Rodrigo Aparecido Silva Maia

Soluções IoT com atuação em Edge Computing na Indústria 4.0: Uma Arquitetura de Referência

Brasil

Fevereiro de 2026

Rodrigo Aparecido Silva Maia

Soluções IoT com atuação em Edge Computing na Indústria 4.0: Uma Arquitetura de Referência

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Universidade Federal de Itajubá – UNIFEI

Programa de Pós-Graduação em Ciência e Tecnologia Computação

Orientador: Bruno Guazzelli Batista

Brasil

Fevereiro de 2026

Agradecimentos

Primeiramente, agradeço a Deus pela saúde e pela força concedidas para concluir esta etapa. À minha família, dedico minha gratidão, especialmente à minha mãe, Maria Lourdes, pelo apoio constante. Ao meu pai, Olívio (*in memoriam*), meu eterno agradecimento pelo incentivo que me trouxe até aqui; embora não esteja presente fisicamente, sua memória é parte fundamental desta conquista.

Agradeço aos professores e professoras do Programa de Pós-Graduação em Ciência e Tecnologia da Computação da UNIFEI por compartilharem seus conhecimentos e proporcionarem meu crescimento profissional e acadêmico.

Agradeço ao Prof. Adler Diniz de Souza pela importante contribuição na produção da Revisão Sistemática da Literatura presente neste trabalho.

Meus sinceros agradecimentos ao meu orientador, Prof. Bruno Guazzelli Batista, pela orientação, paciência e valiosas contribuições fundamentais para a realização desta pesquisa.

Resumo

A demanda por soluções de *Edge Computing* na Indústria 4.0 é crescente, especialmente em ambientes de produção que exigem tomada de decisão em tempo real. No entanto, a adoção de *Edge Computing* em redes industriais baseadas em Internet das Coisas (IoT - *Internet of Things*) impõe desafios significativos, como interoperabilidade entre dispositivose softwares, baixa latência e escalabilidade. Com o objetivo de apoiar arquitetos de software na definição de arquiteturas para esse contexto, este trabalho propõe uma Arquitetura de Referência (RA), denominada ED-operator RA. A construção da ED-operator RA baseou-se em uma Revisão Sistemática da Literatura (RSL), por meio da qual foram identificados artefatos relevantes, como requisitos não funcionais, soluções de software, *middlewares* e arquiteturas de referência existentes. A arquitetura proposta foi modelada a partir de diferentes visões arquiteturais, incluindo contexto, contêineres e módulos, implantação, fluxo de controle e fluxo de dados. Para garantir a validade da arquitetura, o trabalho apresenta uma avaliação arquitetural fundamentada no método SAAM (*Scenario-Based Architecture Analysis Method*), apoiada por um experimento realizado com um protótipo da ED-operator RA. Dessa forma, a ED-operator RA demonstrou em ambiente simulado a capacidade de atender de forma consistente os requisitos definidos, conciliando escalabilidade, interoperabilidade, segurança, privacidade e baixa latência em ambientes IoT baseados em *Edge Computing*. Além disso, o estudo disponibiliza uma metodologia estruturada e reproduzível para a construção e avaliação de arquiteturas de referência.

Palavras-chave: Edge Computing, Indústria 4.0, Internet das Coisas, IoT, IIoT, Arquitetura de Referência, SAAM

Abstract

The demand for Edge Computing solutions in Industry 4.0 is increasing, especially in production environments that require real-time decision-making. However, implementing Edge Computing in industrial networks based on the Internet of Things (IoT) poses significant challenges, such as interoperability, low latency, and scalability. To support software architects in defining architectures for this context, this work proposes a Reference Architecture (RA) named ED-operator RA. The development of ED-operator RA was based on a Systematic Literature Review (SLR), through which relevant artifacts were identified, including non-functional requirements, software solutions, middleware, and other reference architectures previously proposed in the literature. The proposed architecture was modeled using multiple architectural views, including context view, container and module view, deployment view, control flow view, and data flow view. To ensure the validity of the architecture, this work presents an architectural evaluation based on the SAAM (Scenario-Based Architecture Analysis Method), supported by an experiment conducted using a prototype of the ED-operator RA. As a result, the ED-operator RA consistently demonstrated compliance with the defined requirements, reconciling scalability, interoperability, security, privacy, and low latency in IoT environments based on Edge Computing. Furthermore, this study provides a structured and reproducible methodology for the development and evaluation of reference architectures.

Keywords: Edge Computing, Industry 4.0, Internet of Things, IoT, IIoT, Reference Architectures, SAAM

Lista de ilustrações

Figura 1 – Cloud, Fog e Edge Computing 18
Figura 2 – Fluxo de trabalho definido para extrair e selecionar os artigos. 24
Figura 3 – Distribuição das fontes 24
Figura 4 – Arquitetura FIWARE, adaptado de (FIWARE FOUNDATION E.V, 2024). 32
Figura 5 – Estrutura RAMI 4.0, adaptado de (MIRANI et al., 2022). 35
Figura 6 – Camadas IIRA, adaptado de (PEDONE; MEZGÁR, 2018). 37
Figura 7 – FAR-Edge RA (SITTóN-CANDANEDO et al., 2020). 38
Figura 8 – Contexto da Arquitetura de Referência 49
Figura 9 – Contêineres da Arquitetura de Referência 50
Figura 10 – Implantação da Arquitetura de Referência 53
Figura 11 – Exemplo da ED-operator RA em Modo *Cluster*. 54
Figura 12 – Fluxo de controle de implantação e atualização de microsserviços na ED-operator RA 55
Figura 13 – Envio de dados 60
Figura 14 – Recebimento de dados 60
Figura 15 – Envio de dados agregados para nuvem 61
Figura 16 – Implementação do Protótipo (módulos de software) 64
Figura 17 – Comandos Disponíveis no Protótipo 65
Figura 18 – Bloco de produção no Factory.io 66
Figura 19 – Resumo do experimento 67
Figura 20 – Comparação entre a curva do simulador desenvolvido e a curva apresentada em Nagorny et al. (2018) 68
Figura 21 – Fluxo de operação do PLC 69
Figura 22 – Fluxo de inspeção no Node-RED 69
Figura 23 – Dashboard 003 nas três sequencias de testes 76
Figura 24 – Gráficos de latência das mensagens da IA (MQTT) para a linha nas três sequências de testes 77
Figura 25 – Gráficos de latência de mensagens da linha para câmera nas três sequências de testes 78
Figura 26 – Gráfico sobre o tempo de gravação no banco 79
Figura 27 – Concorrência de armazenamento no dispositivo de processamento de borda 80
Figura 28 – Gráficos de uso de CPU 81
Figura 29 – Gráficos de consumo de memória 81
Figura 30 – Pacotes de dispositivos em TLS 1.2 83

Figura 31 – Latências do teste com 8 blocos de produção com interrompimento da conexão com a nuvem 84

Lista de tabelas

Tabela 1 – Cenário de Qualidade para Interoperabilidade (SNE01) 26
Tabela 2 – Cenário de Qualidade para Baixa latência (SNE02) 27
Tabela 3 – Cenário de Qualidade para Escalabilidade (SNE03) 28
Tabela 4 – Cenário de Qualidade para Segurança e Privacidade (SNE04) 30
Tabela 5 – Cenário de Qualidade para Disponibilidade após falhas (SNE05) 30
Tabela 6 – Cenário de Qualidade para atualizações do sistema (SNE06) 31
Tabela 7 – Perfis de Prioridade 57
Tabela 8 – Cenários de qualidade 72
Tabela 9 – Conjunto de testes 73
Tabela 10 – Resultados de latência 77
Tabela 11 – Análise da taxa de falhas 82
Tabela 12 – Resultados do teste sem a conexão com a nuvem 84
Tabela 13 – Comparação sistemática entre a ED-operator RA e outras arquiteturas 87

Lista de abreviaturas e siglas

API	*Application Programming Interface*
ARID	*Active Reviews for Intermidiate Designs*
ATAM	*Architecture Tradeoff Analysis Method*
CD	*Continuous Delivery* (Entrega Contínua)
CI	*Continuous Integration* (Integração Contínua)
CIN	Critérios de Inclusão
CE	Critérios de Exclusão
ETL	*Extract* (Extração), *Transformation* (Transformação) e *Load* (Carregamento)
IaaS	*Infrastructure as a Service* (Infraestrutura como Serviço)
ICC	*Industrial Internet Consortium Reference Architecture*
IIRA	*Industrial Internet Reference Architecture* e Industry 4.0
IoT	*Internet of Things* (Internet das Coisas)
IIoT	*Industrial Internet of Things* (Internet das Coisas Industrial)
MES	*Manufacturing Execution System*
MIINT	*Middleware for IIoT Platforms INTegration*
MQTT	*Message Queuing Telemetry Transport*
M2M	*Machine to Machine* (Máquina para Máquina)
OPC UA	*Open Platform Communications Unified Architecture*
PaaS	*Platform as a Service* (Plataforma como Serviço)
PICOC	*Population, Intervention, Comparison, Outcome, Context*
PLC	*Programmable Logic Controller*
RA	*Reference Architectures* (Arquiteturas de Referência)
RAMI 4.0	*Reference Architectural Model for Industry 4.0*

REST	*Representational State Transfer*
RQ	*Research Question* (Questões de Pesquisa)
RSL	Revisão Sistemática da Literatura
SaaS	*Software as a Service* (Software como Serviço)
SOA	*Service-Oriented Architecture* (Arquitetura Orientada a Serviços)
SAAM	*Software Architecture Analysis Method*
SLA	*Service Level Agreement* (Acordo de Nível de Serviço)
SSH	*Secure Shell* (Terminal Seguro)
TLS	*Transport Layer Security*

Sumário

1 INTRODUÇÃO 13
1.1 Motivação 14
1.2 Objetivos 15
1.3 Estrutura do Documento 15

2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Considerações Iniciais 17
2.2 Fontes básicas 17
2.3 Metodologia de Pesquisa da Revisão Sistemática da Literatura . . . 20
2.4 Resultados da RSL 23
2.4.1 **RQ1**: Quais são os requisitos não-funcionais padrões em soluções de IoT que utilizam *Edge Computing* na Indústria 4.0? 23
2.4.1.1 Interoperabilidade 25
2.4.1.2 Baixa Latência 26
2.4.1.3 Escalabilidade 27
2.4.1.4 Segurança e Privacidade 29
2.4.1.5 Manutenibilidade e disponibilidade 29
2.4.2 **RQ2**: Quais *middlewares* ou softwares de IoT foram propostos na literatura para dar suporte ao *Edge Computing* na Indústria 4.0? 30
2.4.2.1 *Open Platform Communication Unified Architecture* (OPC UA) 31
2.4.2.2 FIWARE 32
2.4.2.3 Node-RED 33
2.4.3 **RQ3**: Quais arquiteturas de referência foram propostas para sistemas de software IoT com implementação em *Edge Computing* na Indústria 4.0? . . 34
2.5 Considerações Finais 39

3 METODOLOGIA DE DESENVOLVIMENTO 41
3.1 Considerações Iniciais 41
3.2 Passo 01: Decisão do tipo de Arquitetura de Referência 41
3.3 Passo 02: Seleção da Estratégia de Design 42
3.4 Passo 03: Aquisição Empírica de Dados 42
3.5 Passo 04: Construção da Arquitetura de Referência 43
3.6 Passo 05: Permitir Variabilidade na Arquitetura 43
3.7 Passo 06: Avaliar a Arquitetura de Referência 44
3.8 Considerações Finais 45

4 ARQUITETURA DE REFERÊNCIA ED-OPERATOR RA (EDGE OPERATOR REFERENCE ARCHITECTURE) 47
4.1 Considerações Iniciais 47
4.2 Análise do contexto 47
4.3 Visão de Contexto 48
4.4 Visão de contêineres e módulos 49
4.5 Visão de Implantação 52
4.5.1 Operador e Orquestrador 52
4.5.2 Compartilhamento de processo com a nuvem 56
4.5.3 Estratégias de escalabilidade 57
4.5.4 Armazenamento 58
4.6 Visões de fluxo de dados 59
4.7 Rationale 61
4.8 Considerações Finais 62

5 PROTOTIPAÇÃO E EXPERIMENTAÇÃO 63
5.1 Considerações Iniciais 63
5.2 Protótipo 63
5.3 Simulação Industrial 66
5.3.1 Agregação de Dados 70
5.4 Avaliação Individual dos Cenários 71
5.4.1 *Dashboard* MQTT para PLC (DASH003) 71
5.4.2 *Dashboard* PLC para câmera (DASH004) 73
5.5 Séries de testes 73
5.6 Considerações Finais 74

6 RESULTADOS DA AVALIAÇÃO ARQUITETURAL 75
6.1 Considerações Iniciais 75
6.2 Interoperabilidade (SNE01) 75
6.3 Baixa latência (SNE02) 76
6.4 Escalabilidade (SNE03) 80
6.5 Segurança e Privacidade (S04) 83
6.6 Disponibilidade após falhas (SNE05) 83
6.7 Manutenibilidade (SNE06) 85
6.8 Avaliação das interações entre cenários 85
6.9 Avaliação Comparativa com Arquiteturas de Referência Existentes . 86
6.10 Considerações Finais 88

7 CONCLUSÃO 89
7.1 Estudos Futuros 90

7.2 Publicações 91

REFERÊNCIAS 92

1 Introdução

O conceito de Indústria 4.0 emergiu em 2013, na Alemanha, como uma tentativa de promover a digitalização de todo o processo industrial, visando maior eficiência, flexibilidade, integração, previsibilidade e inteligência nos processos de manufatura. No entanto, o ambiente industrial é complexo e de difícil integração, no qual sistemas e dispositivos legados devem interoperar com sistemas e dispositivos modernos. Além da integração vertical que ocorre dentro da fábrica, a Indústria 4.0 também busca conectar toda a cadeia industrial, implementando uma colaboração entre diferentes organizações, a fim de promover uma integração de ponta a ponta em toda a cadeia industrial (DINTÉN; MARTÍNEZ; ZORRILLA, 2024).

Paralelamente, a Internet das Coisas (IoT – *Internet of Things*) tem revolucionado o mundo ao permitir que objetos físicos se conectem à Internet, possibilitando a interação com usuários e com outros dispositivos, sem a necessidade de intervenção humana. A capacidade de comunicação *Machine to Machine* (M2M) permite que sistemas físicos operem de forma inteligente, automatizando processos que antes necessitavam de intervenção humana. Essas características tornam a IoT uma ferramenta fundamental para a implementação da Indústria 4.0 (MIRANI et al., 2022).

Embora a aplicação da IoT em diversos contextos, como o industrial, traga inúmeras vantagens, a falta de poder computacional para lidar com grandes quantidades de dispositivos se apresentava como um desafio para sistemas de IoT. Com a chegada da computação em nuvem, foi possível mitigar o desafio de escalabilidade presente nos sistemas IoT (VENANZI et al., 2021). No entanto, as aplicações industriais podem exigir sistemas em *real-time*, com processamentos rápidos e baixa latência, o que representa um desafio para sistemas baseados em nuvem (SITTÓN-CANDANEDO et al., 2019).

Para superar o desafio da baixa latência em sistemas baseados em nuvem, surgiu o conceito de *edge computing* ou computação de borda. A *edge computing* propõe a transferência de recursos computacionais e de armazenamento da nuvem para a borda da rede, com o objetivo de oferecer serviços mais próximos da origem dos dados e, consequentemente, diminuir a latência das respostas (LOPEZ et al., 2015).

A colaboração entre computação em nuvem e a computação na borda se torna essencial em sistemas com as características que a IIoT (Industrial IoT) apresenta. No entanto, a incorporação de múltiplos conceitos em um único sistema, como os associados à Internet das Coisas, pode torná-lo complexo e dificultar a compreensão por parte das equipes de desenvolvimento e manutenção.

Nesse contexto, as arquiteturas de referência (RAs - *Reference Architecture*) são

fundamentais para facilitar o entendimento e a comunicação das decisões tomadas durante o desenvolvimento de software, especialmente no que tange às preocupações sistêmicas, frequentemente associadas aos requisitos não-funcionais, como interoperabilidade, seguração e privacidade. As RAs encapsulam a essência das arquiteturas já existentes, visando atender às futuras necessidades de evolução e fornecendo suporte no desenvolvimento de novas arquiteturas (CLOUTIER et al., 2010).

Dessa forma, a utilização de arquiteturas de referência para facilitar a compreensão, esclarecer pontos importantes e promover a implementação de conceitos tecnológicos é comum na literatura. Um exemplo é a arquitetura RAMI 4.0, proposta pela Alemanha com o objetivo de promover e estruturar a Indústria 4.0 (DINTÉN; MARTÍNEZ; ZORRILLA, 2024).

1.1 Motivação

A crescente adoção da IoT industrial tem impulsionado a criação de sistemas cada vez mais complexos, distribuídos e heterogêneos. Esta diversidade de tecnologias, protocolos e requisitos não funcionais típicos desses sistemas dificulta a compreensão de todas as variáveis envolvidas, prejudicando a tomada de decisão por parte de arquitetos e desenvolvedores de software.

As aplicações industriais são frequentemente caracterizadas por restrições temporais rígidas, o que poderia inviabilizar, em alguns casos, a dependência exclusiva da computação em nuvem, devido à latência introduzida nas respostas. Dessa forma, a adoção do paradigma de *edge computing* se torna essencial, pois essa tecnologia propõe o deslocamento de parte do processamento para mais próximo da origem dos dados, com o objetivo de reduzir a latência, melhorar a disponibilidade e permitir decisões mais ágeis em ambientes industriais (BUTTE; BUTTE, 2022).

A incorporação conjunta dos conceitos de computação em nuvem e computação na borda em sistemas de IoT industriais, embora promissora, aumenta ainda mais a complexidade arquitetural, dificultando a compreensão por parte das equipes de desenvolvimento. Diante disso, arquiteturas de referência se apresentam como instrumentos valiosos ao fornecerem uma forma eficaz de comunicar as decisões tomadas durante o desenvolvimento de software, especialmente no que tange às preocupações sistêmicas (GALSTER; AVGERIOU, 2011).

Apesar da literatura já apresentar RAs para sistemas IoT que contemplam dispositivos de processamento de borda, frequentemente, carecem de detalhamento adequado de como tais dispositivos devem ser integrados à rede. Além disso, não apresentam quais recursos e responsabilidades devem ser atribuídos a eles.

Dessa forma, este trabalho busca preencher essa lacuna por meio da definição de uma arquitetura de referência voltada ao desenvolvimento de softwares — ou módulos de software — para IoT industrial, com foco nos dispositivos de processamento de borda. A proposta especifica os recursos necessários e os serviços esperados desses dispositivos, com o objetivo de apoiar a construção de soluções mais eficientes e aderentes aos requisitos da Indústria 4.0.

1.2 Objetivos

Com base no contexto apresentado, este trabalho tem como objetivo propor uma arquitetura de referência para soluções de IoT Industrial na *edge computing*. Essa arquitetura busca auxiliar os arquitetos de software na tomada de decisões arquiteturais, visando a escalabilidade e o desempenho das soluções de IoT na Indústria. Para a elaboração da arquitetura de referência, este trabalho propõe cumprir três objetivos específicos:

1. Realizar uma revisão sistemática da literatura para analisar o estado da arte sobre o tema e extrair itens importantes, como: os conceitos fundamentais relacionados a IoT, *edge computing* e indústria 4.0, soluções IoT na indústria 4.0 que demandam processamento de borda, requisitos de qualidade comuns e cenários de qualidade capazes de validar a conformidade das soluções com as necessidades de um ambiente industrial;

2. Com base na metodologia proposta e nos artefatos encontrados na revisão, elaborar uma arquitetura de referência voltada ao papel dos dispositivos de processamento de borda em redes de IoT industrial.

3. Desenvolver um protótipo para validar a arquitetura de referência proposta, por meio de experimentos que estimulem e testem os aspectos mencionados nos cenários de qualidade identificados na literatura, demostrando a conformidade da arquitetura.

1.3 Estrutura do Documento

O restante deste documento está dividido em sete capítulos, que abrangem o estudo sistemático dos trabalhos disponíveis na área, a proposta da arquitetura de referência e a avaliação dessa arquitetura por meio de experimentos com um protótipo.

O Capítulo 2 apresenta a fundamentação teórica do estudo e a revisão sistemática da literatura, descrevendo a metodologia de pesquisa adotada e os resultados obtidos.

O Capítulo 3 descreve as metodologias adotadas para a elaboração da arquitetura, definindo todo o processo arquitetural.

O Capítulo 4 discute os artefatos encontrados durante a revisão sistemática que podem ser utilizados na elaboração da arquitetura, além de apresentar a arquitetura proposta neste trabalho, especificando os detalhes da modelagem de diferentes visões arquiteturais. As visões apresentadas incluem: visão de contexto, visão de contêiner e módulos, visão de implantação, visões de fluxo de controle e visões de fluxo de dados.

O Capítulo 5 detalha o desenvolvimento do protótipo, baseado na arquitetura de referência proposta pelo estudo, e a simulação que foi utilizada nos testes. Este capítulo também apresenta os testes executados durante a prototipação.

O Capitulo 6 apresenta os dados obtidos durante as experimentações e com eles discute a conformidade da arquitetura de referência com os cenários de qualidade desenvolvidos durante a revisão sistemática.

Por fim, o Capítulo 7 apresenta a conclusão do estudo e oportunidades para estudos futuros.

2 Fundamentação Teórica

2.1 Considerações Iniciais

Toda a fundamentação teórica deste estudo será apresentada neste capítulo, sendo dividida em duas partes. A primeira parte aborda os fundamentos básicos do estudo, incluindo temas como *edge computing*, *cloud computing*, entre outros.

Com o objetivo de aprofundar a fundamentação sobre os temas do estudo e identificar artefatos que contribuam para o desenvolvimento da arquitetura de referência, foi conduzida uma revisão sistemática da literatura (RSL), seguindo o modelo PICOC proposto por Wohlin et al. (2012). Dessa forma, este capítulo apresenta a RSL em duas seções: a segunda seção descreve a metodologia empregada, juntamente com o protocolo de revisão, e a terceira seção apresenta e discute os resultados obtidos.

2.2 Fontes básicas

A IIoT é um conceito da indústria 4.0 que se baseia nos princípios da IoT para aumentar a eficiência dos processos industriais, por meio da integração entre os sistemas de informação e os sistemas de operação (VENANZI et al., 2021; MIRANI et al., 2022).

Apesar das vantagens proporcionadas pela aplicação da IoT no contexto industrial, a limitação de poder computacional para lidar com grandes quantidades de dispositivos ainda representa um desafio para os sistemas de IIoT. Para mitigar esse problema, a computação em nuvem se apresenta como uma ferramenta fundamental (VENANZI et al., 2021).

A computação em nuvem é um modelo de computação no qual recursos dinamicamente escaláveis e virtualizados são fornecidos como serviços sob demanda em redes públicas e privadas (HUAWEI TECHNOLOGIES CO. Ltd., 2023). A computação em nuvem possui os seguintes modelos de serviços (HUAWEI TECHNOLOGIES CO. Ltd., 2023):

- IaaS (*Infrastructure as a Service*) - Oferece infraestrutura básica como servidores, redes e armazenamento. Permite que o usuário gerencie o sistema operacional, e tenha controle total do ambiente;
- PaaS (*Platform as a Service*) - Fornece uma plataforma completa para desenvolvimento, o provedor gerencia a infraestrutura e o sistema operacional, e o usuário foca apenas no desenvolvimento e execução de suas aplicações;

- SaaS (*Software as a Service*) - Disponibiliza aplicações prontas para uso, acessadas pela internet, com todo o gerenciamento feito pelo provedor;

Apesar das vantagens da computação em nuvem para sistemas IoT em geral, aplicações de IIoT são, em sua maioria, sistemas de tempo real que exigem processamento rápido e baixa latência. Essa restrição temporal rígida pode, em alguns casos, inviabilizar o uso da computação em nuvem devido à latência introduzida nas respostas (SITTÓN-CANDANEDO et al., 2019).

A adoção dos paradigmas *edge computing* e *fog computing* surgem como alternativas para atender a essas restrições, pois propõem o deslocamento de parte do processamento para mais próximo da origem dos dados em níveis diferentes. Com isso, busca-se reduzir a latência, melhorar a disponibilidade e permitir decisões mais ágeis em ambientes industriais (BUTTE; BUTTE, 2022) (SHERLEKAR; STARLY; COHEN, 2019).

A *fog computing* ou computação em névoa opera entre a nuvem e a borda, sendo capaz de executar tarefas semelhantes às realizadas na nuvem, como processos de virtualização (AAZAM; ZEADALLY; HARRAS, 2018). Essa capacidade deve-se ao fato de que a *fog computing* se baseia em pequenos *data centers*, enquanto a *edge computing* ou computação de borda mantém seu processamento dentro da rede IoT por meio de dispositivos mais complexos (MILADINOVIC et al., 2021). Dessa forma, a *fog computing* apresenta um nível de elasticidade inferior ao da *cloud computing*, porém superior ao da *edge computing*. Em termos de latência, a *fog computing* situa-se entre os dois extremos, pois apresenta latência maior que a da *edge computing*, mas ainda menor que a da *cloud computing*. A Figura 1 ilustra os três paradigmas.

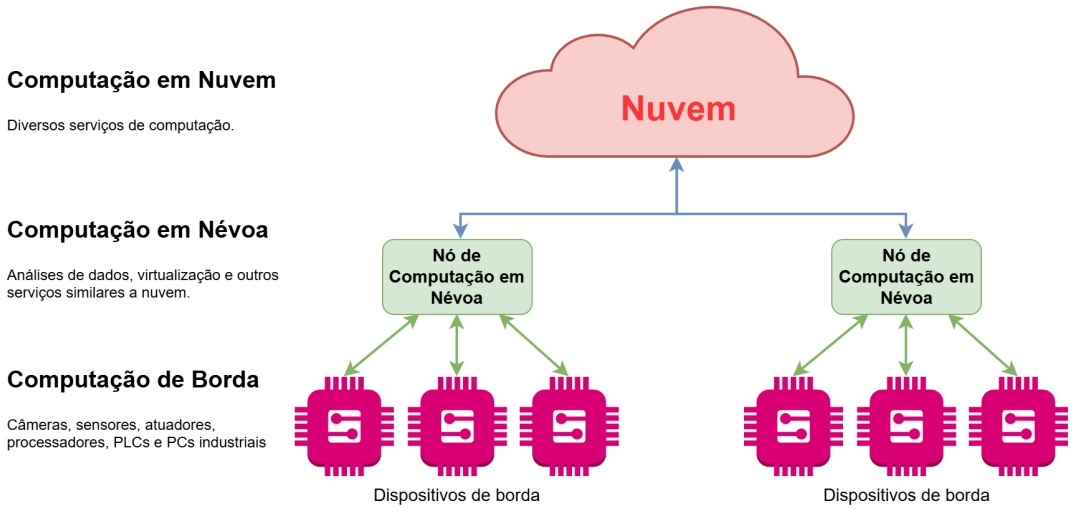

Figura 1 – Cloud, Fog e Edge Computing

Como este estudo tem como objetivo criar uma arquitetura que minimize ao máximo a latência das respostas, o foco é direcionado ao paradigma de *edge computing*.

A Figura 1 também apresenta dispositivos comumente encontrados em redes de IoT Industrial, como câmeras, sensores, atuadores, PCs industriais (IPC) e controladores lógicos programáveis (PLCs - *Programmable Logic Controller*). Os PLCs são dispositivos industriais de alta confiabilidade utilizados para realizar controles determinísticos em ambientes industriais. Eles são capazes de executar operações aritméticas, lógica discreta, contagem, sequenciamento e temporização (PHILLIPS et al., 2013).

Os dispositivos industriais são acompanhados por um sistema de Supervisão, Controle e Aquisição de Dados (SCADA — *Supervisory Control and Data Acquisition*), o qual é responsável pela coleta de dados, pelo controle dos dispositivos e pela disponibilização de interfaces para supervisão (SHERLEKAR; STARLY; COHEN, 2019) (BAZI et al., 2023) (CH. et al., 2018).

Além dos sistemas SCADA, que têm como foco a supervisão técnica do processo, o ambiente industrial ainda possui o Sistema de Execução da Manufatura (MES — *Manufacturing Execution System*), responsável por armazenar dados e executar ordens de produção. E os sistemas MES que atuam como intermediários entre os sistemas SCADA e os Sistemas de Planejamento de Recursos Empresariais (ERP — *Enterprise Resource Planning*) (ALMADA-LOBO, 2016)(PEDONE; MEZGÁR, 2018). ERPs são os sistemas de nível mais alto da hierarquia industrial, responsáveis por todo o planejamento corporativo, incluindo compras, gestão de estoques e finanças (AL-JAROODI; MOHAMED; JAWHAR, 2018).

Segundo Villar et al. (2024), a IIoT e a *edge computing* fazem parte do escopo de tecnologias da Industria 5.0 juntamente com gêmeos digitais, *Robotic* IoT, Inteligência Artificial e tecnologias 5G. O termo Industria 5.0 foi consolidado quando a Comissão Européia publicou oficialmente o relatório de Breque, Nul e Petridis (2021). O relatório define os principais pilares desse conceito:

1. **Sustentabilidade** - Preservação do meio-ambiente e uso consciente do recursos naturais do planeta;

2. ***Human-centric*** - Garante que pessoas, máquinas e sistemas estejam integrados, promovendo a colaboração entre trabalhadores e tecnologias avançadas, como IA, automação e robôs colaborativos, por meio do desenvolvimento de competências técnicas. O empoderamento humano, que busca torná-los menos substituíveis, melhorar as condições de trabalho e ampliar a participação nos processos produtivos, também constitui um dos objetivos centrais desse pilar;

3. **Resiliência** - Capacidade de resistir e se adaptar a choques externos (como a pandemia de COVID-19) por meio de modelos mais circulares, tecnologias digitais emergentes e gestão eficiente de energia e recursos.

A *edge computing* tem sido empregada em setores como manufatura, mineração, processamento e transporte, sendo fortemente dependente de dispositivos. No entanto, essa abordagem pode aumentar a complexidade da infraestrutura, além de impor limitações à manutenção do sistema e à capacidade de armazenamento (MUZELAK; SKOVRANEK, 2022). Nesse contexto, uma arquitetura bem definida torna-se fundamental para facilitar o entendimento, a comunicação e a análise das vantagens e desvantagens associadas às decisões arquiteturais ao longo do desenvolvimento de software.

Com o objetivo de apoiar arquitetos no desenvolvimento de suas soluções, este trabalho propõe uma arquitetura de referência para aplicações de IoT Industrial baseadas em *edge computing*. Para alcançar esse objetivo e mapear o estado da arte da área, foi conduzida uma Revisão Sistemática da Literatura.

2.3 Metodologia de Pesquisa da Revisão Sistemática da Literatura

A metodologia utilizada nesta pesquisa segue as diretrizes propostas por Petersen et al. (2008) para estudos de mapeamento sistemático em engenharia de software. O processo desenvolvido compreende as seguintes etapas principais: (*i*) Definição dos objetivos e questões de pesquisa, (*ii*) Desenvolvimento do protocolo de revisão, que inclui a definição dos critérios de inclusão/exclusão, a definição das bases de dados, definição das estratégias de busca e dos formulários de qualidade e extração dos dados, (*iii*) Seleção de estudos relevantes, (*iv*) Extração de dados e mapeamento, e (*v*) Análise e discussão dos resultados.

Para planejar e conduzir a SLR, foi utilizada o Parsifal (PARSIFAL, 2024) uma ferramenta *online* que auxilia na execução de revisões, especialmente no contexto da engenharia de software. A ferramenta oferece uma interface intuitiva para organizar artefatos (listas de verificação de qualidade e formulários de extração de dados).

Para auxiliar no desenvolvimento das questões de pesquisa e na definição dos parâmetros de busca foi elaborado um modelo PICOC (WOHLIN et al., 2012) que possui uma separação lógica que ajuda na definição das questões de pesquisa e dos parâmetros de busca. O nome PICOC é um acrônimo para as separações *Population* (*Who?*), *Intervention* (*What or How*), *Comparison* (*Compared to what?*), *Outcome* (*What are the expected results or effects?*) e *Context* (WOHLIN et al., 2012). Modelo PICOC:

- **População:** Soluções de IoT com características específicas (por exemplo, escalabilidade, capacidade de processamento de dados, tratamento de latência) que utilizam tecnologias de *edge computing* na Indústria 4.0.
- **Intervenção:** Soluções de IoT que suportam tecnologias de *edge computing* na Indústria 4.0.

- **Comparação:** Não se aplica.
- **Resultado:** Identificação de requisitos não-funcionais (RFNs) padrão (por exemplo, escalabilidade, confiabilidade, latência) para soluções de IoT na Indústria 4.0, possíveis *frameworks* de *middleware* de IoT adequados para *edge Computing*, e recomendações específicas para a integração de tecnologias no enfrentamento dos desafios da Indústria 4.0.
- **Contexto:** *edge Computing* na Indústria 4.0

Para a definição da *string* de busca, foram selecionadas palavras-chave com base no modelo PICOC. Essas palavras-chave, juntamente com seus termos relacionados, foram organizadas e combinadas para formar uma string de busca específica para cada repositório, conforme descrito:

- **Scopus** : *TITLE-ABS-KEY (("edge computing"OR "fog computing"OR "edge-cloud computing"OR "cloud computing") AND ("industry 4.0") AND ("Iot"OR "internet of things") AND ("middleware"OR "software solutions"OR "Reference architectures"OR "Non-functional requirements"))*
- **IEEE Xplore**: *(("edge computing"OR "fog computing"OR "edge-cloud computing"OR "cloud computing") AND "industry 4.0"AND ("IoT"OR "internet of things") AND ("middleware"OR "software solutions"OR "reference architectures"OR "non-functional requirements"))*

Esta revisão utilizou como bases de dados a Scopus (ELSEVIER's, 2025) e IEEE *Xplore* (IEEE, 2025) por sua relevância e abrangência na área. A Scopus cobre ampla literatura científica revisada por pares, enquanto a IEEE Xplore é referência em Engenharia e Computação.

Para que os estudos identificados fossem adicionados a RSL, eles deveriam atender aos seguintes critérios de inclusão (CIN):

1. Revisões sistemáticas completas;
2. Considerar apenas a versão completa no caso de múltiplas versões do mesmo trabalho;
3. Artigos publicados de 2009 até Janeiro de 2025, quando a RSL foi executada;
4. Estudos que abordem características arquiteturais ou específicas;

Os estudos que não atenderam a esses critérios foram excluídos, juntamente com aqueles que se enquadravam nos critérios de exclusão (CE), listados a seguir:

1. Estudos que não estejam em inglês;
2. Estudos duplicados;
3. Estudos anteriores a 2009 e posteriores a Janeiro de 2025,=;

Os trabalhos aceitos foram avaliados com base em um formulário de qualidade composto por seis questões:

1. A fonte selecionada aborda de forma abrangente soluções de IoT com *Edge Computing* na Indústria 4.0?
2. A publicação selecionada é recente e relevante para as pesquisas atuais em IoT e *Edge Computing* na Indústria 4.0?
3. A fonte fornece dados precisos e detalhados sobre requisitos não-funcionais ou arquiteturas de referência relevantes para soluções de *Edge Computing*?
4. A fonte apresenta dados relevantes ou soluções arquiteturais para os requisitos não-funcionais de baixa latência ou escalabilidade?
5. A fonte selecionada fornece exemplos práticos ou estudos de caso que demonstrem como as soluções de software propostas ou a arquitetura de referência implementam ou atendem aos requisitos não-funcionais?

Os critérios desse formulário foram elaborados para medir a capacidade dos trabalhos de responder às questões de pesquisa deste estudo. Cada questão de qualidade pode ser respondida de três maneiras: "Sim"(1 ponto), "Parcialmente"(0,5 ponto) ou "Não"(0 pontos). Para prosseguir para a fase de extração de dados o trabalho aceito precisa alcançar três pontos de qualidade.

Após a avaliação de qualidade, os dados foram extraídos com base nos questionamentos apresentados a seguir.

1. A fonte apresenta soluções de software relevantes para o contexto?
2. Entre as soluções de software apresentadas pela fonte, existe algum *middleware* IoT ou *middleware* de *Edge Computing*? Se sim, quais?
3. Quais softwares apresentados pela fonte são passíveis de engenharia reversa?
4. Quais softwares apresentados pela fonte possuem documentação detalhada sobre as decisões arquiteturais tomadas?

5. A fonte apresenta um ou mais requisitos não-funcionais comuns para o contexto? Quais?
6. É possível identificar na fonte um ou mais requisitos não-funcionais comuns para o contexto? Quais?
7. A fonte apresenta ou propõe alguma solução que atenda a um requisito não funcional específico? Se sim, qual a solução? Para qual requisito não funcional?
8. A fonte selecionada propõe ou apresenta uma arquitetura de referência relacionada ao tema de pesquisa? Se sim, qual?
9. A fonte selecionada apresenta os requisitos não-funcionais que são atendidos pela arquitetura de referência proposta?
10. As arquiteturas de referência ou as soluções propostas/apresentadas pela fonte atendem aos requisitos não-funcionais de latência e escalabilidade?

As perguntas do formulário de extração de dados estão direcionadas à identificação de artefatos que serão utilizados no desenvolvimento da arquitetura de referência, considerando possíveis soluções de software, outras arquiteturas e atributos de qualidade frequentemente associados aos requisitos não-funcionais. Apesar de não ser o foco da RSL, requisitos funcionais foram também considerados.

2.4 Resultados da RSL

A Figura 2 apresenta, com mais detalhes, o fluxo de trabalho e os resultados alcançados.

No total foram encontradas 90 fontes pelas *strings* de busca, essas fontes são recentes e estão distribuídas entre os anos de 2017 e 2025. Conforme o gráfico da Figura 3, os anos de 2018, 2019 e 2022 foram os que apresentam o maior número de fontes. No entanto, apenas 36 fontes foram aceitas compor o estudo, sendo a maioria excluída na etapa de avaliação de qualidade por não atender às questões de pesquisa definidas. Entre as fontes aceitas, os anos om maior representatividade foram os anos de 2019 e 2022.

2.4.1 **RQ1**: Quais são os requisitos não-funcionais padrões em soluções de IoT que utilizam *Edge Computing* na Indústria 4.0?

Apenas oito requisitos não-funcionais foram mencionados por cinco fontes. Entre eles, os dois RFNs mais relevantes para este estudo, baixa latência e escalabilidade, além dos requisitos de interoperabilidade, segurança e privacidade.

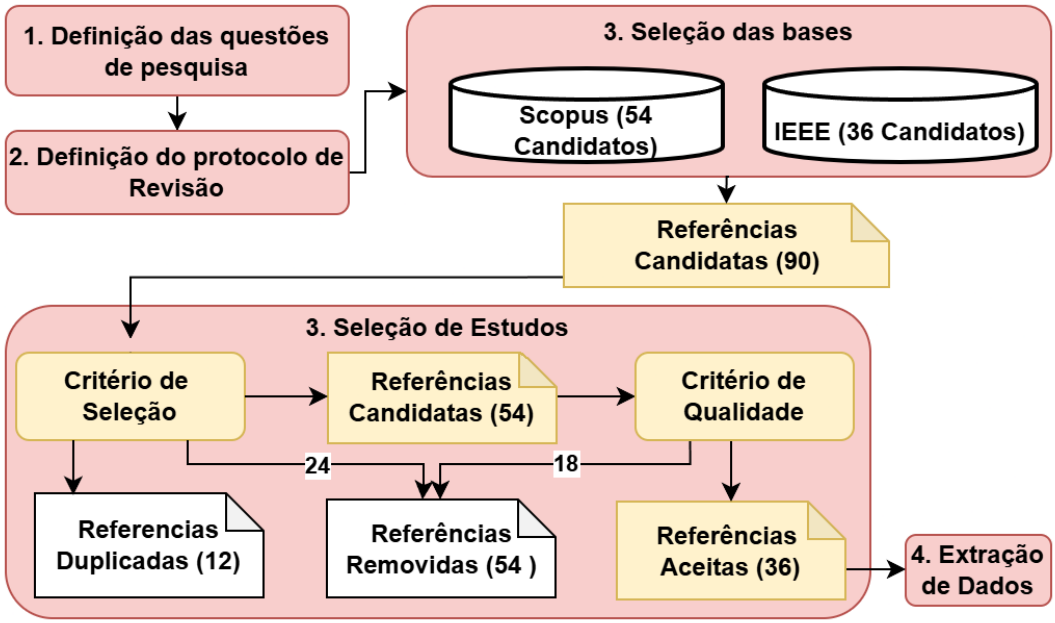

Figura 2 – Fluxo de trabalho definido para extrair e selecionar os artigos.

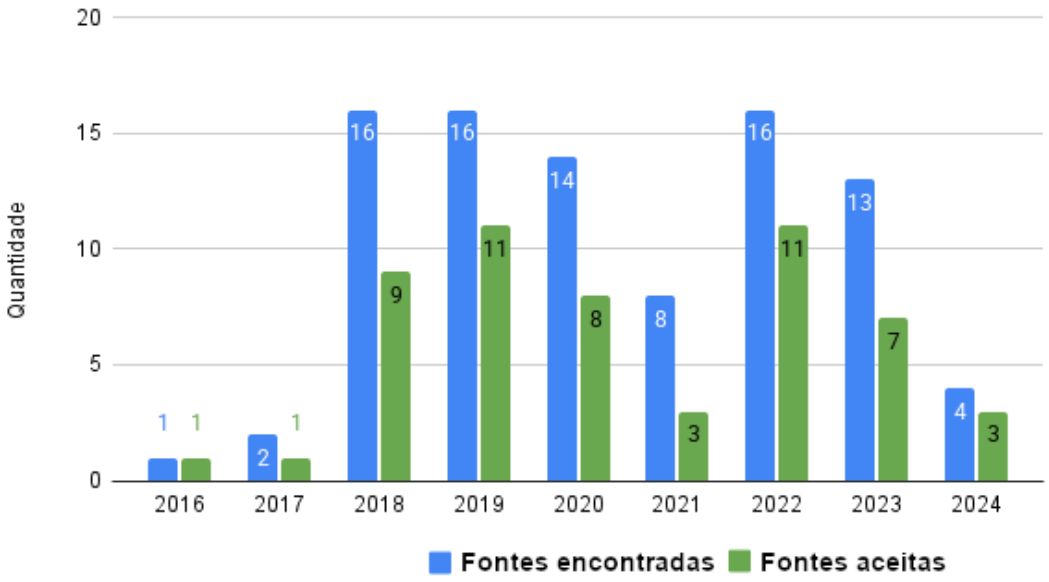

Figura 3 – Distribuição das fontes

Para responder esta questão de pesquisa e tornar os requisitos de qualidade testáveis, foram elaborados cenários de qualidade conforme o modelo SAAM (BASS; CLEMENTS; KAZMAN, 2012). Este modelo é composto por seis itens:

- Estímulo: Uma ação ou evento que impacta no sistema;
- Fonte do Estímulo: De onde o estímulo se origina;
- Ambiente: Estado do sistema ao receber o estímulo;

- Artefato: A parte do sistema que receberá o estimulo;
- Resposta: A resposta que o sistema apresentou ao estímulo;
- Medida da Resposta: Definição de como medir a resposta do sistema.

Mais detalhes sobre o modelo SAAM serão apresentados no Capitulo 3.

2.4.1.1 Interoperabilidade

Um dos requisitos de qualidade mais mencionados pelas fontes analisadas é a interoperabilidade. Esse fato se justifica porque, em ambientes de IoT industrial, dispositivos como câmeras, microfones, sensores e atuadores interagem com equipamentos específicos da indústria, tais como PLCs, robôs, sistemas de transporte, PCs industriais e diversas máquinas (VATER; HARSCHEIDT; KNOLL, 2019).

Além da variedade de dispositivos, as redes de IoT industrial precisam lidar com componentes provenientes de diferentes fornecedores, que também devem operar de forma integrada para alcançar um objetivo comum. Com isso, é essencial garantir uma comunicação eficiente e uniforme entre os dispositivos (CH. et al., 2018).

Com o intuito de viabilizar essa comunicação entre dispositivos heterogêneos, o trabalho de Ch. et al. (2018) se baseia em padrões de mercado, especificamente no protocolo MQTT (*Message Queuing Telemetry Transport*). Habib et al. (2022) e Barth, Balakrishna e Willner (2022) propõem soluções para interoperabilidade utilizando o protocolo OPC UA (*Open Platform Communication Unified Architecture*). No entanto, Aldalur et al. (2020) argumenta que a ampla diversidade e heterogeneidade dos sistemas industriais podem dificultar a adoção de um único padrão de mercado, tornando a integração um desafio complexo. Para contornar esse problema, o trabalho de Aldalur et al. (2020) propõe a utilização de *middlewares* IoT (por exemplo, FIWARE e Amazon AWS IoT) como estratégia para garantir a interoperabilidade do sistema.

Segundo Matevska e Soldin (2023), a utilização de *gateways* e *middlewares* em arquiteturas IoT é essencial para promover a interoperabilidade entre dispositivos em redes industriais, corroborando as conclusões de Aldalur et al. (2020). Com base nesse conceito, Matevska e Soldin (2023) propõe um modelo de *gateway* capaz de traduzir diferentes protocolos e formatos para um formato unificado. Assim, por exemplo, uma mensagem enviada por um dispositivo que utiliza MQTT é convertida para um formato ou protocolo comum da plataforma, o qual pode ser novamente traduzido para o protocolo e formato específico de outro dispositivo que necessite dessa informação. O *gateway* é projetado de forma extensível, permitindo que as traduções sejam realizadas de acordo com a heterogeneidade dos dispositivos presentes na rede.

Com a crescente popularização de dispositivos de processamento embarcado, soluções de software baseadas em *middlewares* e *gateways* tornam-se cada vez mais viáveis dentro do paradigma de *edge computing* (STRLJIC; VOLLMANN; RIEDEL, 2020). Um exemplo dessa abordagem é o *Middleware for IIoT platforms INTegration* (MIINT), proposto por Venanzi et al. (2021). Baseado em uma arquitetura de microsserviços, o MIINT promove a interoperabilidade entre plataformas de nuvem e dispositivos IoT, permitindo a leitura e interpretação de múltiplos protocolos. Essa abordagem facilita a comunicação entre sistemas heterogêneos e contribui para uma integração mais eficiente no contexto industrial (VENANZI et al., 2021).

A Tabela 1 apresenta um cenário de qualidade baseado nas questões sobre interoperabilidade encontradas nesta revisão.

Tabela 1 – Cenário de Qualidade para Interoperabilidade (SNE01)

Item	**Possíveis Valores**
Fonte	Comunicação entre dispositivos de diferentes protocolos (MQTT, OPC-UA, HTTP - Hypertext Transfer Protocol - e etc.)
Estímulo	Envio contínuo de dados e atuações entre dispositivos de protocolos distintos.
Artefato	Módulos de interoperabilidade, dispositivo de processamento local, câmeras, microfones, sensores, atuadores PLCs, robôs, sistemas de transporte, PCs industriais e entre outros.
Ambiente	Sistema em seu funcionamento normal e com variação de carga.
Resposta	A mensagem é interpretada corretamente por outro dispositivo.
Medida	O número de mensagens interpretadas deve se manter acima dos 99%.

2.4.1.2 Baixa Latência

A utilização da computação em nuvem em ambientes IIoT é mais adequada para tarefas que não possuem restrições rígidas de tempo, uma vez que o envio de grandes volumes de dados pode gerar problemas de largura de banda e latência (MUZELAK; SKOVRANEK, 2022). Para aplicações com restrições temporais mais rigorosas, a *edge computing* se torna essencial, pois esse conceito busca trazer os recursos de computação em nuvem para mais próximo da rede de dispositivos físicos, reduzindo assim a latência das respostas (ALDALUR et al., 2020).

Um exemplo da adoção da *edge computing* no ambiente industrial para melhorar a latência é apresentado em Muzelak e Skovranek (2022), que propõe um sistema baseado em visão computacional para supervisionar ambientes industriais e prevenir ou mitigar acidentes de trabalho. Nesse estudo, os autores utilizam um Raspberry Pi 4 como dispositivo de processamento local para executar modelos de aprendizado de máquina, eliminando a necessidade de processamento na nuvem para essa finalidade e, consequentemente,

reduzindo o tempo de resposta dos modelos. Destaca-se ainda que os dados processados são predominantemente multimídia, incluindo vídeos e imagens capturados diretamente da linha de produção. Esse modelo proposto por Muzelak e Skovranek (2022) difere dos cenários comuns de IIoT, nos quais os dados predominantes correspondem à telemetria de dispositivos (PLCs, sensores e atuadores), caracterizada por fluxos contínuos de séries temporais utilizados para monitoramento e controle operacional.

O trabalho de Sherlekar, Starly e Cohen (2019) também faz uso do Raspberry Pi como dispositivo de processamento na borda, porém com o objetivo de comprimir os dados antes de enviá-los para a nuvem. Essa abordagem reduz a carga de transmissão na rede e minimiza a latência nas respostas da nuvem.

Além da distribuição do processamento entre borda e nuvem, a escolha do protocolo de comunicação também impacta diretamente a latência dos sistemas industriais. Nesse contexto, Strljic, Vollmann e Riedel (2020) apresenta uma comparação entre diferentes tecnologias de comunicação, destacando os protocolos *Fieldbus*, OPC UA e gRPC (*Google Remote Procedure Call* - Chamada de Procedimento Remoto da Google), como os mais eficientes para aplicações que exigem baixa latência, seguidos por MQTT e pelo software Apache Kafka.

A Tabela 2 apresenta um cenário de qualidade para baixa latência baseado principalmente nos trabalhos de Sherlekar, Starly e Cohen (2019) e Muzelak e Skovranek (2022).

Tabela 2 – Cenário de Qualidade para Baixa latência (SNE02)

Item	**Possíveis Valores**
Fonte	Comunicação entre dispositivos
Estímulo	Envio contínuo de dados e atuações entre dispositivos
Artefato	Módulos de interoperabilidade, dispositivo de processamento local e outros dispositivos.
Ambiente	Sistema em seu funcionamento normal e com variação de carga.
Resposta	A mensagem é recebida pelos dispositivos de processamento da rede.
Medida	O latência máxima de 99,8% das mensagens deve ser 300ms (BARTH; BALAKRISHNA; WILLNER, 2022).

2.4.1.3 Escalabilidade

Um sistema de computação de borda pode operar com múltiplos dispositivos com capacidade de processamento, que se gerenciados corretamente, podem garantir a escalabilidade dos sistemas *edge*. Nesse contexto, é possível afirmar que a *edge computing* possui uma capacidade natural para escalabilidade. No entanto, por se tratar de uma rede composta por dispositivos físicos distribuídos em campo, a manutenção do sistema torna-se mais complexa em comparação com a *cloud computing* (MUZELAK; SKOVRANEK, 2022).

Embora a escalabilidade via software tenha suas limitações, em redes IoT a escalabilidade também envolve o aumento do número de dispositivos físicos. Dessa forma, um sistema de borda escalável deve facilitar a adição de novos dispositivos, tanto para atender a novos requisitos quanto para expandir a capacidade de processamento. Para garantir a escalabilidade da conectividade, tecnologias baseadas no modelo *pub/sub*, como o Apache Kafka, podem ser úteis, pois criam um desacoplamento entre a rede e os dispositivos (VATER; HARSCHEIDT; KNOLL, 2019). Ainda sobre o crescimento da rede, Matevska e Soldin (2023) reforça que o uso de *gateways* e *middlewares* representa um desafio, pois essas soluções devem ser capazes de suportar um aumento no número de dispositivos sem comprometer a qualidade dos serviços prestados.

Diante dessas limitações, Vater, Harscheidt e Knoll (2019) sugere que a cooperação entre *edge computing* e *cloud computing* pode ser uma solução eficaz para garantir a escalabilidade do software em redes IoT Industrial. No entanto, o trabalho carece de exemplos práticos. Já Nicholson et al. (2019) apresenta o conceito de *microcloud*, que propõe o uso de tecnologias de computação em nuvem em escala reduzida dentro da *edge computing*. Segundo os autores, organizações podem implementar múltiplas *microclouds* em diferentes fábricas e localidades para formar um ecossistema integrado. Seguindo essa abordagem, Nicholson et al. (2019) também apresentam o BRAIN-IoT, um sistema federado que permite a implantação, orquestração e monitoramento de aplicações distribuídas em redes IoT.

Baseados nas discussões supracitadas, a Tabela 3 apresenta um cenário de qualidade para escalabilidade.

Tabela 3 – Cenário de Qualidade para Escalabilidade (SNE03)

Item	**Possíveis Valores**
Fonte	Comunicação entre dispositivos, traduções de protocolo, tratamento de fluxos locais, execuções de modelos de Inteligência Artificial, armazenamento, consulta, agregações de dados e demais processamentos que possam ser delegados de forma extensível.
Estímulo	Execução simultânea de múltiplos processamentos, incluindo fluxos de dados, IA, consultas e armazenamento, variando a carga dinamicamente.
Artefato	Dispositivos de processamento local e plataforma da nuvem.
Ambiente	Sistema em seu funcionamento normal e com variação de carga.
Resposta	Os processamentos devem ser concluídos dentro do tempo esperado para a carga de trabalho definida e se necessário processos devem ser executados na nuvem.
Medida	A taxa de processamentos falhos ou com atraso deve ser inferior a 1%.

2.4.1.4 Segurança e Privacidade

A *edge computing*, quando adotada, tende a naturalmente trazer melhorias de segurança e privacidade, já que não há a necessidade de compartilhar os dados com meios externos para processá-los, evitando assim vazamento de dados e minimizando as oportunidades de ataques (BAZI et al., 2023). Alinhado a essa afirmação está o trabalho de Muzelak e Skovranek (2022), no qual os autores executam modelos de visão computacional na borda para supervisionar ambientes industriais. O vídeo em tempo real obtido pelas câmeras não é compartilhado com a nuvem. O compartilhamento acontece apenas quando um evento de anomalia é identificado e assim a imagem da anomalia e os dados obtidos são compartilhados. Segundo Aldalur et al. (2020) outra tecnologia que naturalmente traz ganhos consideráveis nos aspectos de segurança e privacidade é a conteinerização, já que a mesma proporciona um maior isolamento dos módulos. Nesse sentido, modelos de arquiteturas baseadas em microsserviços podem ser uma ótima opção, já que essa tecnologia possui uma forte integração com tecnologias de conteinerização e orquestração.

Apesar da segurança proporcionada pela *edge computing* através do não compartilhamento de dados, alguns trabalhos ressaltam a importância da adoção de outras tecnologias para aprimorar a segurança e privacidade. O trabalho de Kim, Lee e Jeong (2019) apresenta a importância das tecnologias de criptografia dentro das redes IoT. A pesquisa compara o nível de utilização de técnicas de criptografia em softwares de código aberto para IoT. Seguindo essa abordagem existem a arquitetura FORA, proposta em Barzegaran e Pop (2023), e a *Reference architectures for big data-intensive applications*, mencionada em Dintén, Martínez e Zorrilla (2024), que adotam o protocolo TLS (*Transport Layer Security*) na comunicação dos seus serviços essenciais. Já os trabalhos discutidos em Sittón-Candanedo et al. (2019) e Mirani et al. (2022) apresentam tecnologias de *blockchain* como uma ferramenta para garantir transparência, confiabilidade e auditabilidade dos dados.

Baseados principalmente nos trabalhos de Kim, Lee e Jeong (2019), Dintén, Martínez e Zorrilla (2024) e Barzegaran e Pop (2023), a Tabela 4 apresenta um cenário de qualidade com foco em autenticação e criptografia dos dados.

2.4.1.5 Manutenibilidade e disponibilidade

Os conceitos de *Edge computing* e sistemas distribuídos oferecem de forma intrínseca, suporte à melhoria de disponibilidade. Isso ocorre porque o sistema pode continuar processando dados localmente, mesmo em casos de falhas de conectividade com a nuvem (MIRANI et al., 2022)(BARLETTA; CINQUE; MARTINO, 2022). Nesse contexto, a Tabela 5 apresenta o cenário de qualidade relacionado à Disponibilidade frente à falhas.

A manutenção de sistemas de IoT pode ser complexa, já que geralmente envolve

Tabela 4 – Cenário de Qualidade para Segurança e Privacidade (SNE04)

Item	**Possíveis Valores**
Fonte	Processamento distribuído na borda e controle de acesso aos dados.
Estímulo	Acesso a dados sensíveis ou compartilhamento de informações entre dispositivos e sistemas externos.
Artefato	Mecanismos de autenticação, controle de acesso e criptografia implementados no *Edge Device* e dispositivos IoT.
Ambiente	Sistema operando normalmente, com dispositivos processando e compartilhando dados de forma distribuída.
Resposta	O acesso aos dados deve ser permitido apenas a entidades autorizadas, e a comunicação deve ocorrer de forma segura.
Medida	100% das tentativas de acesso não autorizadas devem ser bloqueadas, e a transmissão de dados deve ser criptografada (TLS 1.2 ou superior).

ações humanas que precisam ser realizadas em campo. Esse tipo de manutenção impacta diretamente a disponibilidade do sistema, pois tende a ser mais lenta que ações automatizadas. Por isso, minimizar manutenções em campo por meio de serviços automatizados torna-se essencial em redes IoT (VATER; HARSCHEIDT; KNOLL, 2019) (ALDALUR et al., 2020). Os conceitos de *edge computing* contribuem para a execução de serviços automatizados, já que os dispositivos na borda da rede são capazes de realizar processos complexos de manutenção. Seguindo esse contexto, a Tabela 6 apresenta um cenário de qualidade voltado à manutenção e à disponibilidade contínua do sistema.

Tabela 5 – Cenário de Qualidade para Disponibilidade após falhas (SNE05)

Item	**Possíveis Valores**
Fonte	Processamento distribuído na borda.
Estímulo	Falha na conectividade com a nuvem ou degradação da rede.
Artefato	Dispositivo de processamento local (*Edge Device*) e demais dispositivos IoT conectados.
Ambiente	Sistema operando normalmente, mas sujeito à falhas temporárias de conexão com a nuvem.
Resposta	O sistema deve continuar operando autonomamente, garantindo a execução das funções essenciais.
Medida	Pelo menos 99% das funções críticas devem permanecer operacionais durante falhas de conectividade com a nuvem.

2.4.2 **RQ2**: Quais *middlewares* ou softwares de IoT foram propostos na literatura para dar suporte ao *Edge Computing* na Indústria 4.0?

Dos estudos aceitos e qualificados, 30 mencionaram soluções de software. A maioria dessas soluções está relacionada à comunicação entre dispositivos (M2M) e entre dispositivos

Tabela 6 – Cenário de Qualidade para atualizações do sistema (SNE06)

Item	**Possíveis Valores**
Fonte	Dispositivos de borda com capacidade de processamento e gerenciamento de atualizações.
Estímulo	Execução de atualizações remotas de *firmware* e software, incluindo modelos de IA embarcados e aplicações personalizadas.
Artefato	Dispositivo de processamento local (*Edge Device*) e dispositivos IoT conectados a ele.
Ambiente	Sistema em funcionamento normal, com dispositivos recebendo atualizações sem interrupção crítica nos serviços.
Resposta	As atualizações são distribuídas corretamente para os dispositivos conectados, garantindo continuidade operacional.
Medida	Pelo menos 98% das atualizações devem ser concluídas com sucesso.

e plataformas. As soluções de comunicação encontradas englobam tanto protocolos quanto paradigmas de comunicação.

O protocolo MQTT foi a solução mais mencionada, com 12 citações. Outro protocolo frequentemente citado foi o HTTP, com 5 menções. Paradigmas de comunicação como OPC UA e Pub/Sub também foram destacados, sendo mencionados 10 e 4 vezes, respectivamente.

Entre os estudos que mencionam alguma solução de software, 14 deles apresentam ou propõem algum *middleware* IoT ou de *Edge computing*. Um desses estudos é apresentado no artigo de Salis et al. (2023) que propõe uma arquitetura de referência baseada nas arquiteturas *Reference Architectural Model Industry* 4.0 (RAMI 4.0) e *Industrial Internet Reference Architecture* (IIRA), que utiliza a FIWARE (FIWARE FOUNDATION E.V, 2024) como middleware IoT. A FIWARE, citada em Salis et al. (2023), e AWS IoT Core (AMAZON WEB SERVICES, 2024) são os *middlewares* mais citados entre os estudos, cada um com 4 citações. No entanto, apenas a FIWARE, por ser de código aberto, possui artefatos que podem ser utilizados para engenharia reversa, permitindo a extração de decisões arquiteturais.

2.4.2.1 *Open Platform Communication Unified Architecture* (OPC UA)

A OPC UA é uma arquitetura de comunicação projetada para a Indústria 4.0, que se baseia no paradigma de comunicação cliente-servidor. Proposta pela *OPC Foundation*, a OPC UA foi desenvolvida com o objetivo de estabelecer um padrão de comunicação entre componentes industriais dentro de uma rede M2M. Esse paradigma de comunicação busca a interoperabilidade em tempo real entre os componentes de um sistema, garantindo que diferentes dispositivos e sistemas possam trocar informações de forma eficiente e segura (HABIB et al., 2022).

A OPC UA implementa um modelo de informações integrado conforme a norma IEC 61131-3, que padroniza a programação de PLCs (VATER; HARSCHEIDT; KNOLL,

2019). Para que haja integração entre a OPC UA e outros domínios, é necessária a utilização de um protocolo de *middleware*, como o MQTT ou o REST (*Representational State Transfer*), um exemplo dessa prática é o estudo de Habib et al. (2022).

No contexto de *edge computing*, o servidor OPC UA pode ser integrado com *edge gateways* e/ou com dispositivos *edge* (VATER; HARSCHEIDT; KNOLL, 2019). De acordo com Illa e Padhi (2018), como a OPC UA é um *middleware* que facilita a integração entre o MES e os PLCs, sua integração com a *edge computing* pode permitir que o sistema tome decisões mais rapidamente, graças à baixa latência proporcionada pela computação de borda.

2.4.2.2 FIWARE

A FIWARE é uma implementação *open source* de padrões definidos pela *FIWARE Foundation*, que facilita o desenvolvimento de soluções inteligentes, portáteis e interoperáveis. A plataforma FIWARE foi projetada para suportar aplicações IoT em diversos contextos, como *Smart AgriFood, Smart Cities, Smart Energy, Smart Industry e Smart Water* (FIWARE FOUNDATION E.V, 2024).

O principal componente de qualquer solução que utiliza a FIWARE é o *FIWARE Context Broker Generic Enabler*, que fornece uma função fundamental de integração de dispositivos com sensores e atuadores. A comunicação entre os dispositivos pode acontecer diretamente com o *back-end* (*cloud*) da FIWARE, assim como pode ser intermediada por nós de *edge computing* (KIM; LEE; JEONG, 2019).

A representação da estrutura visual da arquitetura disponível na plataforma pode ser vista na Figura 4.

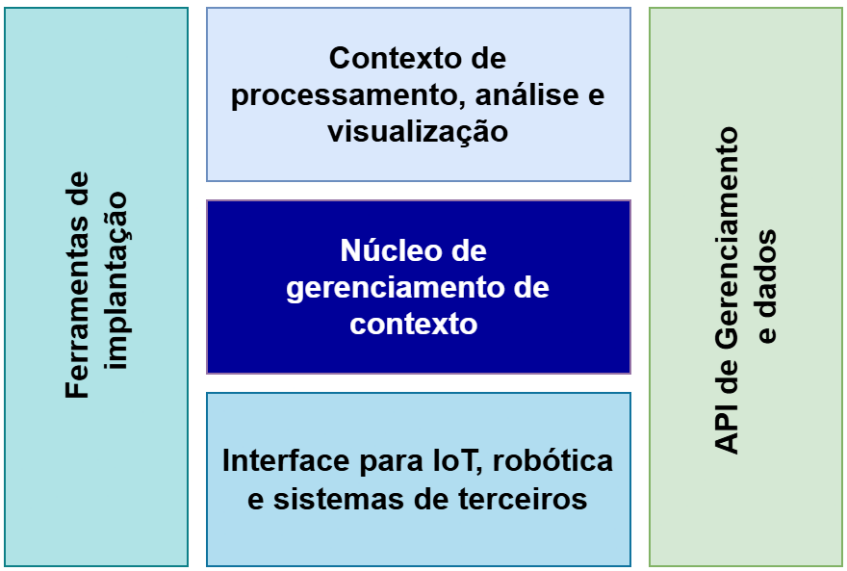

Figura 4 – Arquitetura FIWARE, adaptado de (FIWARE FOUNDATION E.V, 2024).

Conforme apresentado na Figura 4, a arquitetura da FIWARE é dividida em cinco partes principais. A seguir, uma breve descrição dos módulos de acordo com (FIWARE FOUNDATION E.V, 2024):

- Contexto de processamento análise e visualização - Este módulo se refere ao domínio específico, sendo responsável pelas funções de processamento do domínio, análise e monitoramento dos dados.
- Núcleo de gerenciamento de contexto - Este módulo fornece funções de gerenciamento de dispositivos e envio de comandos para a rede IoT.
- Interface para IoT, robótica e sistemas de terceiros - Este módulo opera como interface do sistema com os dispositivos IoT. É responsável pela padronização dos dados provenientes de dispositivos que utilizam diferentes protocolos de comunicação. Os componentes responsáveis por essa padronização são chamados de Agentes IoT (SALIS et al., 2023).
- Ferramentas de Implantação - Este módulo inclui as ferramentas de implantação, como Kubernetes (KUBERNETES, 2025), Docker (DOCKER, 2025) e Ansible (ANSIBLE, 2025).
- API de gereciamento e dados - Este módulo é a interface que fornece dados para outros sistemas que desejam se integrar com a FIWARE.

Os Agentes IoT são componentes de software responsáveis por converter protocolos e formatos de dados provenientes de dispositivos heterogêneos para o protocolo e o modelo de dados definidos pelo arquiteto durante o desenvolvimento da aplicação. Esse conceito, adotado na FIWARE, é semelhante ao apresentado no estudo Matevska e Soldin (2023), que propõe um modelo de *gateway* capaz de traduzir múltiplos protocolos para um formato unificado.

2.4.2.3 Node-RED

O Node-RED é uma ferramenta de programação baseada em fluxo, criada pela *Emerging Technology Services* da IBM e atualmente mantida pela *OpenJS Foundation* (NODE-RED CONTRIBUTORS, 2024). O Node-RED é amplamente utilizado em protótipos de IoT devido à sua portabilidade, permitindo sua implantação em diversos ambientes, como dispositivos embarcados, para o gerenciamento de fluxos de processos e dados (FERRARI et al., 2018).

2.4.3 **RQ3**: Quais arquiteturas de referência foram propostas para sistemas de software IoT com implementação em *Edge Computing* na Indústria 4.0?

Dos trabalhos aceitos e qualificados, 28 propõem ou apresentam arquiteturas de referência para Indústria 4.0. As mais citadas são a RAMI 4.0 e a IIRA, mencionadas em 10 e 7 trabalhos, respectivamente. Essas menções variam desde simples apresentações da RAMI 4.0 ou da IIRA, até propostas de novas arquiteturas baseadas em uma ou ambas. A Arquitetura Orientada a Serviços (SOA - *Service-oriented architecture*) foi mencionada em 6 trabalhos como base para arquiteturas com ênfase em Indústria 4.0 e/ou *Edge computing*. Um exemplo é o estudo apresentado em Vater, Harscheidt e Knoll (2019), que destaca a SOA como uma solução importante para o desacoplamento e a escalabilidade da arquitetura proposta no estudo.

A RAMI 4.0 foi desenvolvida pela Alemanha para promover e estruturar a Indústria 4.0. Dessa forma, ela atende às padronizações DIN SPEC 91345:2016 e IEC/PAS 63088:2017 que tem como o objetivo justamente a modernização dos processos de automação industrial. Esta RA considera os produtos como parte da rede industrial e promove a ideia de que as funções de um sistema de Indústria 4.0 devem ser distribuídas por toda a rede (MIRANI et al., 2022). Ao mesmo tempo, busca criar um desacoplamento entre seus componentes internos e fornecedores. Para atingir esse objetivo, adota uma estratégia baseada em SOA (AHELEROFF et al., 2021).

A RAMI 4.0 combina três tipos distintos de visões arquiteturais: a visão hierárquica, a visão em camadas e a visão do ciclo de vida (PEDONE; MEZGÁR, 2018). Na Figura 5 é possível ver as três visões arquiteturais representadas como dimensões de um cubo.

A seguir é apresentada a descrição dos níveis hierárquicos da RAMI 4.0, de acordo com Mirani et al. (2022):

- **Produto** - O produto é o resultado final da fabricação industrial;
- **Dispositivo de campo** - São os dispositivos que coletam dados do ambiente industrial, como atuadores e sensores;
- **Dispositivo de controle** - São os dispositivos que enviam comandos de controle para operação do sistema de manufatura, como PLCs e DCs (Dispositivos de Controle);
- **Estação** - É o local de onde os usuários administrativos monitoram o processo industrial;
- **Central de trabalho** - É onde são armazenados dados, informações e análises baseadas em percepções históricas do MES;

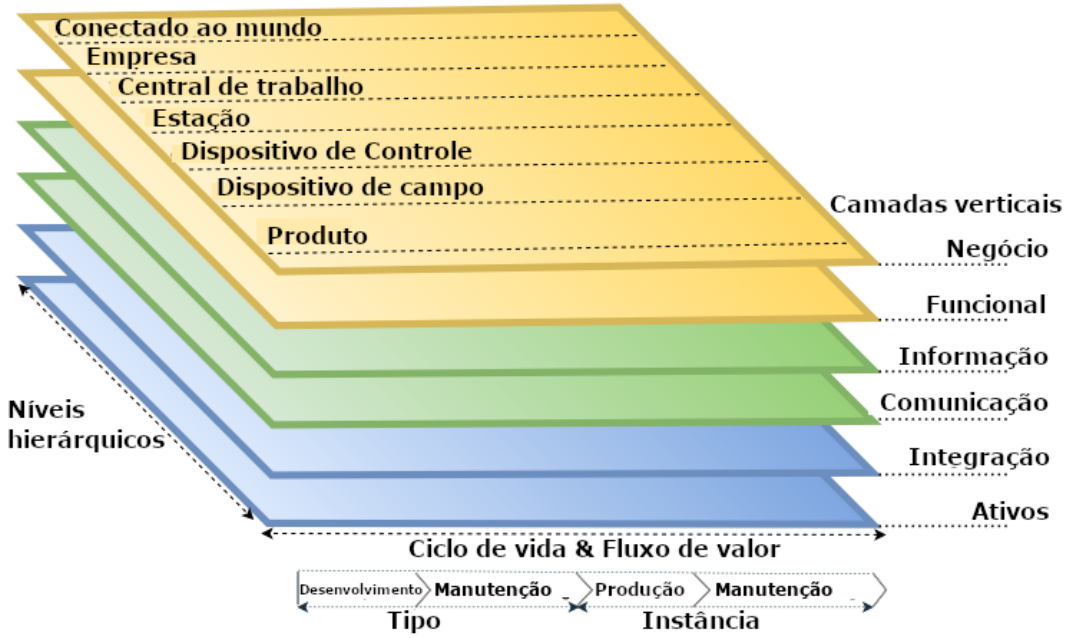

Figura 5 – Estrutura RAMI 4.0, adaptado de (MIRANI et al., 2022).

- **Empresa** - Este é o nível empresarial onde decisões comerciais são tomadas com base no acompanhamento da produção, pedidos, despesas e custos, além de gerenciar e planejar a fabricação;
- **Conectado ao mundo** - Através da internet, o sistema está conectado com toda a cadeia de suprimentos das indústrias externas.

Enquanto os níveis hierárquicos representam diversas funções de tecnologia de informação (TI) e controle corporativo, o eixo de ciclo de vida integra os ciclos de vida de produtos, máquinas, pedidos e fábricas, junto com o conceito de "fluxo de valor" da norma IEC 62890. As seis camadas verticais da RAMI 4.0 definem os componentes de TI empregados no sistema, como protocolos, aplicações de negócio e demais tecnologias (PEDONE; MEZGÁR, 2018). Segundo Mirani et al. (2022) as seis camadas verticais da RAMI 4.0 são:

- **Ativos** - Esta é a camada de mais baixo nível, onde estão presentes os componentes físicos de um ambiente industrial, como PLCs, sensores e atuadores.
- **Integração** - Esta camada fornece as informações geradas pelos ativos para as camadas superiores e permite o comando e controle desses ativos.
- **Comunicação** - Esta camada é responsável pela comunicação na rede, ou seja, pela interação entre os elementos das camadas inferiores e superiores através de padrões e protocolos.

- **Informação** - É a camada que garante a qualidade e a integridade dos dados recebidos das camadas inferiores e que fornece informações estruturadas para as camadas superiores.
- **Funcional** - Esta camada é responsável por tomar decisões com base na análise dos dados recebidos dos elementos da camada de ativos.
- **Negócio** - Esta é a camada mais superior, onde acontece o monitoramento em tempo real dos serviços através de interações dos usuários com as aplicações.

Os estudos de Sittón-Candanedo et al. (2019) e Sittón-Candanedo et al. (2020) apresentam as arquiteturas *"FAR-Edge RA"e "Industrial Internet Consortium RA"(ICC)* como arquiteturas de *Edge computing* para a Indústria 4.0.

Diferente da RAMI 4.0 que possui um escopo abrangente que inclui o negócio e os produtos, as arquiteturas IIRA e Far-Edge possuem um escopo mais técnico com ênfases em operações industriais, incluindo a manufatura.

A IIRA é uma arquitetura de referência aberta e de propósito geral, projetada para a Indústria 4.0. Ela se fundamenta nas padronizações ISO/IEEE/IEC 42010 e tem como objetivo central promover a interoperabilidade industrial (PEDONE; MEZGÁR, 2018). A IIRA incentiva o uso de *machine learning* para a análise de dados coletados em ambientes industriais, com o objetivo de otimizar os sistemas, fornecer *insights* valiosos e aprimorar os sistemas de controle (MIRANI et al., 2022).

A IIRA apresenta quatro visões arquiteturais. A primeira delas, denominada Empresarial, foca nas decisões e objetivos de negócio, concentrando-se em casos de uso específicos. A segunda visão, chamada de "Usabilidade", define como o sistema deve operar, baseada nos requisitos estabelecidos na camada "empresarial" (VILLAR et al., 2024).

A visão "Funcional" da IIRA (Figura 6) apresenta os blocos de funcionalidade de um Sistema Industrial Inteligente, denominados pela IIRA como domínios. Esses domínios são responsáveis por definir os fluxos de dados e controles da aplicação (PEDONE; MEZGÁR, 2018). A seguir, uma breve descrição dos blocos de funcionalidade de acordo com Mirani et al. (2022):

- **Domínio de Controle**: Neste domínio estão presentes as funcionalidades de um sistema de controle industrial, que inclui funções de sensoriamento, atuação e comunicação.
- **Domínio Operacional**: Este domínio contém as tarefas de gerenciamento e operação dos recursos de implantação para o domínio de controle.
- **Domínio de Informação**: Neste domínio estão concentradas as funções de tratamento, processamento, coleta e análise de dados do sistema.

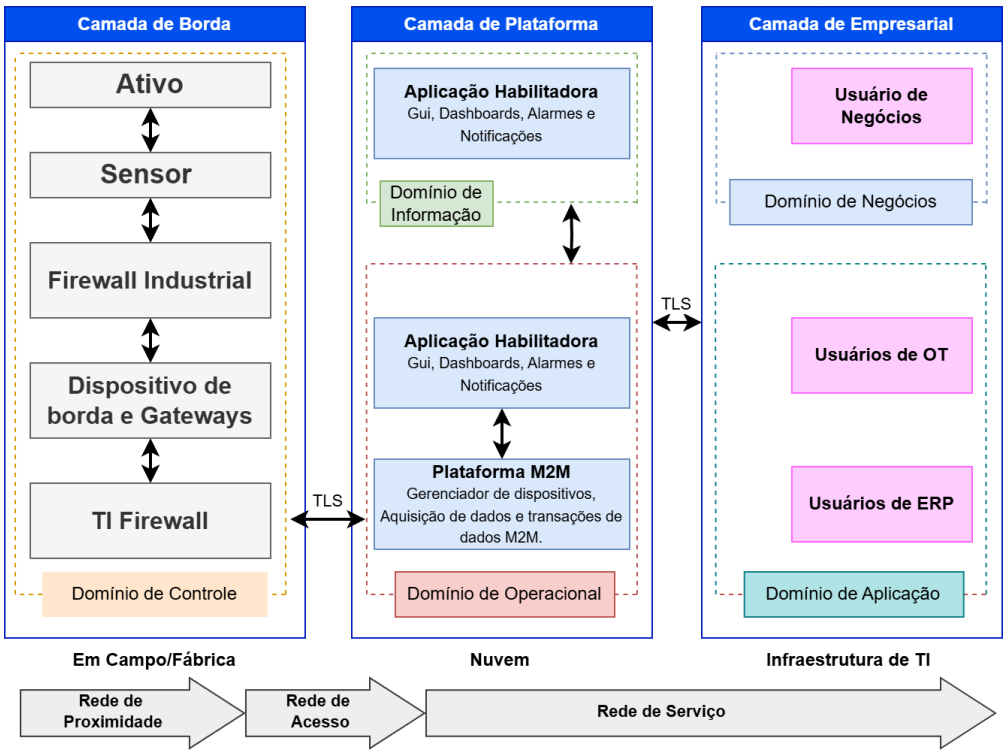

Figura 6 – Camadas IIRA, adaptado de (PEDONE; MEZGÁR, 2018).

- **Domínio de Aplicação**: Este domínio agrupa funções que implementam a lógica e as regras da aplicação. Exemplos de funções incluem interfaces APIs e interfaces de usuário.
- **Domínio de Negócio**: Este é o domínio onde estão presentes as funções de negócio, como pagamentos, ERPs e MES.

Segundo a IIRA, os detalhes tecnológicos para a implementação e integração dos componentes instanciados na visão Funcional são apresentados na visão de Implementação. Nessa visão estão incluídos os detalhes sobre os protocolos de comunicação, arquiteturas de nós computacionais e outras especificações de hardware e software (VILLAR et al., 2024).

A IIRA distribui as suas visões em três camadas de software: a camada de Borda, Plataforma e Empresarial. A seguir, uma breve descrição de cada uma delas, conforme Pedone e MezgÁr (2018):

- **Borda**: Camada responsável pela coleta dos dados dos nós da borda, incluindo sensores, *gateways* e outros tipos de dispositivos físicos, através de uma rede de proximidade.

- **Plataforma**: Nesta camada estão alocados os domínios de operação e informação. Os recursos desta camada incluem funções de gerenciamento, busca e análise de dados.
- **Empresarial**: Composta pelo domínio de negócio, esta camada abrange as aplicações específicas do negócio, além dos sistemas de ERPs e MES

A FAR-Edge RA é uma arquitetura de referência baseada em *Edge Computing* e Tecnologias de *Ledger* Distribuído (DLT - *Distributed Ledger Technology*), ou seja, tecnologias relacionadas à blockchain. Esta RA foi desenvolvida no âmbito do projeto *H2020 FAR-Edge*, como uma resposta ao desafio de criar arquiteturas de automação descentralizadas para cenários industriais. A Far-edge é baseada nos conceitos de escopos e camadas. Os escopos representam diversos elementos em uma fábrica como máquinas, dispositivos em campo e sistemas industriais de gestão e automação. Já as camadas oferecem uma separação lógica dos componentes e suas relações (SITTóN-CANDANEDO et al., 2020).

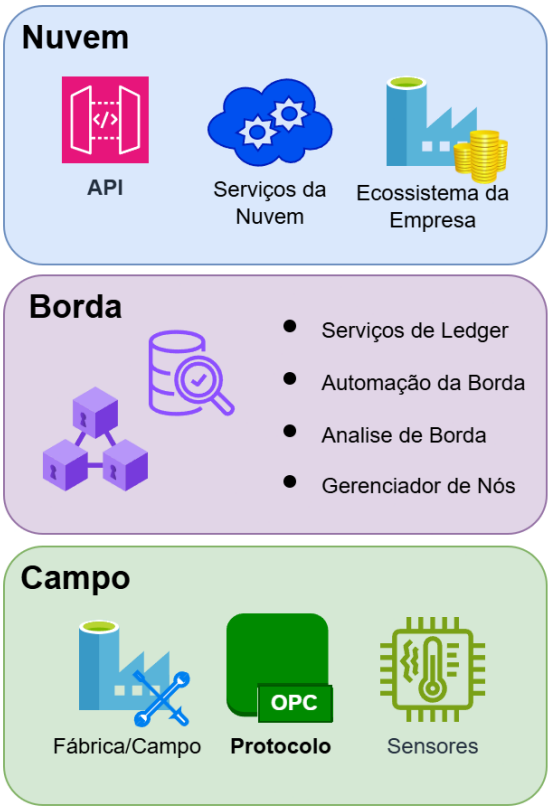

Figura 7 – FAR-Edge RA (SITTóN-CANDANEDO et al., 2020).

A Figura 7 ilustra as camadas da arquitetura FAR-Edge RA. A seguir, uma breve descrição das três camadas (SITTóN-CANDANEDO et al., 2020):

- **Campo** - Esta é a camada de nível mais baixo, representando os dispositivos físicos presentes em uma fábrica, como sensores, atuadores e máquinas.
- **Borda** - Esta camada contém os componentes de *Edge Computing*, como *edge gateways* e outros equipamentos que realizam processamento local próximo à fonte de dados.
- **Nuvem** - Esta camada representa os componentes de nuvem, incluindo servidores, serviços, *clusters* e outros recursos.

Enquanto a IIRA abrange o modelo de negócios das organizações por meio de sua camada "empresarial", a arquitetura Far-edge foca mais nos aspectos técnicos. A Far-edge possui uma camada específica para os elementos de campo, como sensores e atuadores, separando-os dos dispositivos mais inteligentes presentes na camada de "borda".

As camadas de "nuvem" da Far-edge e de "plataforma" da IIRA apresentam importantes semelhanças, pois ambas se concentram em elementos que operam na nuvem. O *middleware* FIWARE é um exemplo de tecnologia que pode ser incluída nessas camadas em uma aplicação real. No entanto, os elementos denominados Agentes IoT no FIWARE poderiam ser deslocados para a camada de "borda" dessas arquiteturas, visando reduzir a latência na comunicação com os dispositivos. Outros componentes de software analisados neste estudo, como o protocolo MQTT e o paradigma OPC UA, são posicionados na camada de "campo" da Far-edge.

2.5 Considerações Finais

Este capítulo apresentou toda fundamentação teórica do estudo desde elementos e termos mais básicos, como Industria 4.0, *edge computing*, *cloud computing* e entre outros, até artefatos mais específicos que foram identificados na RSL.

Por meio da RSL identificou-se que o principal motivo para a adoção de *Edge computing* na IoT Industrial é a redução das latências no processamento de fluxos críticos. Ao analisar as arquiteturas de referência identificadas na RSL, como Far-Edge e IIRA, observou-se que, apesar de preverem o uso de dispositivos de processamento na borda, há uma carência de definições sobre esse tipo de dispositivo. Questões relacionadas com responsabilidades, requisitos e usabilidade desse tipo de dispositivos se mantêm aberto. Dessa forma, esse estudo buscar através de uma arquitetura de referência preencher essa lacuna da literatura.

A aquisição de dados empíricos é uma das etapas propostas por Galster e Avgeriou (2011) para o desenvolvimento de novas arquiteturas de referência. Para esse fim, a RSL apresentada nesse capítulo, adquiriu informações preciosas para a produção e avaliação

de uma RA. Essas informações consistem em requisitos funcionais genéricos, requisitos não-funcionais, atributos de qualidade, cenários, soluções de software, protocolos de comunicação e outras arquiteturas de referência.

O próximo capítulo explora a metodologia adotada juntamente com os métodos utilizados por ela para produzir, apresentar e avaliar a RA proposta por esse estudo.

3 Metodologia de Desenvolvimento

3.1 Considerações Iniciais

O Capitulo 2 apresentou os conceitos básicos para essa pesquisa e uma RSL para identificar requisitos não funcionais comuns nessas soluções, com foco em latência e escalabilidade. Esses requisitos podem ser aplicados tanto na criação de uma RA preliminar quanto na melhoria de uma RA clássica. Além disso, a RSL visa obter artefatos como soluções de software e arquiteturas de referência para a Indústria 4.0 que integrem *Edge Computing*, auxiliando no desenvolvimento de arquiteturas clássicas.

Já neste Capítulo serão apresentadas as abordagens adotadas para o desenvolvimento da arquitetura de referência ED-operator RA. A principal abordagem adotada por este trabalho foi a apresentada em Galster e Avgeriou (2011), a qual define seis etapas para a produção de uma RA: (*i*) decisão sobre o tipo de arquitetura de referência, (*ii*) seleção da estratégia de design, (*iii*) aquisição empírica de dados, (*iv*) construção da arquitetura de referência, (*v*) incorporação de variabilidade na arquitetura e (vi) avaliação da arquitetura de referência.

3.2 Passo 01: Decisão do tipo de Arquitetura de Referência

Segundo a metodologia proposta em Galster e Avgeriou (2011), o primeiro passo busca definir o tipo de arquitetura que será produzida. Para classificar os tipos de RAs, três dimensões são utilizadas como base (Angelov, Grefen e Greefhorst (2009)):

- "Por que": refere-se à motivação para o desenvolvimento de RAs, que geralmente envolve facilitação ou padronizar softwares;
- "Onde": distingue entre RAs criadas para uma única organização ou para múltiplas organizações;
- "Quando": considera a temporalidade e divide as RAs em dois tipos: a clássica, que se baseia em conhecimento prévio proveniente de projetos implementados ou de outras RAs, e a preliminar, que é desenvolvida antes da implementação de um sistema real.

Utilizando as bases apresentadas, é possível definir cinco tipos de arquiteturas de referência:

1. Tipo 01: Arquiteturas clássicas de padronização a serem implementadas em várias organizações;

2. Tipo 02: Arquiteturas clássicas de padronização a serem implementadas em uma única organização;

3. Tipo 03: Arquiteturas clássicas de referência de facilitação para várias organizações projetadas por uma organização de software em cooperação com organizações de usuários;

4. Tipo 04: Arquiteturas clássicas de facilitação projetadas para serem implementadas em uma única organização;

5. Tipo 05: Arquiteturas preliminares de facilitação projetadas para serem implementadas em várias organizações.

A arquitetura de referência proposta por este estudo é uma arquitetura do Tipo 01, Arquiteturas clássicas de padronização a serem implementadas em várias organizações.

3.3 Passo 02: Seleção da Estratégia de Design

A dimensão "quando" influencia diretamente a estratégia de design, determinando se a arquitetura de referência será "criada do zero", em casos de RAs preliminares, ou "projetada com base em artefatos de arquiteturas existentes", no caso de arquiteturas clássicas (ANGELOV; GREFEN; GREEFHORST, 2009).

A RA proposta foi projetada com base em artefatos de arquiteturas existentes, identificados por meio de pesquisas e materiais previamente levantados. Dessa forma, trata-se de uma arquitetura orientada por pesquisa.

3.4 Passo 03: Aquisição Empírica de Dados

É neste passo que ocorre a coleta de dados que vão fundamentar a construção da arquitetura de referência. O objetivo é reunir evidências provenientes de fontes teóricas e práticas que permitam identificar soluções arquiteturais recorrentes, componentes, responsabilidades, relações e restrições relevantes ao domínio de aplicação. A aquisição dos dados é conduzida a partir de métodos empíricos como revisões sistemáticas da literatura, análise de arquiteturas existentes, estudos de caso, documentação técnica e padrões. Os resultados obtidos nesta etapa são utilizados como insumos diretos para as fases subsequentes de modelagem e definição da arquitetura de referência.

Para a produção da arquitetura de referência proposta por esse estudo foram utilizados os artefatos encontrados na RSL apresentada no Capítulo 2, que incluem requisitos não-funcionais, tecnologias (MQTT, OPC UA, pub/sub e Node-RED), e as

arquiteturas FIWARE, IIRA e Far-edge. Além dos tipos de dados empregados nesse tipo de aplicação, como dados de telemetria de equipamentos e imagens.

3.5 Passo 04: Construção da Arquitetura de Referência

Para a construção de arquiteturas de referência, o quarto passo propõe a adoção dos conceitos estabelecidos pela norma ISO/IEC/IEEE 42010:2011 (ISO/IEC/IEEE, 2011). Contudo, essa norma foi revisada e publicada em 2022 como ISO/IEC/IEEE 42010:2022 (ISO/IEC/IEEE, 2022).

Apesar da atualização, os conceitos fundamentais de *viewpoints* e *architectural views*, discutidos por Galster e Avgeriou (2011), foram mantidos e aprimorados. Assim, esta pesquisa adota a versão mais recente da norma.

Os *view points* orientam a modelagem da arquitetura, garantindo que diferentes preocupações dos *stakeholders* sejam consideradas. Já as *architectural views* são representações derivadas desses *viewpoints*, fornecendo diferentes formas de compreender a arquitetura sob ângulos distintos, como estrutura, comportamento, desempenho e implementação (ISO/IEC/IEEE, 2022).

Para guiar o desenvolvimento das *architectural views* da RA proposta, foi utilizado o modelo C4 que, define quatro níveis de abstração: Contexto, Contêiner, Componente e Código, cada um representando uma *architectural views* que facilita a comunicação da arquitetura para diferentes públicos (BROWN, 2011).

Embora o modelo C4 não defina formalmente *viewpoints*, seus níveis de abstração naturalmente podem ser correlacionados com esse conceito (BROWN, 2011). Por possuir um foco técnico e uma abordagem simplificada, o modelo C4 foi adotado no desenvolvimento da Arquitetura de Referência proposta. Essa escolha se justifica pelo perfil da própria RA, que possui um escopo técnico amplo e não se restringe a domínios específicos dentro do contexto da IoT Industrial.

3.6 Passo 05: Permitir Variabilidade na Arquitetura

As arquiteturas de referência são produzidas com o intuito de serem utilizadas como base para o desenvolvimento de outras arquiteturas, que podem abranger diferentes domínios, contextos e tecnologias. Nesse sentido, o quinto passo da abordagem busca incorporar elementos que permitam a variabilidade na arquitetura proposta.

Segundo Galster e Avgeriou (2011), existem três formas de possibilitar a variabilidade em uma RA:

1. Anotação de RA - Refere-se à marcação de elementos dentro dos modelos de ar-

quitetura com informações relacionadas à variabilidade. Pode ser feito por meio de atributos, marcações ou regras, que permitam a identificação de quais partes da arquitetura são variáveis e sob quais condições;

2. Variabilidade dos modelos - São utilizados para representar explicitamente os pontos de variação e suas dependências dentro de uma RA. Eles podem ser aplicados em diferentes visões arquiteturais para indicar como e onde ocorrem as variações no sistema.

3. Variabilidade das visões - São visualizações específicas que focam nos aspectos de variabilidade da arquitetura.

Essas três formas podem ser utilizadas separadamente ou em conjunto. Originalmente, o método proposto por Galster e Avgeriou (2011) sugere o uso da norma ISO/IEC/IEEE 42010:2011 (ISO/IEC/IEEE, 2011) para auxiliar nessa etapa, uma vez que a norma permite tratar aspectos que promovem variabilidade em uma arquitetura como preocupações. Dessa forma, é possível definir visões específicas que considerem essas características. Essa abordagem permanece válida na ISO/IEC/IEEE 42010:2022 (ISO/IEC/IEEE, 2022), uma vez que os conceitos centrais de preocupações, pontos de vista e visões arquiteturais foram mantidos. Assim, como no passo anterior, esta pesquisa adota a versão mais recente da norma.

3.7 Passo 06: Avaliar a Arquitetura de Referência

O sexto passo consiste na avaliação da arquitetura de referência proposta, a qual pode ser conduzida por meio de um *checklist* que verifica aspectos fundamentais da arquitetura, ou através de testes aplicados a protótipos baseados na arquitetura que deseja-se avaliar. O trabalho Bass, Clements e Kazman (2012) apresenta três métodos de avaliação de arquitetura:

- ATAM: *Architecture Tradeoff Analysis Method* - É um método qualitativo e colaborativo que envolve diversos participantes, como arquitetos, *stakeholders* e avaliadores. Seu foco está na análise dos atributos de qualidade da arquitetura (como desempenho, escalabilidade, segurança, entre outros) e na identificação dos *trade-offs* existentes entre esses atributos.

- SAAM: *Software Architecture Analysis Method* - O objetivo é avaliar as arquiteturas de software com base em cenários de uso, focando na capacidade da arquitetura de atender aos requisitos de qualidade. Esse método é separado em seis etapas principais:

1. **Desenvolvimento de cenários**: Etapa em que são elaborados os cenários de qualidade, descrevendo as funcionalidades relevantes e os requisitos da arquitetura;
2. **Descrição da arquitetura**: Envolve o desenvolvimento, apresentação e documentação da arquitetura proposta. Esta etapa geralmente ocorre de forma interativa e incremental, em paralelo com a elaboração dos cenários;
3. **Priorização dos cenários**: Os cenários são classificados com base no suporte direto ou indireto fornecido pela arquitetura, e posteriormente priorizados;
4. **Avaliação individual dos cenários**: Cada cenário priorizado é analisado em relação à sua aderência e suporte pela arquitetura;
5. **Avaliação das interações entre cenários**: Visa analisar como os cenários interagem entre si, sendo fundamental para a identificação de possíveis *trade-offs* entre os atributos de qualidade;
6. **Geração da avaliação geral**: As informações obtidas nas etapas anteriores são consolidadas, podendo-se atribuir pontuações a cada cenário. Estas pontuações possibilitam comparações futuras entre diferentes arquiteturas.

- ARID: *Active Reviews for Intermidiate Designs* - Este método baseia-se em revisar e validar os designs intermediários de software, focando em aspectos práticos do projeto que já estão em andamento.

Neste trabalho, optou-se pela adoção do método SAAM, por ser o mais simples entre os apresentados e por não demandar a participação de múltiplos avaliadores. Outro fator determinante foi a facilidade de integração do SAAM com a abordagem de prototipação adotada.

No contexto deste trabalho, a primeira etapa foi conduzida durante o levantamento bibliográfico descrito no Capítulo 2, no qual os cenários de qualidade são apresentados. A segunda etapa será detalhada no Capítulo 4. As etapas subsequentes serão realizadas com o suporte de um experimento baseado em um protótipo da arquitetura, cujo detalhamento será apresentado no capítulo dedicado à avaliação dos cenários.

3.8 Considerações Finais

Este capítulo apresentou a metodologia adotada no desenvolvimento da arquitetura de referência ED-operator, bem como os conceitos necessários para sua compreensão completa, como *viewpoints*, *architectural views*, modelo C4 e o método de avaliação SAAM. Além disso, foram discutidos os tipos de arquiteturas existentes, classificando a arquitetura ED-operator como do Tipo 01 — arquiteturas clássicas de padronização.

O Capítulo 4 apresentará os resultados dos passo 4 (Construção da Arquitetura de Referência) e 5 (Permitir Variabilidade na Arquitetura). Em outras palavras, será descrita a arquitetura de referência proposta por este estudo.

4 Arquitetura de Referência ED-operator RA (EDGE Operator Reference Architecture)

4.1 Considerações Iniciais

Este capítulo apresenta os resultados obtidos com a aplicação dos passos 4 (Construção da Arquitetura de Referência) e 5 (Permitir Variabilidade na Arquitetura) da metodologia de desenvolvimento descrita no Capítulo 3. A primeira seção trata da análise das informações mais relevantes obtidas ao longo do estudo. Em seguida, são apresentadas as visões arquiteturais e o capítulo é finalizado com o racional da arquitetura.

O racional da arquitetura aborda, de forma concisa, as principais decisões tomadas durante o desenvolvimento da proposta, bem como suas respectivas justificativas.

4.2 Análise do contexto

Ao analisar as arquiteturas de referência identificadas na RSL, como Far-Edge e IIRA, bem como os estudos apresentados em (MATEVSKA; SOLDIN, 2023), (STRLJIC; VOLLMANN; RIEDEL, 2020), (VENANZI et al., 2021) e (ALDALUR et al., 2020), observou-se que, embora prevejam o uso de dispositivos de processamento na borda, há uma carência de definições sobre esse tipo de dispositivo. Esses artefatos especificam quais elementos podem estar presentes nas redes IoT com processamento de borda, mas não se aprofundam em como implementá-los na prática, o que deixa margem para diferentes interpretações e abordagens personalizadas. Nesse sentido, a arquitetura proposta por este trabalho foi baseada na utilização de dispositivos dedicados exclusivamente ao processamento dentro de uma rede IoT Industrial.

A RSL apresentada no Capítulo 2 identificou que o principal motivo para a adoção de *Edge computing* na IoT Industrial é a redução da latência no processamento de fluxos críticos. No entanto, a implementação de *edge computing* com dispositivos exclusivos para processamento também pode permitir que outras funções essenciais para a IoT Industrial, como a interoperabilidade entre diferentes tecnologias e protocolos, ocorram na borda. Um exemplo é o deslocamento dos agentes IoT — módulos de software da FIWARE, específicos para a interoperabilidade — para a borda.

O deslocamento dos agentes IoT da nuvem para a borda permite que, ao contrário do que ocorre em arquiteturas como AWS IoT e FIWARE padrão, a tradução de protocolos seja realizada diretamente no ambiente de campo, em vez de depender de processamento

remoto na nuvem. Isso significa que um sensor que opera em OPC UA pode se comunicar diretamente com um atuador MQTT sem precisar enviar os dados para um servidor central antes da conversão, tornando a comunicação mais rápida e eficiente.

Com o processamento ocorrendo na borda, a necessidade de compartilhamento de dados com a nuvem diminui drasticamente, impactando diretamente na diminuição do tráfego de rede e na demanda por armazenamento. Além disso, essa abordagem proporciona ganhos significativos em termos de privacidade e segurança dos dados. A utilização de dispositivos exclusivos para processamento na rede permite a implementação de sistemas de autorização mais robustos para o compartilhamento de informações, como apresentado em Nicholson et al. (2019).

Outro benefício do processamento distribuído na borda é a continuidade operacional do sistema mesmo em situações de indisponibilidade da conectividade com a nuvem. Em setores como a manufatura, essa autonomia pode ser essencial para garantir o funcionamento contínuo do sistema, mesmo sob condições adversas.

Apesar de a pesquisa focar nos requisitos não funcionais, alguns requisitos funcionais genéricos orientaram a construção da arquitetura de referência proposta, como aquisição de dados industriais, envio de atuações da nuvem para a rede IoT, operação autônoma, tradução de protocolos e formatos e intermediação da comunicação entre dispositivos.

Outro requisito funcional relevante para a construção da arquitetura é a capacidade de realizar atualizações remotas de implantação, permitindo a adoção de *pipelines* de Integração Contínua (CI — *Continuous Integration*) e Entrega Contínua (CD — *Continuous Delivery*) na borda. Isso facilita a implantação e a manutenção de modelos de IA embarcados e de aplicações personalizadas sem a necessidade de intervenção manual. Além de suportar suas próprias atualizações, esse tipo de dispositivo pode atuar como intermediário na atualização de outros dispositivos IoT com menor capacidade computacional, gerenciando a distribuição de *firmware* e software para sensores e atuadores conectados, funcionando como um *hub* de atualizações na rede local.

4.3 Visão de Contexto

Seguindo o modelo C4 proposto em Brown (2011), o primeiro ponto de vista arquitetural é o de contexto, voltado tanto para o público técnico quanto não técnico, dentro e fora da equipe de desenvolvimento de software.

A arquitetura proposta concentra-se na definição dos recursos necessários no dispositivo de processamento, mas, em seu contexto, estão presentes as plataformas de IoT de nuvem. Independentemente da plataforma usada, seja ela AWS IoT ou FIWARE, o sistema do dispositivo deve funcionar da mesma maneira.

Além do sistema do dispositivo de borda e da plataforma IoT, os usuários do sistema IoT e os dispositivos são componentes presentes no contexto da arquitetura. A Figura 8 apresenta a visão arquitetural do ponto de vista de contexto.

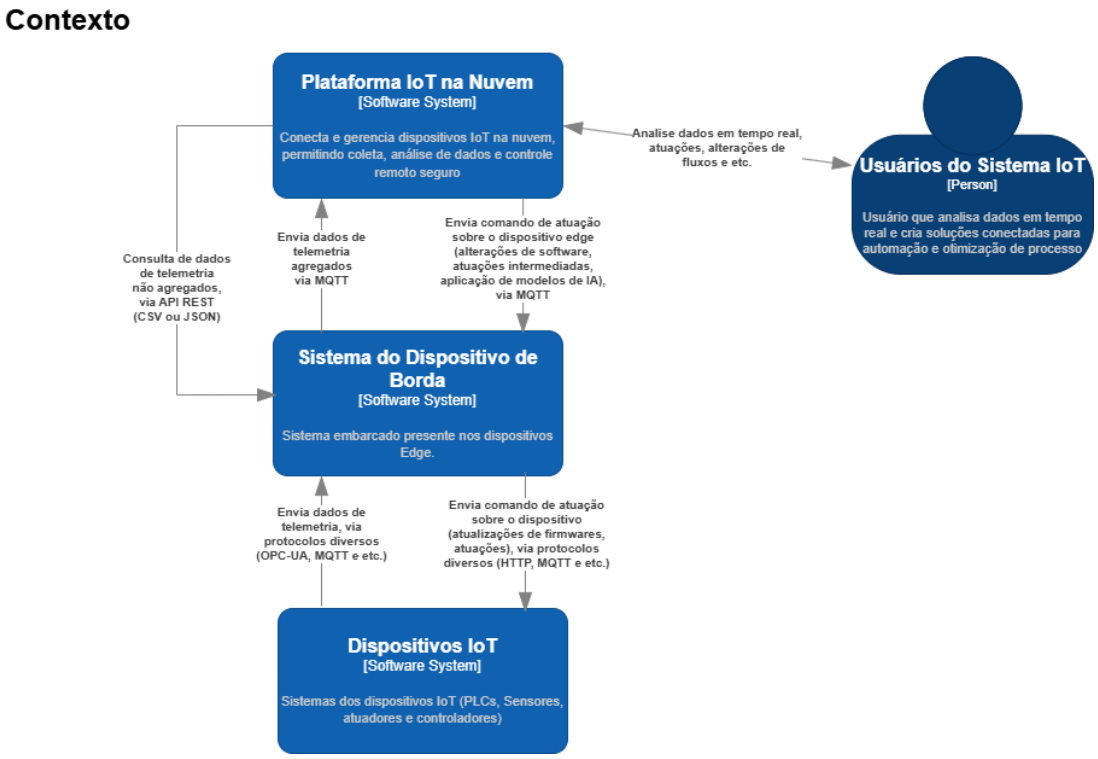

Figura 8 – Contexto da Arquitetura de Referência

Os dispositivos IoT devem se comunicar primeiro com o sistema de borda, independentemente do protocolo utilizado. Esse sistema será responsável por processar os dados localmente e enviá-los, de forma agregada, para a nuvem, assim que possível. Uma vez armazenados na nuvem, os dados poderão ser acessados pelo usuário. Caso o usuário necessite consultar dados históricos não agregados, a plataforma na nuvem poderá acessar diretamente a API do dispositivo de borda.

4.4 Visão de contêineres e módulos

Essa visão arquitetural tem como público-alvo profissionais técnicos, tanto internos quanto externos à equipe de desenvolvimento de software, incluindo arquitetos de software, desenvolvedores e equipes de operações e suporte.

Essa arquitetura foi baseada no modelo de microsserviços, pois esse modelo possui uma sinergia com tecnologias de conteinerização e orquestração. Essas tecnologias são essenciais para garantir flexibilidade na manutenção, atualização e customização do sistema de forma remota, com segurança e robustez (ALDALUR et al., 2020). Dessa forma, os contêineres apresentados nessa visão também representam, parcialmente, os módulos do

sistema como um todo. A Figura 9 apresenta a visão arquitetural do ponto de vista dos contêineres.

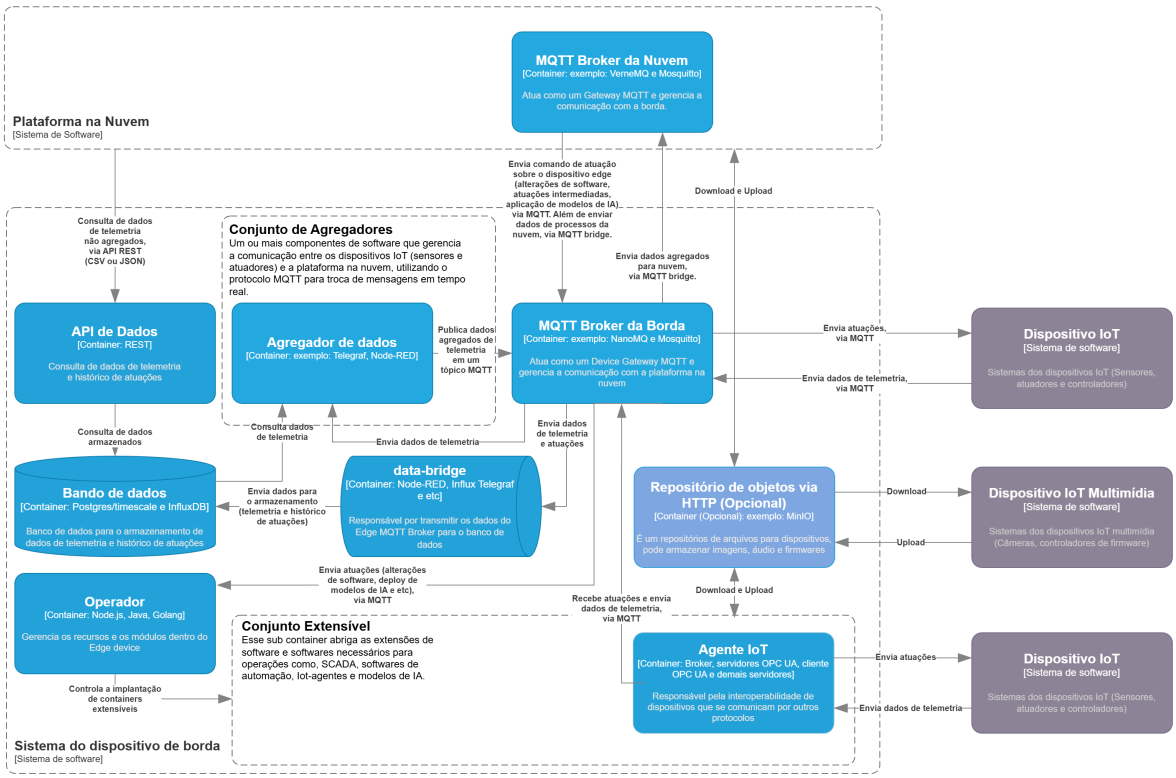

Figura 9 – Contêineres da Arquitetura de Referência

O sistema do dispositivo de processamento não é apenas um *gateway* IoT como geralmente é utilizado neste tipo de sistema, ele é uma *microcloud* dentro da rede IoT como apresentado em Nicholson et al. (2019). Dessa forma, este sistema pode conter diferentes contêineres para a prestação de vários tipos de serviços necessários dentro do domínio em que o sistema está inserido. No entanto, alguns contêineres são básicos e deverão estar presentes em todas as implementações desta arquitetura. Uma breve descrição dos contêineres básicos é apresentada a seguir:

- **MQTT *Broker* da Borda** - Este contêiner é um *gateway* IoT, sendo responsável por padronizar a comunicação do dispositivo de processamento de borda com a plataforma de nuvem, além de estabelecer a comunicação com os dispositivos IoT que utilizam o protocolo MQTT;
- **MQTT *Broker* da Nuvem** — Este contêiner está localizado na nuvem e atua como receptor das mensagens provenientes da borda por meio do modo *bridge* do MQTT. Além disso, é responsável pelo envio de comandos e dados gerados por processos executados no ambiente em nuvem;

- **Agente IoT** - O Agente IoT é um conceito importado da arquitetura da FIWARE, que possui similaridade com o conceito de *gateway* de Matevska e Soldin (2023). Tanto os agentes quanto os *gateways* são responsáveis por traduzir comunicações em diferentes protocolos para o protocolo padrão dos sistemas nos quais são implementados. No caso da arquitetura proposta neste trabalho, os Agentes-IoT são serviços adicionais que fornecem tradução de diversos protocolos para o protocolo MQTT;

- **Data-bridge** - É um software de ETL (Extração, Transformação e Carregamento - *Load*) responsável por captar os dados de telemetria dos dispositivos que são trafegados pelo MQTT *Broker* e armazenar no banco de dados do dispositivo de processamento através ou não de uma regra estabelecida;

- **Banco de dados** - Este contêiner é responsável por armazenar os dados de telemetria dos dispositivos que são trafegados pelo MQTT *Broker*, com o objetivo de criar um histórico de dados. A captação de dados de telemetria provenientes de equipamentos e dispositivos (PLCs, sensores e câmaras injetoras) é uma característica típica da IIoT. Dessa forma, a arquitetura recomenda a utilização de um banco de dados capaz de suportar o armazenamento de séries temporais.

- **Agregadores de dados** - Representa o conjunto de contêineres responsáveis por ler os dados do MQTT *Broker*, do *data bridge* ou do banco de dados, com o objetivo de agregá-los. Após a agregação, esses dados são enviados para o armazenamento na nuvem. A agregação será realizada conforme as regras definidas pelo domínio do sistema. Como exemplo, pode-se citar um sistema de controle de temperatura que monitora, segundo a segundo, a temperatura de um ambiente, mas envia à nuvem apenas métricas agregadas, como média móvel, desvio padrão e demais indicadores estatísticos por minuto. Outro exemplo é um sistema que realiza consultas periódicas e, na maior parte do tempo, recebe valores inalterados, transmitindo para a nuvem apenas os pontos de mudança relevantes.

- **API de dados** - Uma API, preferencialmente no padrão REST, que permite a consulta dos dados de telemetria armazenados no banco de dados por outros sistemas, como as plataformas de IoT na nuvem. Além disso, deve permitir a exportação de dados para serviços de *backup* que possam existir dentro do sistema;

- **Bloco extensível** - O bloco extensível representa os contêineres que não são parte da arquitetura básica e que podem ser adicionados conforme o domínio da aplicação. Neste bloco, estão incluídos os Agentes-IoT, que são, por natureza, serviços extensíveis sob demanda, além de possíveis modelos de IA, os quais devem ser empacotados como contêineres para seu correto funcionamento;

- **Operador** - Este é o microsserviço responsável por controlar todo o sistema do dispositivo de processamento de borda, incluindo a implantação e atualização dos

microsserviços básicos e customizados. Serão apresentados mais detalhes sobre esse componente na próxima seção.

- **Repositório de objetos via HTTP** - Este recurso é responsável por armazenar os arquivos gerados pelos dispositivos multimídia, como vídeos, imagens, áudios e *datalogs*. Os uploads e downloads de arquivos devem ser realizados por meio do protocolo HTTP. Além disso, o repositório pode ser utilizado para armazenar imagens de contêineres, logs e outros tipos de arquivos que possam demandar alguma iteração com a nuvem. Caso a rede IoT local não possua dispositivos multimídia, esse componente pode não ser implantado. Da mesma forma, caso existam dispositivos que não operem sobre o protocolo HTTP, o conceito de Agente IoT pode ser adotado.

4.5 Visão de Implantação

A visão de implantação, representada na Figura 10, consiste em um modelo estático que descreve como os contêineres serão implantados no dispositivo de processamento de borda. O público-alvo desta visão arquitetural é composto por desenvolvedores, arquitetos de software e profissionais de infraestrutura.

O dispositivo deve estar conectado tanto à rede da plataforma em nuvem quanto à rede local IoT. A rede da nuvem é pública e precisa ser protegida. Toda comunicação com a nuvem, seja via *MQTT Broker* ou pela API de dados, deve ocorrer por meio de conexões criptografadas, como conexões TLS ou similares, permitindo acesso apenas a sistemas externos confiáveis. Em uma rede IoT pode haver dispositivos com baixa capacidade de processamento que são incapazes de lidar com técnicas de criptografia. Dessa forma, para que os dados desses dispositivos não sejam expostos, a rede onde os dispositivos se encontram (representada na Figura 10 pelo bloco Rede Local IoT) deve ser privada.

4.5.1 Operador e Orquestrador

Os contêineres do sistema no dispositivo de processamento de borda devem ser implantados de forma controlada por um orquestrador, de modo a garantir a escalabilidade do sistema até o limite do hardware disponível. A implantação de customizações e a atualização dos microsserviços serão realizadas por meio de um Operador, que escuta eventos do sistema e aciona as medidas apropriadas a serem executadas pelo Orquestrador.

Essa separação entre Operador e Orquestrador é essencial para garantir a manutenibilidade do sistema. O Orquestrador atua no nível do sistema operacional, o que torna processos como a atualização remota de software e a recuperação após falhas de atualização mais complexos e sensíveis. Ao separar as funcionalidades em dois componentes e manter a lógica principal do sistema concentrada no Operador, que é executado em um contêiner,

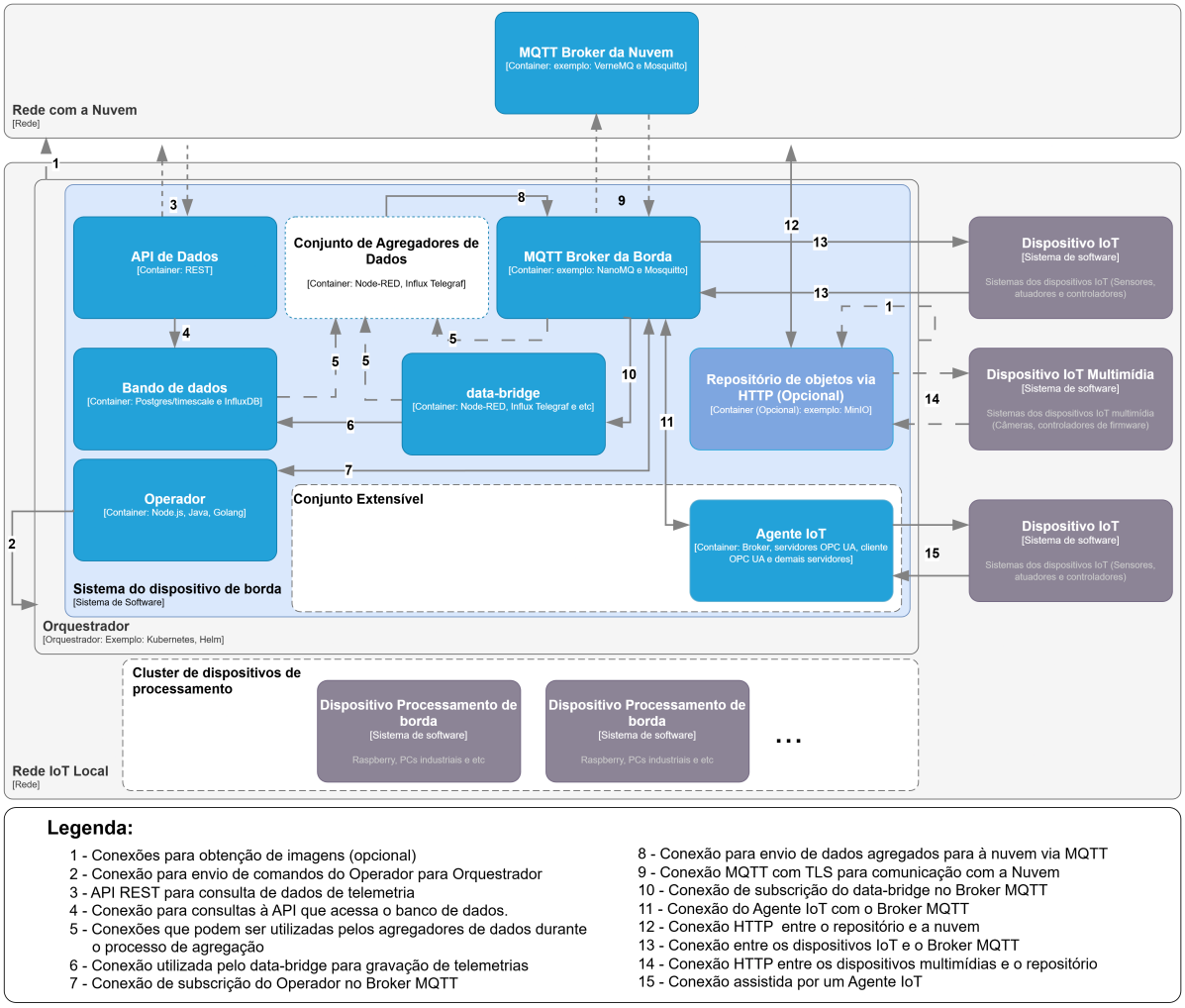

Figura 10 – Implantação da Arquitetura de Referência

torna-se possível facilitar os processos de atualização e de recuperação em caso de falhas do componente. Além disso, o uso de contêineres contribui para reduzir a probabilidade de erros, devido ao isolamento entre componentes que esse modelo de execução proporciona.

Os comandos direcionados aos dispositivos de borda devem ser enviados pela plataforma em nuvem, como atuações, por meio do *MQTT Broker*, e respondidos de forma assíncrona, como dados de dispositivo. Dessa forma, os dispositivos de processamento de borda são percebidos pela plataforma em nuvem como mais um dispositivo na rede IoT, o que facilita a integração com qualquer plataforma.

Conforme ilustrado na visão de implantação da Figura 10, o orquestrador pode estabelecer uma conexão com a nuvem para realizar o download das imagens dos contêineres que serão executados no sistema, ou a nuvem pode armazenar essas imagens no repositório multimídia do sistema, caso esse recurso esteja sendo utilizado na implementação. Arquivos de backup, logs e dados relativos à recuperação de desastres também podem ser armazenados nesse repositório antes de serem enviados para a nuvem. Dessa

forma, as operações podem ser executadas mesmo quando a conexão com a nuvem estiver comprometida.

A implantação de um sistema baseado na ED-operator RA pode ocorrer em dois modos. Esses modos diferem apenas quando múltiplos dispositivos de processamento estão disponíveis na rede. No primeiro modo, os componentes são implantados de forma unitária em cada dispositivo, ou seja, cada nó possui seu próprio Operador e seu próprio Orquestrador, como apresentado na Figura 9.

No segundo modo, os dispositivos compartilham o Operador e o Orquestrador em um *cluster*. Nessa configuração, o arquiteto pode optar por distribuir mais de uma instâncias do Operador, do banco de dados e do Broker MQTT em diferentes nós da rede, como apresentado na Figura 11, garantindo, assim, a disponibilidade imediata do sistema mesmo em caso de falha de um dos dispositivos de processamento local. Os demais componentes padrão da ED-Operator RA não exigem essa atenção especial, pois não interagem diretamente com a nuvem para atender comandos essenciais do sistema, o que dispensa a necessidade de resposta imediata por parte deles. No entanto, o orquestrador deve recompor esses componentes o mais rapidamente possível.

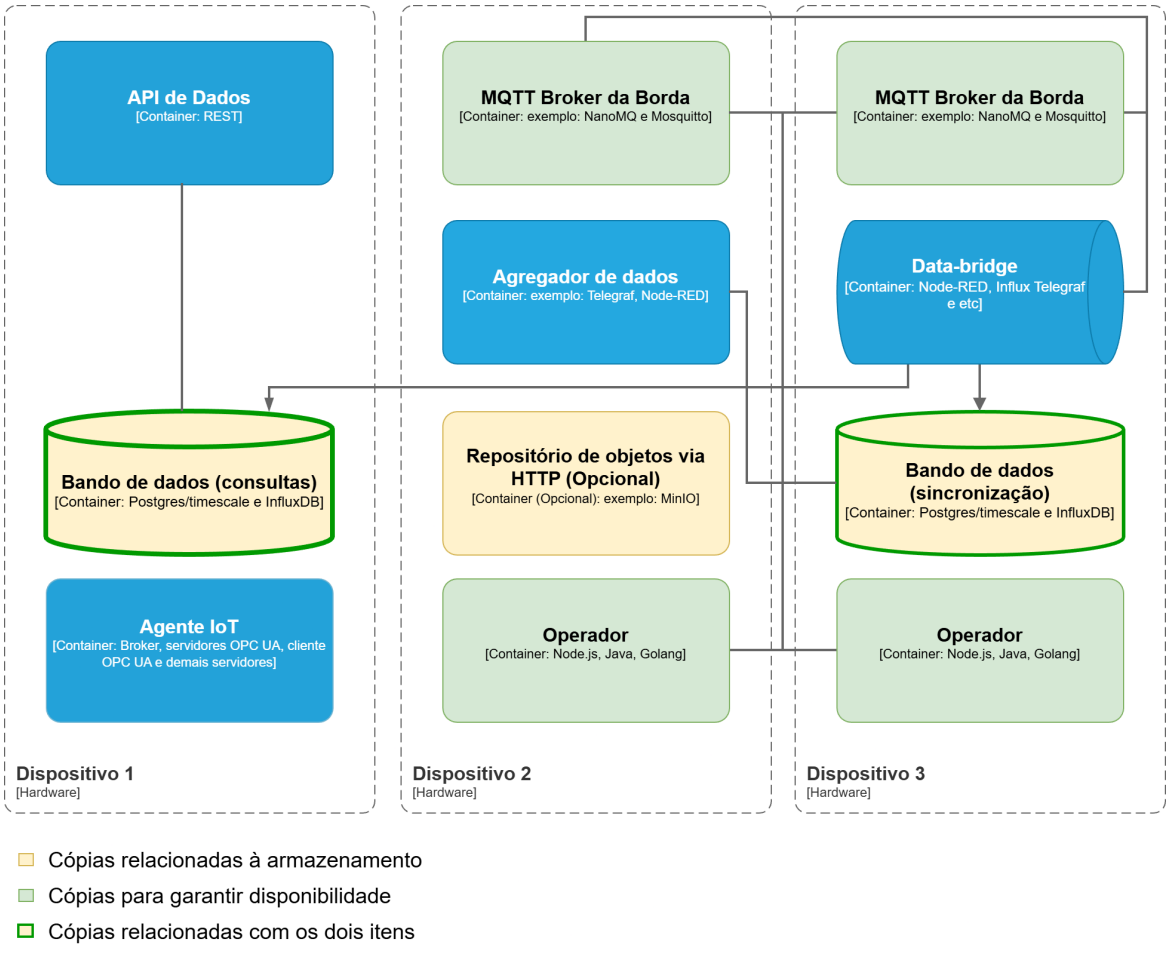

Figura 11 – Exemplo da ED-operator RA em Modo *Cluster*.

A Figura 12 apresenta uma visão mais técnica das operações de implantação e de atualização de microsserviços descritas nesta seção. A plataforma na nuvem publica uma atuação de implantação no *MQTT Broker* do dispositivo de processamento, o Operador recebe essa atuação via subscrição MQTT e emite um evento no orquestrador, que por sua vez conecta ao repositório de imagens na nuvem e faz o download da imagem e aplica a implantação e ao final publica um resposta no *MQTT Broker* para que ela seja enviada para nuvem.

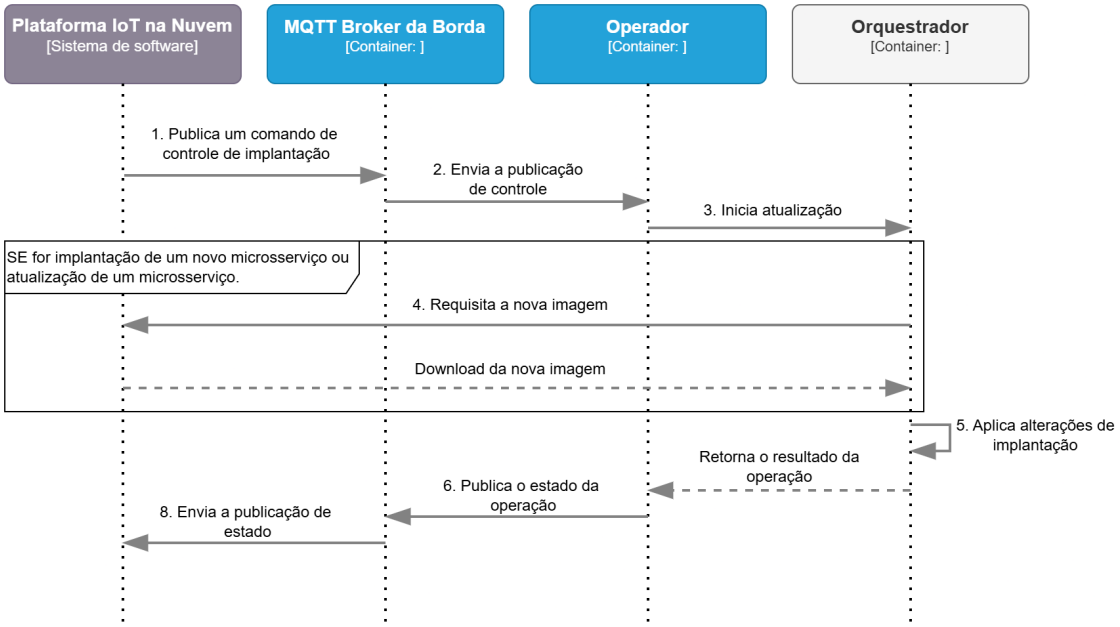

Figura 12 – Fluxo de controle de implantação e atualização de microsserviços na ED-operator RA

Os eventos de atuação sobre o dispositivo *Edge*, publicados no *MQTT broker*, devem conter as seguintes informações:

- **ID do dispositivo** — Um identificador único para o dispositivo;
- **ID de correlação da Mensagem** — Um identificador único para a mensagem;
- **Lista de comandos:**
  - **ID de correlação do Comando** — Um identificador único para a operação;
  - **Tipo de comando** — O tipo de comando que deve ser executado;
  - **Argumentos do comando** — Informações necessárias para a execução do comando.

Para o tráfego dessas informações, podem ser utilizados formatos como JSON, XML, YAML ou equivalentes. Para execução de *pipelines* de atualizações, formatos específicos para esse tipo de operação também podem ser adotados.

Os softwares que utilizam esta arquitetura de referência devem implementar os seguintes cinco comandos principais:

- ***Deploy*** — Realiza a implantação de um serviço. Os argumentos deste comando geralmente incluem: nome da implantação, versão, imagem e perfil de prioridade;
- ***Update-Deploy*** — Atualiza os parâmetros de implantação de um serviço. Os argumentos geralmente incluem: nome da implantação, versão e imagem;
- ***Undeploy*** — Realiza a remoção (desimplantação) de um serviço. Os argumentos geralmente incluem: nome da implantação e versão.
- ***Patch*** — Envia um contêiner que está executando na borda para ser executado na nuvem.
- ***Dispatch*** — Notifica a nuvem para interromper a execução de um contêiner.

Outros comandos podem ser adicionados conforme o domínio de atuação da aplicação, bem como de acordo com os microsserviços de extensão.

4.5.2 Compartilhamento de processo com a nuvem

O cenário SNE03, identificado no Capítulo 2, permite que serviços executados na borda sejam migrados para a nuvem. Para viabilizar essa operação, é necessário que os componentes executados em sistemas que implementam a ED-operator RA priorizem o protocolo MQTT como meio de comunicação entre seus pares. Dessa forma, quando um contêiner é deslocado para a nuvem, ele ainda é capaz de se comunicar com os componentes que continuam na borda por meio do modo *bridge* do MQTT, que replica na borda as mensagens produzidas na nuvem.

Apesar das vantagens oferecidas pelo compartilhamento de processos com a nuvem, alguns microsserviços apresentam limitações nesse contexto. Esse é o caso dos agentes IoT, que, ao serem movidos para a nuvem, tendem a aumentar a latência na comunicação com os dispositivos de campo, podendo comprometer a conformidade da arquitetura com o cenário de baixa latência (SNE02).

Outra situação de risco prevista ocorre quando há indisponibilidade da nuvem, conforme representado pelo cenário SNE05. Nessa situação, o sistema pode ser paralisado caso não haja recursos computacionais suficientes na borda, comprometendo a conformidade da arquitetura com o SNE05, que trata da disponibilidade do sistema. Isso também pode limitar os SLAs (*Service Level Agreement*) dos sistemas que utilizam a ED-operator RA.

Com o objetivo de mitigar essas situações de indisponibilidade e alta latência, esta arquitetura de referência propõe o uso de um perfil de prioridade para cada microsserviço

executado na borda. Dessa forma, em cenários de indisponibilidade, serviços com prioridade inferior podem ser interrompidos por meio de preempção. O perfil de prioridade também permite limitar quais serviços podem ser compartilhados com a nuvem, garantindo que serviços sensíveis à alta latência sejam atribuídos a prioridades mais altas. A Tabela 7 apresenta os perfis de prioridade e os respectivos serviços que os compõem.

Tabela 7 – Perfis de Prioridade

Perfil	**Descrição**
Padrão	Esse é o perfil de prioridade mais alto. É utilizado pelos micro-serviços essenciais do sistema, como Broker MQTT da Borda, data-bridge, Operador, agregador de dados, banco de dados e API de dados
Agente IoT	Esse é o perfil de prioridade dos agentes de tradução de protocolos.
Os perfis específicos de domínio ou de implementação estão alocados aqui.	
Serviço Simples	Esse é o perfil com a menor prioridade do sistema.

Durante a implementação, pode haver a necessidade de criar perfis específicos para o domínio ou para a própria implementação. Esses perfis devem ter menor prioridade em relação ao perfil *Agente IoT* e maior do que o perfil *Serviço Simples*.

4.5.3 Estratégias de escalabilidade

Uma das melhores formas de realizar escalabilidade em um sistema é a horizontal. A escalabilidade horizontal baseia-se em adicionar outras instâncias de um sistema em paralelo; essas novas instâncias podem ser executadas dentro de um contêiner ou de uma outra máquina. Em sistemas de nuvem que utilizam o Kubernetes, a forma mais clássica de buscar escalabilidade é através da criação de novas cópias de contêineres conforme o sistema dentro do contêiner se aproxima de limites estabelecidos, como o uso de CPU ou o consumo de memória RAM.

Dentro da ED-operator RA duas estratégias de escalabilidade serão adotadas. A primeira estratégia é aplicada aos microsserviços que não têm relação direta com os dispositivos da rede IoT, como o operador, *data-bridge*, banco de dados, API de dados e os agregadores de dados. Ela consiste na estratégia clássica baseada apenas em limites de uso de CPU e memória primária.

A segunda estratégia aplicada aos serviços que diretamente interagem com os dispositivos da rede IoT, como MQTT *Broker* da borda e os Agentes IoT, consiste em criar novas instâncias conforme os limites de uso do CPU, uso de memória e o tempo de latência das mensagens. Dessa forma, quando a média da latência das mensagens aumentar, novas instâncias dos microsserviços podem ser provisionadas para lidar com esse cenário.

O objetivo dessa estratégia é manter a latência inferior a 300 milissegundos, conforme definido no cenário de qualidade SNE02.

4.5.4 Armazenamento

O tipo de armazenamento adotado no hardware pode impactar diretamente a velocidade com que os dados são agregados e enviados. Existem diversas opções de armazenamento, como HDs, SSDs SATA, SSDs M.2 e cartões de memória de diferentes categorias. Equipamentos que utilizam HDs ou cartões de memória (como é comum em dispositivos como o Raspberry Pi) tendem a apresentar tempos de acesso mais elevados, o que pode afetar o desempenho das consultas ao banco de dados e comprometer a sincronia com a nuvem.

No caso dos Raspberries, o cartão da classe V30 é o mínimo recomendado para garantir um funcionamento adequado. Entretanto, se o sistema precisar responder a um volume elevado de consultas ou se a sincronia com a nuvem for uma operação crítica, dispositivos de armazenamento mais rápidos, como SSDs SATA ou SSDs M.2, devem ser adotados. Com esses tipos de armazenamento, o sistema atinge o desempenho necessário para atender aos cenários de qualidade definidos.

A ED-operator RA é projetada para manter o funcionamento mesmo durante períodos de indisponibilidade da nuvem. Após a restauração da conectividade, torna-se necessária a sincronização de dados. Esse processo envolve consultas extensas ao banco de dados do sistema, o que, em dispositivos de armazenamento de menor velocidade, pode impactar na operação do sistema.

Para hardwares de menor velocidade, como cartões de memória ou HDs, é recomendado efetuar a sincronização de forma particionada, reduzindo a carga de leitura. Em contrapartida, para SSDs SATA, SSDs M.2 ou outros dispositivos de maior desempenho, pode ser viável realizar sincronizações mais amplas. No entanto, consultas oriundas da nuvem ainda podem comprometer a aderência do sistema aos cenários de qualidade estabelecidos.

Com o objetivo de otimizar o armazenamento em sistemas que usam a arquitetura de referência ED-operator RA operando no modo *cluster*, o arquiteto pode configurar a implantação do repositório de arquivos em um dispositivo que não possua uma instância do banco de dados, conforme ilustrado na Figura 11. Essa configuração adicional permite otimizar o uso do armazenamento, evitando que operações de *upload* e *download* de arquivos interfiram em processos de sincronização ou em outras consultas realizadas pelo sistema.

Ainda no modo *cluster*, as instâncias do banco de dados podem ser divididas em instâncias dedicadas à leitura e a outras destinadas às operações de escrita e sincronização,

garantindo que consultas de dados não impactem o funcionamento operacional do sistema. A Figura 11 ilustra um exemplo dessa prática.

As estratégias de particionamento de dados adotadas pela ED-operator RA são inspiradas nos trabalhos de Fouto, Preguiça e Leitão (2024) e Fu, Qiu e Wang (2019), que buscam otimizar o armazenamento de dados por meio da distribuição das informações entre múltiplos nós de armazenamento.

Para sistemas de implantação unitária, ou quando o arquiteto deseja aplicar melhorias adicionais em ambientes operando em modo *cluster*, soluções baseadas em *striping* de dados podem ser utilizadas, como aquelas implementadas por protocolos RAID, que permitem a distribuição de blocos de dados entre múltiplos dispositivos de armazenamento. Nesse contexto, um equipamento equipado com dois SSDs pode particionar operações de leitura e escrita entre ambos os dispositivos, explorando paralelismo e aumentando o desempenho no acesso a dados.

É possível observar estratégias conceitualmente equivalentes na literatura. Um exemplo é o estudo de Cai et al. (2025), que propõe um *key-value store* persistente capaz de distribuir dados e índices entre DRAM e memórias não voláteis, explorando o paralelismo de acesso na hierarquia de memória. Embora atue no nível de memória, essa abordagem é funcionalmente equivalente ao *striping*, ao particionar dados entre múltiplos recursos e permitir leituras concorrentes, com o objetivo de escalabilidade e baixa latência.

No entanto, abordagens dessa natureza podem comprometer a conformidade do sistema com o cenário SNE06, uma vez que a ED-operator RA tem seu modelo de atualização baseado na conteinerização. As soluções baseadas em *striping* implementadas em nível de sistema operacional geralmente requerem a instalação de componentes específicos fora do ambiente conteinerizado. Em consequência, tais componentes não podem ser atualizados por meio dos mecanismos da arquitetura, exigindo intervenções em campo ou formas externas de acesso ao sistema operacional, como conexões via SSH.

4.6 Visões de fluxo de dados

Nesta seção, são apresentadas as visões técnicas do sistema, direcionadas a desenvolvedores e arquitetos de software, com foco nos principais fluxos de dados. Como essas visões estão mais próximas da implementação real, como na Figura 12, foram utilizados diagramas de sequência para representá-las.

A primeira visão, representada na Figura 13, descreve o fluxo de envio de dados. Nesse processo, os dispositivos IoT transmitem informações para o dispositivo de processamento de borda, que as recebe por meio de um broker MQTT ou, caso utilizem outro protocolo, por meio de um Agente IoT. Em seguida, os dados são persistidos em um banco

de dados para consultas futuras.

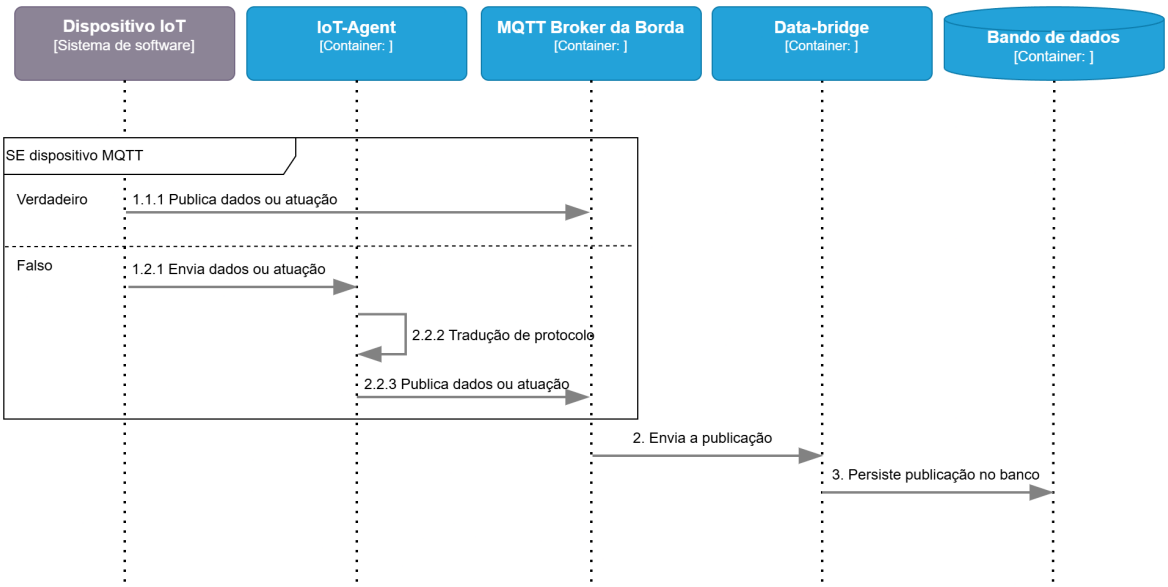

Figura 13 – Envio de dados

Já as atuações oriundas da nuvem são recebidas pelo broker MQTT, que descriptografa a mensagem e a encaminha aos serviços subscritos. Caso a atuação seja destinada a um dispositivo que não utiliza MQTT, o Agente IoT responsável por traduzir o protocolo correspondente receberá a mensagem por meio de sua subscrição e a encaminhará ao dispositivo. As mensagens de atuação provenientes da nuvem também são persistidas no banco de dados para fins históricos, conforme ilustrado na Figura 14.

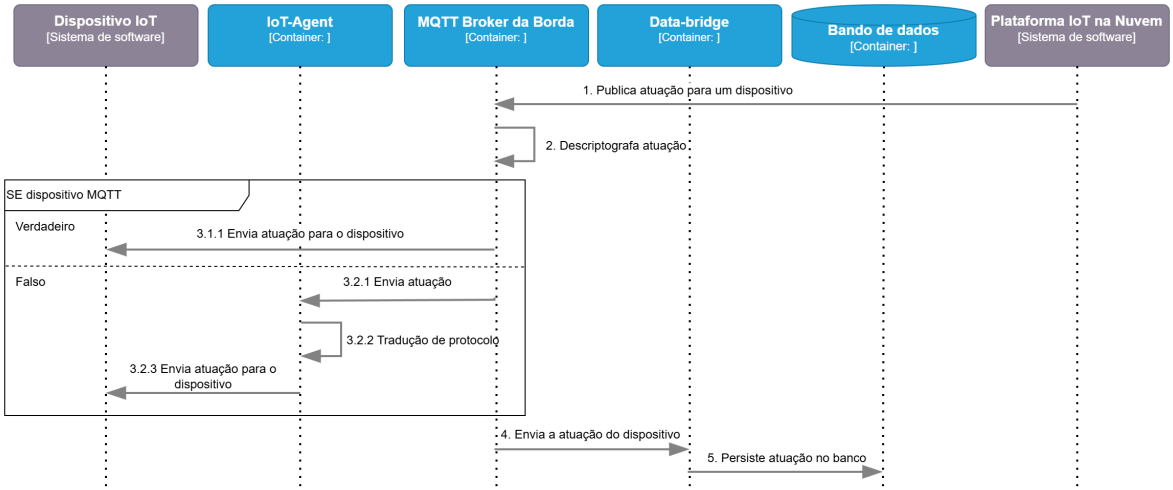

Figura 14 – Recebimento de dados

O envio de dados para a nuvem é realizado de forma agregada, conforme regras definidas pelos desenvolvedores. O serviço agregador pode utilizar outros microsserviços como fontes de dados, conforme a necessidade da aplicação. Os dados agregados são

então enviados para a plataforma em nuvem por meio do protocolo MQTT. A Figura 15 apresenta um diagrama ilustrando esse fluxo, no qual o banco de dados é utilizado como fonte de dados para a agregação.

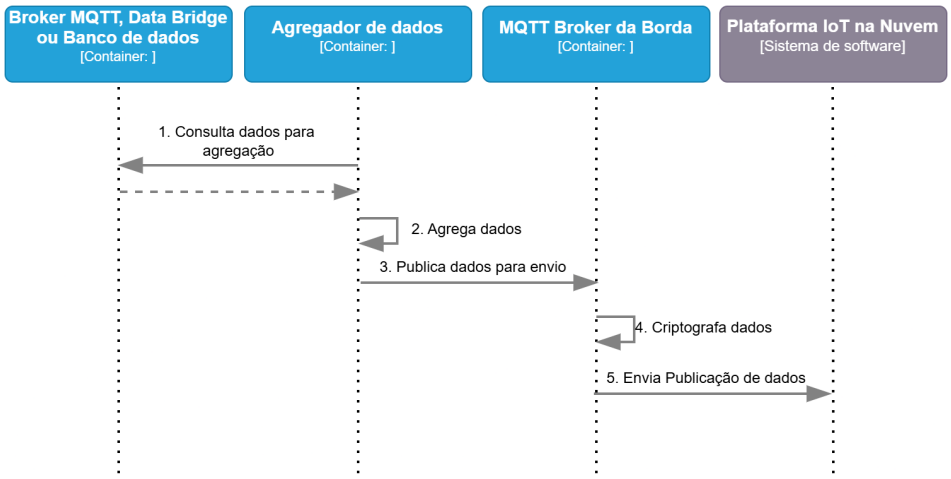

Figura 15 – Envio de dados agregados para nuvem

É recomendado o uso de um mecanismo de encaminhamento de dados conhecido como *bridge*. Nesse modo de operação, o *broker* localizado na borda estabelece uma conexão como cliente com o *broker* da nuvem, sem interromper o atendimento aos clientes da borda. As publicações realizadas nos tópicos destinados à nuvem são replicadas do ambiente de borda para o ambiente em nuvem. Essa forma de operação também viabiliza que componentes em borda interessados em dados agregados possam se subscrever nesses tópicos, permitindo o consumo local das informações antes ou independentemente do envio à nuvem. Uma vez que os dados estejam publicados no *broker* de borda é possível configurar o *data-bridge* para gravar esses dados no banco para consultas futuras.

4.7 Rationale

Esta seção apresenta as principais decisões tomadas durante o desenvolvimento da proposta, bem como suas respectivas justificativas.

A adoção do modelo de microsserviços para a ED-operator RA foi motivada pela facilidade de integração desse tipo de arquitetura com tecnologias de conteinerização e orquestração, bem como pela capacidade de modularização. A conteinerização, aliada à orquestração, oferece um ambiente seguro e prático para a manutenção, a atualização e a customização remota do sistema.

Para a comunicação, optou-se pelo MQTT como protocolo padrão da arquitetura. Essa decisão decorreu do fato de este protocolo ser o mais citado na RSL apresentada no Capítulo 2 e de ser frequentemente definido na literatura como padrão de mercado.

Embora o protocolo ofereça suporte a grandes volumes de dados, nesta arquitetura foi decidido que a transmissão de dados ocorrerá de forma agregada. O envio de dados agregados reduz o tráfego de rede e contribui para mitigar problemas relacionados à *Big Data*. Além disso, a agregação aumenta a privacidade em relação aos dados brutos e reduz a exposição do funcionamento interno do sistema, considerando que, a partir dos dados brutos, um atacante poderia inferir o comportamento do sistema.

Para transmitir um arquivo via MQTT, é necessário codificar os *bytes* em formatos como Base64. Nos testes práticos realizados neste estudo, a transmissão de arquivos por MQTT gerou gargalos no broker, evidenciando a necessidade de um protocolo mais apropriado para esse tipo de operação. Diante disso, o protocolo HTTP foi escolhido para o tráfego de arquivos, pois ele é o terceiro protocolo mais citado da RSL e o mais citado para esse tipo de operação.

4.8 Considerações Finais

Este capítulo apresentou a Arquitetura de Referência ED-operator RA, proposta para permitir o uso de dispositivos dedicados ao processamento em redes IIoT. Foram descritos as decisões e motivações que fundamentam a proposta, as visões arquiteturais baseadas no modelo C4 e os principais fluxos de dados do sistema. A adoção do processamento na borda, com agregação de dados e suporte a atualizações remotas, visa atender aos desafios de latência, confiabilidade e privacidade típicos de ambientes industriais.

No Capítulo 6, serão apresentados os próximos passos do estudo, com foco na avaliação da arquitetura proposta e na construção de um protótipo que valide seus conceitos.

5 Prototipação e Experimentação

5.1 Considerações Iniciais

No capítulo anterior, a proposta da arquitetura de referência ED-operator RA foi apresentada, acompanhada da descrição das decisões arquiteturais e das motivações que a fundamentam, bem como das visões arquiteturais baseadas no modelo C4 e dos principais fluxos de dados do sistema. Além disso, foram descritas as recomendações relacionadas ao armazenamento, à escalabilidade e ao compartilhamento de processos com a nuvem.

Seguindo a metodologia empregada, este capítulo apresenta todo o conteúdo necessário para viabilizar a replicação do processo de prototipação, por meio de uma descrição detalhada das etapas envolvidas, incluindo o desenvolvimento do protótipo, a simulação e os testes. Também apresenta as ferramentas utilizadas ao longo desse processo. As informações apresentadas neste capítulo também podem ser empregadas como base para aprimoramentos futuros na prototipação e na simulação.

5.2 Protótipo

O protótipo da ED-operator RA possui todos os elementos necessários ao seu funcionamento, além de um Agente IoT OPC UA, alocado como um elemento extensível do sistema. Para sua construção, foram utilizados softwares *open-source*, gratuitos e desenvolvidos pelo próprio autor.

O protótipo foi desenvolvido para operar em modo unitário. Essa abordagem foi adotada porque é possível inferir que, se o modo unitário atende aos cenários de qualidade estabelecidos, o modo *cluster* também tende a estar em conformidade, uma vez que este oferece maior capacidade de processamento e melhor desempenho. O contrário, entretanto, não pode ser afirmado, pois a conformidade observada no modo *cluster* não garante que o modo unitário apresente desempenho suficiente para atender aos mesmos cenários de qualidade. A Figura 16 apresenta o diagrama de implementação do protótipo.

O *broker* MQTT utilizado foi o NanoMQ (NANOMQ, 2025), um *broker open-source* projetado para operar em ambientes de *Edge Computing*. O banco de dados escolhido foi o InfluxDB (INFLUXDATA, 2025a), um banco de dados NoSQL de séries temporais que, na versão 2.7, é *open-source*. O InfluxDB também oferece uma interface REST para consulta dos dados armazenados, permitindo seu uso como API. Para extrair os dados do *broker* MQTT e persisti-los no banco de dados de séries temporais, utilizou-se a ferramenta Telegraf (INFLUXDATA, 2025b), um ETL comum para telemetria em sistemas de nuvem.

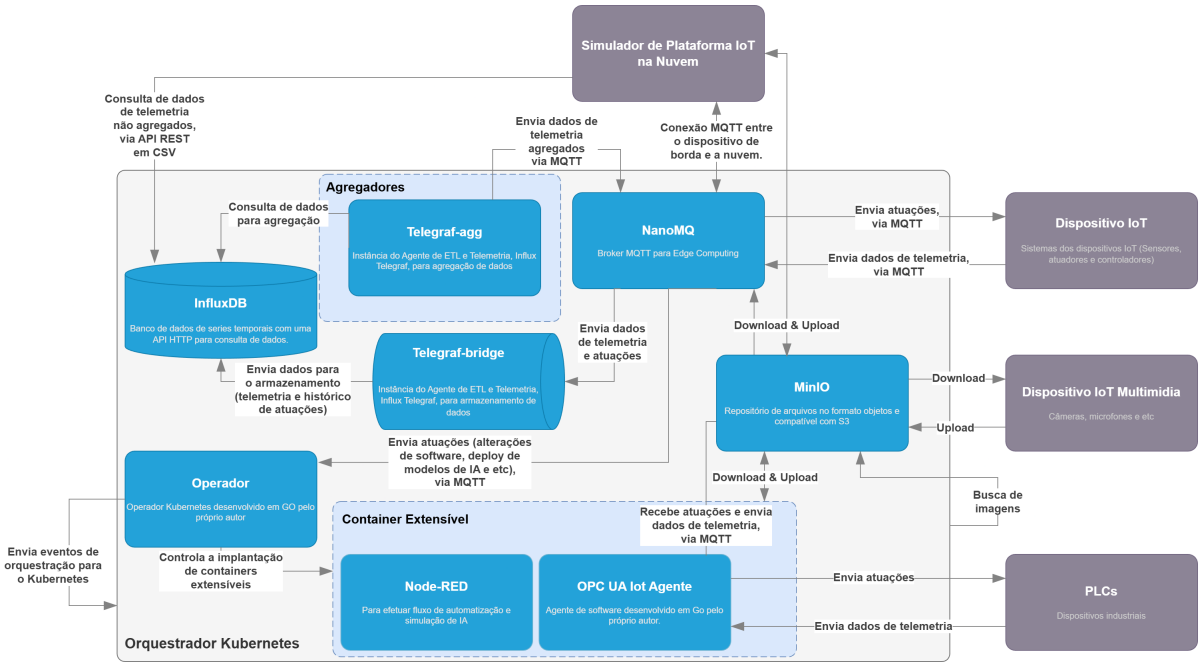

Figura 16 – Implementação do Protótipo (módulos de software)

O Telegraf também é responsável pela agregação dos dados trafegados no sistema.

Para o papel de repositório de arquivos multimídia, utilizou-se o MinIO (MINIO, INC., 2025), uma ferramenta compatível com o padrão Amazon S3 (AMAZON WEB SERVICES, 2025). O S3 é um serviço de armazenamento baseado em objetos, no qual cada objeto é composto por dados do arquivo, seus metadados e uma chave de identificação para buscas. Os objetos são agrupados em *buckets*, nos quais podem ser versionados e ter seu controle de acesso configurado. Além disso, o serviço oferece mecanismos de gerenciamento do ciclo de vida, como a definição de políticas de expiração para persistência temporária (AMAZON WEB SERVICES, 2025).

No protótipo desenvolvido, o MinIO também foi responsável por armazenar as imagens dos contêineres no formato ZIP, atuando como repositório centralizado do sistema, conforme uma das opções previstas pela ED-operator RA para o armazenamento de imagens.

O Operador e o Agente OPC UA foram desenvolvidos pelo próprio autor utilizando a linguagem Go (GRIESEMER; PIKE; THOMPSON, 2009). Essa linguagem foi escolhida devido à sua boa interoperabilidade com o orquestrador Kubernetes, que é a ferramenta de orquestração adotada neste protótipo. Outra ferramenta empregada no contêiner extensível é o Node-RED (NODE-RED CONTRIBUTORS, 2024), selecionada por sua flexibilidade na implementação de fluxos lógicos e de simulações de comunicação entre dispositivos.

No protótipo, a comunicação com a nuvem para a atuação deve ser feita por meio do tópico *device/config* no formato JSON. As mensagens devem possuir dois atributos,

Comandos	Descrição	Argumentos
deploy	**Realiza a implantação de um serviço.**	• **name -** Nome da implantação; • **image -** Imagem do contêiner a ser implantado; • **port -** Porta de conexão do contêiner; • **priorityProfile -** Perfil de prioridade do serviço; • **internalPort** - Porta do serviço para o cluster; • **externalPort** - Porta externa do serviço; • **requestMemory** - Mínimo de memória necessário para o serviço executar; • **limitMemory** - Máximo de memória que pode ser alocada para o serviço; • **requestCPU** - Cotas de CPU minima para o serviço executar; • **limitCPU** - Máximo de cotas de CPU que pode ser alocada para o serviço;
update	**Realiza a atualização na implantação de um serviço.**	
list_deploy	**Lista todas as implantações.**	
undeploy	**Remove a implantação de um serviço.**	• **name -** Nome da implantação; • **image -** Imagem do contêiner a ser implantado;
register_opc	**Registra um dispositivo OPC.**	• **url -** Url do dispositivo OPC na rede IoT Local; • **nodeId -** ID dos dispositivo OPC; • **interval** - Intervalo de consulta de dados
list_opc	**Lista todos os dispositivos OPC registrados.**	
unregister_opc	**Remove um dispositivo OPC.**	• **nodeId -** ID dos dispositivo OPC;

Figura 17 – Comandos Disponíveis no Protótipo

deviceId para identificação do dispositivo-alvo e o *commands* para definir quais comandos serão executados. A Figura 17 apresenta os comandos disponíveis no protótipo, com seus respectivos argumentos.

Todos os resultados dos comandos devem ser publicados no tópico *device/results*. Esse tópico também é responsável por trafegar as mensagens de *Acknowledgement* (ACK) que serão fundamentais para o cálculo da latência das mensagens. Todas as mensagens de ACK são estruturadas em formato JSON e contêm dois atributos essenciais:

- *timestamp* — representa o instante de recebimento da mensagem;
- *correlation* — armazena o *timestamp* da mensagem original enviada, permitindo o cálculo do tempo de resposta.

Os dados provenientes dos dispositivos devem ser publicados em tópicos com o

prefixo *device/data*, enquanto os dados agregados são encaminhados para a nuvem por meio do tópico *cloud/data* que opera no modo *bridge* com o *broker* da nuvem, conforme especificado pela arquitetura referência proposta.

5.3 Simulação Industrial

A simulação industrial foi dividida em conjuntos, referidos neste estudo como blocos de produção. Cada bloco de produção é composto por um PLC responsável pelo controle da linha, uma câmara quente de injeção plástica e uma câmera de inspeção. A simulação da linha foi realizada com o software Factory.io (REAL GAMES UNIPESSOAL LTDA., 2025), conforme apresentado na Figura 18.

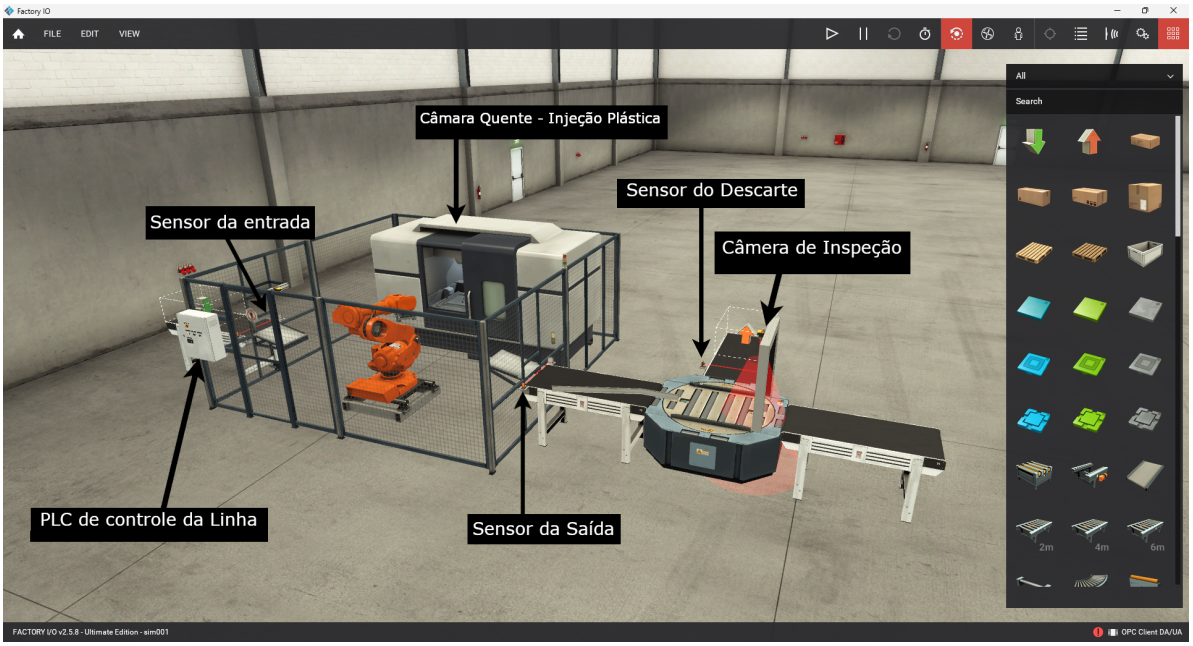

Figura 18 – Bloco de produção no Factory.io

A câmera de inspeção foi simulada por meio de um *script* em Python executado em um contêiner Docker. Esse *script* recebe, via MQTT, dados sobre o estado da linha de operação e confirma o recebimento de todas as mensagens por meio de mensagens ACKs. Quando esses dados indicam que uma peça está posicionada na estação de inspeção, o sistema simula uma captura de imagem, envia o arquivo gerado para o repositório multimídia e publica, no *broker* MQTT.

Esse *script* foi necessário porque a câmera do Factory I/O opera com objetos internos específicos do próprio simulador, não realizando a captura efetiva de imagens. Dessa forma, tornou-se necessário implementar um mecanismo adicional para simular o processo de aquisição de imagem.

A Figura 19 apresenta o experimento completo.

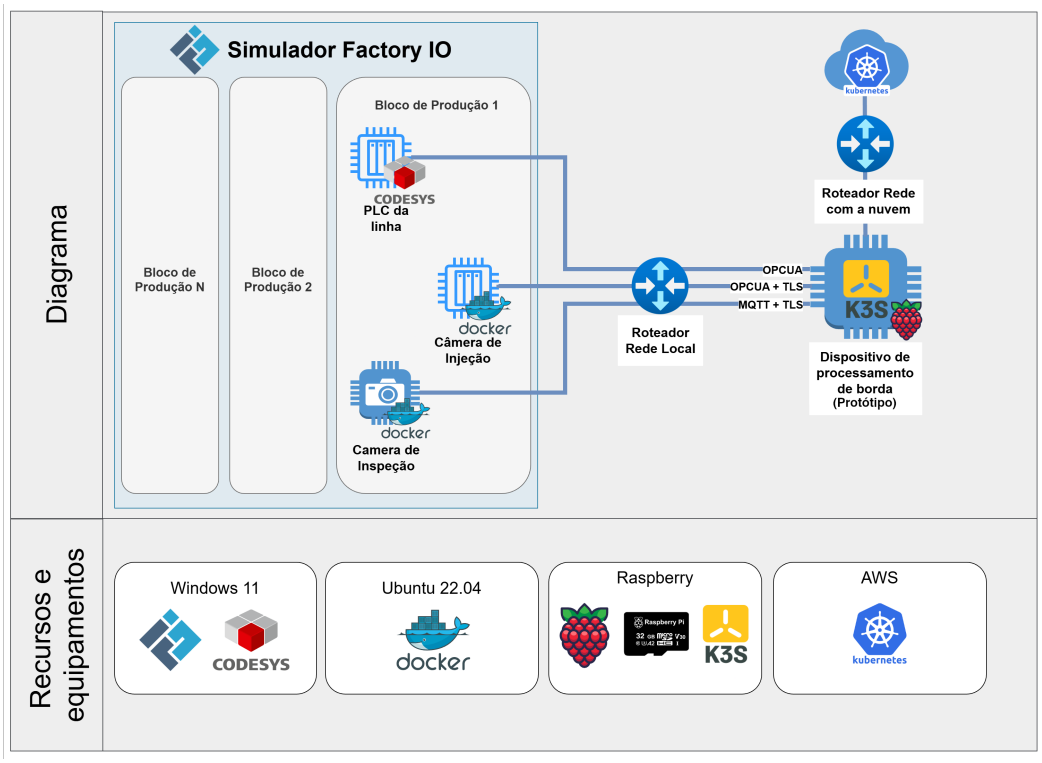

Figura 19 – Resumo do experimento

Os dados da câmara quente foram gerados por um *script* em Go, que armazena as informações em um servidor OPC UA próprio, implementado com a biblioteca *open source* goopcua (THE GOPCUA TEAM, 2025). A geração dos dados baseou-se nos trabalhos de Nagorny et al. (2018) e Jung et al. (2021), ambos com o objetivo de otimizar o processo de inspeção de peças produzidas por injeção plástica. O estudo de Nagorny et al. (2018) utiliza redes neurais e algoritmos de regressão clássicos para prever atributos de qualidade das peças com base em medições industriais, enquanto Jung et al. (2021) aplica algoritmos de aprendizado de máquina (*Machine Learning*) para o mesmo propósito.

O objetivo foi utilizar os dados desses dois estudos para criar uma simulação mais próxima da realidade, embora não com precisão total, uma vez que o foco dos experimentos é avaliar as interações do sistema e a velocidade com que elas ocorrem. Na Figura 20 é possível visualizar a curva apresentada em Nagorny et al. (2018) e a curva apresentada pelo simulador produzido neste estudo.

Na Figura 20, os dois gráficos superiores representam a posição do bico injetor e a pressão aplicada durante a injeção. Os gráficos inferiores apresentam a pressão e a temperatura do molde. Observa-se, que os gráficos superiores apresentam comportamento semelhante, uma vez que foram fortemente influenciados pelo estudo de Nagorny et al. (2018). Em contrapartida, os gráficos inferiores apresentam maiores diferenças, pois se baseiam majoritariamente no estudo de Jung et al. (2021).

A lógica do PLC foi desenvolvida no CODESYS (CODESYS GmbH, 2025), utili-

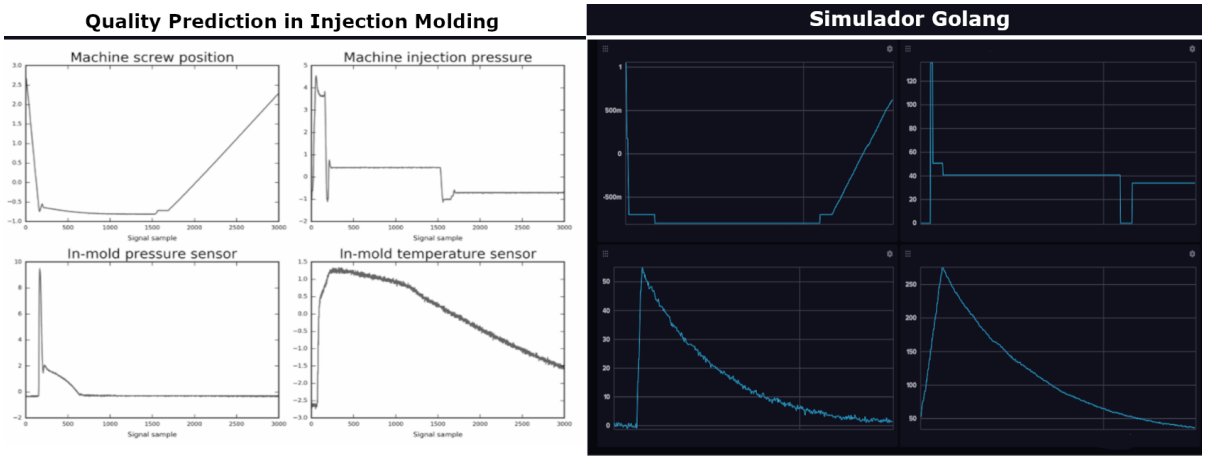

Figura 20 – Comparação entre a curva do simulador desenvolvido e a curva apresentada em Nagorny et al. (2018)

zando o diagrama *Ladder*. Essa linguagem, amplamente empregada em PLCs, representa de forma intuitiva o fluxo lógico do sistema, similar a um circuito elétrico de relés. Em geral, o diagrama *Ladder* descreve bem anomalias e variações na condução de sinais entre dispositivos, ainda que seja menos explícito na formação desses impulsos (JOHNSON; DENES, 2008).

O PLC utilizado nos experimentos foi emulado por meio do ambiente CODESYS. O programa de controle foi desenvolvido em conformidade com a norma IEC 61131-3 (IEC, 2013), especificando tarefas com período de varredura configurado e prioridades definidas. A comunicação entre o Factory I/O e o CODESYS foi estabelecida via OPC UA, assim como a comunicação PLC e dispositivo de processamento de borda (Raspberry PI).

A Figura 21 apresenta o fluxo de operação do PLC, cuja lógica empregada na linha é simples: o sistema contabiliza o número de peças que entram e saem, libera uma nova peça a cada saída e, quando a câmera envia uma mensagem de descarte, aciona a esteira e registra a quantidade de itens efetivamente descartados.

Para simular e implementar o ciclo de inspeção realizado por uma IA ou um algoritmo de *machine learning*, foi desenvolvido pelo próprio autor um fluxo no Node-RED, apresentado na Figura 22. Esse fluxo recebe a mensagem MQTT da câmera com o caminho para a imagem, busca a imagem no MinIO e simula a tomada de decisão de uma IA.

Nesse experimento, visando incentivar a comunicação entre os dispositivos, a taxa de refugo foi fixada em 10%. Dessa forma, espera-se aproximadamente dez atuações de descarte por PLC da linha ao longo dos testes.

Para a execução das simulações, foram utilizados três equipamentos físicos. O primeiro, destinado ao papel de dispositivo de processamento local, consistiu em um Raspberry Pi 5, equipado com:

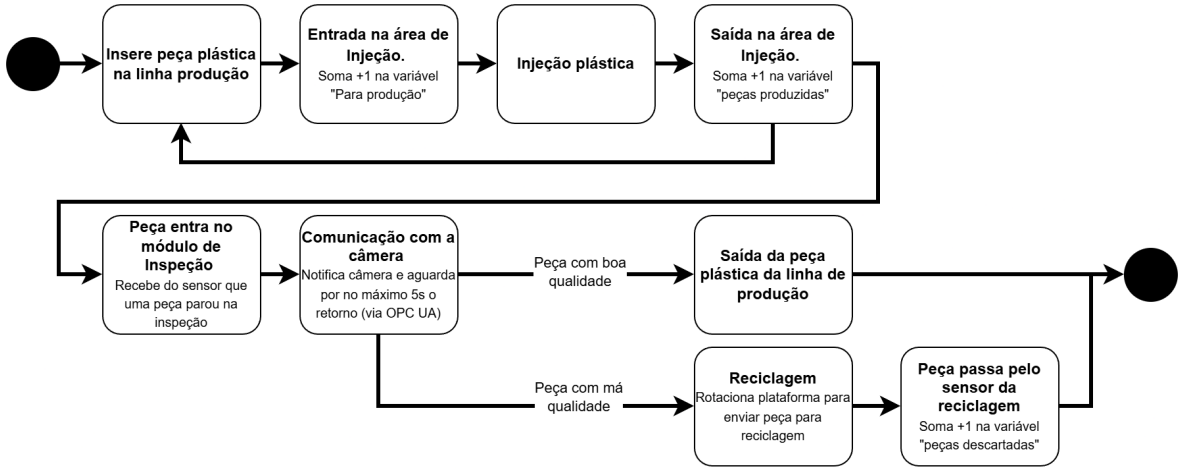

Figura 21 – Fluxo de operação do PLC

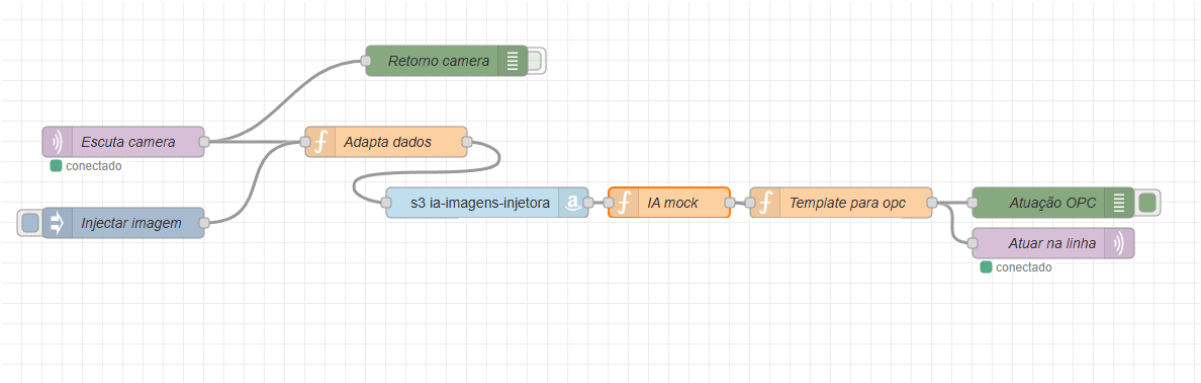

Figura 22 – Fluxo de inspeção no Node-RED

- Processador ARM *quad-core* Cortex-A76 de 64 bits e 2.4 Ghz de frequência de operação;
- 4 GB de memória SDRAM LPDDR4X;
- Cartão de memória Sandisk Micro SD Extreme Pro da classe V30 de 128 GB.

Esse dispositivo operou com Linux Alpine (ALPINE LINUX DEVELOPMENT TEAM, 2025) e executou um cluster K3S (RANCHER LABS, 2025). O Alpine Linux é uma distribuição Linux leve, com poucas dependências instaladas, originalmente projetada para uso em contêineres. Apesar de seu foco em ambientes de conteinerização, o Alpine Linux também disponibiliza versões para processadores AMD, Intel e ARM, como o Raspberry Pi (ALPINE LINUX DEVELOPMENT TEAM, 2025).

Já o k3s é uma distribuição minimalista do Kubernetes, desenvolvida para execução em dispositivos de processamento de borda (RANCHER LABS, 2025). Ambos os elementos foram selecionados com base na abordagem simplificada adotada por essas ferramentas. Para fins de comparação, o Alpine Linux apresentou um consumo médio de

aproximadamente 100 megabytes de memória após sua instalação, enquanto o consumo combinado do sistema com o k3s foi em torno de 700 megabytes.

O segundo equipamento foi responsável pela execução do Factory I/O e do CODESYS. Trata-se de uma máquina com:

- Processador *six-core* AMD Ryzen 5 5600H, com 12 *threads*, frequência de operação de 3.3 GHz a 4.2 GHz;
- 16 GB de memória DDR4, com frequência de operação em 3200 Mhz;
- GPU Nvidia RTX 3050 mobile com 4 GB de VRAM GDDR5.
- Sistema operacional Windows 11 24H2.

Essa mesma máquina também hospedou uma máquina virtual Ubuntu 22.04, utilizada para a execução dos contêineres Docker que simulavam os dispositivos. A máquina virtual foi configurada com 4 núcleos de CPU e 6 GB de memória RAM.

Por fim, a simulação da camada de nuvem foi realizada em uma máquina virtual na AWS, equipada com a ferramenta K8S.

5.3.1 Agregação de Dados

A agregação de dados dessa simulação ocorreu no intervalo de vinte segundos, e as seguintes regras foram adotadas:

- PLC – Para as variáveis que representam a entrada de itens na área de injeção, produção e descarte, foram enviados os valores máximos recebidos em uma janela de um segundo;
- Câmera – Foram enviados apenas os dados de mudança quando a câmera alterou seu estado (apenas dois valores: ligado e desligado);
- IA – Foram enviados todos os dados de inspeção; nesse caso, não foi necessário agregar os dados, já que os mesmos foram poucos;
- Câmara de injeção plástica – Foram enviados os valores máximos e mínimos alcançados dentro de cada janela de 500 milissegundos, bem como a média móvel, também em uma janela de 500 milissegundos, de todas as quatro variáveis apresentadas na Figura 20. Todos os valores gerados pela câmara quando ela não estava operando foram filtrados.

5.4 Avaliação Individual dos Cenários

Os cenários de qualidade foram definidos com base nos atributos identificados na revisão sistemática da literatura apresentada no Capítulo 2. A Tabela 8 reúne novamente esses cenários, acompanhados de suas respectivas métricas e dos testes necessários.

Como mencionado ainda neste capítulo, a conformidade foi avaliada utilizando o sistema no modo unitário do ED-operator RA. Para cada cenário, estabeleceu-se uma métrica, quantitativa ou qualitativa, destinada a avaliar a conformidade da arquitetura, bem como testes específicos para exercitar cada aspecto considerado.

Para viabilizar a análise, foram gerados gráficos a partir de consultas específicas ao banco de dados InfluxDB. Com o objetivo de facilitar a avaliação dos resultados, os diferentes gráficos produzidos foram organizados em *dashboards*.

O primeiro *dashboard* (DASH001) apresenta os dados de uso de memória e de processamento do dispositivo de processamento local pelo ED-operator RA, enquanto o segundo *dashboard* (DASH002) exibe o tempo de gravação de cada mensagem no banco de dados. Os dois *dashboards* seguintes possuem maior complexidade e são detalhados nas seções subsequentes.

5.4.1 *Dashboard* MQTT para PLC (DASH003)

O *dashboard* referente à comunicação MQTT-PLC apresenta as métricas necessárias para validar as ações executadas no PLC, resultantes da conversão de mensagens MQTT em operações de escrita via protocolo OPC UA. Nele são exibidos o número de atuações realizadas no PLC e a quantidade de ACKs retornados pelo próprio PLC. A equivalência entre esses valores indica ausência de falhas na comunicação entre o dispositivo MQTT e o dispositivo OPC UA.

As atuações enviadas pelo MQTT para o PLC ocorrem após as inspeções, com o objetivo de informar se o item inspecionado deve ser descartado ou não. Para evidenciar o funcionamento correto da linha, o *dashboard* também exibe o número de comandos de descarte enviados e o número de descartes efetivamente registrados pelo sensor na saída da estação. Como mencionado anteriormente, essas atuações estão diretamente relacionadas ao processo de inspeção. Por esse motivo, o *dashboard* inclui ainda a quantidade de itens produzidos por cada bloco, cuja soma deve corresponder ao total de atuações pós-inspeção.

Com esses dados, é possível avaliar a conformidade do sistema com o cenário SNE01. Além disso, o *dashboard* apoia a verificação do SNE02 por meio de um gráfico que mostra a diferença temporal entre o envio da mensagem de atuação e o envio do ACK correspondente.

Tabela 8 – Cenários de qualidade

Cenários	**Descrição**	**Métricas**	**Descrição dos testes**
Interoperabilidade (SNE01)	Garante que dispositivos com protocolos diferentes (como MQTT, OPC-UA, HTTP) consigam se comunicar corretamente.	O número de mensagens interpretadas deve se manter acima dos 99%.	A avaliação utiliza um ambiente simulado com monitoramento contínuo, no qual a conformidade é garantida através da comparação entre as mensagens enviadas e as ACKs.
Baixa Latência (SNE02)	Foca na redução do tempo de resposta entre dispositivos, com o uso de edge computing e compressão de dados.	A latência de 99.8% das mensagens entre dispositivos deve ser inferior a 300ms.	Reutilizar o teste do cenário SNE01, monitorando os dados e calculando o tempo de resposta pela diferença entre o envio da mensagem e o recebimento indicado no ACK.
Escalabilidade (SNE03)	Avalia a capacidade do sistema de escalar em número de dispositivos e volume de processamento, por meio da cooperação entre *edge* e *cloud*.	A taxa de processamentos falhos ou com atraso deve ser inferior a 1%.	Reutilizar o teste do cenário SNE01, aumentando o número de dispositivos de modo a aumentar a taxa de solicitações e avaliar a capacidade do sistema de lidar com cargas crescentes.
Segurança e Privacidade (SNE04)	Trata da proteção dos dados por meio de autenticação e criptografia (ex: TLS 1.2+).	-	A conformidade com os requisitos de segurança e criptografia é demonstrada pelo uso de certificados digitais e protocolos que asseguram a confidencialidade e autenticidade da comunicação.
Disponibilidade após falhas (SNE05)	Avalia a capacidade do sistema de continuar operando em caso de falhas de conexão com a nuvem	99% das funções críticas devem permanecer operacionais.	Executar os testes dos cenários SNE01, SNE02 e SNE03 sem a conexão com a nuvem.
Manutenibilidade (SNE06)	Visa reduzir a necessidade de manutenção manual através de automações executadas na borda de forma remota.	O sistema deve realizar atualizações sem causar indisponibilidade.	A conformidade com esse requisito é evidenciada por meio da execução sequencial de comandos remotos.

5.4.2 *Dashboard* PLC para câmera (DASH004)

O *dashboard* referente à comunicação PLC-câmera adota uma estratégia semelhante à do *dashboard* anterior, apresentando a quantidade de mensagens enviadas do PLC para a câmera e a quantidade de mensagens de retorno enviadas pela câmera. Além disso, exibe um gráfico que mostra a diferença temporal entre o envio da mensagem pelo PLC e o recebimento do ACK correspondente.

Entretanto, essa etapa do experimento apresentou problemas de sincronização temporal entre o PLC e a câmera, o que é típico de sistemas distribuídos. Para possibilitar a geração do gráfico, o cálculo da diferença temporal não utilizou o momento do envio do ACK pelo dispositivo, e sim o momento em que o ACK foi registrado no banco de dados. Assim, caso essa diferença se mantenha abaixo do valor definido no cenário SNE02 (300 ms), pode-se inferir que o ACK foi efetivamente enviado dentro desse intervalo, o que valida o cenário.

5.5 Séries de testes

Para possibilitar a avaliação da escalabilidade do sistema, atributo associado ao cenário SNE03, o experimento foi dividido em três séries. A duração de cada série foi de uma hora e, em cada uma, a quantidade de dispositivos foi incrementada para elevar a taxa de solicitações ao sistema. A Tabela 9 apresenta os valores correspondentes a cada conjunto experimental.

Tabela 9 – Conjunto de testes

Blocos	**Tipo de Dispositivo**	**Qtd. (Disp.)**	**Freq. (Hz)**	**MPS (Disp.)**	**MPH (Disp.)**	**MPH (Total)**
1	PLC	1	25	25	$\sim$ 90.000	$\sim$ 540.100
	Injetora	1	100	100	$\sim$ 360.000	
	Câmera (ACK)	1	25	25	$\sim$ 90.000	
	IA (MQTT-OPC)	1	-	-	$\sim$ 100	
2	PLC	4	25	100	$\sim$ 360.000	$\sim$ 1.728.400
	Injetora	4	70	280	$\sim$ 1.008.000	
	Câmera (ACK)	4	25	100	$\sim$ 360.000	
	IA (MQTT-OPC)	1	-	-	$\sim$ 400	
3	PLC	8	25	200	$\sim$ 720.000	$\sim$ 2.880.800
	Injetora	8	50	400	$\sim$ 1.440.000	
	Câmera (ACK)	8	25	200	$\sim$ 720.000	
	IA (MQTT-OPC)	1	-	-	$\sim$ 800	

As mensagens provenientes das câmeras correspondem a mensagens de confirmação (ACK) destinadas ao servidor OPC UA. As mensagens de atuação são geradas pela IA a partir das imagens captadas pelas câmeras durante a operação de inspeção de peças e

ocorrem a cada ciclo de produção, estimado em aproximadamente 100 itens por hora. Os valores totais apresentados na tabela são aproximados, uma vez que as mensagens estão sujeitas a variações temporais e atrasos inerentes ao ambiente experimental.

As frequências das mensagens foram definidas de modo a provocar uma situação de estresse no protótipo da ED-operator RA, ao mesmo tempo em que se garantiu o correto funcionamento dos clientes MQTT e OPC UA, respeitando seus limites operacionais e evitando a introdução de ruídos no experimento. Um exemplo dessa adaptação refere-se às câmeras de injeção, cujo valor de captura foi reduzido a cada série de testes. Essa medida tornou-se necessária porque todos os simuladores foram executados no mesmo computador e, com o aumento do número de mensagens, o volume de requisições passou a ser excessivo para os clientes, o que ocasionou atrasos no envio e no processamento das mensagens.

Além das séries apresentadas neste capítulo, durante os experimentos podem ser realizadas outras séries específicas com o objetivo de analisar pontos de atenção ou investigar eventuais anomalias identificadas.

5.6 Considerações Finais

Neste capítulo, foi apresentado o protótipo desenvolvido, juntamente com as ferramentas que viabilizaram sua implementação, tais como o banco de dados InfluxDB, o ETL Influx Telegraf, o orquestrador Kubernetes, entre outras. Esse protótipo foi utilizado nos experimentos de avaliação da arquitetura e, por esse motivo, foi desenvolvido em conformidade com as recomendações da arquitetura de referência ED-operator RA proposta neste estudo.

O capítulo também descreveu a simulação industrial adotada e os testes de desempenho elaborados com base nas métricas associadas aos cenários de qualidade identificados na literatura. A descrição detalhada do desenvolvimento do protótipo, da simulação e dos testes de carga tem como objetivo garantir a replicabilidade de todo o processo experimental.

No próximo capítulo, serão apresentados os resultados obtidos a partir dos experimentos e a análise de conformidade da arquitetura ED-operator RA em relação aos cenários de qualidade definidos na RSL do Capítulo 2.

6 Resultados da Avaliação Arquitetural

6.1 Considerações Iniciais

O capítulo anterior apresentou o protótipo utilizado nos experimentos, bem como a descrição do desenho experimental. Também foram detalhadas as ferramentas empregadas no processo de prototipação, com o objetivo de viabilizar a reprodutibilidade do estudo.

O objetivo deste capítulo é demonstrar a conformidade do sistema implementado com a ED-operator RA em relação aos requisitos estabelecidos. As primeiras seções apresentam os dados obtidos nos experimentos, relacionando-os à avaliação individual dos cenários de qualidade definidos no Capítulo 2. Cada cenário é analisado de acordo com suas métricas específicas, permitindo verificar em que medida o sistema atende aos requisitos previstos. A última seção discute os conflitos identificados a partir das interações entre esses cenários de qualidade.

6.2 Interoperabilidade (SNE01)

O cenário de interoperabilidade estabelece que dispositivos que utilizam diferentes protocolos de comunicação devem ser capazes de estabelecer uma comunicação correta e eficiente, garantindo que o percentual de mensagens devidamente interpretadas permaneça acima de 99%.

Conforme a Figura 23 que apresenta dados do DASH003, a quantidade de mensagens publicadas no broker MQTT e o número de mensagens de confirmação recebidas no OPC-UA foram idênticas em todas as sequências de testes. Esse comportamento indica que nenhuma mensagem deixou de ser reconhecida pelos dispositivos durante a execução dos experimentos.

Além disso, cada mensagem de confirmação (ACKs) inclui o valor temporal referente ao momento de envio da mensagem original, informação que somente pode ser produzida caso o dispositivo tenha interpretado corretamente o conteúdo recebido. Portanto, o simples fato do ACK conter esse dado já evidencia que a mensagem foi processada adequadamente. Outro elemento que reforça essa afirmação é a equivalência entre a quantidade de comandos de descarte emitidos e o número real de itens descartados na linha, conforme apresentado nas tabelas “Número de atuações” e “Descarte”, organizadas pelos IDs dos dispositivos (dd0001 e dd0002). Tal comportamento não seria possível caso algum dispositivo apresentasse falhas na interpretação das mensagens provenientes de outros dispositivos.

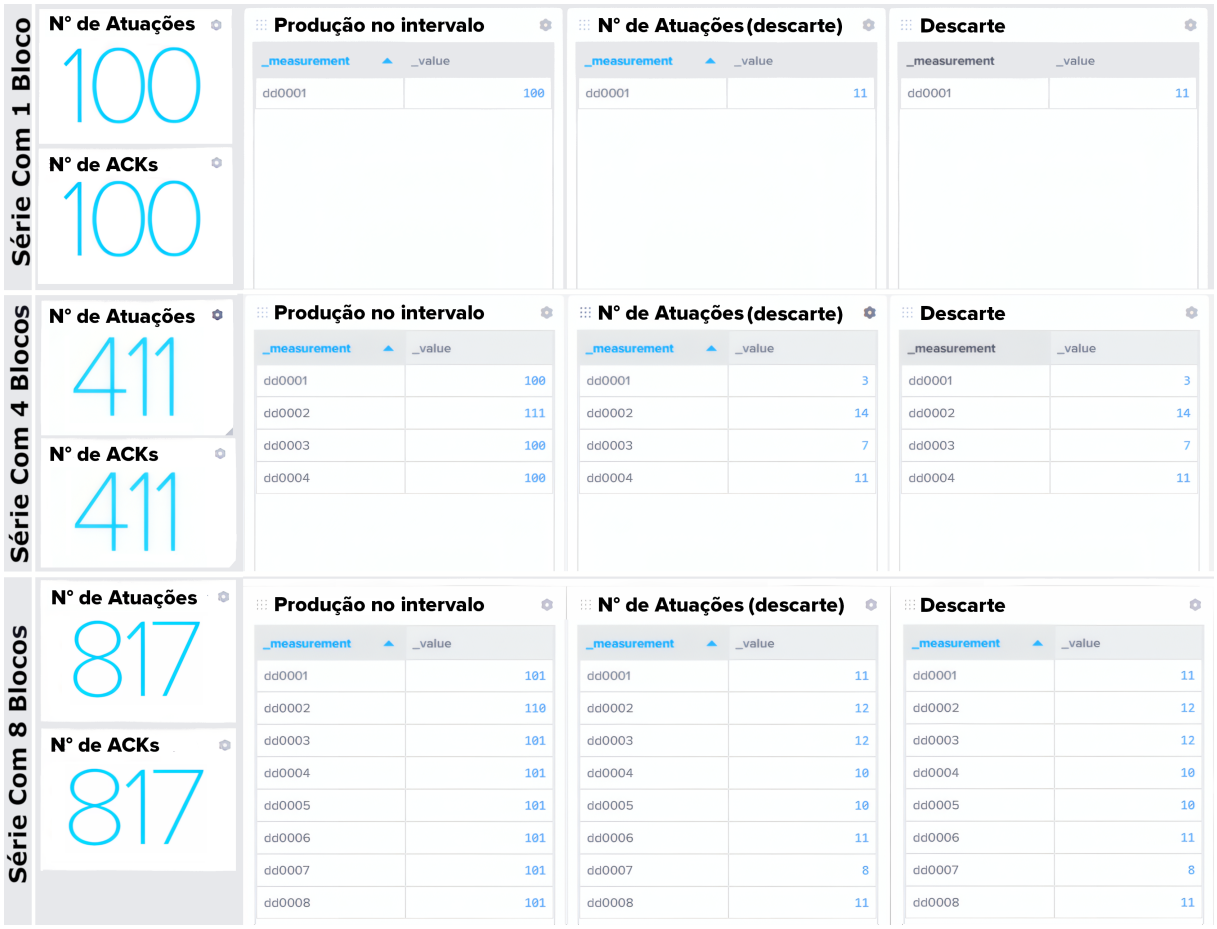

Figura 23 – Dashboard 003 nas três sequencias de testes

6.3 Baixa latência (SNE02)

O cenário de baixa latência estabelece que mais de 99,8% das mensagens devem ser transmitidas entre dispositivos em menos de 300 milissegundos. A Figura 24 apresenta séries temporais em formato de gráfico de linhas, nas quais cada série representa os tempos de latência das mensagens enviadas pela IA ao PLC em cada sequência de testes.

Observa-se que todos os pontos das séries permanecem abaixo da linha vermelha, que indica o limite de 300 milissegundos. É possível também observar que conforme o número de dispositivos aumenta, e consequentemente o número de mensagens, um pequeno nível de instabilidade é adicionado.

Para uma análise mais completa do cenário proposto, é necessário considerar também as mensagens enviadas pela câmara de injeção e as atuações enviadas pelo PLC para a câmera, conforme apresentado na Figura 25.

Como especificado no Capítulo 5, os testes referentes ao envio de mensagens do PLC para a câmara sofreram problemas de sincronização temporal. Dessa forma, utilizou-se o momento de gravação no banco de dados para o cálculo da latência. No entanto, esse

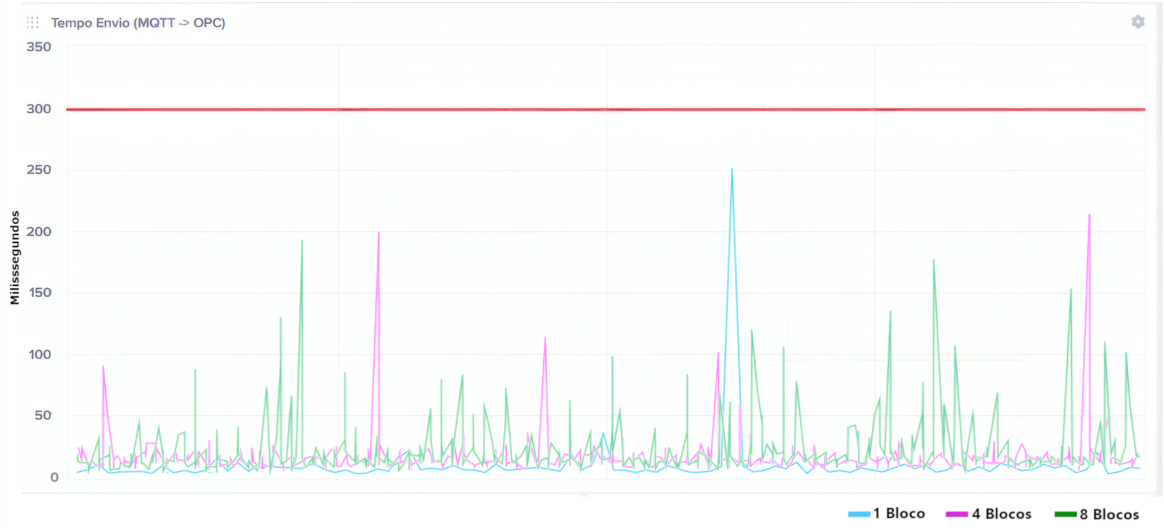

Figura 24 – Gráficos de latência das mensagens da IA (MQTT) para a linha nas três sequências de testes

cálculo apresenta valores adicionais de latência, uma vez que os dados consideram algumas operações realizadas após o recebimento da mensagem.

Diferentemente da Figura 24, a Figura 25 demonstra que todas as séries apresentaram mensagens com latência superior a 300 milissegundos (mensagens não-conformes). Entretanto, a Tabela 10 apresenta os dados consolidados das séries, sem considerar as mensagens de ACKs. Em todos os casos, o valor observado permaneceu abaixo do limite de 0,2% de tolerância estabelecido pelo cenário SNE02, indicando que a arquitetura atendeu aos requisitos desse cenário.

Tabela 10 – Resultados de latência

	N. de Mensagens	**N. Mensagens não-conformes**	**Não-conforme (%)**
Série 1 (1 bloco)			
OPC-MQTT	89006	76	0,085%
MQTT-OPC	100	0	0%
Injetora	327177	24	0,007%
Total	416283	**100**	**0,024%**
Série 2 (4 blocos)			
OPC-MQTT	356593	162	0,045%
MQTT-OPC	411	0	0%
Injetora	967435	76	0,007%
Total	1324439	**238**	**0,018%**
Série 3 (8 blocos)			
OPC-MQTT	708854	410	0,057%
MQTT-OPC	817	0	0%
Injetora	1394802	429	0,03%
Total	2104473	**839**	**0,040%**

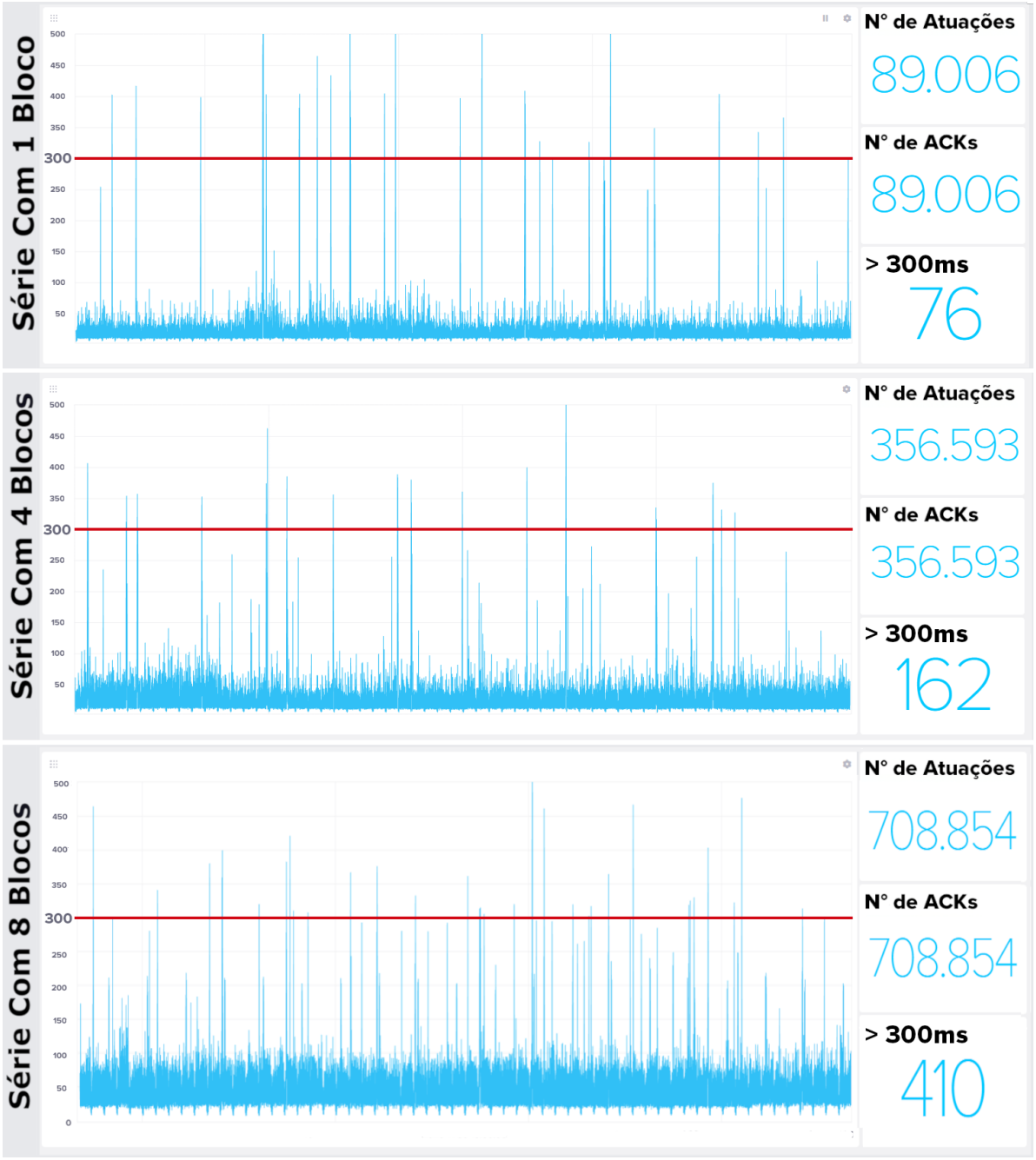

Figura 25 – Gráficos de latência de mensagens da linha para câmera nas três sequências de testes

Apesar de o cenário ter sido atendido, o armazenamento de dados se apresentou como um ponto de atenção para o arquiteto, conforme previsto pela arquitetura ED-operator RA.

Nos testes, como apresentado na Figura 26, foi possível observar que, em alguns momentos, os tempos de gravação no banco ultrapassaram 300 milissegundos. Embora esse comportamento não tenha causado prejuízos de desempenho no protótipo desenvolvido, outros sistemas podem ser impactados, o que pode provocar aumentos de latência na comunicação entre os dispositivos.

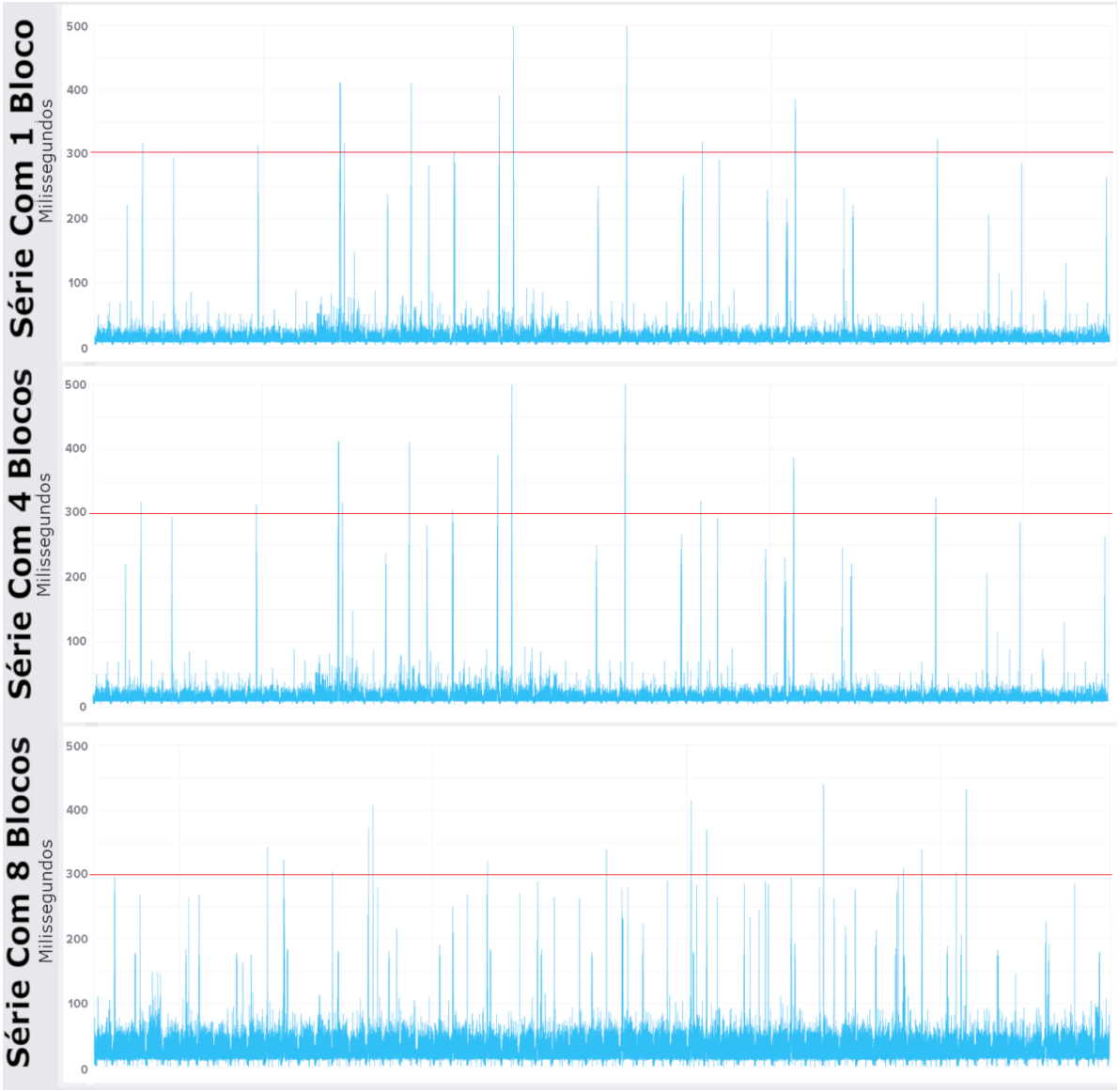

Figura 26 – Gráfico sobre o tempo de gravação no banco

Outro aspecto relevante a ser analisado é a concorrência por recursos de armazenamento estimulada por este experimento, em função da gravação das imagens geradas pela câmera no repositório. A Figura 27 apresenta um gráfico de linha com a série de médias móveis do tempo de gravação de arquivos, considerando uma janela de 500 milissegundos, para os testes com 8 blocos. Para fins de comparação, o gráfico também exibe o número de confirmações de uploads de imagens na janela de 500 milissegundos.

As áreas destacadas em amarelo na Figura 27 indicam possíveis momentos de concorrência por recursos de armazenamento. Nessas áreas, observa-se que, à medida que o tempo de latência de gravação no banco de dados aumenta, em algumas ocasiões o número de confirmações de upload diminui de seis arquivos para um intervalo entre uma e quatro confirmações. Esse comportamento pode estar associado ao atraso na gravação causado pelo maior número de imagens em processo de upload (superior a sete), o que também

atrasa a conclusão do próprio envio das imagens. Outro padrão identificado nas séries é que, pouco antes do número de confirmações de upload atingir sete, já se observa um aumento na latência de gravação no banco de dados.

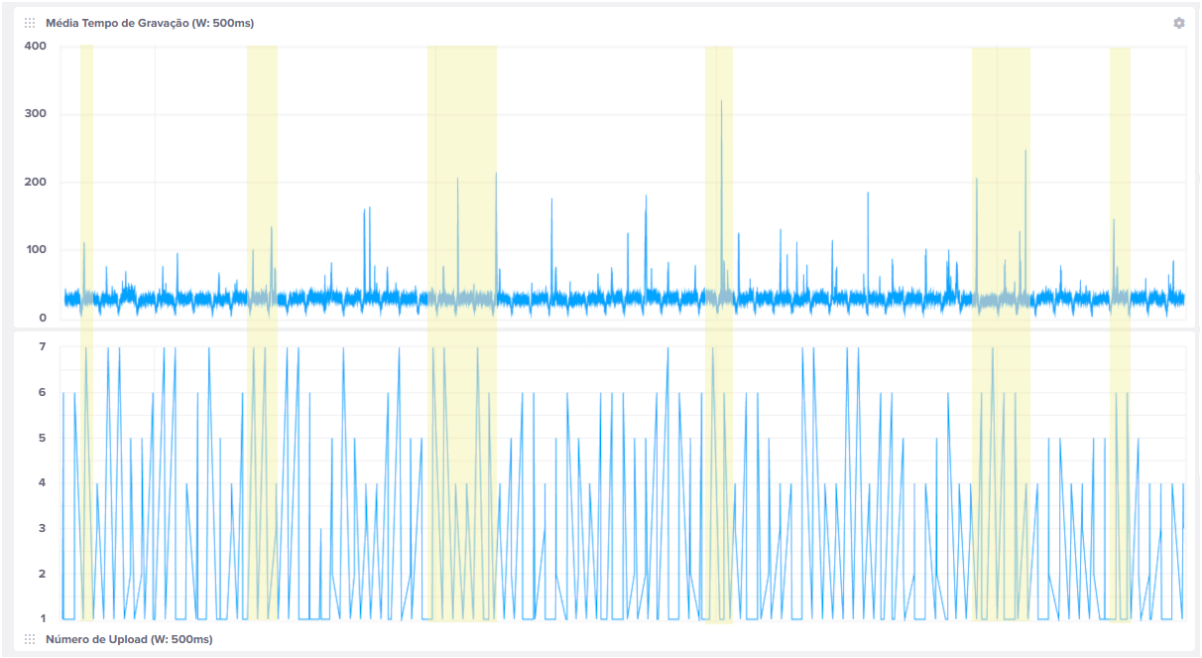

Figura 27 – Concorrência de armazenamento no dispositivo de processamento de borda

Com base nos dados apresentados na Figura 27, é possível afirmar que, para o protótipo utilizado, um limite considerado saudável é de, no máximo, seis imagens a cada 500 milissegundos. Sempre que o sistema ultrapassa esse limiar, surgem problemas de latência na gravação. Quando sete imagens são enviadas simultaneamente, os impactos ainda são moderados, afetando principalmente o tempo de gravação no banco de dados. Acima desse valor, observa-se um aumento significativo da latência tanto no banco de dados quanto no repositório de arquivos, o que caracteriza um comportamento mais crítico do sistema.

No entanto, a ED-operator RA antecipa esse tipo de cenário e recomenda a adoção de soluções de armazenamento com maior desempenho quando necessário. Em outros casos, sugere-se o uso do modo *cluster*, no qual os componentes que demandam recursos de armazenamento são distribuídos em dispositivos distintos, reduzindo a concorrência e mitigando impactos.

6.4 Escalabilidade (SNE03)

O cenário de qualidade relacionado à escalabilidade avalia a capacidade do sistema de se expandir, tanto em número de dispositivos quanto em volume de processamento (IA, dados, entre outros), sem perda de desempenho. As Figuras 28 e 29 apresentam o DASH001,

responsável por exibir os dados de uso do processador e da memória RAM do *cluster*. O uso de CPU é captado na unidade padrão do Kubernetes, o mili-núcleo, que representa frações do tempo de uso de um núcleo de processamento. Por exemplo, 100 mili-núcleos indicam que um núcleo foi utilizado por 100 milissegundos, enquanto 2000 mili-núcleos representam o uso de dois núcleos por 1000 milissegundos cada (KUBERNETES, 2025).

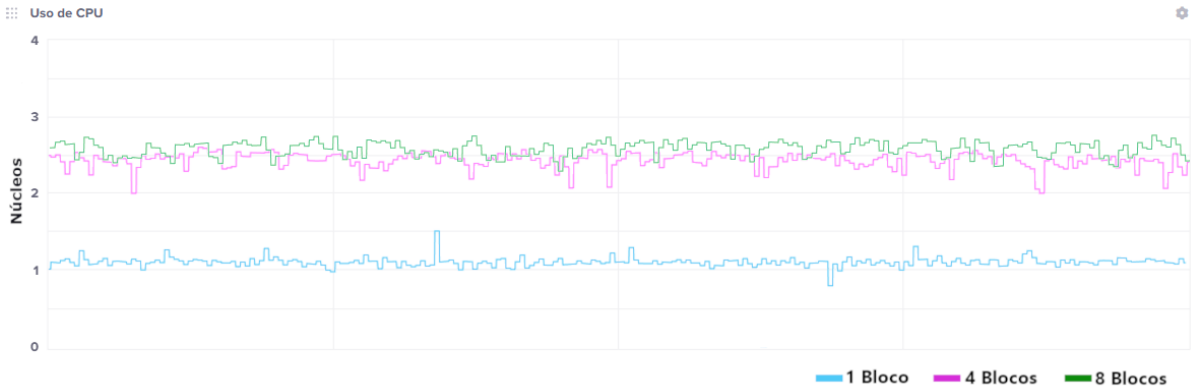

Figura 28 – Gráficos de uso de CPU

Nos dois gráficos exibidos nas Figuras 28 e 29 (DASH001), é possível observar três séries temporais representadas em formato de gráfico de linha, cada uma correspondente ao uso de CPU ou de memória durante cada sequência de testes. Em cada sequência, uma quantidade maior de dispositivos é utilizada justamente para avaliar a escalabilidade do sistema.

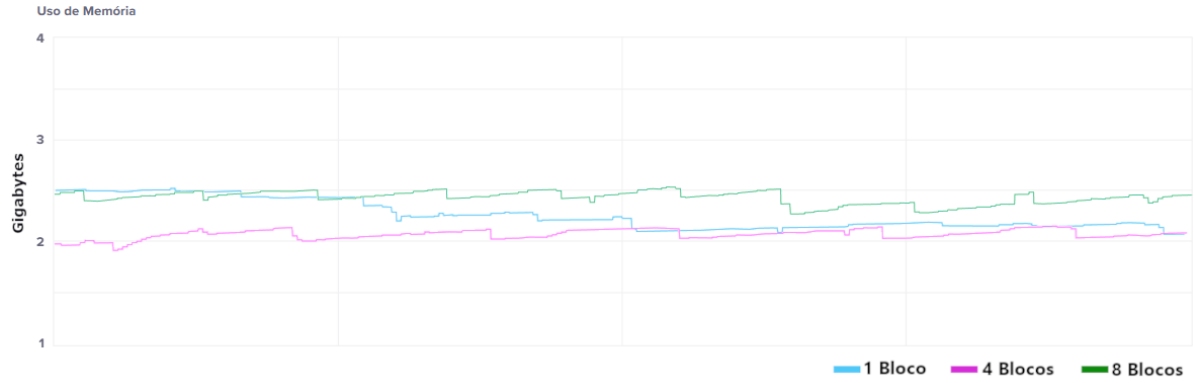

Figura 29 – Gráficos de consumo de memória

Os resultados mostram que, em todas as sequências, o uso de memória do *cluster* permaneceu estável, enquanto o uso de CPU apresentou aumentos mais expressivos. Ainda assim, mesmo na sequência com o maior número de dispositivos, o processador não foi levado ao limite, indicando margem para a adição de ainda mais dispositivos à rede.

Em todas as séries experimentais não foi observada perda de mensagens. No entanto, foram registradas mensagens cuja latência ultrapassou o limite de 300 ms, sendo estas classificadas como falhas para fins de avaliação do cenário SNE03.

A Tabela 11 apresenta o número total de mensagens processadas e a taxa de falhas correspondente a cada série. Embora tenham ocorrido mensagens com atraso superior ao limite definido, a taxa de falhas permaneceu inferior a 0,05%, valor significativamente abaixo do limite de tolerância estabelecido para o cenário (1%). Esses resultados indicam que o sistema manteve estabilidade operacional sob variação de carga, atendendo ao requisito de escalabilidade definido.

Tabela 11 – Análise da taxa de falhas

Séries	**Mensagens**	**Mensagens Falhas**	**Mensagens Atrasadas**	**Taxa de Falhas (%)**
1 (1 Bloco)	416283	0	100	**0,024**
2 (4 Blocos)	1324439	0	238	**0,018**
3 (8 Blocos)	2104473	0	839	**0,040**

A estratégia de escalabilidade baseada nos limites de CPU, memória e latência, proposta para os componentes da arquitetura que possuem acesso direto aos dispositivos da rede IoT, atuou de forma efetiva nas séries com quatro e oito blocos de produção. Assim que o teste foi iniciado nessas séries, uma nova instância do serviço foi adicionada, uma vez que a instância original ultrapassou o limite de consumo de CPU estabelecido. Apesar dos pontos de não conformidade, as médias de latência se mantiveram baixas; assim, não houve aumento de instâncias por meio desse critério.

O protótipo foi configurado para disponibilizar ao Agente IoT OPC UA apenas um núcleo do processador. Assim, é possível garantir mais recursos computacionais para essa operação, criando novas instâncias do agente. Por essa razão, o gráfico de uso de CPU apresentado na Figura 28 se limita ao valor um no eixo Y na série de apenas 1 bloco.

Apesar da estratégia de escalabilidade adotada pela ED-operator RA demonstrar ser eficaz, ela permanece limitada pela quantidade de dispositivos disponíveis para executar os processamentos, um cenário típico de sistemas baseados em edge computing.

Por esse motivo, esse cenário considera o uso cooperativo de computação em nuvem como estratégia de mitigação. A arquitetura ED-operator RA propõe essa cooperação por meio do compartilhamento da execução de contêineres com a nuvem. Esse compartilhamento ocorre quando os recursos do *cluster* não são suficientes para manter todos os contêineres em execução. Nessa situação, os contêineres de menor prioridade, conforme o perfil de prioridade definido pelo ED-operator RA, são preemptados pelo Kubernetes. O operador do sistema monitora os eventos de preempção e envia para a nuvem, por meio do comando *dispatch*, o processo que foi preemptado na borda, garantindo sua continuidade de execução no ambiente de nuvem.

6.5 Segurança e Privacidade (S04)

O cenário de qualidade relacionado à segurança e à privacidade aborda a proteção dos dados por meio de mecanismos de autenticação e criptografia. A conformidade com esses requisitos é evidenciada pela utilização de TLS 1.2 em conjunto com o protocolo MQTT pela câmera de inspeção, bem como pelo uso do modo de segurança "SignAndEncrypt" no OPC UA empregado pela câmera da injeção. A Figura 30 ilustra os pacotes criptografados transmitidos pela câmera de inspeção, capturados por meio da ferramenta Wireshark.

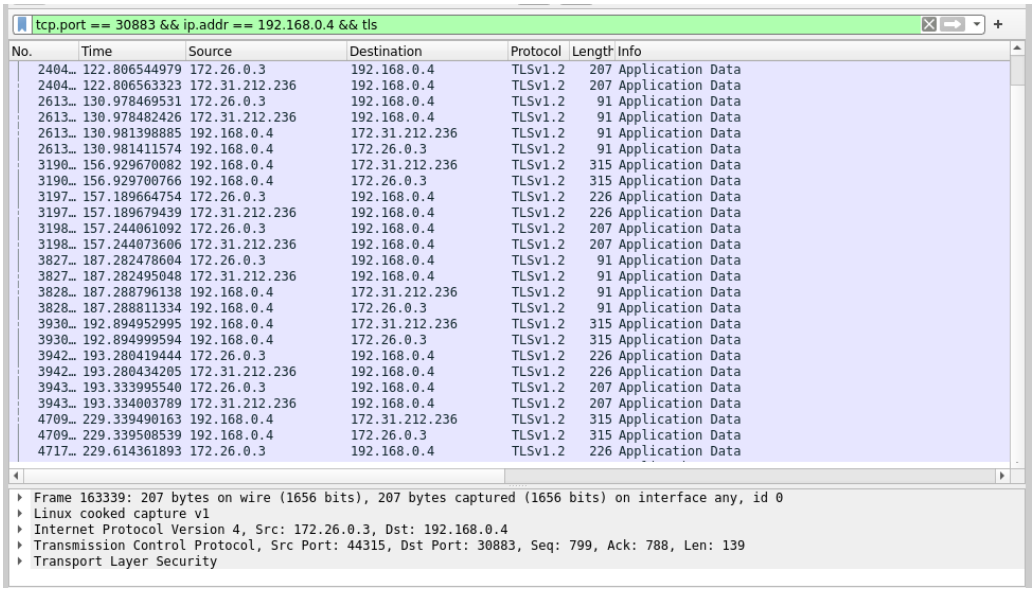

Figura 30 – Pacotes de dispositivos em TLS 1.2

Além disso, dispositivos com menor capacidade de processamento ou de natureza legada podem recorrer a redes locais isoladas e seguras como forma de mitigar riscos de segurança. Esse é o caso do PLC da linha, que opera utilizando o modo "None" do OPC UA.

6.6 Disponibilidade após falhas (SNE05)

Esse cenário avalia a capacidade do sistema de continuar operando localmente mesmo em caso de falhas na conexão com a nuvem. Para isso, foi realizado um teste em que a conexão foi intencionalmente interrompida. A Figura 31 apresenta um experimento com duração de quarenta minutos, em que a área sombreada em rosa representa o período em que o sistema permaneceu desconectado da nuvem.

A série temporal não apresenta falhas entre esses dois pontos, e todas as mensagens recebidas foram devidamente respondidas com ACK, demonstrando que o sistema permanece funcional mesmo na ausência de comunicação com a nuvem. A Tabela 12 apresenta

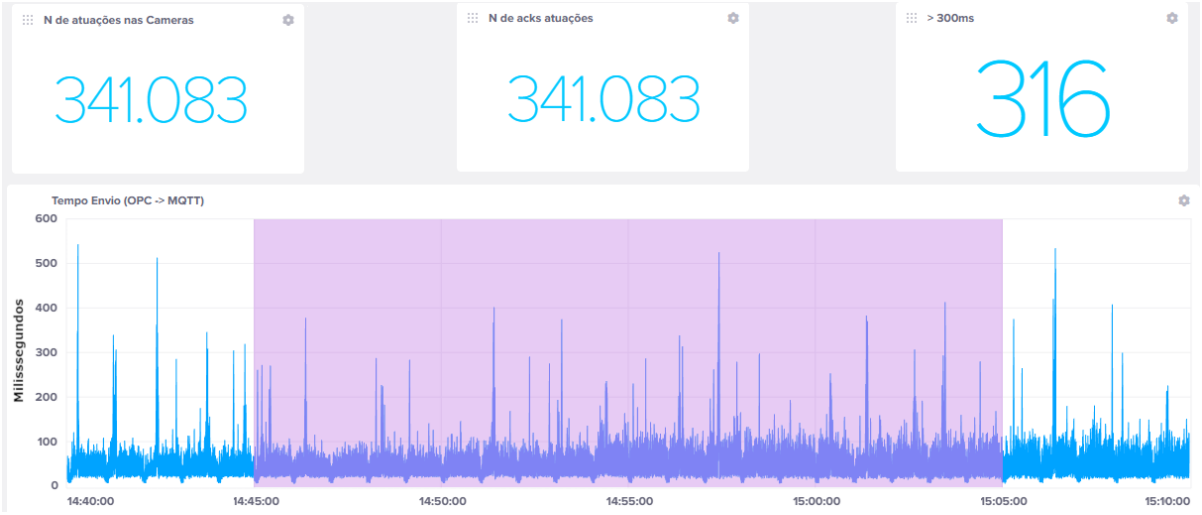

Figura 31 – Latências do teste com 8 blocos de produção com interrompimento da conexão com a nuvem

as informações de latência necessárias para demonstrar a conformidade do sistema com o cenário SNE02, mesmo sem conexão com a nuvem.

Tabela 12 – Resultados do teste sem a conexão com a nuvem

	N. de mensagens	**N. Mensagens Não-conformes**	**Não-conforme (%)**
OPC-MQTT	341.083	316	0,093%
MQTT-OPC	404	0	0%
Injetora	660.273	204	0,031%
Total	**1.001.760**	**520**	**0,051%**

Durante o teste, foram enviadas 1.001.760 mensagens, das quais 520 apresentaram latência superior a 300 milissegundos, o que corresponde a aproximadamente 0,051% do total. Esse valor encontra-se dentro da tolerância estabelecida pelo cenário SNE02.

Após o restabelecimento da conexão com a nuvem, não foi observado aumento anormal na latência das mensagens, mesmo com a utilização de um cartão de memória classe V30, cujo desempenho inferior poderia impactar negativamente no sistema. Esse resultado foi alcançado graças ao sincronismo particionado recomendado pela arquitetura de referência ED-operator RA (Subseção 4.5.4).

Com base nos resultados do experimento de interrupção da conexão com a nuvem, é possível afirmar que o sistema derivado da arquitetura de referência ED-operator RA mantém sua conformidade com o cenário SNE05.

6.7 Manutenibilidade (SNE06)

Esse cenário exige que o sistema seja capaz de realizar atualizações de forma remota, sem causar indisponibilidade, por meio de automações de CI/CD executadas localmente na borda. Essa capacidade é viabilizada pelo Operador, que recebe mensagens via MQTT enviadas pela nuvem e as traduz em comandos para o Kubernetes. Com os comandos apresentados na Figura 17, é possível realizar a implantação, a atualização ou a remoção de serviços diretamente no *cluster* de borda.

Os microsserviços customizados também podem disponibilizar comandos próprios para alteração do seu próprio estado ou de suas configurações. Um exemplo é o Agente IoT OPC UA, que oferece comandos para adicionar, listar e remover dispositivos da lista de servidores OPC UA conhecidos.

Para garantir a disponibilidade do sistema, as atualizações seguem uma estratégia de *rollout* que mantém a versão antiga em execução até que uma instância da nova versão esteja pronta e funcional. Dessa forma, caso ocorra uma falha durante o processo de atualização, os dispositivos da rede IoT não sofrerão impacto direto. A arquitetura ED-operator RA propõe o uso de contêineres, o que reforça a disponibilidade do sistema, já que o isolamento entre contêineres assegura, na maioria dos casos, que uma instância com mau funcionamento não interfira em outra que esteja operando corretamente.

6.8 Avaliação das interações entre cenários

Todos os cenários apresentados nesse capítulo são cenários diretos, ou seja, espera-se que a arquitetura de referência proposta por este estudo ofereça suporte direto a eles. Contudo, ao buscar atender a múltiplos cenários simultaneamente, podem surgir conflitos entre eles. A análise dos seis cenários identificou quatro possíveis conflitos, descritos a seguir:

1. O processo de tradução de protocolos proposto no cenário SNE01 gera um leve aumento na latência das respostas do sistema, o que pode interferir no desempenho esperado pelo cenário SNE02;

2. O cenário SNE03 permite a transferência de serviços para a nuvem. No entanto, se os serviços de tradução forem movidos, o cenário SNE02 poderá ser afetado de forma substancial;

3. O uso de criptografia, conforme proposto no cenário SNE04, também acarreta um aumento na latência do sistema, podendo, assim, interferir no cenário SNE02;

4. Em situações de indisponibilidade da nuvem, conforme representado no cenário SNE05, a funcionalidade de compartilhamento de processos com a nuvem (SNE03) é comprometida por falta de comunicação.

Os testes apresentados na seção anterior demonstram que o impacto do primeiro e do terceiro conflito é baixo. A principal evidência dessa afirmação pode ser observada na Figura 25, que apresenta latências baixas na comunicação entre o PLC da linha e a câmera. Durante essa comunicação, ambos os processos críticos mencionados nos conflitos são empregados. As mensagens enviadas do PLC para a câmera passam por conversão de OPC UA para MQTT e, em seguida, são criptografadas para transmissão via *broker* até a câmera, que precisa decifrá-las. Ainda assim, as latências observadas mantêm a arquitetura em conformidade com o cenário SNE02.

No entanto, o Agente IoT permanece como um ponto de atenção, pois sua eficiência pode impactar diretamente o comportamento do sistema. A arquitetura ED-operator RA prevê e recomenda práticas que, quando adotadas adequadamente, podem mitigar possíveis gargalos nesse componente.

O segundo conflito apresenta maior gravidade, pois pode comprometer a aderência ao cenário SNE02. No entanto, a adoção da política de perfil de priorização pela ED-operator RA mitiga os riscos desse cenário, já que essa ferramenta permite dar preferência ao processamento local aos microsserviços essenciais, como os Agentes IoT. Dessa forma, em situações de sobrecarga, apenas os microsserviços de menor prioridade poderiam ser delegados à nuvem.

O quarto conflito tem sua gravidade ampliada caso a borda também esteja sobrecarregada no mesmo momento; pode haver uma indisponibilidade total do sistema. Para mitigar esse risco, o Operador pode optar por interromper microsserviços com um perfil de prioridade baixa, mantendo o funcionamento parcial do sistema. Alternativamente, os sistemas podem compartilhar serviços entre dispositivos de borda, como forma de reduzir a dependência da nuvem como previsto pela arquitetura.

6.9 Avaliação Comparativa com Arquiteturas de Referência Existentes

A RSL apresentada no Capítulo 2 evidenciou uma lacuna na literatura no que se refere ao papel do dispositivo de processamento de borda. Embora seu uso seja previsto, as demais arquiteturas de referência não especificam quais tipos de processos podem ser executados por esse dispositivo nem de que forma isso deve ocorrer.

Nesse contexto, a arquitetura de referência proposta neste estudo atribui ao dispositivo de processamento de borda o papel de *microcloud*, conforme mencionado por

Nicholson et al. (2019). A Tabela 13 destaca a principal distinção entre a ED-operator RA e as arquiteturas de referência industriais existentes. Enquanto o RAMI 4.0 e a IIRA fornecem primordialmente *frameworks* conceituais que não abrangem o escopo técnico, e o FAR-Edge introduz considerações parciais mas sem especificar mecanismos de implantação operacional, estratégias de gerenciamento de ciclo de vida ou estratégias de interoperabilidade no nível necessário para uma implementação direta.

Tabela 13 – Comparação sistemática entre a ED-operator RA e outras arquiteturas

Preocupação	**RAMI 4.0**	**IIRA**	**FAR-Edge**	**ED-operator RA**
Dispositivo de edge tratado como ambiente de execução (micro-nuvem)	N/A	N/A	Parcial	**Modelo explícito de execução com microsserviços conteinerizados**
Gerenciamento de implantação e ciclo de vida no edge	N/A	N/A	Parcial	**Implantação centrada em Operator com gerenciamento remoto do ciclo de vida**
Estratégia de orquestração de contêineres	N/A	N/A	Parcial	**Orquestração baseada em conteinerização integrada à arquitetura**
Pipeline de tradução de protocolos	Conceitual	Conceitual	Parcial	**Componente arquitetural explícito com fluxo de dados definido**
Mecanismo de atualização remota e manutenção	N/A	N/A	N/A	**Separação Operator/Orchestrator permitindo atualizações remotas**
Estratégia de offloading edge-nuvem	Conceitual	Conceitual	Parcial	**Offloading baseado em prioridade integrado à arquitetura**
Tratamento de dados multimídia no edge	N/A	N/A	N/A	**Repositório dedicado para arquivos multimídia**
Nível de detalhamento técnico de implantação	Conceitual	Conceitual	Parcial	**Especificação em nível de contêiner, protocolo e implantação**

Ao adotar tecnologias de conteinerização e permitir atualizações remotas de serviços e aplicações, a ED-operator possibilita que a camada de borda ofereça funcionalidades tradicionalmente associadas à nuvem. Essa característica a diferencia das demais arquiteturas encontradas na literatura, como RAMI 4.0, IIRA e Far-Edge RA, por aprofundar o nível técnico da implementação. Dessa forma, a arquitetura também fornece uma visão

complementar a outras arquiteturas, facilitando a adoção de *edge computing* também em cenários industriais que já utilizam essas RAs.

6.10 Considerações Finais

Com base nos dados obtidos durante o experimento controlado, foi possível avaliar o nível de conformidade da arquitetura em relação aos cenários SNE01, SNE02, SNE03 e SNE05. Este capítulo também apresentou as análises referentes aos cenários SNE04 e SNE06, cuja conformidade pôde ser verificada por meio de avaliações objetivas das capacidades do sistema, como a viabilização de processos de CI/CD e o suporte à criptografia para melhorias de segurança e privacidade. Os resultados apontam que a arquitetura ED-operator RA, quando empregada corretamente e seguindo as suas visões e recomendações, dentro de uma ambiente controlado é capaz de produzir um software alinhado aos atributos de qualidade exigidos em ambientes de IoT Industrial.

Apesar dos resultados positivos obtidos com a ED-operator RA, a validação foi conduzida em um ambiente experimental com rede controlada. Em cenários reais de implantação, podem ocorrer discrepâncias decorrentes de variabilidades inerentes à infraestrutura, carga de tráfego e condições operacionais.

O capítulo também discutiu os conflitos existentes entre os cenários, que podem eventualmente comprometer a conformidade da ED-operator RA com um ou mais cenários de qualidade. Entre eles, o cenário SNE002 foi identificado como o mais crítico. Entretanto, os dados demonstram que, mesmo diante desses conflitos, é possível manter a conformidade geral da arquitetura.

As análises realizadas ainda abrem espaço para melhorias futuras, seja para reduzir os pontos de não conformidade identificados, seja para aprimorar o desempenho geral do sistema.

7 Conclusão

A IoT é uma ferramenta fundamental na Indústria 4.0, pois permite que objetos físicos se conectem à Internet, reduzindo a necessidade de intervenções humanas nos sistemas. Essa conectividade possibilita a comunicação *Machine to Machine*. No entanto, a IoT, principalmente no contexto industrial, exige considerável poder computacional para lidar com grandes volumes de dados provenientes de diversos dispositivos, demanda que pode ser atendida pela computação em nuvem. Ainda assim, aplicações industriais possuem restrições rígidas de tempo, o que representa um desafio significativo para sistemas baseados exclusivamente em nuvem.

Para superar o desafio da alta latência associada à computação em nuvem, este estudo adotou o conceito de *edge computing* com o objetivo de propor uma arquitetura de referência capaz de disponibilizar mecanismos que reduzam a latência das mensagens em sistemas de IoT Industrial. Ao elaborar essa arquitetura de referência, o estudo também contribui como material de apoio à discussão sobre *edge computing* no contexto industrial, um tema que vem ganhando relevância nos últimos anos, conforme evidenciado pela Revisão Sistemática da Literatura apresentada no Capítulo 2.

O processo de criação foi fundamentado no estudo de (GALSTER; AVGERIOU, 2011), que propõe seis etapas para a construção de uma arquitetura de referência. Durante esse processo, foi definido o tipo da arquitetura, classificando a arquitetura de referência ED-operator RA como uma arquitetura do Tipo 01, referente às arquiteturas clássicas de padronização.

Para atender à terceira etapa proposta por (GALSTER; AVGERIOU, 2011) e analisar o estado da arte relacionado ao tema, foi realizada uma Revisão Sistemática da Literatura (RSL). A partir dessa revisão, foi possível identificar informações essenciais para a criação da arquitetura de referência, incluindo padrões e protocolos de comunicação relevantes ao contexto de pesquisa, além de softwares e *middlewares* IoT que oferecem artefatos necessários e suficientes para a extração de decisões arquiteturais. A RSL permitiu contextualizar os arquitetos de software no domínio abordado pelo estudo, oferecendo-lhes uma visão estruturada e fundamentada sobre o tema.

A ED-operator RA é um modelo de arquitetura criado para facilitar o desenvolvimento de softwares industriais, oferecendo suporte a arquitetos que desejam construir sistemas baseados no paradigma de *Edge Computing*. Essa arquitetura define o dispositivo de processamento como um provedor de *microcloud*, permitindo que tarefas normalmente executadas apenas na nuvem também sejam realizadas na borda. Com isso, a ED-operator RA apresenta um caráter dinâmico, possibilitando que os sistemas que seguem suas

diretrizes se adaptem aos diferentes cenários encontrados na indústria.

Um ponto de atenção da ED-operator RA é a simplificação das atividades de manutenção, tornando viáveis atualizações remotas e reduzindo a necessidade de intervenção humana em campo. Todos esses elementos são projetados com o objetivo de não aumentar a latência do sistema.

Os resultados mostram que a ED-operator RA atende de maneira consistente aos requisitos definidos, apresentando capacidade de escalabilidade e promovendo interoperabilidade na rede IoT, ao mesmo tempo em que mantém a baixa latência característica de sistemas baseados em computação de borda, além de assegurar a segurança e privacidade através de criptografia.

Outra contribuição deste estudo, é o fato de poder apoiar arquitetos de software interessados em desenvolver sua própria arquitetura de referência, uma vez que apresenta uma metodologia estruturada e reproduzível. A metodologia pode ser reaproveitada integralmente ou adaptada de forma parcial, pois oferece um conjunto de ferramentas úteis para a obtenção empírica de dados, assim como para a construção e avaliação de arquiteturas de referência.

Este estudo também pode ser utilizado para apresentar, de forma estruturada, todo o processo de desenvolvimento de uma arquitetura de referência, ou mesmo de uma arquitetura concreta aplicada em sistemas reais, servindo como material de apoio para indivíduos que não estejam familiarizados com esse tipo de processo.

Em suma, os artefatos produzidos por este estudo contribuem para o avanço do paradigma de *edge computing* na indústria, incentivando sua adoção por meio de uma arquitetura de referência que apresenta, de forma clara e objetiva, as decisões envolvidas no desenvolvimento de sistemas industriais baseados nesse modelo. Em um cenário cada vez mais dinâmico, no qual respostas rápidas são fundamentais, as arquiteturas de referência tornam-se ferramentas essenciais para orientar soluções tecnológicas consistentes e eficientes.

7.1 Estudos Futuros

As discussões e análises conduzidas ao longo deste trabalho podem servir como base para futuras pesquisas, fomentando o desenvolvimento dos diversos temas abordados e ampliando o entendimento sobre práticas, desafios e oportunidades no contexto da computação de borda aplicada à Indústria 4.0. Esses resultados também podem apoiar estudos voltados à evolução desse paradigma no contexto da Indústria 5.0, em que a integração entre humanos, máquinas e IA tende a se tornar ainda mais relevante.

Embora a arquitetura ED-operator RA tenha demonstrado conformidade com todos

os cenários de qualidade propostos, ainda existem limitações que podem ser exploradas em trabalhos futuros. O protótipo utilizado nos experimentos foi desenvolvido para operar em modo unitário, dessa forma seria necessário conduzir novos experimentos para avaliar a conformidade da arquitetura em modo *cluster*, verificando se as recomendações propostas pela ED-operator RA permanecem válidas nesse contexto distribuído.

Além disso, o estudo não apresenta uma comparação de desempenho entre sistemas baseados exclusivamente em nuvem e o protótipo operando sob o paradigma de *Edge Computing*. Tal comparação poderia fornecer evidências quantitativas adicionais sobre os ganhos obtidos com a abordagem proposta.

Outro fator relevante que representa um passo essencial para consolidar a maturidade e a aplicabilidade prática da arquitetura seria a aplicação da ED-operator RA em um ambiente industrial real.

Como a ED-operator RA apresenta características importantes para qualquer rede IoT, surge a oportunidade de empregá-la em contextos associados ou até mesmo distintos do ambiente industrial. Sua capacidade de realizar atualizações remotas, aliada à operação distribuída e ao processamento na borda, reforça seu potencial de adoção em outros domínios. Um exemplo é o contexto de *smart farm*, em que dispositivos distribuídos por áreas agrícolas coletam e processam dados em campo, possibilitando respostas mais rápidas e maior autonomia operacional. Outro cenário promissor envolve redes de sensores instaladas em regiões de difícil acesso, como áreas de preservação ambiental, onde a operação autônoma e o processamento local são fundamentais para monitoramento contínuo.

Outro ponto de melhoria é a adoção de técnicas capazes de localizar dispositivos em campo, permitindo que, em cenários com múltiplos dispositivos de processamento disponíveis, o sistema estabeleça a conexão com o nó mais próximo. Essa capacidade de seleção dinâmica pode reduzir ainda mais a latência apresenta pelo sistema, aprimorando o desempenho da arquitetura.

7.2 Publicações

A revisão sistemática da literatura apresentada no Capítulo 2 deste estudo gerou um artigo que foi publicado na conferência *The Sixteenth International Conference on Information, Intelligence, Systems and Applications* com o nome "IoT Solutions with Edge Computing in Industry 4.0: A review".

Referências

AAZAM, M.; ZEADALLY, S.; HARRAS, K. A. Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Transactions on Industrial Informatics*, IEEE Computer Society, v. 14, n. 10, p. 4674 – 4682, 2018. ISSN 15513203. Citado na página 18.

AHELEROFF, S. et al. Digital twin as a service (dtaas) in industry 4.0: An architecture reference model. *Advanced Engineering Informatics*, Elsevier Ltd, v. 47, 2021. ISSN 14740346. Citado na página 34.

AL-JAROODI, J.; MOHAMED, N.; JAWHAR, I. A service-oriented middleware framework for manufacturing industry 4.0. *ACM SIGBED Review*, Association for Computing Machinery, v. 15, n. 5, p. 29 – 36, 2018. ISSN 15513688. Citado na página 19.

ALDALUR, I. et al. Advantages of arrowhead framework for the machine tooling industry. In: *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. [S.l.: s.n.], 2020. p. 4511–4518. ISSN 2577-1647. Citado 6 vezes nas páginas 25, 26, 29, 30, 47 e 49.

ALMADA-LOBO, F. The industry 4.0 revolution and the future of manufacturing execution systems (mes). *Journal of Innovation Management*, v. 3, p. 16–21, 01 2016. Citado na página 19.

ALPINE LINUX DEVELOPMENT TEAM. *Alpine Linux – Abount*. 2025. <https://www.alpinelinux.org/about>. Acesso em: 17 nov. 2025. Citado na página 69.

AMAZON WEB SERVICES. *AWS IoT Core*. 2024. <https://aws.amazon.com/pt/iot-core/>. Acessado em 22 de junho de 2024. Citado na página 31.

AMAZON WEB SERVICES. *Amazon S3 objects overview*. 2025. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingObjects.html>. Acesso em: 13 dez. 2025. Citado na página 64.

ANGELOV, S.; GREFEN, P.; GREEFHORST, D. A classification of software reference architectures: Analyzing their success and effectiveness. In: *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*. [S.l.: s.n.], 2009. p. 141–150. Citado 2 vezes nas páginas 41 e 42.

ANSIBLE. *Documentação da Comunidade Ansible*. 2025. <https://docs.ansible.com/>. Acessado em: 8 de maio de 2025. Disponível em: <https://docs.ansible.com/>. Citado na página 33.

BARLETTA, M.; CINQUE, M.; MARTINO, C. D. Sla-driven software orchestration in industry 4.0. *IEEE Internet of Things Magazine*, v. 5, n. 4, p. 136–141, December 2022. ISSN 2576-3199. Citado na página 29.

BARTH, A.; BALAKRISHNA, B.; WILLNER, A. Configurable mapping of ethercat field-level devices to opc ua. In: *2022 International Young Engineers Forum (YEF-ECE)*. [S.l.: s.n.], 2022. p. 57–62. Citado 2 vezes nas páginas 25 e 27.

BARZEGARAN, M.; POP, P. The fora european training network on fog computing for robotics and industrial automation. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. [S.l.: s.n.], 2023. p. 1–6. ISSN 1558-1101. Citado na página 29.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2012. ISBN 0321815734. Citado 2 vezes nas páginas 24 e 44.

BAZI, N. E. et al. Generic multi-layered digital-twin-framework-enabled asset lifecycle management for the sustainable mining industry. *Sustainability*, v. 15, p. 3470, 02 2023. Citado 2 vezes nas páginas 19 e 29.

BREQUE, M.; NUL, L. D.; PETRIDIS, A. *Industry 5.0: Towards a sustainable, human-centric and resilient European industry*. [S.l.], 2021. Accessed: 2026-01. Disponível em: <https://op.europa.eu/en/publication-detail/-/publication/468a892a-5097-11eb-b59f-01aa75ed71a1>. Citado na página 19.

BROWN, S. *The C4 model for visualising software architecture*. 2011. <https://c4model.com/>. Accessed: 2025-04-06. Citado 2 vezes nas páginas 43 e 48.

BUTTE, V. K.; BUTTE, S. An end to end edge to cloud data and analytics strategy. In: *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2022. p. 1–6. Citado 2 vezes nas páginas 14 e 18.

CAI, M. et al. Scaling persistent in-memory key-value stores over modern tiered, heterogeneous memory hierarchies. *IEEE Transactions on Computers*, v. 74, n. 2, p. 495–509, 2025. Citado na página 59.

CH., G. D. S. et al. Open middleware proposal for iot focused on industry 4.0. In: *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*. [S.l.: s.n.], 2018. p. 1–6. Citado 2 vezes nas páginas 19 e 25.

CLOUTIER, R. et al. The concept of reference architectures. *Systems Engineering*, Wiley, v. 13, n. 1, p. 14–27, 2010. Citado na página 14.

CODESYS GmbH. *CODESYS GROUP: The comprehensive software suite for automation technology*. 2025. <https://www.codesys.com/>. CODESYS é a plataforma de desenvolvimento líder para programar controladores industriais (PLCs) de acordo com a norma IEC 61131-3. Acesso em: 12 de novembro de 2025. Citado na página 67.

DIAZ-CACHO, M. et al. Educational test-bed for maintenance 4.0. In: *2022 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.: s.n.], 2022. p. 1310–1315. ISSN 2165-9567. Nenhuma citação no texto.

DINTÉN, R.; MARTÍNEZ, P. L.; ZORRILLA, M. Model-based tool for the design, configuration and deployment of data-intensive applications in hybrid environments: An industry 4.0 case study. *Journal of Industrial Information Integration*, v. 41, p. 100668, 2024. ISSN 2452-414X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2452414X24001122>. Citado 3 vezes nas páginas 13, 14 e 29.

DOCKER. *Docker: Plataforma para Desenvolvimento Acelerado de Aplicações em Contêineres*. 2025. <https://www.docker.com/>. Acessado em: 8 de maio de 2025. Disponível em: <https://www.docker.com/>. Citado na página 33.

ELSEVIER's. *Scopus Database*. 2025. <https://www.scopus.com>. Elsevier's abstract and citation database. Citado na página 21.

FERRARI, P. et al. Delay estimation of industrial iot applications based on messaging protocols. *IEEE Transactions on Instrumentation and Measurement*, v. 67, n. 9, p. 2188–2199, Sep. 2018. ISSN 1557-9662. Citado na página 33.

FIWARE FOUNDATION E.V. *FIWARE Foundation*. 2024. <https://www.fiware.org/>. Acessado em 22 de junho de 2024. Citado 4 vezes nas páginas 5, 31, 32 e 33.

FOURNARIS, A. P. et al. Introducing hardware-based intelligence and reconfigurability on industrial iot edge nodes. *IEEE Design & Test*, v. 36, n. 4, p. 15–23, Aug 2019. ISSN 2168-2364. Nenhuma citação no texto.

FOUTO, P.; PREGUIçA, N.; LEITãO, J. Large-scale causal data replication for stateful edge applications. In: *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2024. p. 209–220. Citado na página 59.

FU, Y.; QIU, X.; WANG, J. F2mc: Enhancing data storage services with fog-tomulticloud hybrid computing. In: *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. [S.l.: s.n.], 2019. p. 1–6. Citado na página 59.

GALSTER, M.; AVGERIOU, P. Empirically-grounded reference architectures: a proposal. In: *Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS*. [S.l.]: ACM, 2011. p. 153–158. Citado 6 vezes nas páginas 14, 39, 41, 43, 44 e 89.

GRIESEMER, R.; PIKE, R.; THOMPSON, K. *The Go Programming Language*. 2009. <https://golang.org>. Acesso em: 5 ago. 2025. Citado na página 64.

HABIB, K. et al. An aggregated data integration approach to the web and cloud platforms through a modular rest-based opc ua middleware. *Sensors*, MDPI, v. 22, n. 5, 2022. ISSN 14248220. Citado 3 vezes nas páginas 25, 31 e 32.

HAMM, A.; WILLNER, A.; SCHIEFERDECKER, I. Edge computing: A comprehensive survey of current initiatives and a roadmap for a sustainable edge computing development. In: . [S.l.: s.n.], 2019. Nenhuma citação no texto.

HUAWEI TECHNOLOGIES CO. Ltd. *Cloud Computing Technology*. Singapore: Springer, 2023. ISBN 978-981-19-3026-3. Disponível em: <https://link.springer.com/book/10.1007/978-981-19-3026-3>. Citado na página 17.

IEC. *IEC 61131-3:2013 – Programmable Controllers – Part 3: Programming Languages*. Geneva, Switzerland, 2013. Citado na página 68.

IEEE. *IEEE Xplore Digital Library*. 2025. <https://ieeexplore.ieee.org>. Digital library for engineering and technology research. Citado na página 21.

ILLA, P. K.; PADHI, N. Practical guide to smart factory transition using iot, big data and edge analytics. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 6, p. 55162 – 55170, 2018. ISSN 21693536. Citado na página 32.

INFLUXDATA. *InfluxData: The platform for building and operating time series applications*. 2025. <https://www.influxdata.com/>. Acesso em 5 de agosto de 2025. Citado na página 63.

INFLUXDATA. *Telegraf: An open source plugin-driven agent for collecting, processing, aggregating, and reporting time series data*. [S.l.], 2025. Acesso em 5 de agosto de 2025. Citado na página 63.

ISO/IEC/IEEE. *ISO/IEC/IEEE 42010:2011 — Systems and software engineering — Architecture description*. Geneva, Switzerland: International Organization for Standardization, 2011. Citado 2 vezes nas páginas 43 e 44.

ISO/IEC/IEEE. *ISO/IEC/IEEE 42010:2022 — Systems and software engineering — Architecture description*. Geneva, Switzerland: International Organization for Standardization, 2022. Citado 2 vezes nas páginas 43 e 44.

JOHNSON, N. P.; DENES, P. The ladder diagram (a 100+ year history). *The American Journal of Cardiology*, v. 101, n. 12, p. 1801–1804, 2008. ISSN 0002-9149. Disponível em: <https://www.sciencedirect.com/science/article/pii/S000291490800386X>. Citado na página 68.

JUNG, H. et al. Application of machine learning techniques in injection molding quality prediction: Implications on sustainable manufacturing industry. *Sustainability*, v. 13, n. 8, 2021. ISSN 2071-1050. Disponível em: <https://www.mdpi.com/2071-1050/13/8/4120>. Citado na página 67.

KIM, M.; LEE, J.; JEONG, J. Open source based industrial iot platforms for smart factory: Concept, comparison and challenges. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, v. 11624 LNCS, p. 105 – 120, 2019. ISSN 03029743. Citado 2 vezes nas páginas 29 e 32.

KUBERNETES. *Documentação oficial do Kubernetes*. 2025. <https://kubernetes.io/pt-br/>. Acessado em: 8 de maio de 2025. Disponível em: <https://kubernetes.io/pt-br/>. Citado 2 vezes nas páginas 33 e 81.

KULATUNGA, N. A.; SAMPAIO, D. J. B. S. Iot platforms and hardware integrations for industry 4.0 applications. In: *2019 14th Conference on Industrial and Information Systems (ICIIS)*. [S.l.: s.n.], 2019. p. 209–214. ISSN 2164-7011. Nenhuma citação no texto.

LOPEZ, P. G. et al. Edge-centric computing: Vision and challenges. *SIGCOMM Comput. Commun. Rev.*, Association for Computing Machinery, New York, NY, USA, v. 45, n. 5, p. 37–42, set. 2015. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/2831347.2831354>. Citado na página 13.

MATEVSKA, J.; SOLDIN, M. Enabling iot connectivity and interoperability by using automated gateways. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Science and Business Media Deutschland GmbH, v. 13928 LNCS, p. 300 – 317, 2023. ISSN 03029743. Citado 5 vezes nas páginas 25, 28, 33, 47 e 51.

MILADINOVIC, I. et al. Multi-access edge computing: An overview and latency evaluation. In: . [S.l.: s.n.], 2021. p. 744–748. Citado na página 18.

MINIO, INC. *MinIO: S3 Compatible, Exascale Object Store for AI*. [S.l.], 2025. Acesso em: 12 de novembro de 2025. Citado na página 64.

MIRANI, A. A. et al. Key challenges and emerging technologies in industrial iot architectures: A review. *Sensors*, MDPI, v. 22, n. 15, 2022. ISSN 1424-8220. Citado 7 vezes nas páginas 5, 13, 17, 29, 34, 35 e 36.

MOHAMED, N.; AL-JAROODI, J.; LAZAROVA-MOLNAR, S. Leveraging the capabilities of industry 4.0 for improving energy efficiency in smart factories. *IEEE Access*, v. 7, p. 18008–18020, 2019. ISSN 2169-3536. Nenhuma citação no texto.

MUZELAK, M.; SKOVRANEK, T. Edge computing implementation of safety monitoring system in frame of iiot. In: *2022 23rd International Carpathian Control Conference (ICCC)*. [S.l.: s.n.], 2022. p. 125–129. Citado 4 vezes nas páginas 20, 26, 27 e 29.

NAGORNY, P. et al. Quality prediction in injection molding. *2017 CIVEMSA conference*, 04 2018. Citado 3 vezes nas páginas 5, 67 e 68.

NANOMQ. *NanoMQ: An Ultra-lightweight and Blazing-fast MQTT Broker for IoT Edge*. 2025. <https://nanomq.io/>. Acesso em 5 de agosto de 2025. Citado na página 63.

NICHOLSON, R. et al. Dynamic fog computing platform for event-driven deployment and orchestration of distributed internet of things applications. In: *2019 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4)*. [S.l.: s.n.], 2019. p. 239–246. Citado 4 vezes nas páginas 28, 48, 50 e 87.

NODE-RED CONTRIBUTORS. *Node-RED*. 2024. <https://nodered.org/>. Acessado em 22 de junho de 2024. Citado 2 vezes nas páginas 33 e 64.

PARSIFAL. 2024. <https://parsif.al/>. Accessed: 2024-07-30. Citado na página 20.

PEDONE, G.; MEZGÁR, I. Model similarity evidence and interoperability affinity in cloud-ready industry 4.0 technologies. *Computers in Industry*, Elsevier B.V., v. 100, p. 278 – 286, 2018. ISSN 01663615. Citado 6 vezes nas páginas 5, 19, 34, 35, 36 e 37.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. [S.l.: s.n.], 2008. p. 1–10. Citado na página 20.

PHILLIPS, K. et al. Connecting industrial automation software to a discrete manufacturing plant model for research and education. In: *2013 Africon*. [S.l.: s.n.], 2013. p. 1–5. Citado na página 19.

RANCHER LABS. *K3s: Lightweight Certified Kubernetes Distribution*. 2025. <https://docs.k3s.io/>. Acesso em: 17 nov. 2025. Citado na página 69.

REAL GAMES UNIPESSOAL LTDA. *Next-Gen PLC Training - Factory I/O*. [S.l.], 2025. Factory I/O é um software de simulação de fábrica em 3D para treinamento de PLC (Controlador Lógico Programável). Acesso em: 12 de novembro de 2025. Citado na página 66.

ROSENBERGER, J. et al. Virtual commissioning of distributed systems in the industrial internet of things. *Sensors*, MDPI, v. 23, n. 7, 2023. ISSN 14248220. Nenhuma citação no texto.

SALIS, A. et al. An edge-cloud based reference architecture to support cognitive solutions in process industry. In: . [S.l.: s.n.], 2023. v. 217, p. 20–30. ISSN 1877-0509. 4th International Conference on Industry 4.0 and Smart Manufacturing. Citado 2 vezes nas páginas 31 e 33.

SHERLEKAR, R.; STARLY, B.; COHEN, P. H. Provisioned data distribution for intelligent manufacturing via fog computing. In: . [S.l.: s.n.], 2019. v. 34, p. 893–902. ISSN 2351-9789. 47th SME North American Manufacturing Research Conference, NAMRC 47, Pennsylvania, USA. Citado 3 vezes nas páginas 18, 19 e 27.

SITTÓN-CANDANEDO, I. et al. A review of edge computing reference architectures and a new global edge proposal. *Future Generation Computer Systems*, Elsevier B.V., v. 99, p. 278 – 294, 2019. ISSN 0167739X. Citado 4 vezes nas páginas 13, 18, 29 e 36.

SITTóN-CANDANEDO, I. et al. Edge computing architectures in industry 4.0: A general survey and comparison. *Advances in Intelligent Systems and Computing*, Springer Verlag, v. 950, p. 121 – 131, 2020. ISSN 21945357. Citado 3 vezes nas páginas 5, 36 e 38.

STRLJIC, M. M.; VOLLMANN, C.; RIEDEL, O. Shop-floor service connector - a message-oriented middleware focused on the usability and infrastructure requirements of smes developing smart services. In: *2020 3rd IEEE International Conference on Knowledge Innovation and Invention (ICKII)*. [S.l.: s.n.], 2020. p. 37–40. Citado 3 vezes nas páginas 26, 27 e 47.

THE GOPCUA TEAM. *gopcua/opcua: Native Go OPC-UA library*. 2025. <https://github.com/gopcua/opcua>. Repositório do GitHub. Versão acessada: v0.8.0, Acesso em: 12 de novembro de 2025. Citado na página 67.

VATER, J.; HARSCHEIDT, L.; KNOLL, A. A reference architecture based on edge and cloud computing for smart manufacturing. In: *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. [S.l.: s.n.], 2019. p. 1–7. Citado 5 vezes nas páginas 25, 28, 30, 32 e 34.

VENANZI, R. et al. Miint: Middleware for iiot platforms integration. In: *2021 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2021. p. 1–6. Citado 4 vezes nas páginas 13, 17, 26 e 47.

VILLAR, E. et al. Architectures for industrial aiot applications. *Sensors*, MDPI, v. 24, n. 15, p. 4929, 2024. Citado 3 vezes nas páginas 19, 36 e 37.

WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer, 2012. ISBN 978-3-642-29043-5. Citado 2 vezes nas páginas 17 e 20.

YANG, C. et al. Software-defined cloud manufacturing with edge computing for industry 4.0. In: *2020 International Wireless Communications and Mobile Computing (IWCMC)*. [S.l.: s.n.], 2020. p. 1618–1623. Nenhuma citação no texto.

YANG, L. et al. A social-d2d architecture for people-centric industrial internet of things. In: *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*. [S.l.: s.n.], 2017. p. 744–749. Nenhuma citação no texto.