações no contexto de Sistemas CPS e IoT

Orientador: Prof. Dr. Reuber Regis de Melo

Coorientador: Prof. Dr. Wendley Souza da Silva

SOBRAL

2025

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

B764a Botêlho, Raul de Vasconcelos.
Arquitetura IIoT baseada em Apache Kafka para um Sistema de Alertas em Máquinas Industriais / Raul de Vasconcelos Botêlho. – 2025.
66 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Sobral, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Sobral, 2025.
Orientação: Prof. Dr. Reuber Regis Melo.
Coorientação: Prof. Dr. Wendley Souza da Silva.

1. indústria. 2. mensageria. 3. IIoT. 4. kafka. I. Título.

CDD 621.3

RAUL DE VASCONCELOS BOTÊLHO

ARQUITETURA IIOT BASEADA EM APACHE KAFKA PARA UM SISTEMA DE ALERTAS EM MÁQUINAS INDUSTRIAIS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica e Computação do Programa de Pós-Graduação em Engenharia Elétrica e Computação do da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica e Computação. Área de Concentração: Sistemas Embarcados e suas Aplicações no contexto de Sistemas CPS e IoT

Aprovada em: 27 de Agosto de 2025

BANCA EXAMINADORA

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

Prof. Dr. Wendley Souza da Silva (Coorientador)
Universidade Federal do Ceará (UFC)

Prof.ª Dr. Jermana Lopes de Moraes
Universidade Federal do Ceará (UFC)

Prof. Dr. Ialis Cavalcante de Paula Junior
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Gostaria de expressar minha gratidão aos meus pais Maria Romana e Everson pelo apoio incondicional ao longo de toda minha jornada. Em especial para minha mãe, que dedicou sua vida inteira para garantir que eu tivesse oportunidades de estudo e pudesse me desenvolver plenamente: toda minha conquista também é dela. Ao meu pai, que, desde cedo, acreditou no meu potencial e me presenteou com o meu primeiro computador com a esperança de que, um dia, isso pudesse se transformar em minha profissão.

Agradeço à minha esposa Naianna Morais por todo amor, carinho e compreensão durante todos os anos da minha formação acadêmica. Sua presença constante e apoio emocional foram fundamentais para que eu chegasse até aqui sempre motivado e determinado.

Registro meus sinceros agradecimentos à Grendene S/A por ter confiado no potencial deste trabalho e permitido o uso do seu parque industrial como laboratório de testes, viabilizando alterações de sistema e fornecendo o suporte necessário para as análises realizadas.

Sou grato a todos os professores que fizeram parte da minha formação, especialmente ao corpo docente dos cursos de Engenharia de Computação e Engenharia Elétrica da Universidade Federal do Ceará — Campus Sobral. Com dedicação, competência e compromisso com o ensino, esses profissionais contribuíram para uma formação sólida, técnica e humana em ambas as áreas, e seguem sendo fonte de inspiração por valorizarem a academia, enfrentando suas limitações e buscando aproximá-la cada vez mais das demandas reais do campo de trabalho.

De forma especial, agradeço ao professor Dr. Marco Toledo, cuja visão, exemplo e incentivo foram determinantes para que eu mantivesse o vínculo com a academia mesmo diante dos caminhos profissionais.

RESUMO

A Indústria 4.0 está transformando rapidamente o cenário global da manufatura, integrando sistemas ciber-físicos, automação avançada e análise de dados em tempo real. No Brasil, essa transformação é marcada pela flexibilidade das indústrias na adaptação às novas tecnologias, impulsionada pela necessidade de competitividade em um mercado global. Este estudo examina a aplicação de tecnologias emergentes na área de Internet das Coisas Industrial (IIoT) com o objetivo de proporcionar uma solução flexível que permita a descentralização e digitalização dos fluxos de trabalho de produção. Focando em uma indústria calçadista no nordeste do Brasil, nesse trabalho é realizado um estudo de caso onde é explorado como a adoção dessas tecnologias é crucial para enfrentar os desafios da transformação digital, mantendo a competitividade. A solução proposta envolve o uso do Apache Kafka para facilitar o fluxo contínuo de alertas em tempo real e a integração entre os sistemas de Tecnologia Operacional (TO) e de Tecnologia da Informação (TI). Utilizando comunicação baseada em eventos e formatos de dados flexíveis como JSON, a arquitetura visa aumentar a visibilidade e eficiência dos processos industriais. Os resultados experimentais indicam que o sistema manteve latência inferior a 300 ms em cenários com alta concorrência, sem perdas ou duplicação de mensagens, representando uma redução superior a 95% em relação à configuração anterior, que apresentava atrasos de até dois minutos. Isso demonstra a viabilidade da abordagem para aplicações críticas em ambientes industriais.
A arquitetura proposta mostra-se promissora como base para futuras expansões e integração com outras tecnologias da Indústria 4.0.

Palavras-chave: indústria 4.0; flexibilidade; internet das coisas industrial; apache kafka ; arquitetura orientada a eventos; comunicação em tempo real; mensageria; manufatura digital; visibilidade de dados.

ABSTRACT

Industry 4.0 is rapidly transforming the global manufacturing landscape by integrating cyber-physical systems, advanced automation, and real-time data analysis. In Brazil, this transformation is characterized by the flexibility of industries in adapting to new technologies, driven by the need for competitiveness in a global market. This study examines the application of emerging technologies in the field of Industrial Internet of Things (IIoT) with the goal of providing a flexible solution that enables the decentralization and digitalization of production workflows. Focusing on a footwear industry in northeastern Brazil, this case study explores how the adoption of these technologies is crucial for addressing the challenges of digital transformation and maintaining competitiveness. The proposed solution involves the use of Apache Kafka to enable continuous real-time alert flows and integration between Operational Technology (OT) and Information Technology (IT) systems. Using event-driven communication and flexible data formats such as JSON, the architecture aims to enhance the visibility and efficiency of industrial processes. Experimental results show that the system maintained latency below 300 ms under high-concurrency scenarios, with no data loss or duplication, representing a reduction of over 95% compared to the previous setup, which experienced delays of up to two minutes. This demonstrates the feasibility of the approach for critical applications in industrial environments. The proposed architecture proves to be a promising foundation for future expansions and integration with other Industry 4.0 technologies.

Keywords: industry 4.0; flexibility; industrial internet of things; apache kafka; event-driven architecture; real-time communication; messaging; digital manufacturing; data visibility.

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BI	*Business Intelligence* / Inteligência de Negócios
CDF	*Cumulative Distribution Function* / Função de Distribuição Acumulada
CLP	Controlador Lógico Programável
CNI	Confederação Nacional da Indústria
CPU	*Central Processing Unit* / Unidade Central de Processamento
EAI	*Enterprise Application Integration* / Integração de Aplicações Empresariais
ERP	*Enterprise Resource Planning* / Planejamento de Recursos Empresariais
ESB	*Enterprise Service Bus* / Barramento de Serviços Empresariais
IIoT	*Industrial Internet of Things*
IoT	*Internet of Things*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
MES	*Manufacturing Execution System* / Sistema de Execução da Manufatura
NTP	*Network Time Protocol*
PIB	Produto Interno Bruto
pub/sub	*publish/subscribe*
SOA	*Service-Oriented Architecture* / Arquitetura Orientada a Serviços
TI	Tecnologia da Informação
TO	Tecnologia Operacional
UTC	*Universal Coordinated Time*
VLAN	*Virtual Local Area Network*
WMS	*Warehouse Management System* / Sistema de Gerenciamento de Armazém

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Motivação 11
1.2 Objetivos 12
1.2.1 Objetivo Geral 12
1.2.2 Objetivos Específicos 12
1.3 Organização do Trabalho 13
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 Indústria 4.0 no Cenário Brasileiro 14
2.2 Apache Kafka 16
2.2.1 JSON: JavaScript Object Notation 19
2.3 Trabalhos Relacionados 20
2.3.1 Arquitetura de Ingestão de Dados 20
2.3.2 Gateway IIOT Adaptativo 21
2.3.3 Apache Kafka para Big Data em Fábricas Inteligentes 24
2.3.4 Avaliação de Performance do Kafka em Ambientes de Alto Throughput . . 26
2.3.5 Ferramenta de Testes para Confiabilidade em Kafka 27
2.3.6 Considerações sobre a Aplicação do Kafka na Indústria 4.0 29
3 METODOLOGIA 31
3.1 Visão Geral 31
3.2 Estudo de Caso 33
3.2.1 Cenário Industrial 33
3.2.2 Problemática 35
3.2.3 Proposta 37
3.3 Materiais 38
3.3.1 Arquitetura de Mensageria Kafka 39
3.3.2 Descrição dos Tópicos Kafka 41
3.3.2.1 Tópicos: `machine_alerts` *41*
3.3.2.2 Tópicos: `alerts_action` *43*
3.3.3 Funcionamento do Sistema de Alertas 43
3.3.4 Prototipação 46

3.3.4.1 *Servidor e Consumidores* *47*
3.3.4.2 *Produtores* *49*
3.3.5 ***Medidas de Desempenho*** 51
3.3.5.1 *Latência* *52*
4 **RESULTADOS E DISCUSSÃO** 54
5 **CONCLUSÕES E ATIVIDADES FUTURAS** 61
5.1 **Conclusão e Trabalhos Futuros** 61
5.2 **Atividades Futuras** 62
REFERÊNCIAS 63
APÊNDICES 66
APÊNDICE A – Exemplo de JSON de um alerta 66

1 INTRODUÇÃO

A Indústria 4.0 tem transformado rapidamente o cenário global da manufatura, promovendo a integração de sistemas ciber-físicos, automação avançada e análise de dados em tempo real. Estudos como os de Barata *et al.* (2020) destacam a evolução da mobilidade e da flexibilidade dos processos industriais para atender às exigências da produção moderna. Além disso, Lugert *et al.* (2018) , por sua vez, sugerem que, diante da crescente complexidade dos processos e da incorporação de tecnologias emergentes, metodologias tradicionais como o mapeamento de fluxo de valor, uma técnica usada para analisar e projetar o fluxo de materiais e informações em um processo de produção, precisarão ser adaptadas para manter sua eficácia em ambientes produtivos cada vez mais digitalizados e que demandam monitoramento contínuo.

Essa transformação não é exclusiva de grandes corporações. De acordo com Silva *et al.* (2022), empresas de diferentes portes, incluindo micro e pequenas industrias, têm aderido a esse novo paradigma, implementando diversas tecnologias, como *Internet of Things* (IoT), *big data*, inteligência artificial, computação em nuvem e manufatura aditiva. Essas inovações possibilitam a transição de processos analógicos e centralizados para fluxos de trabalho digitais e descentralizados, integrando máquinas, fornecedores, clientes e dados, com foco em flexibilidade, produtividade e competitividade. Contudo, essa transição exige planejamento estratégico rigoroso, já que os investimentos são elevados e devem ser direcionados de forma precisa para garantir retorno sustentável e evitar obsolescência precoce.

No contexto brasileiro, a Indústria 4.0 representa simultaneamente desafios e oportunidades. Segundo Contador *et al.* (2020), a flexibilidade e a capacidade de adaptação das empresas nacionais são fatores críticos para enfrentar as dificuldades impostas por essa nova era industrial. Apesar dessas barreiras, a adoção de tecnologias emergentes pode proporcionar ganhos substanciais em produtividade e personalização em massa, alavancando a competitividade do setor industrial brasileiro. Nesse cenário, as soluções baseadas em *Industrial Internet of Things* (IIoT) tornam-se essenciais para otimizar processos, reduzir custos e aumentar a capacidade de resposta às dinâmicas do mercado.

IIoT consiste na integração de sensores, dispositivos e máquinas em uma rede de comunicação que permite a coleta e análise de dados em tempo real (SISINNI *et al.*, 2018). Essa integração é crucial para a transformação digital das indústrias, pois favorece a interoperabilidade entre sistemas e viabiliza uma visão holística das operações. Por isso, arquiteturas de dados flexíveis e escaláveis são imprescindíveis. Nesse cenário, ferramentas como o Apache Kafka

surge como uma ferramenta-chave nesse contexto, no qual trata-se de uma plataforma distribuída de streaming de dados, projetada para lidar com grandes volumes de informação com baixa latência e alta durabilidade, permitindo publicação, assinatura e processamento contínuo de eventos em tempo real (SHAPIRA *et al.*, 2021).

Diante deste panorama, este trabalho busca explorar as potencialidades do Apache Kafka na implementação de soluções IIoT, evidenciando como essa tecnologia pode contribuir para a modernização dos processos industriais. A seguir, a seção de Motivação detalha as razões que justificam a escolha deste tema e sua relevância para o contexto atual da indústria.

1.1 Motivação

O estado do Ceará apresenta um cenário industrial relevante para a aplicação de tecnologias da Indústria 4.0. De acordo com a Confederação Nacional da Indústria (CNI), a indústria representa 20,5% do Produto Interno Bruto (PIB) estadual, totalizando cerca de R$ 34,2 bilhões. Dentro desse contexto, o setor coureiro-calçadista, embora responda por 7,2% do valor, destaca-se por sua capacidade de geração de empregos e impacto na balança comercial cearense, especialmente em municípios do interior que dependem fortemente dessa atividade econômica (INDÚSTRIA, 2024).

A competitividade do setor, no entanto, é fortemente impactada por fatores como sazonalidade da demanda, variações nos modelos de produção, necessidade de personalização em massa e pressões por prazos curtos de entrega (MESQUITA *et al.*, 2016). Tais dinâmicas exigem maior flexibilidade operacional e agilidade na tomada de decisões. Segundo Contador *et al.* (2020), flexibilidade e adaptabilidade são elementos essenciais para a sobrevivência das indústrias brasileiras diante das exigências impostas pela Indústria 4.0.

Nesse cenário, a experiência profissional do autor como engenheiro de automação em uma indústria calçadista de grande porte no Ceará permite observar diretamente os desafios enfrentados na prática. A rápida implantação de soluções de automação e de IIoT tem gerado um volume crescente de dados provenientes de sistemas heterogêneos, tais como controladores industriais, sensores, sistemas supervisórios e bancos de dados legados. A ausência de uma arquitetura integrada dificulta a visibilidade em tempo real dos processos produtivos e compromete a capacidade de resposta da gestão industrial.

Frente a esse desafio, torna-se necessário implementar soluções que estejam alinhadas aos princípios da Indústria 4.0 — especialmente no que diz respeito à conectividade,

interoperabilidade e modularidade — sem recorrer, no entanto, a reengenharias complexas e onerosas. Isso se justifica pelo fato de que a maioria das indústrias locais opera com margens reduzidas e não pode arcar com grandes paradas de produção para reestruturação completa de seus sistemas (AHMED *et al.*, 2023).

Uma alternativa viável para esse cenário é o uso de arquiteturas de software altamente desacopladas, capazes de integrar dados de sistemas de Tecnologia Operacional (TO) e Tecnologia da Informação (TI), enriquecer essas informações com modelos de ativos industriais e disseminar os dados em diferentes protocolos e formatos. Plataformas como o Apache Kafka se destacam por possibilitar a transmissão eficiente de eventos em tempo real(BOSI *et al.*, 2020), com baixa latência e suporte a formatos flexíveis como *JavaScript Object Notation* (JSON), tornando-se uma ferramenta estratégica para viabilizar a transformação digital no setor calçadista cearense.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver e implementar uma arquitetura de mensageria baseada em eventos, com o uso do Apache Kafka, para aprimorar a visibilidade e integração dos dados provenientes dos sistemas de TO e TI em uma indústria calçadista de grande porte.

1.2.2 Objetivos Específicos

- Descrever a relevância da integração de dados provenientes dos sistemas de TO e TI para o contexto de uma indústria calçadista;
- Avaliar os ganhos operacionais e de desempenho obtidas com a utilização do Apache Kafka em uma arquitetura de mensageria baseada em eventos;
- Analisar a necessidade da adoção de arquiteturas IIoT para otimização de processos e aumento da competitividade industrial;
- Investigar os impactos da implementação de uma arquitetura de mensageria baseada em eventos no sistema de alertas fabril, considerando aspectos de performance e escalabilidade.

1.3 Organização do Trabalho

Este trabalho está organizado em capítulos que cobrem desde a Fundamentação Teórica até os Resultados e Conclusões da pesquisa. Iniciando pelo Capítulo 2, onde será apresentada a Fundamentação Teórica, que discute os conceitos centrais da Indústria 4.0, abordando o cenário brasileiro. Além disso, serão explorados os Sistemas IIoT e a tecnologia Apache Kafka, essenciais para a compreensão da proposta deste trabalho. Também será feita uma revisão de Trabalhos Relacionados, onde são analisados artigos e pesquisas relevantes para a temática abordada, com diferentes arquiteturas e soluções tecnológicas aplicadas em ambientes industriais.

O Capítulo 3 descreve a Metodologia adotada neste trabalho. Trazendo uma visão geral sobre os desafios da integração de dados na era da Indústria 4.0 e a necessidade de arquiteturas baseadas em eventos para otimizar a coleta e análise de dados em tempo real. Em seguida, serão detalhados os materiais e métodos utilizados, incluindo a pesquisa de modelos de arquiteturas, requisitos do sistema e comparativos, culminando em um estudo de caso aplicado a uma indústria calçadista, onde a proposta de arquitetura será implementada e avaliada.

Serão apresentados os Resultados Preliminares obtidos a partir da implementação da arquitetura proposta no Capítulo 4. Este capítulo incluirá uma análise preliminar dos dados coletados, o desempenho do sistema em termos de latência, escalabilidade, e a eficácia na comunicação de alertas em tempo real para a cadeia de ajuda.

Finalmente, o Capítulo 5 traz as Conclusões e Atividades Futuras onde serão discutidas as principais contribuições do trabalho, as limitações encontradas durante a pesquisa, e sugestões para futuras investigações e melhorias na arquitetura proposta.

2 FUNDAMENTAÇÃO TEÓRICA

A Fundamentação Teórica deste trabalho busca contextualizar os conceitos e tecnologias essenciais para a compreensão da proposta abordada. O objetivo é oferecer um embasamento sólido sobre a Indústria 4.0, com ênfase no cenário brasileiro, e explorar as ferramentas tecnológicas que viabilizam a integração de dados e otimização dos processos industriais, como o Apache Kafka e os Sistemas IIoT. Além disso, será feita uma análise de trabalhos e soluções existentes que aplicam tais tecnologias em contextos industriais, com foco em arquitetura de dados e análise em tempo real. A seguir, serão detalhados os principais conceitos e estudos que fundamentam a investigação realizada neste trabalho.

2.1 Indústria 4.0 no Cenário Brasileiro

O cenário da Indústria 4.0 no Brasil apresenta uma série de desafios e oportunidades que refletem a complexidade de implementar tecnologias emergentes em um ambiente industrial marcado por desigualdades regionais e questões estruturais. O trabalho de Contador *et al.* (2020), trouxe diversas informações relevantes sobre o cenário brasileiro. O autor questionou 39 lideres de industrias brasileiras e destacou 28 desafios e 23 oportunidades. Embora o conceito de Indústria 4.0 esteja ganhando destaque no país, a transição efetiva para esse novo paradigma tecnológico ainda enfrenta barreiras significativas. O desafio mais importante é a necessidade de construir ou reformar a infraestrutura tecnológica que, em muitas regiões, é insuficiente para suportar as exigências da Indústria 4.0. A necessidade de investir em equipamentos avançados, sistemas de TI integrados e redes de comunicação robustas é uma realidade que muitas empresas ainda não conseguem atender plenamente pois sua implantação pode levar muito tempo a ponto de trazer o risco da mesma se tornar obsoleta rapidamente. Conforme apontado por Oliveira *et al.* (2009) e Babenko *et al.* (2019), a resistência organizacional e a complexidade dos processos podem aumentar o tempo necessário para implantação, tornando mais provável o risco de obsolescência precoce. Além disso, a coexistência de tecnologias legadas e novas soluções digitais cria uma barreira para a integração e automação completas dos processos produtivos.

Outro desafio destacado é a capacitação da força de trabalho. A Indústria 4.0 requer habilidades específicas em áreas como IoT, *big data* e automação, que ainda são escassas no mercado brasileiro. Essa dificuldade de encontrar profissionais qualificados ocupou a sexta posição entre os principais obstáculos para a adoção dessas tecnologias no país Contador *et al.*

(2020), apesar da notória escassez de especialistas em tecnologias avançadas no Brasil reforçada pelo relatório de "Tendências Globais de RH no setor de Manufatura – 2023"pela Gi Group Holding (2023). Além disso, Contador *et al.* (2020) observou que a cultura organizacional em muitas indústrias brasileiras ainda é resistente a mudanças, com uma tendência a valorizar métodos tradicionais de produção e gestão, o que retarda a transição para um ambiente mais digital e automatizado.

Como complemento, o trabalho de Medeiros (2021) oferece uma perspectiva adicional sobre a implementação da Indústria 4.0 no Brasil, baseada em entrevistas com representantes de indústrias brasileiras e uma análise da teoria substantiva. O estudo revela que a implementação da Indústria 4.0 é afetada por uma combinação de condições externas e internas. Entre as condições externas, destacam-se a conjuntura econômica, a presença ativa de ecossistemas de inovação, e o ambiente competitivo, incluindo as mudanças constantes no mercado consumidor. Internamente, a cultura organizacional e a capacidade dos líderes e da força de trabalho em identificar oportunidades de inovação são fatores críticos.

Medeiros (2021) sugere que o Brasil está passando por uma corrida de transformação digital, onde é essencial tornar a Indústria 4.0 uma prioridade estratégica. As ações recomendadas para enfrentar os desafios incluem buscar parcerias, diagnosticar operações para identificar lacunas, desenvolver e implementar projetos de inovação, e gerenciar as incertezas associadas a esses projetos. As consequências esperadas da adoção da Indústria 4.0 incluem melhorias nas condições de trabalho, aumento da eficiência produtiva, retorno sobre o investimento, maior competitividade e valor agregado aos produtos e serviços. No entanto, uma consequência negativa dessa corrida é a proliferação de soluções tecnológicas diversificadas e implantadas sem o devido planejamento, que muitas vezes apresentam dificuldades de integração. Esse cenário resulta em um ambiente industrial onde há uma multiplicidade de sistemas que não se comunicam bem entre si, trazendo desafios significativos na hora de escalar e integrar as tecnologias. Essas dificuldades aumentam o retrabalho e a complexidade operacional, tornando o processo de adoção e escalabilidade mais oneroso e menos eficiente. Diante desse cenário de fragmentação e dificuldades de integração entre sistemas industriais, torna-se essencial adotar soluções que facilitem a comunicação e o fluxo de dados de maneira escalável e eficiente. É nesse contexto que o Apache Kafka se destaca como uma plataforma robusta para unificar e otimizar a troca de informações na Indústria 4.0.

2.2 Apache Kafka

Apache Kafka é uma plataforma distribuída de streaming de dados projetada para lidar com grandes volumes de informações de forma escalável e eficiente. O livro de Shapira *et al.* (2021) é uma referência abrangente, onde tem o detalhamento do modelo de comunicação utilizado pelo Kafka. Baseado no padrão *publish/subscribe (pub/sub)*, sendo uma arquitetura onde os produtores publicam mensagens em tópicos sem direcioná-las explicitamente a consumidores específicos. Os consumidores, por sua vez, se inscrevem em tópicos para receber as mensagens. Um tópico pode ser subdividido em várias partições, o que permite a distribuição dos dados entre múltiplos servidores, garantindo escalabilidade horizontal e aumentando a capacidade de processamento. Cada partição funciona como um *log*, no qual as mensagens são armazenadas de maneira sequencial e durável em campos chamados comumente de *offsets*, garantindo que possam ser lidas posteriormente na mesma ordem.

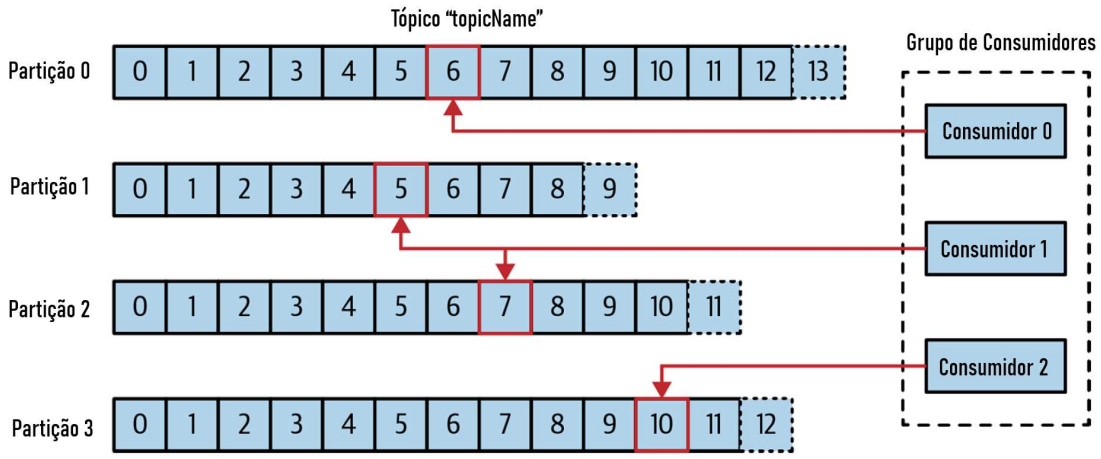
A Figura 1 representa um tópico com múltiplas partições e consumidores obtendo as mensagens nas partições. No Kafka, os tópicos são divididos em partições para permitir o processamento paralelo e aumentar a escalabilidade do sistema. Na imagem, o tópico "*topicName*"possui quatro partições (*Partition 0, Partition 1, Partition 2 e Partition 3*), cada uma contendo mensagens ordenadas sequencialmente (representadas pelos números dentro dos blocos azuis). Os consumidores, organizados em um *Consumer Group*, são responsáveis por processar as mensagens. Cada consumidor dentro do grupo é automaticamente atribuído a uma ou mais partições, garantindo que as mensagens dentro de uma partição sejam consumidas por apenas um consumidor do grupo, mas nunca simultaneamente por múltiplos consumidores.

Na Figura 1, podemos observar:

- *Consumer 0* lendo mensagens da *Partition 0* (posição 6).
- *Consumer 1* consumindo mensagens tanto da *Partition 1* (posição 5) quanto da *Partition 2* (posição 7).
- *Consumer 2* processando mensagens da *Partition 3* (posição 10).

Essa distribuição garante balanceamento de carga e paralelismo na leitura dos dados. Caso um consumidor falhe, as partições que ele consumia podem ser redistribuídas entre os demais consumidores do grupo.

Figura 1 – Representação de um tópico com múltiplas partições e consumidor obtendo mensagens

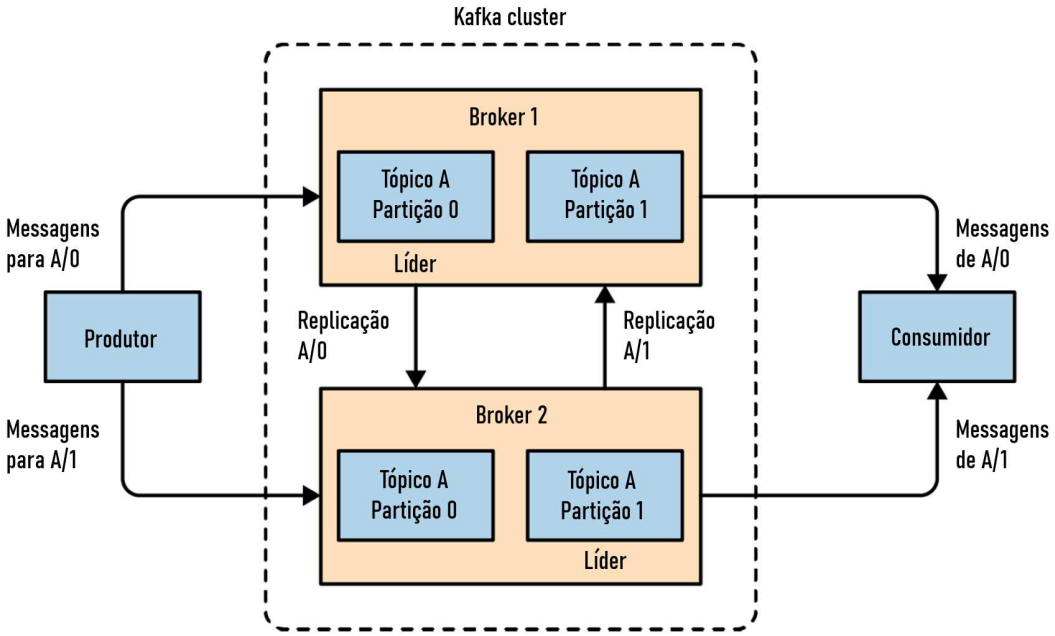


Fonte: Traduzido e adaptado de Shapira *et al.* (2021)

Um dos diferenciais do Kafka é a capacidade de oferecer alta disponibilidade e tolerância a falhas (RAPTIS; PASSARELLA, 2022). As partições podem ser replicadas em diferentes servidores, o que assegura a continuidade das operações mesmo diante da falha de um nó. O sistema também permite que as mensagens sejam consumidas de forma eficiente através de grupos de consumidores. Nesse modelo, cada partição é atribuída a apenas um consumidor por grupo, garantindo que múltiplos consumidores possam operar em paralelo, dividindo a carga de trabalho e aumentando a capacidade de processamento de grandes volumes de dados. Caso um consumidor falhe, o Kafka redistribui automaticamente as partições entre os consumidores restantes do grupo.

A Figura 2 ilustra esse modelo de replicação e consumo de mensagens no *Kafka*. O sistema é composto por múltiplos *brokers*, que armazenam as mensagens divididas em *partições*. Cada *partição* possui um líder responsável por receber os dados do *producer* e coordenar a replicação para outros *brokers*, garantindo alta disponibilidade e tolerância a falhas. No esquema apresentado, os consumidores acessam as mensagens diretamente dos *brokers*, garantindo um fluxo contínuo de dados. Esse mecanismo permite escalabilidade e eficiência no processamento de eventos distribuídos, assegurando a integridade e a disponibilidade das mensagens mesmo em cenários de falha de nós individuais (ESTRADA, 2018).

Figura 2 – Replicação de partições em um *cluster* Kafka



Fonte: Traduzido e adaptado de Shapira *et al.* (2021)

O Kafka oferece um conjunto avançado de *Application Programming Interface* (API) que ampliam sua funcionalidade, como o Kafka Streams, que permite o processamento de fluxos de dados em tempo real, e o Kafka Connect, voltado para integração de dados com outras plataformas. Essas APIs são construídas sobre os componentes fundamentais de produtores e consumidores, mas fornecem um nível mais elevado de abstração, facilitando a implementação de pipelines de dados complexos. O Kafka Streams, por exemplo, permite a transformação de dados em tempo de execução, trazendo consigo todo o conceito de *streaming* de dados(KLEPPMANN, 2016).

A flexibilidade e a interoperabilidade são aspectos fundamentais no tratamento de dados distribuídos (STEEN; TANENBAUM, 2017). Embora o Kafka ofereça mecanismos robustos para a transmissão e o processamento contínuo de eventos, a maneira como os dados são estruturados impacta diretamente na eficiência da comunicação entre sistemas. Nesse contexto, o uso de formatos padronizados, como *Apache Avro*, facilita a compatibilidade entre produtores e consumidores (SHAPIRA *et al.*, 2021). No entanto, há cenários em que um formato mais amplamente adotado e de fácil interpretação se faz necessário, especialmente em aplicações web e integração de serviços. O *JavaScript Object Notation* surge, então, como uma alternativa popular para a serialização de mensagens, permitindo a troca de dados estruturados de maneira simples e legível, conforme será discutido na próxima subseção.

2.2.1 JSON: JavaScript Object Notation

JSON é um formato leve de troca de dados, amplamente utilizado na comunicação entre aplicações e na troca de informações em ambientes web. Desenvolvido com o objetivo de ser flexível e funcional para máquinas e fácil de ler e escrever para seres humanos. De acordo com o RFC 8259 (Internet Engineering Task Force, 2017) a sua sintaxe é derivada da notação de objetos do JavaScript, mas o formato é independente de linguagem, com suporte para diversas linguagens de programação como Python, Java e C++. A simplicidade e a legibilidade do JSON são aspectos destacados no estudo de Wehner *et al.* (2014), que evidenciam sua popularidade como formato de troca de dados em sistemas distribuídos e aplicações web.

O JSON é um texto baseado em uma estrutura de pares chave-valor e em listas ordenadas de valores. Esta estrutura é composta por dois tipos principais de dados: objetos e arrays. As chaves são strings e os valores podem ser strings, números, objetos, arrays, booleanos ou ‘null‘. Já um array JSON é uma lista ordenada de valores, que podem ser de qualquer tipo permitido no formato JSON. Esta flexibilidade permite que o JSON seja utilizado em uma ampla gama de aplicações, desde a configuração de sistemas até a transmissão de dados em tempo real. Abaixo está um exemplo de um objeto JSON representando informações sobre um usuário em um sistema:

```
{
    "nome": "Henrique Soares",
    "idade": 23,
    "email": "henrique.soares@examplo.com",
    "ativo": true,
    "enderecos": [
        {
            "tipo": "residencial",
            "logradouro": "Rua Almirante Sabino, 123",
            "cidade": "Balsas",
            "estado": "MA"
        },
        {
            "tipo": "comercial",
            "logradouro": "Rua Imperador, 200",
            "cidade": "São Luís",
            "estado": "MA"
        }
    ]
}
```

No exemplo acima, o objeto JSON representa um usuário com informações pessoais e de contato. O objeto possui pares chave-valor para "nome", "idade"e "email", onde os valores são uma string, um número e uma string, respectivamente. Além disso, o campo "enderecos"é um array que contém dois objetos, cada um representando um endereço diferente com detalhes como "tipo", "logradouro", "cidade"e "estado". O campo "ativo"é um valor booleano que indica se o usuário está ativo ou não. Esta estrutura permite uma representação clara e organizada dos dados tanto para leitura humana como para utilização para troca de dados entre sistemas.

2.3 Trabalhos Relacionados

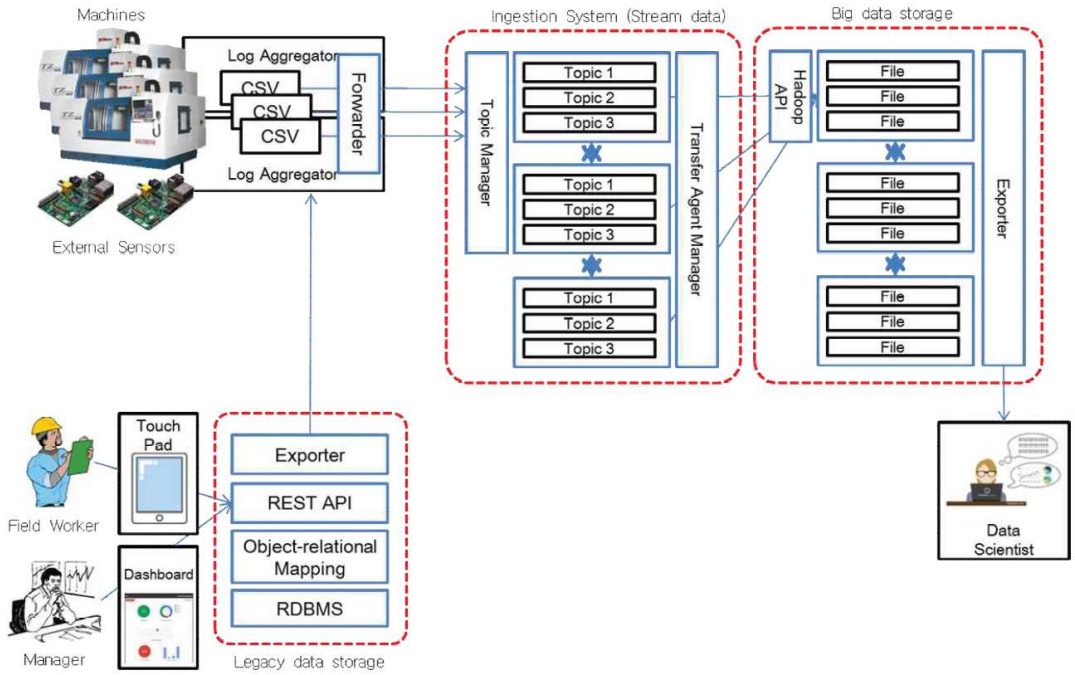
Para embasar a proposta deste trabalho, foi realizada uma revisão de pesquisas e arquiteturas previamente desenvolvidas que tratam do uso de *streaming* de dados e mensageria em ambientes industriais como base técnica para o desenvolvimento. Conforme discutido na Seção 2.2, o *Apache Kafka* tem sido amplamente adotado em cenários de processamento de dados em tempo real, incluindo a Indústria 4.0. Com isso, esta seção apresenta uma análise de abordagens relacionadas, dividindo-as em categorias que contemplam desde arquiteturas de ingestão de dados até aplicações específicas do *Kafka* no contexto industrial.

2.3.1 Arquitetura de Ingestão de Dados

O uso de *streaming* de dados em ambientes industriais tem ganhado destaque devido à capacidade de processar grandes volumes de dados em tempo real. Park e Chi (2016) propuseram a implementação de um sistema de ingestão de dados de alta vazão voltado para *logs* de máquinas na indústria de manufatura. O sistema é arquitetado para lidar com grandes volumes de dados gerados pelas máquinas, utilizando o Apache Kafka como a fila central de mensagens. O Kafka desempenha um papel crítico, permitindo o manuseio e o armazenamento temporário dos fluxos de dados em alta velocidade, organizando-os em tópicos particionados para garantir a escalabilidade e a eficiência do processamento. A Figura 3 exibe uma visão geral da arquitetura de Park e Chi (2016).

Além do Kafka, a solução integra o Apache Flume para gerenciar o processo de ingestão e sincronização dos dados entre o Kafka e o Hadoop Distributed File System (HDFS), onde os dados são armazenados de forma distribuída e escalável. A arquitetura proposta destaca-se pela escalabilidade e pela capacidade de lidar com grande volume de *logs* de máquina,

Figura 3 – Arquitetura de Ingestão de Dados



Fonte: (PARK; CHI, 2016)

garantindo uma gestão centralizada dos dados que facilita análises preditivas e a tomada de decisões em ambientes industriais. Assim, o trabalho de Park e Chi (2016) demonstrou uma abordagem robusta para a ingestão e gerenciamento de dados em tempo real na manufatura, utilizando tecnologias modernas de *big data*. Ele também destacou como o uso de *streaming* melhora significativamente a capacidade de ingerir e processar dados sem causar atrasos, o que é vital para a operação eficiente de fábricas inteligentes.

2.3.2 Gateway IIOT Adaptativo

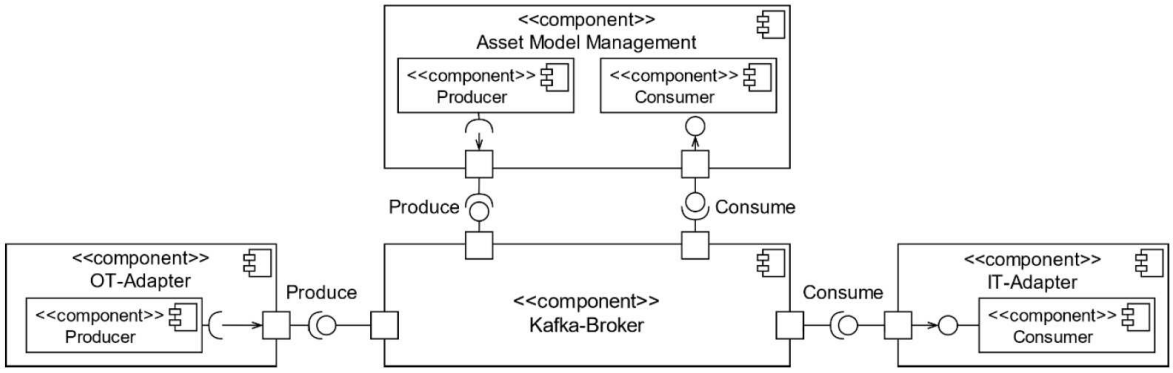
Braunisch *et al.* (2022) propuseram um *gateway* IIoT adaptativo utilizando uma proposta de *hardware* que é capaz de realizar a coleta de dados de máquina, chamado de *OT-Adapter* e publicar usando a plataforma de *streaming* Kafka, demonstrando como a tecnologia pode ser integrada para melhorar a flexibilidade e a adaptabilidade de arquiteturas industriais. O trabalho propõe uma solução inovadora para o uso do Apache Kafka em *gateways* adaptativos de IIoT. Os autores também trouxeram uma sugestão de arquitetura utilizando o Kafka como a plataforma central de *streaming* para gerenciar e processar dados em tempo real, coletados de dispositivos no chão de fábrica.

O autor também incluiu na arquitetura em uma segunda proposta de hardware chamada de *IT-Adapter*, que serve como uma interface entre os dispositivos de campo e a

plataforma de streaming, realizando a coleta, filtragem e agregação de dados em tempo real antes de transmiti-los para os sistemas de TI, como *Enterprise Resource Planning* / Planejamento de Recursos Empresariais (ERP) ou *Manufacturing Execution System* / Sistema de Execução da Manufatura (MES).

A Figura 4 apresenta a arquitetura proposta para integração de dados entre os sistemas operacionais da fábrica (*OT - Operational Technology*) e os sistemas de tecnologia da informação (*IT - Information Technology*), utilizando o *Apache Kafka* como intermediário na comunicação. No lado esquerdo da Figura, o *OT-Adapter* é responsável por coletar dados das máquinas industriais. Ele contém um componente *Producer*, que publica os dados coletados diretamente no *Kafka-Broker*. O *Kafka-Broker*, localizado no centro da Figura, atua como um intermediário confiável para a transmissão dos dados, garantindo que as mensagens publicadas pelos *producers* sejam armazenadas temporariamente e possam ser consumidas de maneira escalável. No lado direito da Figura, o *IT-Adapter* contém um componente *Consumer*, que recupera os dados do *Kafka-Broker* para disponibilizá-los aos sistemas de TI, permitindo análise e uso das informações na gestão da produção.

Figura 4 – Arquitetura IIoT Adaptativa



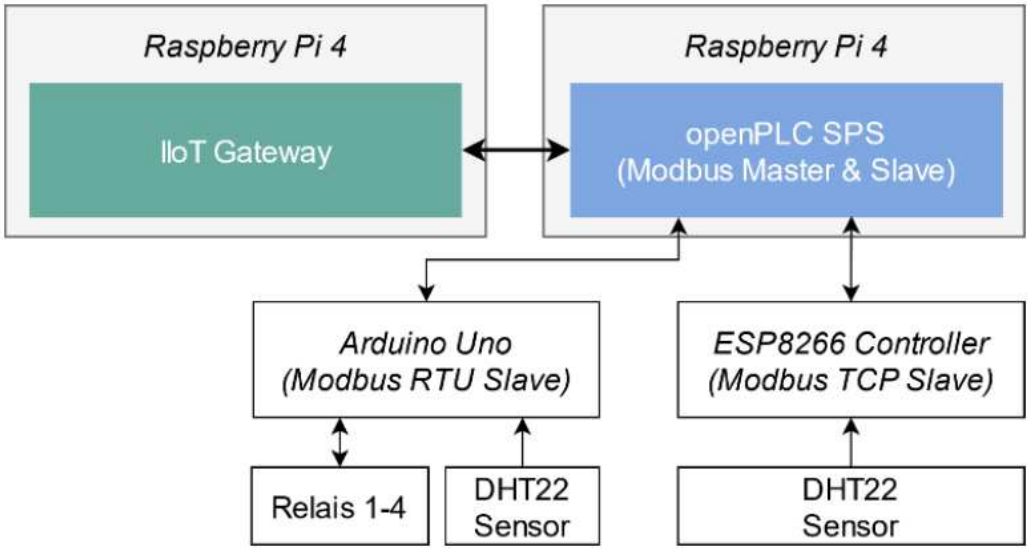
Fonte: (BRAUNISCH *et al.*, 2022)

A arquitetura incluiu ainda o uso de modelos de ativos (*asset models*), que representam as capacidades dos dispositivos conectados e o tipo de dados que geram. Ele está localizado na parte superior da Figura, o módulo *Asset Model Management* contém tanto um *Producer* quanto um *Consumer*, indicando que ele desempenha um papel bidirecional: pode tanto consumir dados do *Kafka* quanto produzir novas informações. Esses modelos permitem que o sistema adapte suas operações de acordo com as condições operacionais e os requisitos de fabricação. Essa flexibilidade resulta em uma solução altamente escalável e eficiente, capaz de priorizar dados relevantes e otimizar o uso da largura de banda e armazenamento, o que é crucial no con-

texto da Indústria 4.0. Desta forma ela garante uma comunicação eficiente e desacoplada entre os sistemas *OT* e *IT*, permitindo a escalabilidade e facilitando a implementação de aplicações industriais baseadas em eventos.

Para avaliar o desempenho do *gateway IIoT* em aplicações críticas em tempo real, os autore realizaram testes de performance utilizando uma prototipação de *hardwares* composta por um Raspberry Pi 4, Arduino Uno e ESP8266, conforme pode ser visto na Figura 5. O cenário de teste envolveu a publicação de dados enviados do *OT-Adapter*, representado pelo conjunto à direita (em azul), para o *IT-Adapter*, representado pelo segundo conjunto à esquerda (em verde).

Figura 5 – Arquitetura IIoT Adaptativa para testes em campo



Fonte: (BRAUNISCH *et al.*, 2022)

Na configuração proposta, um Raspberry Pi 4 executando o *openPLC SPS* opera simultaneamente como mestre e escravo Modbus, permitindo a comunicação com diferentes dispositivos de automação industrial. O *gateway IIoT*, implementado em outro Raspberry Pi 4, é responsável por intermediar essa comunicação, garantindo a interoperabilidade entre os dispositivos e a infraestrutura de TI. O Arduino Uno, configurado como escravo Modbus RTU, está conectado a sensores DHT22 e a um conjunto de relés, simulando a coleta e o acionamento de dispositivos físicos dentro do ambiente industrial. Da mesma forma, o ESP8266 atua como um escravo Modbus TCP, fornecendo leituras do sensor DHT22 via rede sem fio, ampliando a flexibilidade do sistema.

Para verificar a performance, foi calculado a latência do sistema através do tempo necessário para o *OT-Adapter* capturar um dado do conjunto de controladores e publicá-lo no *IT-Adapter*. A latência foi calculada usando a fórmula:

$$\text{Latência} = \Delta t = t_{\text{publish}} - t_{\text{poll}} \tag{2.1}$$

Onde t_{poll} é o *timestamp* gerado pelo *OT-Adapter* imediatamente antes da solicitação ao sensor via Modbus TCP, e t_{publish} é o *timestamp* definido pelo *IT-Adapter* imediatamente após a publicação no *broker* MQTT. A sincronização dos relógios do *gateway* e do PLC foi ignorada para simplificar a análise, focando apenas na duração da transmissão pelo *gateway*.

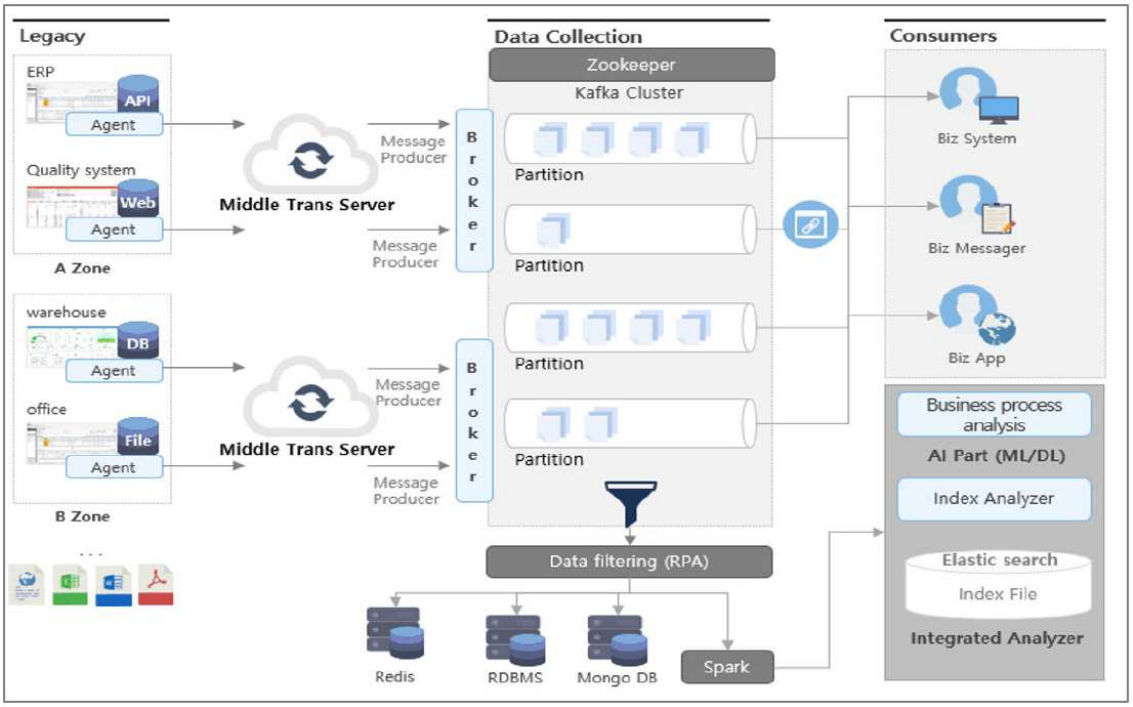
Foram registradas duas séries de medições com um total de 14.000 medições individuais de latência, onde na série com intervalo de *polling* de 100 ms, mais de 90% dos valores foram publicados no *broker* em 10 ms ou menos. A latência máxima observada foi de 122 ms, identificada como um *outlier*, provavelmente devido às limitações de tempo crítico e aos recursos do sistema. Esses resultados indicam que, apesar de algumas variações esporádicas, o sistema é capaz de atender aos requisitos de latência para aplicações práticas, com a maioria dos valores sendo processados rapidamente e eficientemente.

2.3.3 Apache Kafka para Big Data em Fábricas Inteligentes

Park e Huh (2023) propuseram uma arquitetura baseada no Apache Kafka para a coleta e utilização de *Big Data* em ambientes de fábricas inteligentes distribuídas geograficamente. O artigo aborda os desafios encontrados na coleta de grandes volumes de dados gerados por equipamentos, sensores IoT e sistemas de manufatura em fábricas dispersas, onde a troca de informações entre fábricas distantes se torna cada vez mais necessária. A solução apresentada utiliza uma rede em grade baseada em agentes e *brokers* Kafka, permitindo a coleta distribuída de dados e seu posterior refinamento para facilitar a análise integrada.

A arquitetura proposta, exibida na Figura 6, visa superar as limitações das soluções tradicionais, que utilizam sistemas centralizados como *Enterprise Service Bus* / Barramento de Serviços Empresariais (ESB) ou *Enterprise Application Integration* / Integração de Aplicações Empresariais (EAI), frequentemente inadequados para lidar com a diversidade e dispersão dos dados gerados em múltiplas fábricas. O uso do Apache Kafka permite a escalabilidade na coleta e transmissão de dados, garantindo que grandes volumes de dados sejam processados com eficiência.

Figura 6 – Arquitetura proposta para coleta de grandes volumes de dados



Fonte: (PARK; HUH, 2023)

Na arquitetura representada, diferentes fontes de dados, como *ERP*, sistemas de qualidade, bancos de dados industriais e arquivos distribuídos em diferentes zonas de produção, utilizam agentes específicos para capturar e enviar os dados a um servidor intermediário de transporte (*Middle Trans Server*). Esse servidor é responsável por atuar como produtor de mensagens para os *brokers* Kafka, que organizam os dados em múltiplas partições dentro de um cluster gerenciado pelo Zookeeper.

Os dados coletados podem então passar por um processo de filtragem automatizado (*Data Filtering RPA*), antes de serem armazenados em diferentes sistemas, como bancos de dados NoSQL (MongoDB, Redis), bancos de dados relacionais (RDBMS) e plataformas de processamento distribuído, como Apache Spark. Além disso, os dados são consumidos por diferentes sistemas empresariais, aplicativos e serviços de análise avançada, incluindo aprendizado de máquina e análise de processos de negócios.

O estudo também avaliou a performance da arquitetura através de testes de carga em servidores distribuídos, demonstrando que o sistema consegue lidar com até 200GB de dados por agente, sem comprometer a performance de *Central Processing Unit* / Unidade Central de Processamento (CPU) ou memória. A conclusão do estudo destaca que a arquitetura é uma solução robusta para coletar, integrar e analisar dados em larga escala, especialmente em cenários de manufatura distribuída.

2.3.4 Avaliação de Performance do Kafka em Ambientes de Alto Throughput

Noac'h *et al.* (2017) propuseram um estudo aprofundado sobre o desempenho do Apache Kafka em cenários com alto volume de dados (*throughput*). O objetivo central do trabalho foi avaliar como o Kafka se comporta em diferentes condições de carga, configurando parâmetros técnicos como tamanho de lote (*batch size*), número de tópicos, quantidade de consumidores/produtores e número de partições. O foco é oferecer orientações práticas para a implementação de arquiteturas de dados em larga escala, evidenciando os limites de escalabilidade e os gargalos do sistema.

O experimento foi realizado no ambiente *Grid'5000*, uma infraestrutura de pesquisa distribuída e reconfigurável voltada para experimentação em larga escala nas áreas de computação em nuvem, redes, e sistemas distribuídos.Balouek *et al.* (2013) descrevem a infraestrutura *Grid'5000* como uma plataforma experimental, altamente reconfigurável, projetada para experimentos em larga escala com redes, sistemas distribuídos, computação em nuvem e *Big Data*. Ela fornece controle total sobre o sistema operacional, rede e escalabilidade dos recursos, sendo amplamente utilizada na pesquisa acadêmica e industrial. No estudo, foram utilizadas múltiplas máquinas físicas dedicadas, cada uma configurada com instâncias do Kafka, produtores e consumidores, conectadas por uma rede de alta velocidade. Isso garantiu um ambiente controlado, isolado de variações externas, ideal para testes de stress com alta fidelidade.

A arquitetura experimental consistia em múltiplos produtores gerando mensagens que eram publicadas em tópicos distintos, sendo depois consumidas por consumidores dedicados. O sistema foi parametrizado com diferentes configurações de *batch size*, número de partições por tópico, e instâncias concorrentes, com o objetivo de avaliar como esses fatores afetam o *throughput* e a latência da entrega das mensagens. *Batch* refere-se ao agrupamento de múltiplas mensagens antes de serem enviadas ao *broker*, otimizando o uso da rede e do disco.

Os resultados obtidos mostraram que o Kafka apresenta melhor desempenho com *batch sizes* entre 100 e 500 mensagens, mantendo o equilíbrio entre *throughput* e latência. Lotes maiores aumentam a eficiência do envio de mensagens, pois reduzem a sobrecarga de comunicação, embora introduzam latência adicional. Além disso, o sistema mostrou boa escalabilidade até o ponto em que o número de partições por tópico se torna excessivo, o que começa a degradar o desempenho devido à sobrecarga de gerenciamento interno.

Outro resultado importante do estudo foi a identificação de limites práticos: acima de 500.000 mensagens por segundo, o sistema começou a apresentar sinais de saturação, com aumentos expressivos na latência. Isso destaca a importância de um ajuste fino dos parâmetros para manter a performance ideal em aplicações de *streaming* de grande escala.

O estudo concluiu que, embora o Apache Kafka seja altamente eficiente, sua performance depende fortemente da configuração e da carga de trabalho. O uso de um ambiente como o *Grid'5000* permitiu analisar detalhadamente essas dependências, fornecendo subsídios técnicos valiosos para arquitetos de sistemas que desejam implantar pipelines de dados com alta confiabilidade e *throughput.*

2.3.5 Ferramenta de Testes para Confiabilidade em Kafka

O artigo de Wu *et al.* (2019b) apresenta o TRAK, uma ferramenta de teste desenvolvida para avaliar a confiabilidade da entrega de dados no Apache Kafka. Os autores identificaram que, apesar da popularidade do Kafka, há uma escassez de ferramentas voltadas especificamente para testes sistemáticos da confiabilidade de entrega de mensagens em diferentes cenários de uso.

O experimento conduzido foi implementado em um ambiente controlado onde diferentes cenários de falha e variações de configuração foram simulados para testar o comportamento do Kafka. O ambiente de testes consistia em múltiplas instâncias de Kafka, consumidores e produtores, executados em contêineres Docker, permitindo a reprodutibilidade e o controle detalhado das condições de teste. Os autores analisaram como as mensagens eram entregues (ou não) diante de falhas como perda de conexão, reinicialização de componentes e reconfigurações.

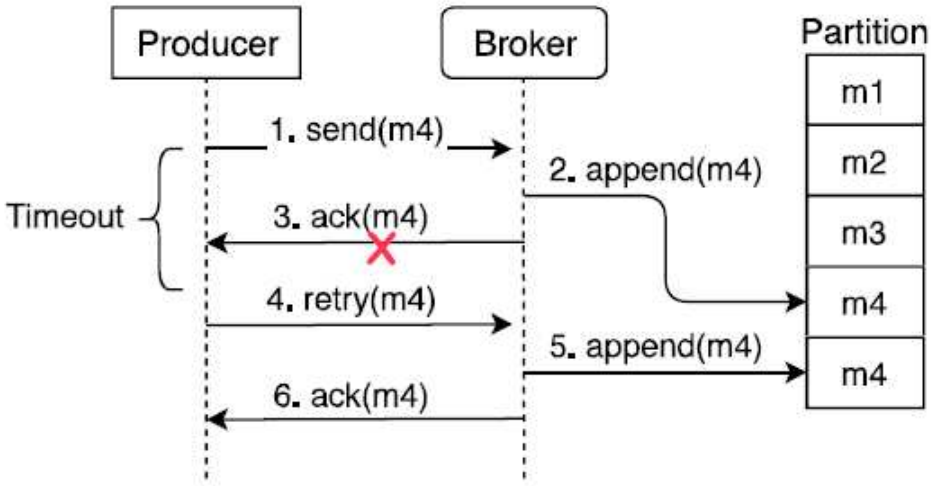
A arquitetura do TRAK é composta por três módulos principais: o gerador de carga (responsável por produzir mensagens para o Kafka), o monitor de falhas (que simula falhas no sistema) e o validador de confiabilidade (que verifica se as mensagens foram entregues corretamente). Essa estrutura modular permitiu aos autores isolar diferentes aspectos do sistema e avaliar, com precisão, a resiliência do Kafka diante de falhas.

A Figura 7 representa os principais componentes do Kafka — producer, broker e consumer — e ilustra como ocorre a entrega de mensagens com garantia de não perda, mesmo na presença de falhas. O diagrama segue o seguinte fluxo:

- O producer envia uma mensagem para o Kafka broker.
- O broker armazena a mensagem com sucesso em seu log.
- O broker envia um acknowledgment (ACK) de volta ao producer, confirmando o recebimento.
- O consumer consome a mensagem do broker.
- O consumer confirma o processamento (commit do offset).

Esse fluxo caracteriza o comportamento conhecido como exactly-once delivery, ou seja, a entrega da mensagem exatamente uma vez. No entanto, a Figura destaca que, caso ocorra uma falha no *consumer* (por exemplo, antes do *commit* do *offset*), a mesma mensagem poderá ser reprocessada após a recuperação do sistema — o que caracteriza o comportamento at-least-once, em que a mensagem é entregue pelo menos uma vez, mas pode ser processada mais de uma vez.

Figura 7 – Fluxo de mensagens no Kafka sob o modelo at-least-once delivery. Garante a entrega mesmo diante de falhas, com possibilidade de duplicação.



Fonte: (WU *et al.*, 2019b)

Os resultados mostraram que a confiabilidade da entrega de mensagens no Kafka pode variar significativamente dependendo das configurações adotadas, como o número de réplicas, o modo de confirmação da escrita (*acknowledgment*), e a tolerância a falhas dos *brokers*. Em um dos principais testes, foram enviadas 500.000 mensagens com a ferramenta TRAK. Os resultados demonstraram zero mensagens perdidas, zero mensagens duplicadas, e uma latência

de entrega entre 5 ms e 10 ms para a maioria das mensagens. Esses números foram obtidos mesmo com a introdução de falhas simuladas durante a execução, evidenciando a robustez do sistema quando corretamente configurado.

Os testes também revelaram que, em situações com alta carga e falhas intermitentes, o Kafka pode apresentar comportamento não determinístico, dificultando a garantia de entrega exata de mensagens (exactly-once delivery). A ferramenta TRAK demonstrou ser eficaz na identificação desses comportamentos e na quantificação do impacto das falhas sobre a entrega de dados.

Concluindo, os autores destacam que o TRAK oferece uma abordagem sistemática para avaliar a confiabilidade de entrega no Kafka e pode ser utilizada tanto por desenvolvedores quanto por operadores para validar configurações e identificar vulnerabilidades. A pesquisa contribui significativamente para o entendimento das limitações operacionais do Kafka em cenários de produção, além de propor uma ferramenta prática e extensível para estudos futuros sobre tolerância a falhas em sistemas de mensageria distribuída.

2.3.6 Considerações sobre a Aplicação do Kafka na Indústria 4.0

Conforme discutido na Seção 2.1, a adoção da Indústria 4.0 no cenário brasileiro enfrenta barreiras estruturais significativas, como a dificuldade de acesso a equipamentos avançados, a escassez de profissionais qualificados e a coexistência de tecnologias legadas com novas soluções digitais. Além disso, o longo tempo necessário para implantar sistemas complexos representa um risco real de obsolescência precoce, conforme apontado por Oliveira *et al.* (2009) e Babenko *et al.* (2019). Tais fatores tornam essencial a busca por soluções tecnológicas que aliem robustez e escalabilidade a uma implantação mais ágil e de menor complexidade.

Os trabalhos analisados neste Capítulo demonstram que o Apache Kafka é uma solução viável e amplamente validada para aplicações industriais de grande escala e alto volume de dados. Os estudos presentes na Seção 2.3 evidenciam o potencial do Kafka em contextos industriais diversos, destacando sua confiabilidade, escalabilidade e tolerância a falhas. Tais arquiteturas demonstraram capacidades de ingestão e processamento em tempo real, entrega confiável de mensagens e resiliência a falhas em ambientes distribuídos.

No entanto, observa-se que essas soluções, embora tecnicamente eficientes, muitas vezes envolvem o uso conjunto de diversas tecnologias de suporte, como HDFS, Apache Flume, gRPC, sistemas NoSQL, e até *hardwares* dedicados, como os *adapters* propostos por Braunisch *et al.* (2022). Esses elementos, embora agreguem valor técnico às soluções, também aumentam a complexidade de implantação e elevam os requisitos de infraestrutura — o que pode representar um desafio adicional para empresas que operam em contextos de baixa maturidade digital, como é o caso de grande parte do parque industrial brasileiro.

Diante desse cenário, esta dissertação propõe uma arquitetura de mensageria baseada em Apache Kafka, com o objetivo de minimizar os riscos de implantação e reduzir a complexidade técnica, tornando-a mais compatível com a realidade de indústrias brasileiras. A partir da análise realizada neste Capítulo, a proposta contempla os seguintes princípios:

- Integração direta com estruturas e sistemas já existentes no ambiente fabril, sem necessidade de substituições ou reconfigurações significativas;
- Eliminação do uso de tecnologias auxiliares complexas, como HDFS ou Flume, priorizando uma solução mais enxuta e operacionalmente viável;
- Utilização de ferramentas e equipamentos acessíveis e amplamente disponíveis, alinhando-se com os recursos típicos das indústrias nacionais;
- Apresentação de um caso prático real, descrito no Capítulo 3, com potencial de retorno visível no curto prazo para operadores e gestores da planta.
- Definição e utilização de métricas de desempenho adequadas para validar a performance da arquitetura em cenários industriais reais.
- Adoção do modelo de entrega *exactly-once delivery*, visando garantir confiabilidade sem sobrecarga de replicações ou reprocessamentos;

Essa abordagem busca contribuir para a adoção gradual e segura da Indústria 4.0 no Brasil, oferecendo um caminho viável de modernização com base em mensageria confiável e de baixo risco.

3 METODOLOGIA

Este Capítulo descreve o processo de desenvolvimento e avaliação da arquitetura proposta. Inicialmente, apresenta-se uma visão geral da metodologia adotada, seguida pela descrição da arquitetura, ferramentas utilizadas e procedimentos de avaliação.

3.1 Visão Geral

Considerando os desafios enfrentados por indústrias no contexto da transformação digital, especialmente diante das exigências da Indústria 4.0, esta pesquisa adota uma abordagem aplicada, de natureza exploratória, com objetivo descritivo. O foco está no desenvolvimento e na avaliação de uma arquitetura IIoT para sistemas de alerta em ambientes industriais.

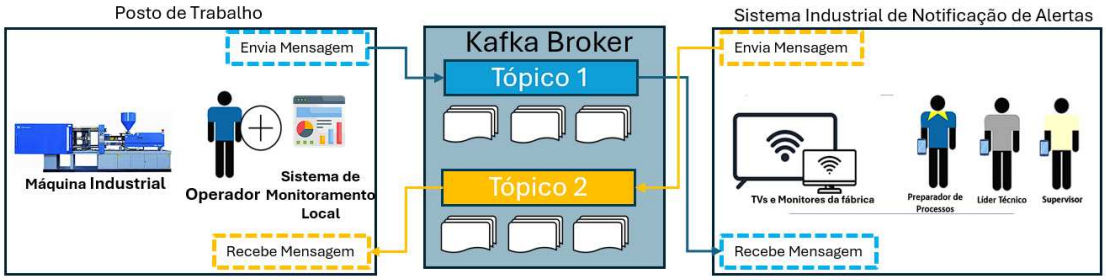
A revolução promovida pela Indústria 4.0 intensificou a necessidade de integração entre processos produtivos, setores administrativos e sistemas de suporte, impulsionando a geração massiva de dados por máquinas, sensores IoT e sistemas industriais heterogêneos. Essa diversidade de fontes e formatos, no entanto, acarreta desafios significativos relacionados à interoperabilidade, visibilidade de dados e latência na comunicação.

Diante desse cenário, o presente trabalho foi estruturado em quatro etapas metodológicas principais:

1. Levantamento dos requisitos funcionais e não funcionais de uma arquitetura voltada ao gerenciamento de alertas em tempo real no contexto industrial, com foco em cenários de falhas operacionais;
2. Desenvolvimento da arquitetura proposta, utilizando tecnologias de código aberto, com ênfase na flexibilidade, escalabilidade horizontal, interoperabilidade entre sistemas legados e modernos, além da tolerância a falhas;
3. Implementação e validação em ambiente industrial real, considerando a coleta contínua de dados provenientes de sensores instalados em máquinas e a geração de mensagens de alerta baseadas em eventos operacionais;
4. Avaliação empírica da arquitetura, com base em métricas como latência, perda de mensagens, duplicação e confiabilidade na entrega, visando verificar a viabilidade técnica e operacional da solução proposta.

Com base nessas etapas, desenvolveu-se uma arquitetura orientada a eventos, baseada no Apache Kafka, que permite a comunicação assíncrona entre máquinas e sistemas corporativos. Essa abordagem busca evitar a sobrecarga de rede causada pelo tráfego massivo de dados, além de eliminar a necessidade de consultas constantes a bancos de dados transacionais. Ao utilizar tópicos Kafka e estrutura padronizada de dados em formato JSON, cada máquina torna-se produtora de mensagens que são processadas em tempo real pelos consumidores autorizados, promovendo uma resposta rápida e eficaz aos eventos registrados no chão de fábrica. A Figura 8 ilustra de forma simplificada a arquitetura de mensageria proposta, que serve como base para essa comunicação orientada a eventos.

Figura 8 – Representação simplificada da arquitetura de mensageria proposta



Fonte: Próprio Autor

As Seções seguintes deste Capítulo detalham a execução dessas quatro etapas. A Seção 3.2 descreve o estudo de caso no ambiente industrial, contextualizando os desafios e justificando os requisitos funcionais e não funcionais que fundamentaram o projeto da arquitetura. Já a Seção 3.3 apresenta os materiais, ferramentas e métodos utilizados, cobrindo desde o desenho da arquitetura com Apache Kafka, passando pela implementação e prototipação em ambiente fabril, até os testes e medições de desempenho realizadas. Dessa forma, o capítulo como um todo documenta de maneira estruturada o processo de concepção, implantação e validação da proposta.

3.2 Estudo de Caso

3.2.1 Cenário Industrial

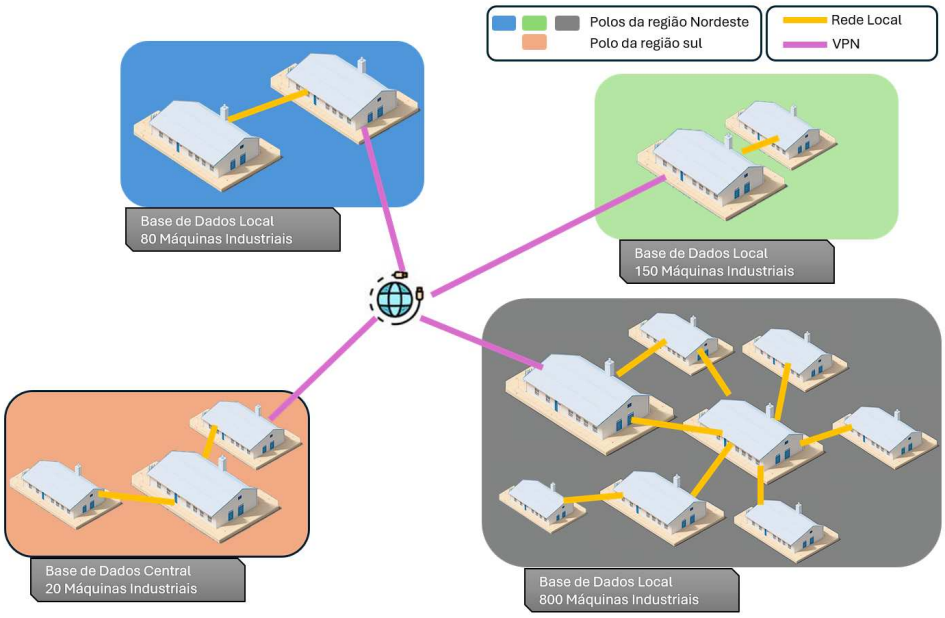
O estudo de caso foi desenvolvido em uma indústria calçadista de grande porte, cujos processos principais envolvem injeção de plástico e montagem de calçados. A empresa já apresenta iniciativas alinhadas à Indústria 4.0, embora vários processos ainda estejam em fase de adaptação, enfrentando desafios como os apontados por Contador *et al.* (2020): infraestrutura deficiente, altos custos de implementação e resistência organizacional.

A empresa possui quatro polos de produção, sendo três na região Nordeste e um na região Sul do Brasil, e utiliza sistemas corporativos como ERP, MES, *Warehouse Management System* / Sistema de Gerenciamento de Armazém (WMS), em sua maioria legados. A fim de viabilizar a integração entre esses sistemas e permitir modernização gradual, foi adotada uma *Service-Oriented Architecture* / Arquitetura Orientada a Serviços (SOA), com o uso de microsserviços e *web services*.

Os sistemas de automação industrial estão em processo contínuo de modernização. Dos quatro polos, aproximadamente 70% da estrutura fabril está concentrada em um único polo, que compreende oito unidades distribuídas em uma área superior a 600.000 m². A infraestrutura de rede foi projetada para suportar essa complexidade, incluindo interconexão entre *racks* de *switches*, segmentação lógica por *Virtual Local Area Network* (VLAN) para aplicações de TI e TO, redes *Wi-Fi mesh* e cerca de 20.000 dispositivos conectados, distribuídos em 40 VLANs. Entre esses dispositivos, mais de 1.000 estão associados ao monitoramento de máquinas industriais semi-automatizadas, que ainda exigem interação humana em partes do ciclo produtivo.

A Figura 9 apresenta uma ilustração da distribuição dos polos industriais da empresa, destacando as interligações principais de rede e a quantidade de pontos de monitoramento em cada unidade fabril. Essa visualização contribui para dimensionar a escala e a complexidade da infraestrutura de comunicação envolvida.

Figura 9 – Ilustração dos polos industriais



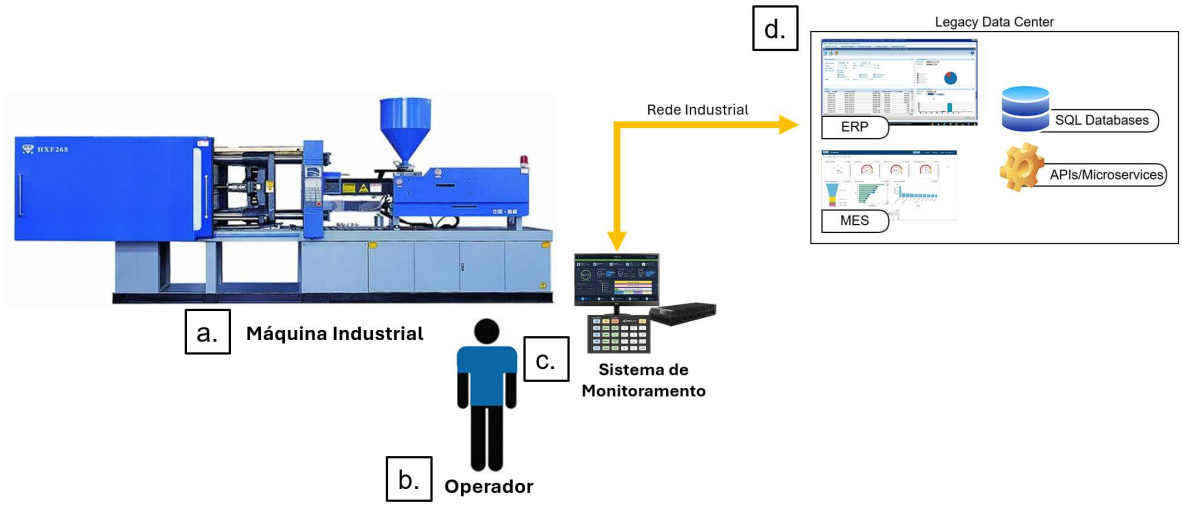
Fonte: Próprio Autor

Nesse contexto de automação e monitoramento, o papel dos operadores continua essencial. Apesar da presença de sistemas automatizados, muitas máquinas exigem ações manuais em determinadas etapas. O operador, profissional que interage diretamente com o maquinário, é responsável não apenas por operá-lo, mas também por consultar e registrar dados de produção por meio de uma instância do sistema MES executada localmente ou do sistema embarcado da própria máquina.

As máquinas industriais são equipadas com Controlador Lógico Programável (CLP)s, sensores, motores, computadores e interfaces de monitoramento, que geram dados tanto para os sistemas automatizados quanto para o registro manual dos operadores. A Figura 10 ilustra um posto de trabalho típico utilizado neste estudo de caso. Na representação, (a.) indica a máquina industrial, (b.) o operador, (c.) o sistema embarcado de monitoramento e (d.) a conexão com os sistemas centrais da fábrica. Essa configuração é generalizada ao longo das unidades fabris e serve de base para o desenvolvimento da arquitetura de mensageria.

Entretanto, esses sistemas de TO e TI ainda apresentam limitações na integração com os sistemas corporativos, os quais concentram os dados de forma centralizada. Esse modelo, embora tradicional, não se mostra adequado para um ambiente com centenas de máquinas dispersas geograficamente, como é o caso desta indústria. Problemas típicos de arquiteturas centralizadas, como aumento da latência e complexidade na integração, têm impacto direto na visibilidade operacional, conforme também apontado por Park e Huh (2023).

Figura 10 – Representação de um posto de trabalho



Fonte: Próprio Autor

3.2.2 Problemática

Como supracitado, mais de 1.000 máquinas industriais são monitoradas por meio de um sistema local integrado diretamente ao MES. Esse sistema permite que o operador interaja com o equipamento, consulte demandas de produção e registre informações relevantes, como contagem de peças, desempenho da máquina, períodos de inatividade, manutenções realizadas e perdas relacionadas à qualidade do produto.

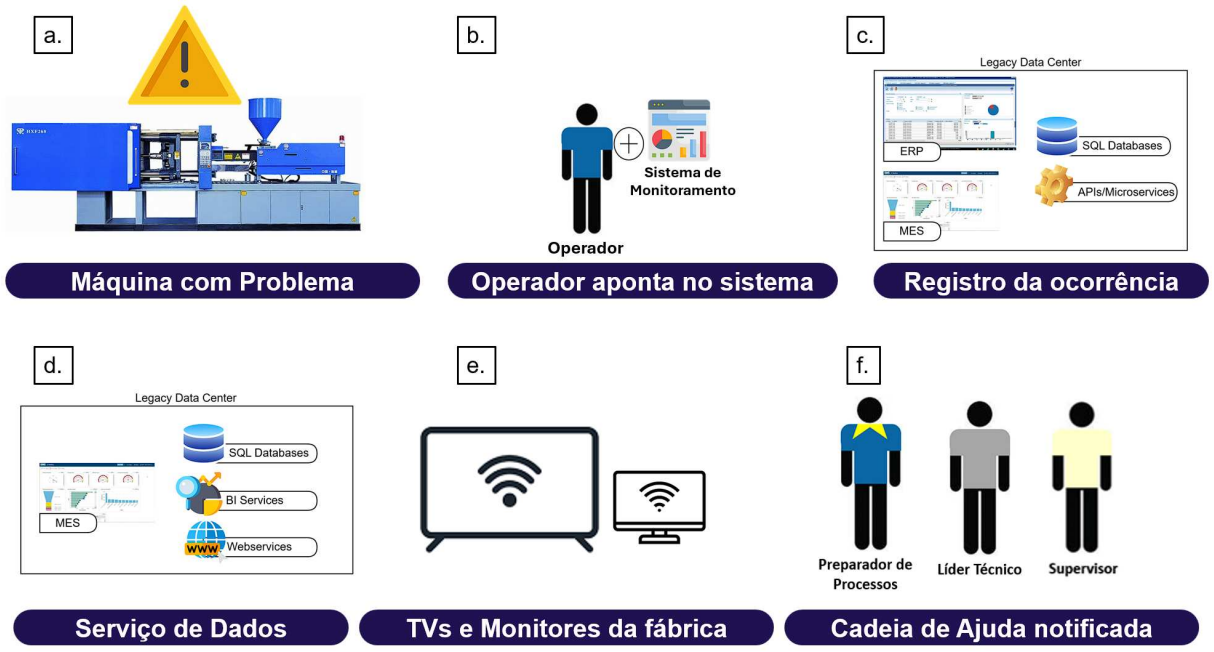
Esses dados são utilizados como base para a geração de indicadores de desempenho do setor produtivo. Ao início e término de cada turno, os líderes recebem boletins por e-mail com um resumo dessas informações, auxiliando no processo de tomada de decisão. No entanto, o processo atual de comunicação de falhas apresenta gargalos importantes.

Grande parte dos problemas nas máquinas é inicialmente identificada e registrada pelo operador. Contudo, esses profissionais nem sempre conseguem diagnosticar a origem da falha de forma precisa. Quando isso ocorre, é necessário acionar outros membros da equipe de apoio à produção composta por funções como Preparador de Processos, Manutenção, Líder Técnico e Supervisores. Esta equipe é chamada neste trabalho como Cadeia de Ajuda.

Esse fluxo de resolução segue um modelo sequencial e hierárquico: o operador inicia o chamado, que pode ser solucionado diretamente por um Preparador de Processos, ou escalado para outros profissionais em função da complexidade do problema como o Líder Técnico, por exemplo, que atua como responsável técnico de múltiplas máquinas.

A Figura 11 ilustra esse processo tradicional de comunicação indireta. O fluxo inicia com a ocorrência de um problema (a.), registrado pelo operador no sistema MES (b.). Essa informação é armazenada em um banco de dados corporativo (c.), sendo processada por sistemas de *Business Intelligence* / Inteligência de Negócios (BI) (d.), que consolidam os dados em relatórios. Posteriormente, os alertas são distribuídos por e-mail ou exibidos em monitores da fábrica (e.), chegando de forma passiva e tardia à Cadeia de Ajuda (f.). Como o fluxo depende de consultas periódicas a painéis e relatórios, não há garantia de que o problema será percebido imediatamente, já que os profissionais envolvidos possuem outras tarefas e nem sempre estão atentos às telas.

Figura 11 – Representação do acionamento indireto da Cadeia de Ajuda



Fonte: Próprio Autor

Essa abordagem compromete a agilidade na identificação e no tratamento de falhas, gerando atrasos que impactam diretamente a produtividade. Além disso, a centralização das informações em sistemas de BI dificulta a segmentação e o direcionamento eficiente dos alertas, tornando o processo pouco reativo e dependente da iniciativa humana para a leitura e resposta.

3.2.3 Proposta

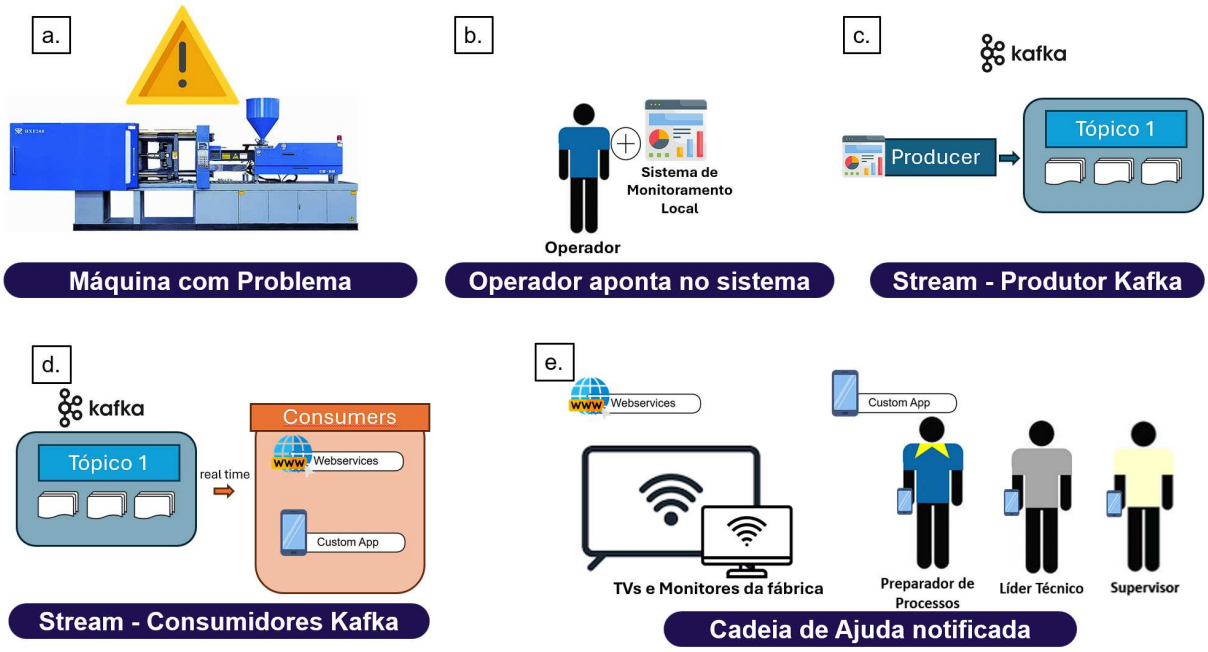
Como discutido na Seção 3.2.2, o modelo atual de comunicação com a Cadeia de Ajuda apresenta limitações quanto à velocidade e à precisão no repasse de informações críticas sobre o funcionamento das máquinas. Cada posto de trabalho fornece dados para um conjunto de sistemas corporativos de maneira centralizada, o que resulta em atrasos e perda de visibilidade operacional em tempo real.

De acordo com registros internos da empresa, apenas no período de um ano foram contabilizados 726.490 eventos de parada de máquinas industriais. Considerando 288 dias úteis, isso representa uma média de aproximadamente 2.522 acionamentos diários. Esses eventos estão armazenados em um banco de dados que também registra ciclos de produção, perdas de qualidade e outras informações que compõem o histórico de cada equipamento.

Dada a magnitude desses números, os problemas relatados pela fábrica — como latências de até 2 minutos entre o registro da parada no MES e sua exibição nos monitores gerenciais, além de apontamentos não registrados em banco devido a falhas de conexão de rede — tornam-se compreensíveis, pois refletem a sobrecarga imposta ao sistema. Além disso, a comunicação ocorre de forma indireta e passiva, sem um canal de notificação que direcione imediatamente o alerta ao profissional responsável. Diante desse cenário, propõe-se neste trabalho a adoção de uma arquitetura orientada a eventos, na qual cada máquina se torna capaz de enviar mensagens em tempo real diretamente à Cadeia de Ajuda, utilizando a plataforma Apache Kafka como barramento de mensageria.

A Figura 12 ilustra o funcionamento da arquitetura proposta. A sequência inicia-se com a identificação do problema na máquina (a.), que é registrada pelo operador no sistema local de monitoramento (b.). Esse evento, além de ser integrado aos sistemas corporativos existentes (MES e ERP), é também enviado para um tópico Kafka no formato JSON (c.), atuando como um produtor de mensagens. Os consumidores desse tópico (d.), como aplicativos móveis e *web services*, recebem o alerta imediatamente e o encaminham para os dispositivos da Cadeia de Ajuda, incluindo *smartphones*, *tablets* ou painéis específicos. Com isso, os profissionais responsáveis (e.) são notificados de forma direta, rápida e contextualizada.

Figura 12 – Representação do acionamento direto da Cadeia de Ajuda por mensageria



Fonte: Próprio Autor

A proposta inclui ainda a utilização de um aplicativo de comunicação corporativa, já adotado pela empresa, como canal oficial para o envio das notificações. As mensagens produzidas pelas máquinas serão consumidas por um serviço intermediário que realizará o direcionamento inteligente das informações para o profissional vinculado ao equipamento, com base em regras definidas pela equipe de planejamento e manutenção.

Essa abordagem visa eliminar os gargalos presentes no modelo centralizado, garantindo maior granularidade na entrega dos alertas, além de permitir a escalabilidade da solução para diferentes unidades fabris, mantendo o desacoplamento entre os sistemas produtores (máquinas) e consumidores (equipe técnica).

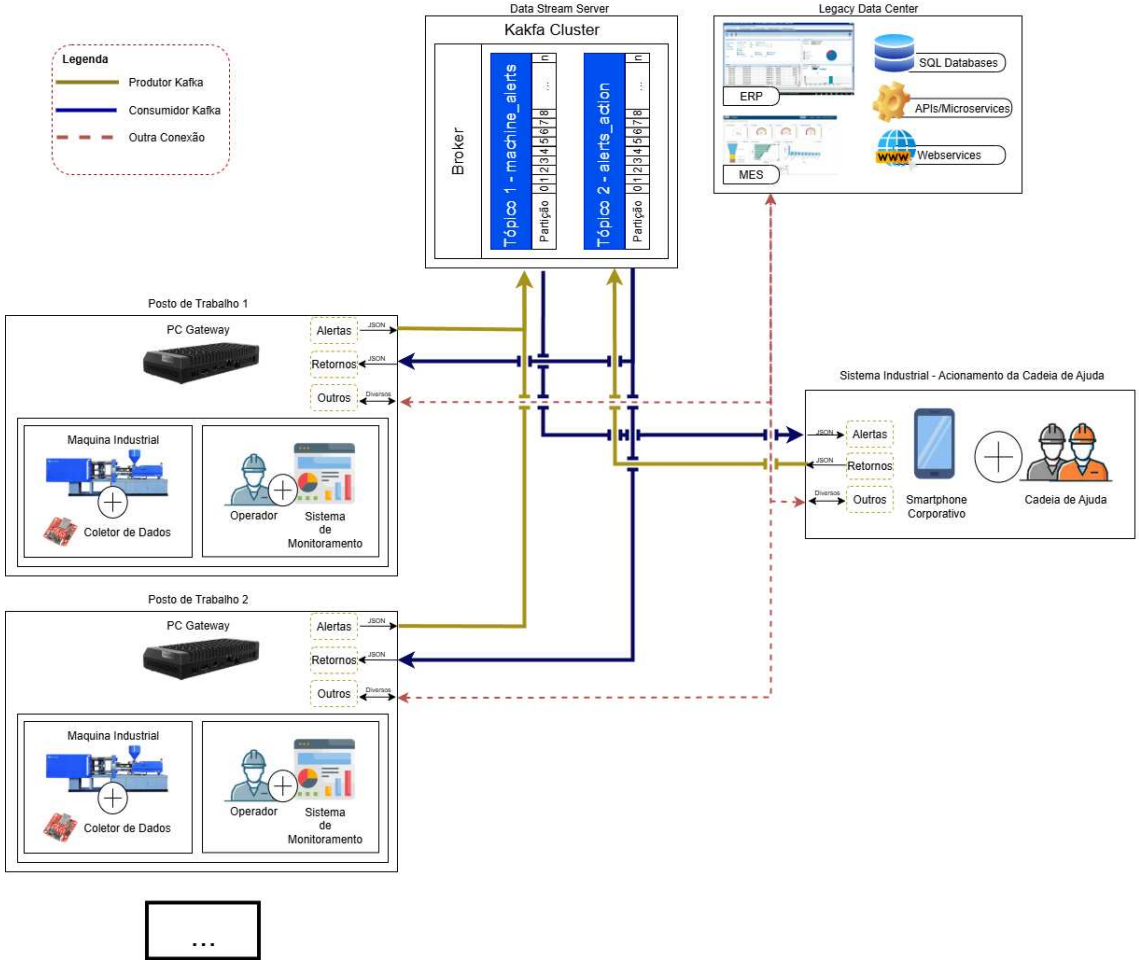
3.3 Materiais

Para validar a arquitetura proposta, foi desenvolvido um protótipo funcional e aplicado no ambiente industrial real. A solução envolveu a criação de um servidor Apache Kafka, responsável por gerenciar o fluxo de mensagens entre os equipamentos industriais e a Cadeia de Ajuda, permitindo o envio de alertas e respostas em tempo real.

3.3.1 Arquitetura de Mensageria Kafka

A arquitetura inicial adotada para o sistema de mensageria está representada na Figura 13. Nela, observa-se um único broker Kafka com dois tópicos: um destinado ao envio de alertas pelas máquinas (*machine_alerts*) e outro para o recebimento de respostas da Cadeia de Ajuda (*alerts_action*). Essa estrutura possibilita a comunicação bidirecional entre os equipamentos industriais e os profissionais responsáveis pelo suporte técnico.

Figura 13 – Arquitetura básica de mensageria para notificação da Cadeia de Ajuda

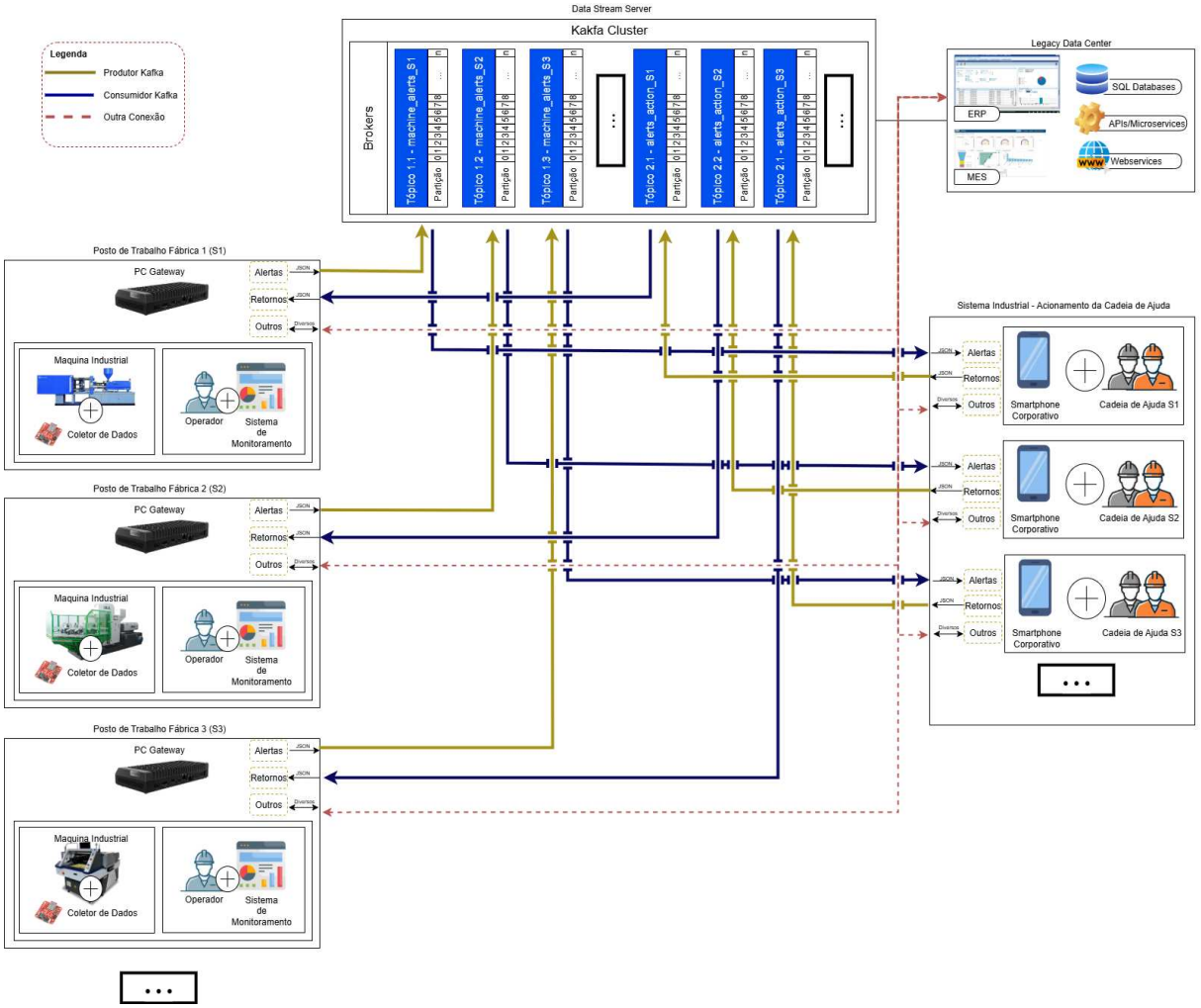


Fonte: Próprio Autor

No entanto, considerando a grande quantidade de máquinas e a diversidade de consumidores no ambiente fabril, essa configuração simples apresenta limitações. Com apenas dois tópicos, todas as mensagens seriam transmitidas em formato de *broadcast*, exigindo que cada aplicação consumidora filtre, localmente, apenas os dados relevantes ao seu contexto. Esse modelo aumenta o volume de processamento desnecessário e compromete a escalabilidade do sistema.

Além disso, como indicado anteriormente na Figura 9, um único polo industrial pode conter até oito unidades fabris distintas, cada uma com alto volume de dados gerados de forma contínua. Para lidar com esse cenário de forma mais eficiente, propõe-se uma arquitetura expandida, apresentada na Figura 14, que utiliza 16 tópicos Kafka, sendo dois por unidade fabril (um para alertas e outro para respostas).

Figura 14 – Arquitetura completa com divisão de tópicos por unidade fabril



Fonte: Próprio Autor

Essa segmentação permite que cada instância da Cadeia de Ajuda consuma apenas as mensagens relacionadas à sua respectiva unidade, reduzindo o tráfego irrelevante e otimizando o desempenho das aplicações. Além disso, essa organização modular viabiliza o reaproveitamento dos eventos por outros sistemas corporativos, como supervisórios, painéis de indicadores ou ferramentas de análise preditiva.

3.3.2 Descrição dos Tópicos Kafka

A arquitetura de mensageria proposta utiliza dois tipos de tópicos Kafka para viabilizar a comunicação entre os equipamentos industriais e a Cadeia de Ajuda. Esses tópicos são organizados em pares para cada unidade fabril, identificados por um `SX`, onde `SX` representa o código da unidade correspondente.

Cada equipamento é identificado por um *ID do Equipamento* (*equip_id*), estruturado conforme a Figura 15, composto por três elementos:

- **Unidade Fabril:** indica a planta de origem do equipamento (ex: `S3`),
- **Subdivisão:** representa o setor ou célula da unidade (ex: `A`),
- **Número do Equipamento:** sequência numérica única dentro da subdivisão (ex: `153`).

Assim, o identificador `S3A153` representa o equipamento nº 153 da subdivisão A da unidade fabril 3. Esse código é utilizado para determinar os tópicos Kafka nos quais o equipamento publica e consome mensagens: `machine_alerts_S3` para o envio de alertas, e `alerts_action_S3` para o recebimento de respostas. Para fins de abstração, pode-se denominar esses tópicos como Tópico 1.X (alertas) e Tópico 2.X (respostas), onde X representa a unidade.

Figura 15 – Estrutura do Identificador de Equipamento (*equip_id*)

Fonte: Próprio Autor

3.3.2.1 Tópicos: `machine_alerts`

Esse tópico é responsável pela publicação de mensagens de alerta originadas dos equipamentos industriais. Cada mensagem é estruturada em formato JSON e contém informações como código e motivo do alerta, identificação do equipamento, data de início, tipo, prioridade, tempo de validade do alerta, histórico de falhas, entre outros.

O corpo da mensagem segue o seguinte modelo:

```
{
    "codigo_alerta": string,
    "motivo_alerta": string,
    "equip_id": string,
    "data_inicio": timestamp,
    "tipo": number,
    "prioridade": number,
    "tempo_intervalo": number,
    "helpchain": [{
        "nome": string,
        "cracha": number,
    }],
    "falhas": [{
        "familia": number,
        "descricao": string
    }],
    "retorno": boolean,
    "url_resposta": string
}
```

A Tabela 1 apresenta a descrição detalhada de cada campo.

Tabela 1 – Descrição dos campos do JSON para o tópico `machine_alerts`

Campo	**Tipo de Dado**	**Descrição**
`codigo_alerta`	string	Código único identificando o alerta gerado pela máquina.
`motivo_alerta`	string	Título descritivo da mensagem de alerta.
`equip_id`	string	Identificador único da máquina que gerou o alerta. Determina qual tópico será enviado como base das suas iniciais.
`prioridade`	number	Categoria do alerta em relação à urgência, sendo 1 o menor e 3 o maior.
`data_inicio`	timestamp	Data e hora de início do alerta.
`tipo`	number	Código numérico que classifica o tipo de alerta (ex.: 1 para falha, 2 para manutenção, 3 para suporte).
`helpchain`	array	Lista de usuários da cadeia de ajuda que receberão o alerta, contendo o nome, número do crachá e prioridade de cada um.
`helpchain.nome`	string	Nome do profissional que receberá o alerta.
`helpchain.cracha`	number	Número do crachá do profissional que receberá o alerta.
`falhas`	array	Lista de falhas atuais ou anteriores que ocasionaram o alerta, contendo a quantidade de ocorrências e descrição de cada falha.
`falhas.descricao`	string	Descrição da falha que ocasionou o alerta.
`falhas.quantidade`	number	Quantidade de ocorrências do alerta.
`retorno`	boolean	Indica se o alerta necessita de uma resposta da cadeia de ajuda para a máquina.
`tempo_intervalo`	number	Tempo, em minutos, até que o alerta expire ou exija uma nova ação.
`url_resposta`	string	URL da API da máquina que gerou o alerta, utilizado para facilmente redirecionar as possíveis respostas do alerta.

Fonte: Próprio Autor.

3.3.2.2 Tópicos: `alerts_action`

Este tópico é utilizado para o envio de respostas por parte da Cadeia de Ajuda às máquinas. As mensagens também seguem o formato JSON e contêm informações como o usuário responsável pela ação, o *equip_id*, o código do alerta ao qual se responde, data da ação, comentário e o tipo de ação executada.

Formato da mensagem:

```
{
    "usuario" {
        "nome": string,
        "cracha": number,
    }
    "equip_id": string,
    "data": timestamp,
    "comentario": string,
    "acao": string
}
```

A Tabela 2 detalha os campos correspondentes.

Tabela 2 – Descrição dos campos do JSON para o tópico `alerts_action`

Campo	Tipo de Dado	Descrição
`usuario`	object	Objeto contendo informações sobre o operador que enviou a resposta.
`usuario.nome`	string	Nome do profissional que respondeu o alerta.
`usuario.cracha`	number	Número do crachá do profissional.
`equip_id`	string	Identificador único da máquina que recebeu a ação do operador.
`codigo_alerta`	string	Código único identificando o alerta que receberá um retorno do profissional.
`data`	timestamp	Data e hora em que a ação foi realizada.
`comentario`	string	Comentário do operador sobre a situação ou a ação realizada.
`acao`	string	Descrição da ação tomada pelo operador (ex.: reiniciar, parar, etc.).

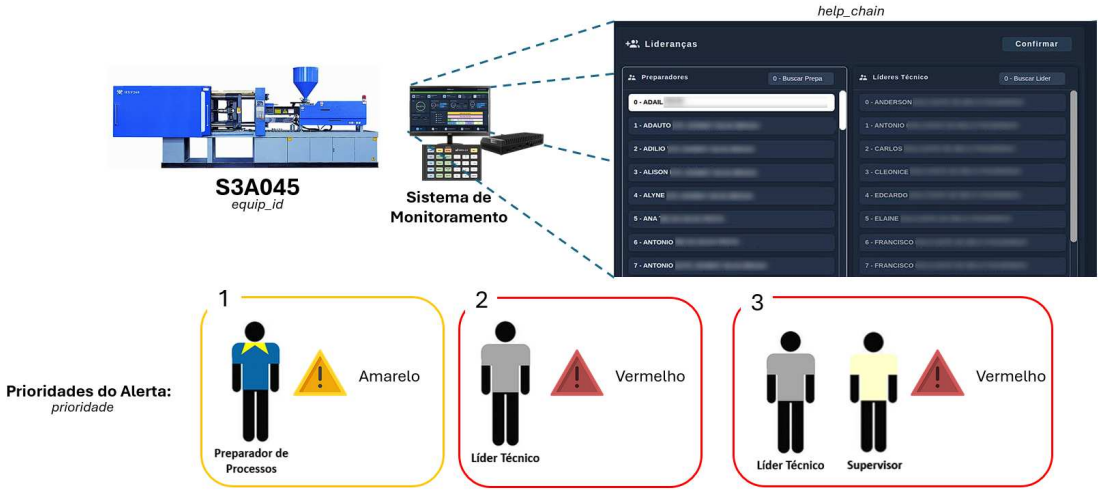
Fonte: Própio Autor.

3.3.3 Funcionamento do Sistema de Alertas

O sistema de alertas foi integrado ao sistema de monitoramento existente no parque fabril, desenvolvido internamente pela empresa e baseado no modelo MES. Esse sistema já contém informações operacionais como status das máquinas, dados de produtividade e identificação dos operadores, o que facilitou a implementação das adaptações necessárias para a arquitetura de mensageria.

Inicialmente, foi criada uma estrutura para o cadastro da Cadeia de Ajuda, permitindo a associação de cada máquina a um grupo de profissionais responsáveis por seu suporte técnico. Essa associação é feita por meio do *número do crachá* do colaborador e pode ser configurada diretamente pelo operador no sistema de monitoramento. A Figura 16 ilustra essa associação, demonstrando como os membros são organizados por equipamento.

Figura 16 – Associação de membros da cadeia de ajuda e prioridades



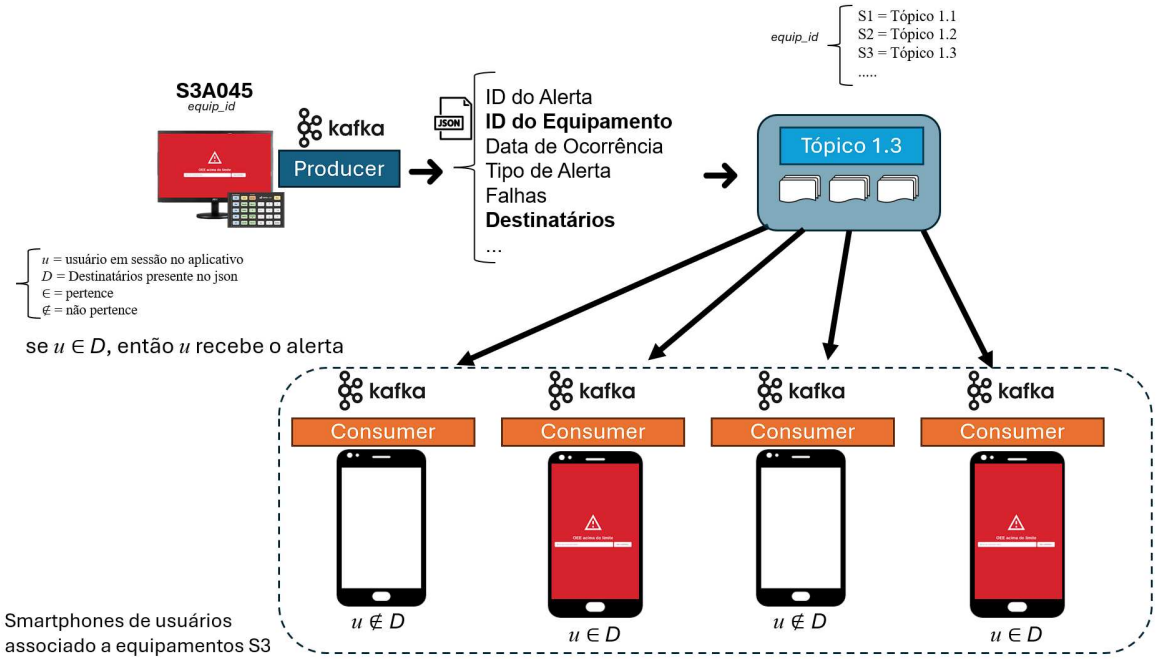
Fonte: Próprio Autor

Além disso, foi implementada a parametrização de eventos que devem gerar alertas, com base em métricas como eficiência da máquina, tempo elevado de setup ou percentual de perdas. Cada evento pode ser configurado com até três níveis de prioridade, os quais determinam o grupo de profissionais a ser acionado e a cor associada ao alerta na interface do sistema. A propriedade `prioridade`, contida na mensagem JSON (ver Tabela 1), representa esse nível de criticidade.

Sempre que um evento parametrizado for detectado, o sistema envia um alerta correspondente. Por exemplo, se o número de perdas de qualidade ultrapassar 10% da produção planejada, será disparado um alerta de prioridade 3 para o líder técnico e o supervisor vinculados àquela máquina. O JSON gerado para esse evento está disponível no Apêndice A.

A Figura 17 apresenta o fluxo de envio do alerta. A máquina atua como produtora Kafka, publicando a mensagem no tópico correspondente (por exemplo, `machine_alerts_S3`). Com base na propriedade `prioridade`, apenas os profissionais listados na propriedade `helpchain` da mensagem receberão a notificação. Cada aplicativo consumidor instalado em *smartphones* consome apenas mensagens da unidade à qual pertence, o que reduz o volume de processamento e evita sobrecarga.

Figura 17 – Fluxo de envio de alerta para a cadeia de ajuda

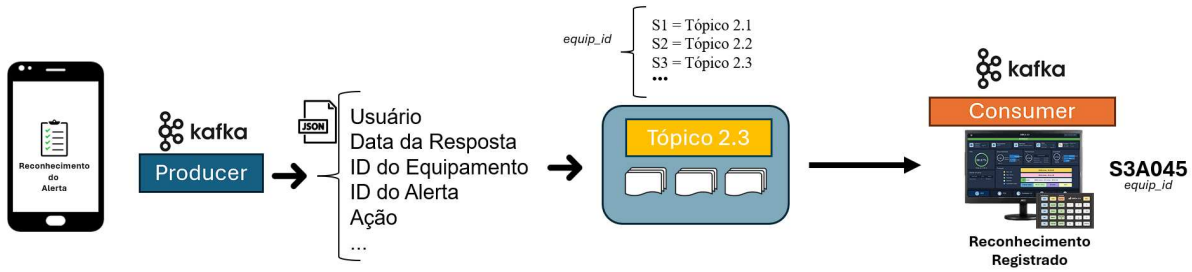


Fonte: Próprio Autor

Alguns alertas exigem resposta obrigatória por parte do destinatário, o que é indicado pela propriedade `retorno` no JSON. Quando esta propriedade estiver com valor verdadeiro (*true*), inicia-se a contagem de tempo definida em `tempo_intervalo`. Caso o tempo expire sem que haja resposta, um novo alerta é emitido automaticamente.

A Figura 18 ilustra o fluxo de retorno da Cadeia de Ajuda para a máquina. Nesse caso, o aplicativo atua como produtor Kafka, enviando a resposta ao tópico `alerts_action_SX`, enquanto a máquina torna-se consumidora dessa mensagem. A associação entre a resposta e o alerta original é realizada por meio da propriedade `codigo_alerta`.

Figura 18 – Fluxo de envio do reconhecimento do alerta para a máquina



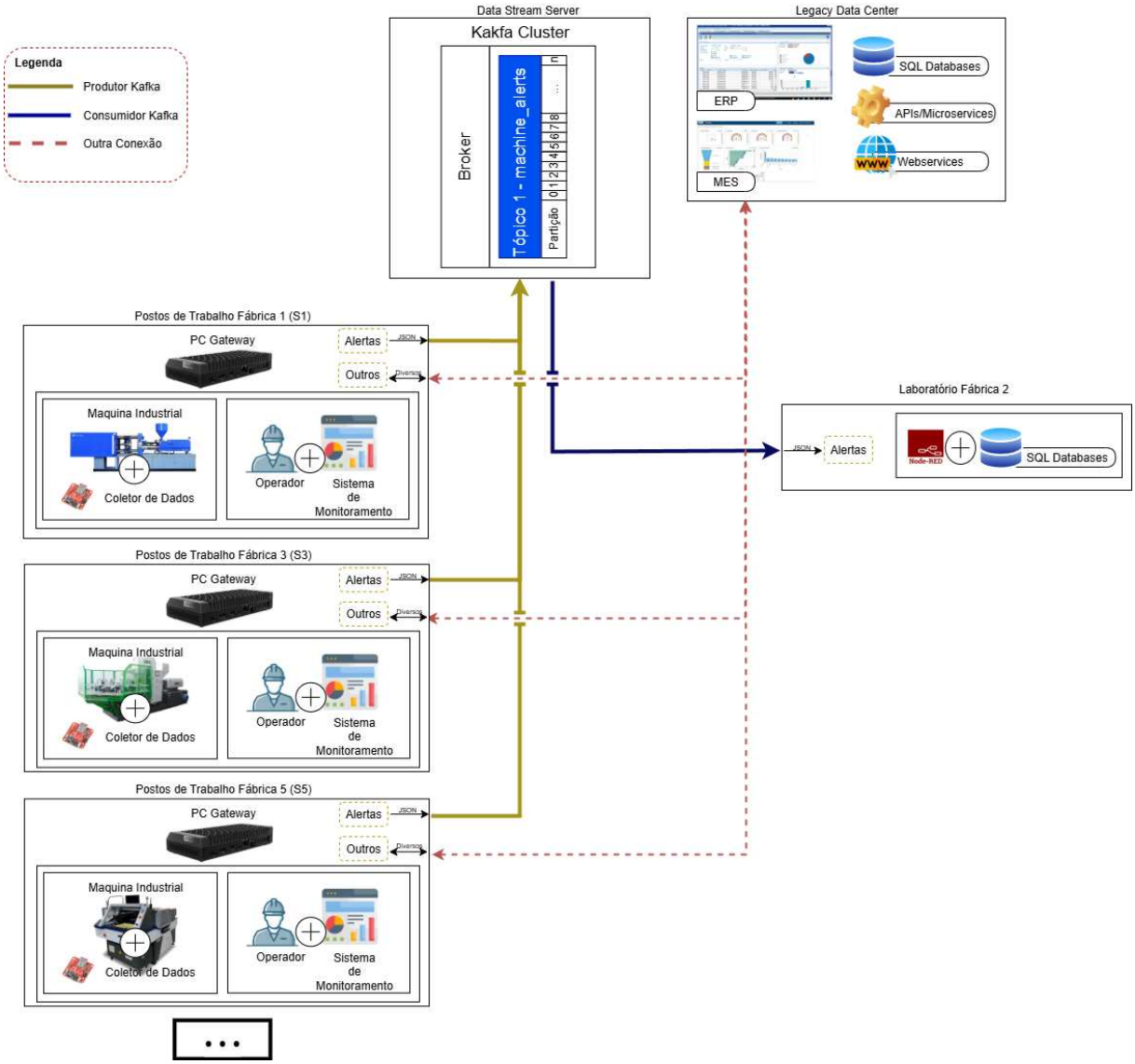
Fonte: Próprio Autor

3.3.4 Prototipação

Com o objetivo de validar a arquitetura de mensageria proposta, foi desenvolvido um protótipo funcional com foco em avaliar a capacidade do Apache Kafka de garantir a entrega de mensagens em fila, sem perdas e com baixa latência. A utilização de tecnologias *open-source* e a compatibilidade com a infraestrutura existente tornaram o protótipo uma solução de baixo custo e alta escalabilidade para testes preliminares.

Durante essa fase, optou-se por não desenvolver o aplicativo móvel final. Em vez disso, priorizou-se a implementação da infraestrutura essencial, conforme ilustrado na Figura 19, simulando a comunicação entre equipamentos e a Cadeia de Ajuda por meio de publicações diretas no Kafka.

Figura 19 – Representação da arquitetura de mensageria para o protótipo



Fonte: Próprio Autor

Para tornar o experimento mais desafiador, o protótipo foi configurado com apenas um único tópico Kafka para o envio de alertas, ignorando temporariamente a segmentação por unidades fabris prevista na arquitetura final (ver Seção 3.3.1). Essa escolha teve como objetivo avaliar o desempenho do sistema em um cenário crítico, com centenas de produtores concorrendo por um mesmo canal de comunicação.

Embora essa simplificação aumente a carga sobre o *broker* e os consumidores, ela permitiu uma análise robusta da capacidade do Kafka de manter confiabilidade e desempenho mesmo sob forte concorrência. Ressalta-se, contudo, que a segmentação em múltiplos tópicos continua recomendada para aplicações em produção, tanto por questões de escalabilidade quanto de organização lógica dos dados. O protótipo serviu, assim, como um ambiente controlado para testar os principais elementos da solução proposta, permitindo a coleta de métricas quantitativas e qualitativas que sustentam a avaliação descrita nas seções subsequentes.

3.3.4.1 Servidor e Consumidores

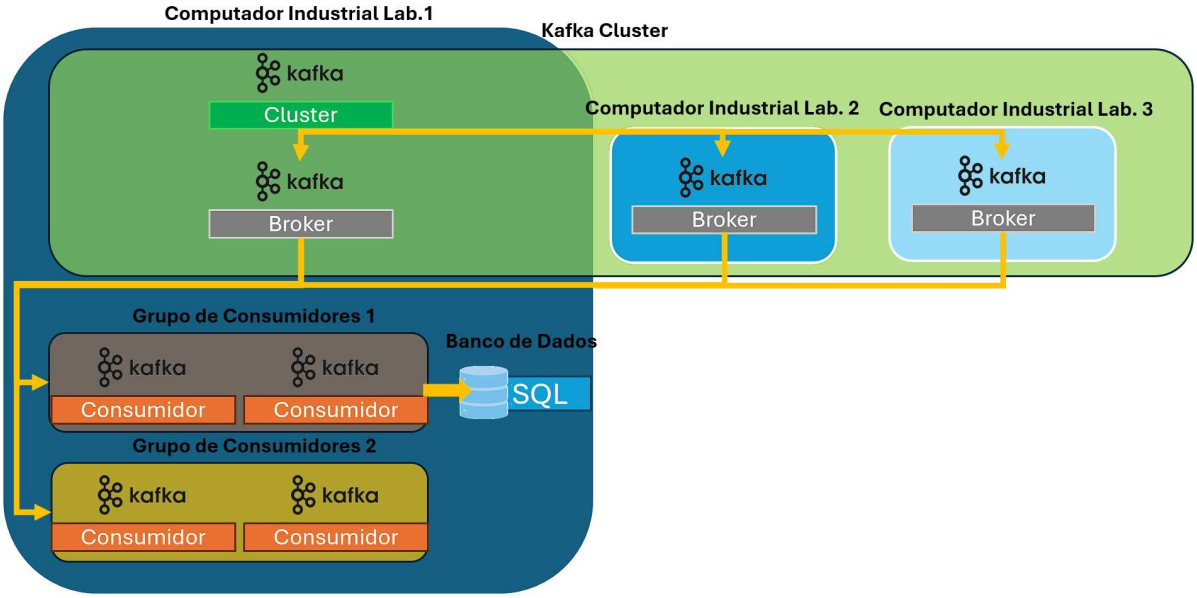
O *cluster* Kafka do protótipo foi implementado com três *brokers*, executados em computadores do modelo Lenovo ThinkCentre M75n IoT, cada um equipado com processador AMD Athlon Silver 3050e, 4 GB de memória RAM e sistema operacional Debian 12. O primeiro equipamento, denominado *Computador Industrial Laboratório 1*, atuou como *broker* principal, enquanto os demais (*Laboratórios 2 e 3*), foram configurados como nós adicionais do *cluster*.

Esses dispositivos foram escolhidos por apresentarem desempenho moderado, baixo consumo energético e, principalmente, por serem os mesmos utilizados na operação real do sistema MES da fábrica. Assim, o ambiente de testes manteve alta representatividade do cenário industrial, sem necessidade de novos investimentos em hardware.

A instalação foi realizada na Fábrica 2, que no período dos testes não possuía máquinas em operação, sendo utilizada por setores administrativos, desenvolvimento de sistemas e engenharia de projetos. Essa unidade está conectada às demais fábricas por meio de rede local, conforme descrito na Seção 3.2.1.

A aplicação de consumo foi desenvolvida com o *Node-RED*, ferramenta de programação visual baseada em *JavaScript* amplamente utilizada em projetos de IoT. A comunicação com o Kafka foi viabilizada pelo *plugin* `node-red-kafka-managerr`, que permite a publicação e o consumo de mensagens diretamente nos tópicos do *cluster*.

Figura 20 – Diagrama da composição do servidor do protótipo



Fonte: Próprio Autor

A Figura 20 apresenta o diagrama de composição do servidor no protótipo, incluindo os três *brokers* Kafka e os grupos de consumidores. O objetivo da aplicação é simular o comportamento dos dispositivos móveis da Cadeia de Ajuda, coletando as mensagens dos tópicos e armazenando-as em um banco de dados relacional para posterior análise. Para isso, foram configurados dois grupos de consumidores, cada um com duas instâncias:

- O Grupo de Consumidores 1 foi responsável pela gravação dos dados no banco SQL;
- O Grupo de Consumidores 2 atuou como monitor, contabilizando as mensagens recebidas com o intuito de verificar a integridade da entrega.

Essa configuração teve como base os conceitos propostos por Wu *et al.* (2019a), que avaliaram a confiabilidade da entrega de dados em sistemas Kafka. O *cluster* Kafka foi configurado com semântica de entrega exatamente-uma (*exactly-once*), habilitando recursos como produtor idempotente e transações, conforme suportado nas versões mais recentes do Kafka (WU *et al.*, 2019b). Esse mecanismo permite que cada mensagem seja persistida exatamente uma vez, mesmo em casos de falha, evitando duplicações durante tentativas de reenvio.

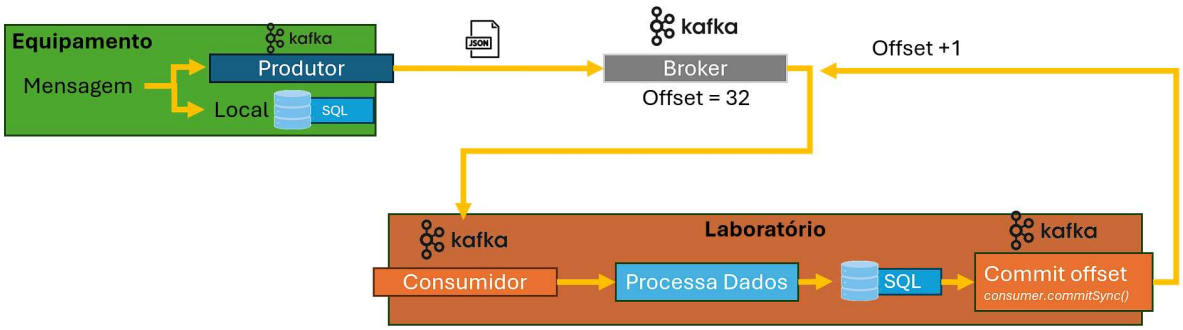
Para garantir a consistência dos dados, apenas um dos grupos de consumidores foi responsável por realizar a gravação das mensagens no banco de dados, servindo de base para as análises de desempenho que serão apresentadas nas seções seguintes. O segundo grupo foi utilizado exclusivamente para monitoramento e comparação da quantidade de mensagens recebidas, com o objetivo de validar a integridade da entrega. A contagem de mensagens foi feita por meio de um contador simples de mensagens consumidas.

Para garantir a consistência dos dados, o *cluster* foi configurado com semântica de entrega "exatamente uma vez"(*exactly-once*), utilizando recursos como produtor idempotente e transações. Nos consumidores, o *auto-commit* foi desativado e o *commit manual* (`commitSync()`) foi adotado, assegurando que o *offset* só fosse atualizado após a persistência bem-sucedida da mensagem no banco.

O parâmetro `max.poll.records` foi ajustado para limitar a quantidade de mensagens processadas por vez, evitando sobrecarga de memória. A Figura 21 ilustra esse fluxo, no qual cada mensagem é processada individualmente, armazenada no banco e, somente após isso, tem seu *offset* confirmado. Essa abordagem segue as boas práticas descritas por Wu *et al.* (2019b) para sistemas com foco em confiabilidade.

Esse controle manual dos *offsets* é particularmente importante em aplicações industriais móveis, onde a possibilidade de desconexões de rede é elevada. Ao garantir a confirmação explícita da entrega, a arquitetura assegura a consistência do sistema mesmo em cenários de falha ou reinício dos consumidores.

Figura 21 – Diagrama de processamento de mensagens com *commit* síncrono de *offset*



Fonte: Próprio Autor

3.3.4.2 Produtores

Para implementar os produtores da arquitetura, foi realizada uma adaptação no sistema de monitoramento proprietário utilizado pela fábrica. Esse sistema, amplamente customizado, foi modificado para incorporar as telas e funcionalidades descritas na Seção 3.3.3, possibilitando a geração automática de mensagens no formato JSON, conforme a estrutura apresentada na Seção 3.3.2.

Cada equipamento industrial atualizado para participar do experimento passou a publicar alertas diretamente no Kafka, atuando como produtor de eventos. Além do envio

da mensagem para o *broker*, o alerta também era registrado localmente em um banco de dados embarcado, permitindo posterior validação da entrega e detecção de possíveis falhas ou duplicações.

A Tabela 3 apresenta a quantidade de equipamentos atualizados por unidade fabril, totalizando 326 dispositivos distribuídos entre cinco plantas industriais. Todos esses equipamentos foram configurados para enviar mensagens para um único tópico Kafka, como descrito na Seção 3.3.4. Essa escolha intencional criou uma situação de carga elevada sobre o sistema, com o objetivo de testar o desempenho do Kafka em um cenário crítico.

Tabela 3 – Quantidade de equipamentos atualizados por fábrica

Fábrica	**Quantidade de Equipamentos**
S1	28
S3	179
S5	2
S6	42
S7	75
Total Geral	**326**

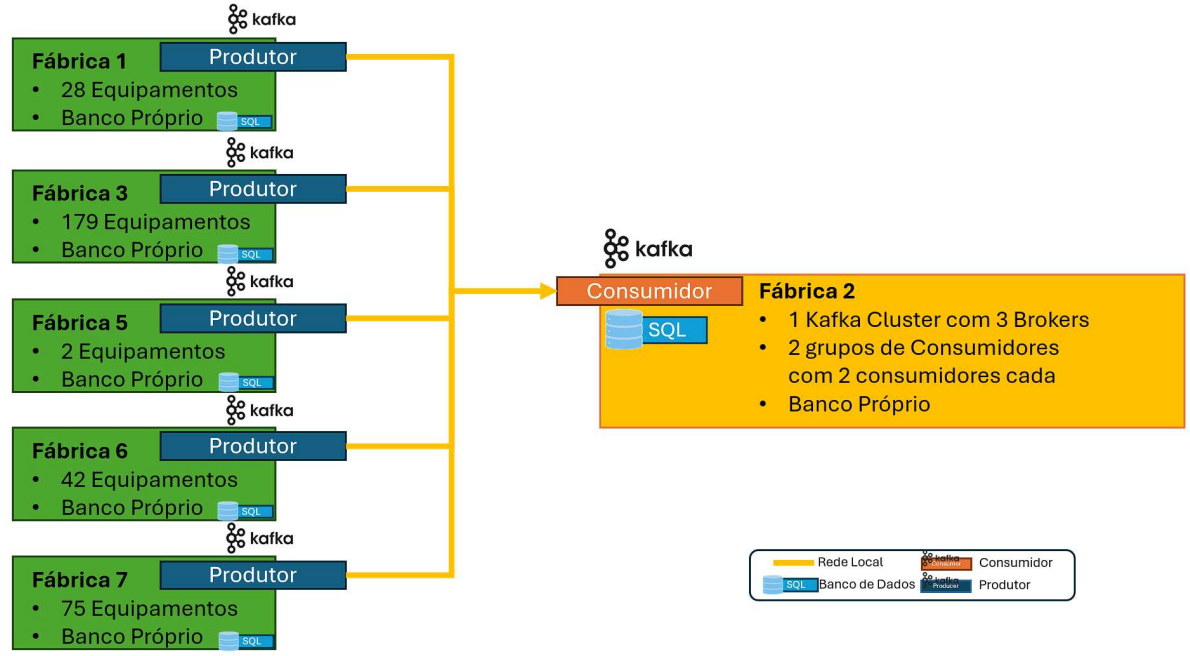
Fonte: Próprio Autor.

Essa duplicação de envio, para o Kafka e armazenamento local, permitiu a realização de uma auditoria cruzada dos dados, garantindo a verificação da integridade da entrega durante a fase de análise. A estratégia também simulou o comportamento esperado em ambiente real, onde os alertas são enviados para a mensageria e registrados em sistemas internos simultaneamente.

A Figura 22 resume o fluxo dos produtores, mostrando à esquerda os equipamentos gerando mensagens que são enviadas ao Kafka e gravadas localmente, enquanto à direita os consumidores processam e registram os dados recebidos. Esse fluxo reproduz o comportamento típico de uma solução IIoT integrada a um sistema industrial, com múltiplos produtores concorrendo por um canal de comunicação compartilhado.

Com essa configuração, foi possível verificar o comportamento do sistema frente a um volume representativo de mensagens, assegurando a validade dos testes de confiabilidade e latência apresentados nas seções seguintes.

Figura 22 – Diagrama do fluxo dos produtores



Fonte: Próprio Autor

3.3.5 Medidas de Desempenho

A avaliação da arquitetura de mensageria proposta foi orientada por três métricas principais de desempenho:

- Mensagens perdidas
- Mensagens duplicadas
- Latência entre produção e consumo das mensagens

As duas primeiras métricas foram tratadas diretamente durante a implementação dos produtores e consumidores (3.3.4.2 e 3.3.4.1) com o uso da semântica de entrega "exatamente uma vez"(*exactly-once*) e da confirmação manual de *offsets* via `commitSync()`.

Essa configuração garante que cada mensagem seja processada apenas uma vez, mesmo em cenários de falha ou reinício de componentes, minimizando o risco de perdas ou duplicações. A validação desses aspectos foi possível por meio da comparação entre os dados armazenados localmente nos equipamentos e os dados consumidos e registrados no banco de dados central.

Já a análise de latência, que é o tempo decorrido entre o envio da mensagem pelo produtor e o seu consumo, é detalhada na próxima subseção (3.3.5.1). Essa métrica é especialmente relevante no contexto desta aplicação, uma vez que o sistema proposto visa alertar em tempo quase real a Cadeia de Ajuda, permitindo respostas rápidas a falhas e eventos operacionais.

3.3.5.1 Latência

A métrica de latência adotada neste trabalho foi definida como o intervalo de tempo entre o envio da mensagem de alerta pelo produtor e o seu recebimento pelo consumidor Kafka. Para garantir a precisão dessa medição, o *timestamp* de envio (`data_inicio`, ver Tabela 1) foi gerado no momento da publicação da mensagem, com base no relógio do próprio computador produtor, sincronizado via *Network Time Protocol* (NTP) com o servidor interno da fábrica. Este *timestamp* foi registrado no formato *Universal Coordinated Time* (UTC) conforme o padrão *International Organization for Standardization* (ISO) 8601.

Esse mesmo servidor NTP também sincroniza os relógios dos sistemas fabris, incluindo os computadores que executam o MES, os relógios de ponto e os dispositivos utilizados no laboratório. Dessa forma, garante-se a consistência temporal entre os registros gerados pelos produtores e pelos consumidores.

A latência foi calculada de forma similar à abordagem descrita por Braunisch *et al.* (2022), conforme a Equação 3.1:

$$\text{Latência} = \Delta t = t_{\text{dataConsumidor}} - t_{\text{dataProdutor}}, \tag{3.1}$$

Em que:

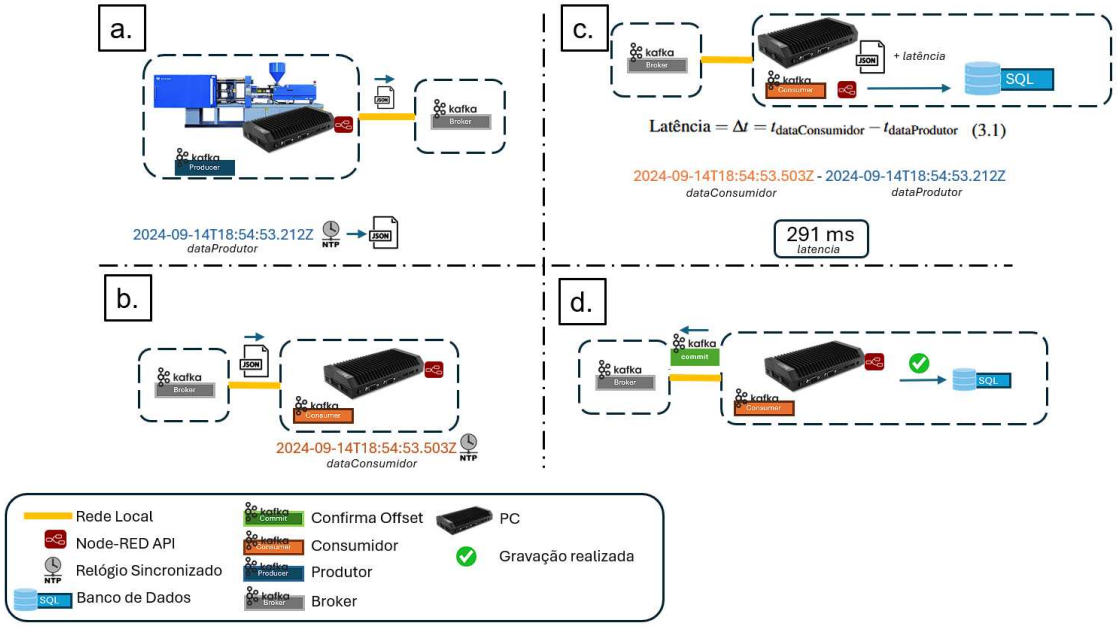
- $t_{\text{dataProdutor}}$ corresponde ao *timestamp* contido na propriedade `data_inicio` do JSON;
- $t_{\text{dataConsumidor}}$ é capturado pelo consumidor no momento da recepção da mensagem.

Ambos os valores foram convertidos para milissegundos e utilizados para o cálculo direto da latência entre produção e consumo.

A Figura 23 ilustra esse processo de medição, destacando as etapas (a) geração do *timestamp* pelo produtor, (b) envio da mensagem ao *broker* Kafka, (c) recepção e captura do *timestamp* pelo consumidor e (d) confirmação da transação com `commitSync()`.

O experimento foi conduzido por aproximadamente oito semanas, entre meados de fevereiro e abril de 2025, período no qual foram registrados 206.252 alertas. Esses dados foram extraídos a partir dos consumidores do protótipo e associados aos respectivos registros armazenados localmente nos produtores, permitindo a análise precisa da latência.

Figura 23 – Cálculo de Latência



Fonte: Próprio Autor

A Tabela 4 apresenta a distribuição das mensagens por unidade fabril. Do total de aproximadamente 800 máquinas monitoradas pela empresa, 326 equipamentos participaram da simulação, representando cerca de 40,75% da carga total do parque industrial. Todas as mensagens foram publicadas em um único tópico Kafka, como descrito na Seção 3.3.4, o que reforça a robustez da solução mesmo sob alta concorrência.

Tabela 4 – Quantidade de mensagens registradas por fábrica durante o experimento

Fábrica	**Quantidade de Mensagens**
S1	9.752
S3	137.418
S5	242
S6	13.916
S7	44.924
Total Geral	**206.252**

Fonte: Próprio Autor.

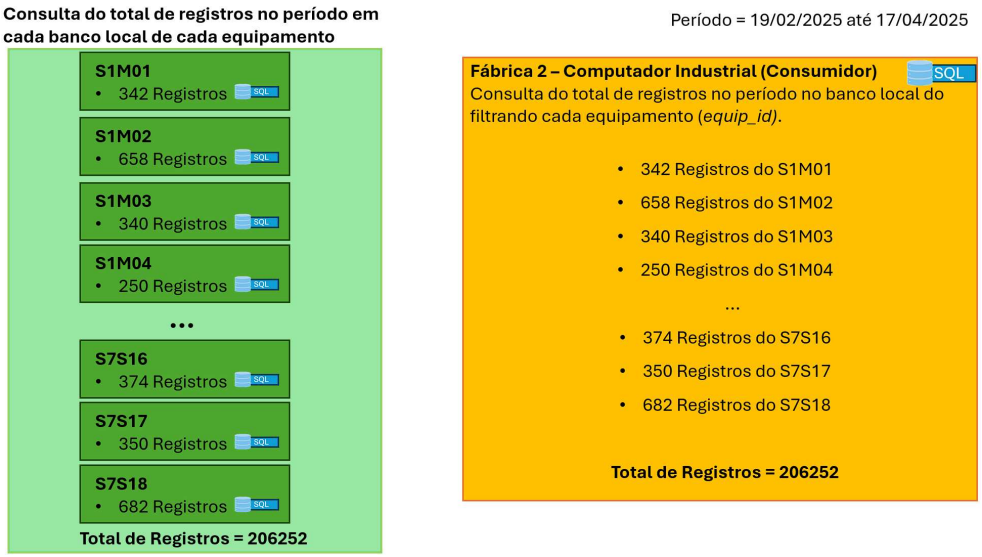
4 RESULTADOS E DISCUSSÃO

A avaliação da arquitetura proposta envolveu a análise das três métricas estabelecidas na Seção 3.3.5: (i) ausência de mensagens duplicadas, (ii) ausência de mensagens perdidas e (iii) latência de comunicação entre produção e consumo de mensagens.

Para as duas primeiras métricas, o experimento foi conduzido conforme os cenários apresentados anteriormente na Seção 3.3.4, com o uso da semântica *exactly-once* e a confirmação de leitura via método `commitSync()` no consumidor Kafka. A validação empírica dessas métricas foi realizada por meio da comparação direta entre os dados registrados localmente em cada equipamento (produtores) e os dados recebidos e armazenados no banco de dados do consumidor central no laboratório.

Todos os equipamentos utilizados no experimento mantinham uma cópia local de seus próprios dados de alerta, os quais eram registrados de forma automática no banco de dados embarcado. Ao final do período de testes, esses bancos locais foram sincronizados e extraídos para facilitar a análise e permitir a contagem total de mensagens produzidas por equipamento. A Figura 24 apresenta o resultado dessa verificação cruzada, evidenciando que os dados recebidos no laboratório (consumidor) coincidem integralmente com os dados produzidos localmente pelos equipamentos.

Figura 24 – Verificação da integridade dos dados: comparação entre registros locais dos produtores e do consumidor



Fonte: Próprio Autor

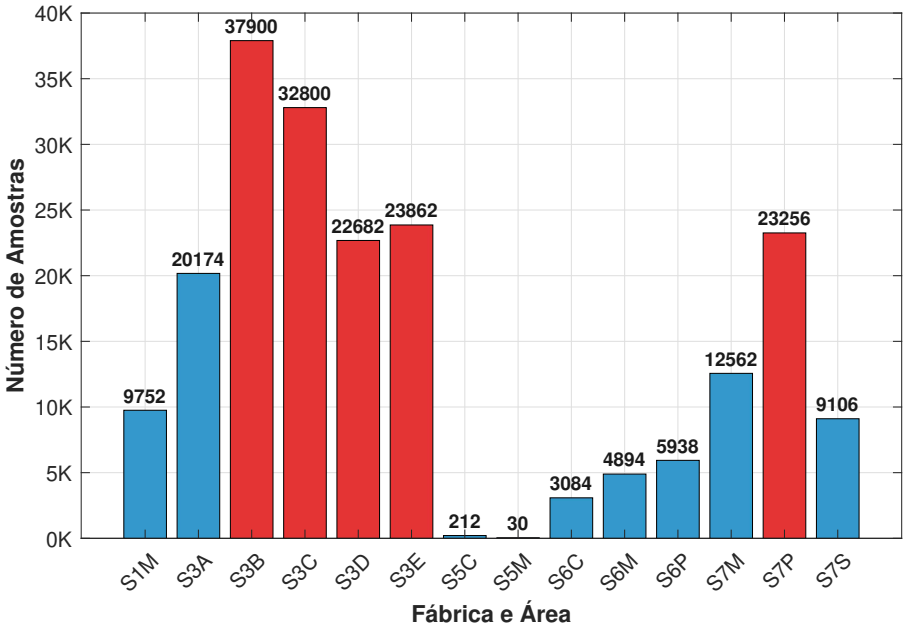
Dessa forma, não foram identificadas mensagens duplicadas ou perdidas durante toda a execução do experimento, corroborando a confiabilidade do canal de comunicação estabelecido

via Kafka mesmo em situação crítica com **40,75%** da carga total do parque industrial operando simultaneamente em um único tópico.

A latência de comunicação foi calculada conforme descrito na Seção 3.3.5, a partir da diferença entre o *timestamp* de produção e o *timestamp* de consumo de cada mensagem. Como cada mensagem processada possui sua respectiva latência associada, foram gerados exatamente **206.252 valores de latência** correspondentes ao total de mensagens transmitidas durante o experimento. Os resultados foram analisados por meio de cinco visualizações complementares, que são discutidas em detalhes a seguir.

A Figura 25 apresenta a distribuição das amostras de latência por unidade fabril e setor. Este gráfico de barras agrupadas exibe a quantidade de amostras coletadas em diversas instalações, organizadas por códigos de setor (por exemplo, S3, S5, S6). As barras azuis e vermelhas representam diferentes classificações ou categorias (por exemplo, setores normais versus críticos). Os setores S3B e S3C registraram os maiores volumes de mensagens, com 37.900 e 32.800 amostras, respectivamente, seguidos pelos setores S3D e S3E. Em contraste, setores como S5C e S5M coletaram apenas 212 e 30 amostras, respectivamente, o que sugere menor atividade ou limitação na captura de dados nessas áreas. Essa visualização fornece uma visão sobre a distribuição de carga de trabalho e auxilia na identificação de setores críticos para o monitoramento e otimização do desempenho do sistema.

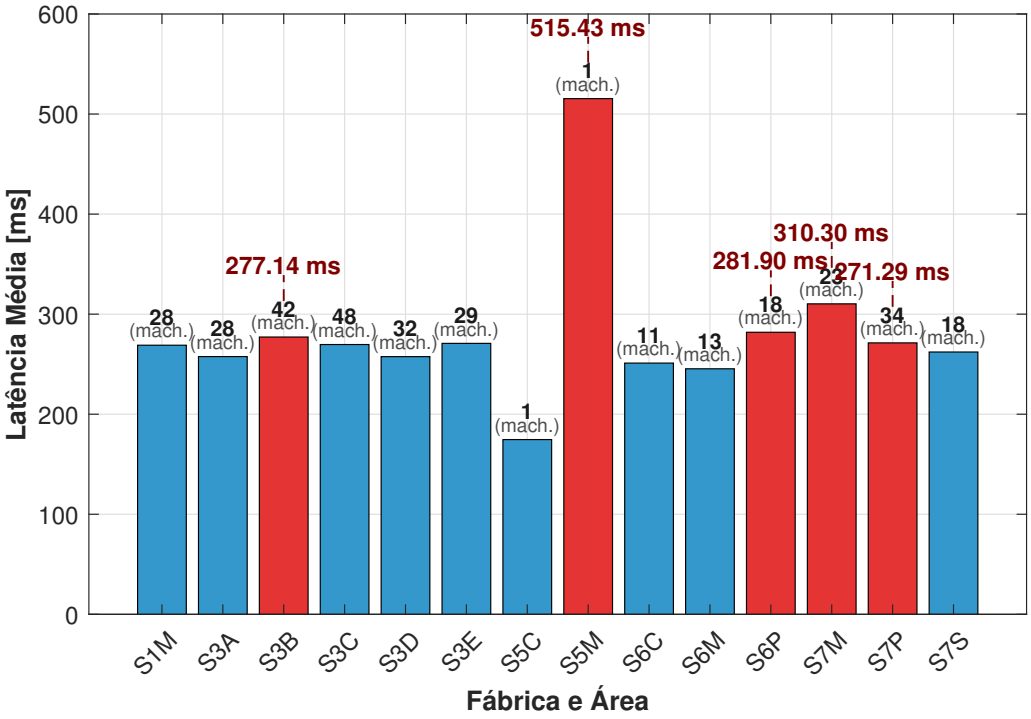
Figura 25 – Distribuição de Amostras por Unidade Fabril e Setor.



Fonte: Próprio Autor

A Figura 26 apresenta a latência média por unidade fabril e setor, juntamente com o número de máquinas monitoradas. A maioria dos setores apresentou latências médias entre 250ms e 310ms. No entanto, o setor S5M apesar de contar com apenas uma máquina monitorada, registrou a maior latência média com 515,43ms, indicando um possível problema localizado de desempenho associado a conexão do equipamento específico. Outros setores com latências médias elevadas incluem S6P e S7M, ambos com valores superiores a 280ms. Setores com maior número de máquinas, como S3B (42 máquinas), S3C (48 máquinas) e S7P (34 máquinas), tendem a apresentar latências médias mais baixas e estáveis, permanecendo próximas ou abaixo de 280ms. Isso sugere que, em ambientes com maior densidade de equipamentos, o sistema mantém um desempenho consistente, possivelmente devido à distribuição de carga e a ciclos de comunicação mais frequentes. Os resultados indicam que picos de latência são mais propensos a ocorrer em setores com menor quantidade de máquinas, onde anomalias individuais têm maior impacto sobre a média. Esses resultados apoiam esforços de otimização direcionados e ajudam a identificar áreas que requerem investigação adicional.

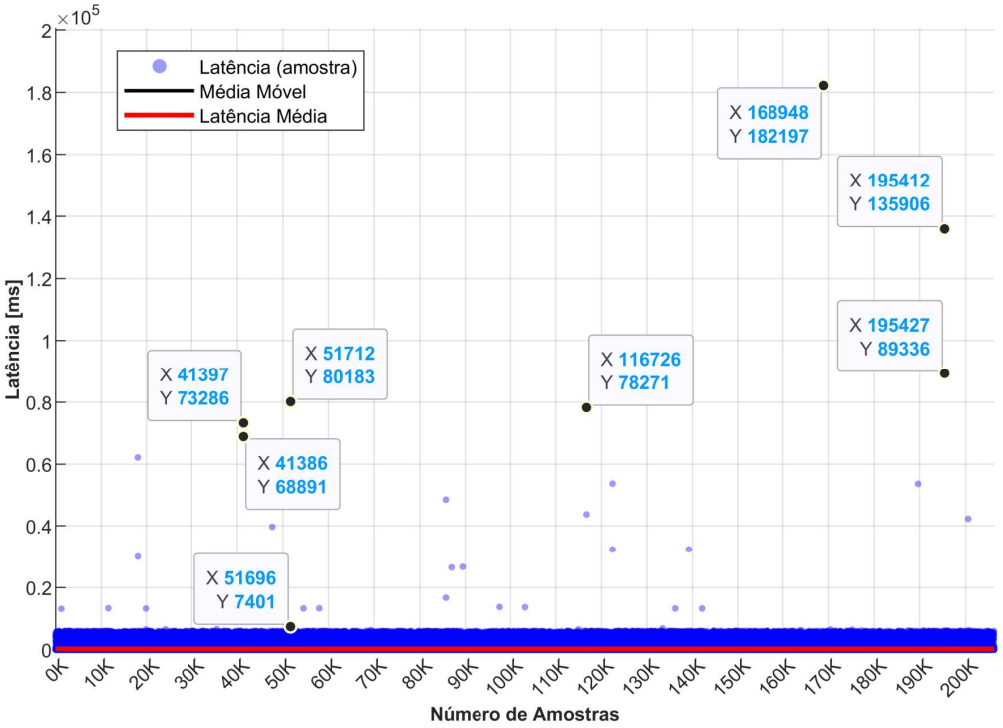
Figura 26 – Latência Média por Unidade Fabril e Setor.



Fonte: Próprio Autor

A Figura 27 apresenta o gráfico de dispersão completo das latências observadas nas 206.252 medições realizadas durante o experimento. Embora a maioria absoluta dos valores se concentre abaixo de 200 ms, observa-se a presença de picos superiores a 7000 ms. Esses valores, que representam menos de 0,05% do total de amostras, foram considerados outliers para fins de visualização, dado seu impacto desproporcional na escala do gráfico e sua baixa representatividade estatística. A decisão de tratá-los como outliers se fundamenta na hipótese de que tais picos decorrem de falhas pontuais de rede ou sobrecarga momentânea de processamento, não refletindo o comportamento típico do sistema. A exclusão desses pontos na visualização subsequente (Figura 28) visa preservar a legibilidade e a análise da distribuição principal, sem comprometer a integridade dos dados ou a validade dos resultados.

Figura 27 – Resultados de Latência do Experimento.

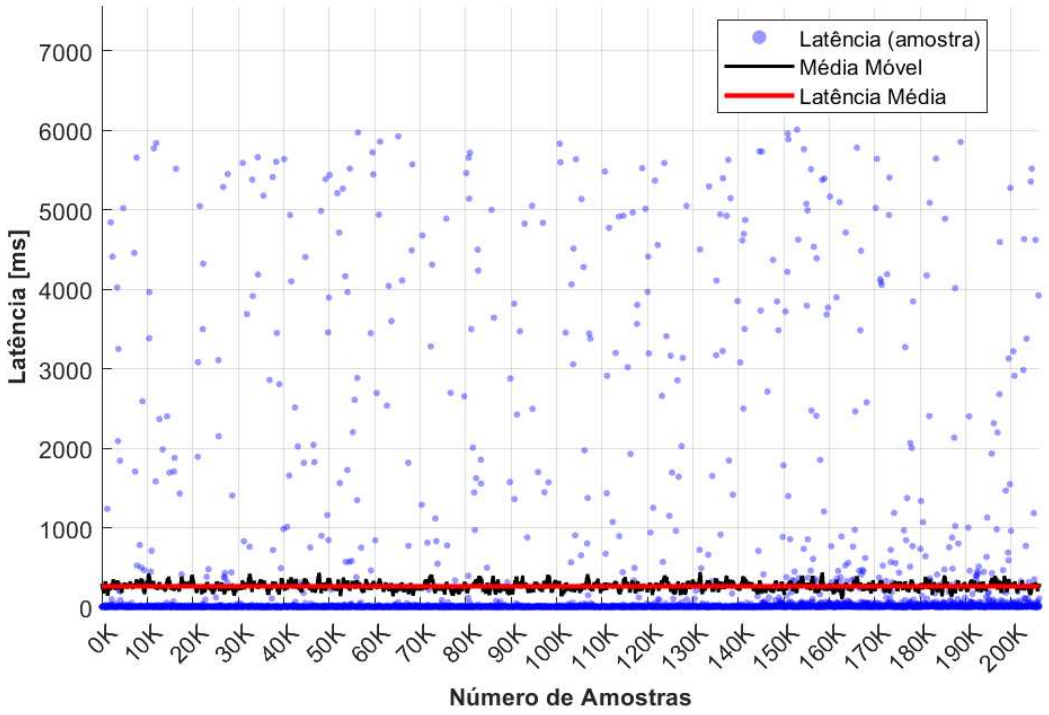


Fonte: Próprio Autor

Dando continuidade à análise, a Figura 28 apresenta uma versão filtrada do gráfico de dispersão, focada nos valores de latência inferiores a 7000 ms. Essa abordagem, além de excluir os outliers previamente discutidos, aplica uma amostragem de 1 a cada 50 pontos para preservar a legibilidade visual. Os pontos azuis representam as latências individuais, enquanto a linha preta indica a média móvel e a linha vermelha corresponde à média global. Nota-se que a maioria das mensagens apresenta latência inferior a 200 ms, com variações pontuais que

raramente ultrapassam 6000 ms. A proximidade entre a média móvel e a média global reforça a estabilidade do sistema sob condições típicas de operação. Essa visualização complementa a anterior sugerindo que, embora picos de latência possam ocorrer (provavelmente devido a anomalias de rede ou de processamento), o comportamento geral do sistema de mensagens é caracterizado por baixa latência e previsibilidade na maioria dos eventos.

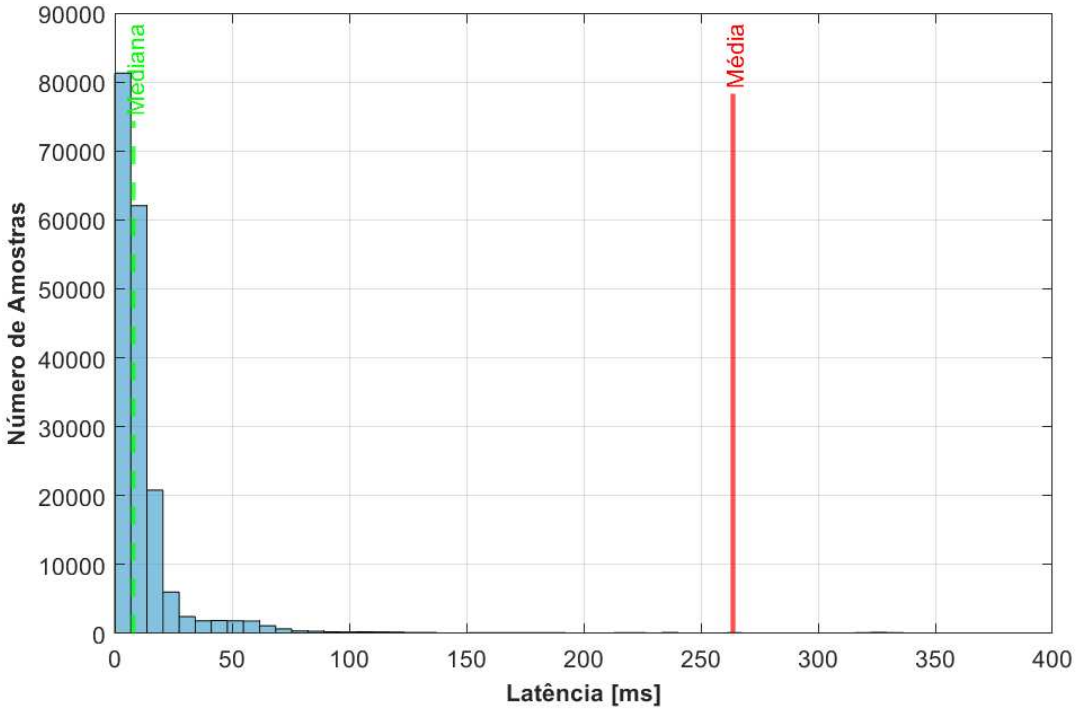
Figura 28 – Resultados de Latência do Experimento com Filtro.



Fonte: Próprio Autor

O histograma de latência apresentado na Figura 29 mostra a distribuição dos valores de latência ao longo de todo o conjunto de dados. Observa-se uma concentração substancial de amostras abaixo de 50ms, evidenciada pela forma assimétrica do histograma, com cauda à direita. A linha tracejada verde indica a mediana da latência, que se encontra bem abaixo da média (representada pela linha vermelha contínua, em aproximadamente 260 ms). Essa assimetria evidencia a presença de *outliers* raros com latências elevadas, que elevam a média sem impactar significativamente o desempenho típico do sistema. De modo geral, o histograma confirma que o sistema é altamente previsível e eficiente, com variabilidade mínima em condições normais de operação.

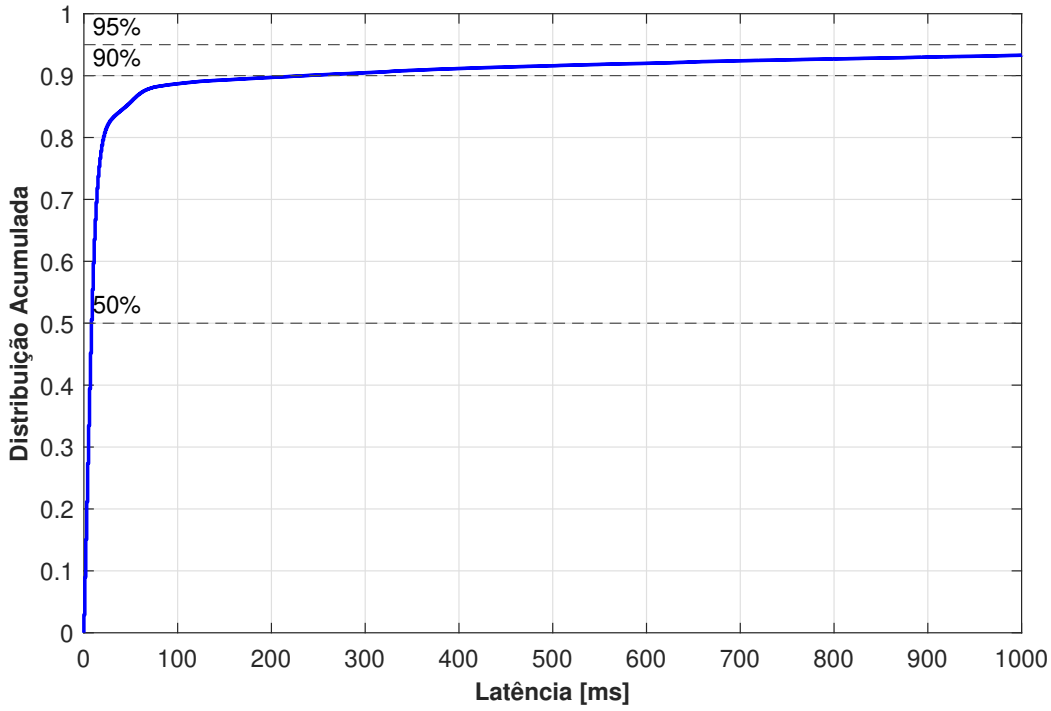
Figura 29 – Histograma das Latências Medidas.



Fonte: Próprio Autor

A *Cumulative Distribution Function* / Função de Distribuição Acumulada (CDF) apresentada na Figura 30 fornece a probabilidade acumulada dos valores de latência ao longo do conjunto de dados. A elevação acentuada no início da curva indica que uma grande proporção das mensagens é processada com latência muito baixa. Especificamente, aproximadamente 90% das mensagens apresentam latência de até 200ms. Essa perspectiva acumulada reforça o comportamento consistentemente de baixa latência do sistema e confirma sua capacidade de lidar com altos volumes de mensagens com tempos de resposta previsíveis.

Figura 30 – Gráfico de Probabilidade Acumulada de Latência (CDF).



Fonte: Próprio Autor

5 CONCLUSÕES E ATIVIDADES FUTURAS

5.1 Conclusão e Trabalhos Futuros

Este trabalho propôs e validou uma arquitetura de mensagens orientada a eventos e de baixo risco, baseada no Apache Kafka, contribuindo para a flexibilidade exigida no cenário atual de Indústria 4.0. A solução proposta vai ao encontro das considerações de Contador *et al.* (2020), que destacou as necessidades e desafios da indústria brasileira ao adotar novas tecnologias. O estudo de caso aqui abordado demonstrou a viabilidade da integração entre sistemas de TI e TO por meio de uma plataforma escalável, de baixo risco e de baixa latência, capaz de lidar com grandes volumes de dados em tempo real com configuração mínima. Os resultados quantitativos mostraram que o sistema manteve a integridade dos dados e baixa latência nas mensagens mesmo sob alta concorrência, validando a robustez do Kafka para aplicações críticas de alerta em contextos de manufatura.

O design modular da arquitetura permite a expansão contínua do sistema de alertas, possibilitando que cada unidade fabril opere de forma independente, ao mesmo tempo em que preserva a interoperabilidade global. Embora o protótipo tenha adotado uma estrutura simplificada, com um único tópico Kafka e sem a camada de aplicação móvel, ele ofereceu um ambiente controlado para avaliar os principais aspectos de desempenho sob condições de estresse. Essa configuração permitiu testar a confiabilidade e a escalabilidade do Kafka diante de um fluxo denso de produtores concorrentes. O sistema foi validado por meio de um protótipo funcional que simulou cenários de alta concorrência, com até 40,75% dos equipamentos fabris transmitindo alertas por meio de um único tópico Kafka. Mesmo nessas condições, o sistema manteve a latência consistentemente abaixo de 300 ms, entregando alertas em tempo real com sucesso, sendo resultados compatíveis com o que foi observado em estudos similares abordados na Seção 2.3.

Ainda assim, algumas limitações foram identificadas. O protótipo não implementou a estratégia de segmentação de tópicos, que prevê dois tópicos por unidade (um para alertas e outro para respostas), e também não implantou a aplicação móvel, a qual desempenha um papel importante por ser um dispositivo com conexão de menor estabilidade que deve aumentar a criticidade dentre as métricas apresentadas na Seção 3.3.5. Além disso, variações de latência observadas em um subconjunto de mensagens indicam a necessidade de uma investigação mais aprofundada da infraestrutura de rede, com o objetivo de identificar possíveis gargalos.

Apesar dessas limitações, a infraestrutura central de mensagens apresentou resultados promissores. Em comparação com a configuração anterior descrita na Seção 3.2.3, na qual foram observadas latências de até dois minutos e perda de dados de maneira frequente, a arquitetura proposta alcançou uma redução de latência superior a 95%, considerando o pior caso limitado a apenas seis segundos (desconsiderando os *outliers*) e sem perda de dados ou duplicação dos mesmos. Essa melhora significativa demonstra a adequação da arquitetura para comunicações críticas baseadas em eventos em ambientes industriais. Esses resultados destacam a confiabilidade e a eficiência do Kafka no processamento de cargas de mensagens em tempo real, mesmo sem a segmentação por tópicos proposta na Seção 3.3.1. O protótipo confirma que a comunicação orientada a eventos pode aumentar significativamente a capacidade de resposta e a confiabilidade dos sistemas de alerta em fábricas, além de oferecer uma base sólida para a implementação de infraestruturas de dados mais escaláveis e inteligentes em contextos industriais.

5.2 Atividades Futuras

Com base nos resultados promissores obtidos pelo protótipo, os trabalhos futuros focarão na implantação da arquitetura completa descrita na Seção 3.3.1 em todas as unidades de fabricação. Essa expansão permitirá uma avaliação mais ampla em um ambiente de escala produtiva, abordando as limitações atuais, como a segmentação de tópicos no Kafka, a integração do aplicativo móvel e o monitoramento estendido das métricas do sistema. E com a maturidade digital que esta implementação trás, se torna possível de aumentar a complexidade com a exploração de tecnologias complementares, como o Apache Flink (KLEPPMANN, 2016) para processamento de fluxos em tempo real e bancos de dados NoSQL mais adequados para dados temporais ou baseados em eventos, pode ampliar ainda mais a adaptabilidade e as capacidades analíticas do sistema, conforme demonstrado em estudos como Park e Chi (2016) e Park e Huh (2023). Esses aprimoramentos visam consolidar a camada de mensagens como um componente fundamental de uma plataforma de dados industrial mais ampla e inteligente.

REFERÊNCIAS

AHMED, S. F.; ALAM, M. S. B.; HOQUE, M.; LAMEESA, A.; AFRIN, S.; FARAH, T.; KABIR, M.; SHAFIULLAH, G.; MUYEEN, S. Industrial internet of things enabled technologies, challenges, and future directions. **Computers and Electrical Engineering**, Elsevier, v. 110, p. 108847, 2023.

BABENKO, V.; LOMOVSKYKH, L.; ORIEKHOVA, A.; KORCHYNSKA, L.; KRUTKO, M.; KONIAIEVA, Y. Features of methods and models in risk management of it projects. **Periodicals of Engineering and Natural Sciences (PEN)**, v. 7, n. 2, p. 629–636, 2019.

BALOUEK, D.; AMARIE, A. C.; CHARRIER, G.; DESPREZ, F.; JEANNOT, E.; JEANVOINE, E.; LÈBRE, A.; MARGERY, D.; NICLAUSSE, N.; NUSSBAUM, L.; RICHARD, O.; PEREZ, C.; QUESNEL, F.; ROHR, C.; SARZYNIEC, L. Adding virtualization capabilities to the grid'5000 testbed. In: IVANOV, I. I.; SINDEREN, M. van; LEYMANN, F.; SHAN, T. (Ed.). **Cloud Computing and Services Science**. Cham: Springer International Publishing, 2013. p. 3–20. ISBN 978-3-319-04519-1.

BARATA, J.; CUNHA, P. R.; COYLE, S. Evolving manufacturing mobility in industry 4.0: the case of process industries. **Journal of manufacturing technology management**, Emerald Publishing Limited, v. 31, n. 1, p. 52–71, 2020.

BOSI, F.; CORRADI, A.; MODICA, G. D.; FOSCHINI, L.; MONTANARI, R.; PATERA, L.; SOLIMANDO, M. Enabling smart manufacturing by empowering data integration with industrial iot support. In: IEEE. **2020 International Conference on Technology and Entrepreneurship (ICTE)**. Bologna, Italy, 2020. p. 1–8.

BRAUNISCH, N.; SCHLESINGER, S.; LEHMANN, R. Adaptive industrial iot gateway using kafka streaming platform. In: IEEE. **2022 IEEE 20th International Conference on Industrial Informatics (INDIN)**. Perth, Australia, 2022. p. 600–605.

CONTADOR, J. C.; SATYRO, W. C.; CONTADOR, J. L.; SPINOLA, M. d. M. Flexibility in the brazilian industry 4.0: Challenges and opportunities. **Global Journal of Flexible Systems Management**, Springer, v. 21, n. Suppl 1, p. 15–31, 2020.

ESTRADA, R. **Apache Kafka Quick Start Guide: Leverage Apache Kafka 2.0 to simplify real-time data processing for distributed applications**. Birmingham, UK: Packt Publishing, 2018. ISBN 9781788992251. Disponível em: <https://books.google.com.br/books?id=DtCBDwAAQBAJ>.

Gi Group Holding. **Tendências Globais de RH no setor de Manufatura – 2023**. 2023. Disponível em: <https://www.gigroupholding.com/portugal/insights/producao-tendencias-globais-de-rh-2023/>.

INDÚSTRIA, C. N. da. **Perfil da Indústria no Estado do Ceará**. 2024. Acesso em: 21 ago. 2024. Disponível em: <https://perfildaindustria.portaldaindustria.com.br/estado/ce>.

Internet Engineering Task Force. **RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format**. 2017. Accessed: [15/09/2024]. Disponível em: <https://www.rfc-editor.org/info/rfc8259>.

KLEPPMANN, M. **Making Sense of Stream Processing: The Philosophy Behind Apache Kafka and Scalable Stream Data Platforms**. Sebastopol, CA, USA: O'Reilly Media, Inc., 2016.

LUGERT, A.; BATZ, A.; WINKLER, H. Empirical assessment of the future adequacy of value stream mapping in manufacturing industries. **Journal of Manufacturing Technology Management**, Emerald Publishing Limited, v. 29, n. 5, p. 886–906, 2018.

MEDEIROS, A. P. de. **Teoria substantiva acerca da implementação da Indústria 4.0 em empresas industriais instaladas no Brasil**. Tese (Doutorado) — Universidade do Oeste de Santa Catarina, Universidade Federal de Santa Catarina, Florianópolis, 2021. Tese de doutorado.

MESQUITA, J. M. C. de; MARTINS, H. C.; DIAS, A. T.; RABELO, A. Impactos da sazonalidade da produção sobre os estoques e lucratividade: Análise do segmento industrial brasileiro. **Contabilidade Vista & Revista**, Universidade Federal de Minas Gerais, v. 27, n. 3, p. 61–80, 2016.

NOAC'H, P. L.; COSTAN, A.; BOUGÉ, L. A performance evaluation of apache kafka in support of big data streaming applications. In: IEEE. **2017 IEEE International Conference on Big Data (Big Data)**. Boston, MA, USA, 2017. p. 4803–4806.

OLIVEIRA, M. M. d.; PONCHIO, M. C.; NETO, M. S.; PIZZINATTO, N. K. Análise dos fatores de resistência na implantação de sistemas de informação na manufatura de eletrônicos. **JISTEM-Journal of Information Systems and Technology Management**, SciELO Brasil, v. 6, p. 507–524, 2009.

PARK, J.; CHI, S.-y. An implementation of a high throughput data ingestion system for machine logs in manufacturing industry. In: IEEE. **2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)**. Vienna, Austria, 2016. p. 117–120.

PARK, S.; HUH, J.-H. A study on big data collecting and utilizing smart factory based grid networking big data using apache kafka. **IEEE Access**, IEEE, 2023.

RAPTIS, T. P.; PASSARELLA, A. On efficiently partitioning a topic in apache kafka. In: IEEE. **2022 International Conference on Computer, Information and Telecommunication Systems (CITS)**. Piraeus, Greece, 2022. p. 1–8.

SHAPIRA, G.; PALINO, T.; SIVARAM, R.; PETTY, K. **Kafka: the definitive guide**. Sebastopol, CA, USA: O'Reilly Media, Inc., 2021.

SILVA, N. A. da; ABREU, J. L.; KLINGENBERG, C. O.; JUNIOR, J. A. V. A.; LACERDA, D. P. Industry 4.0 and micro and small enterprises: systematic literature review and analysis. **Production & Manufacturing Research**, Taylor & Francis, v. 10, n. 1, p. 696–726, 2022.

SISINNI, E.; SAIFULLAH, A.; HAN, S.; JENNEHAG, U.; GIDLUND, M. Industrial internet of things: Challenges, opportunities, and directions. **IEEE transactions on industrial informatics**, IEEE, v. 14, n. 11, p. 4724–4734, 2018.

STEEN, M. V.; TANENBAUM, A. S. **Distributed Systems**. Amsterdam, The Netherlands: Maarten van Steen, Vrije Universiteit Amsterdam, 2017. Disponível em: <https://www.distributed-systems.net/index.php/books/distributed-systems-3rd-edition-2017/>.

WEHNER, P.; PIBERGER, C.; GÖHRINGER, D. Using json to manage communication between services in the internet of things. In: IEEE. **2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)**. Montpellier, France, 2014. p. 1–4.

WU, H.; SHANG, Z.; WOLTER, K. Performance prediction for the apache kafka messaging system. In: IEEE. **2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)**. Zhangjiajie, China, 2019. p. 154–161.

WU, H.; SHANG, Z.; WOLTER, K. Trak: A testing tool for studying the reliability of data delivery in apache kafka. In: IEEE. **2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)**. Berlin, Germany, 2019. p. 394–397.

APÊNDICE A – EXEMPLO DE JSON DE UM ALERTA


```
{
    "codigo_alerta": "5d41402abc4b2a76b9719d911017c592",
    "motivo_alerta": "Perdas de Qualidade superiores a 10%",
    "equip_id": "S3A045",
    "data_inicio": "2024-09-12T14:25:33.000Z",
    "tipo": 3,
    "prioridade": 3,
    "helpchain": [{
            "nome": "João Silva",
            "cracha": 987654
        },
        {
            "nome": "Alberto Dias",
            "cracha": 922650
        }
    ],
    "falhas": [{
            "descricao": "Estria"
            "quantidade": 20
        },
        {
            "descricao": "Carvao"
            "quantidade": 15
        }
    ],
    "retorno": true,
    "tempo_intervalo": 20,
    "url_resposta": "http://enderecoAPI:3000/response/5d41402abc4b2a7"
}
```