UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Vitor Alan de Lima Santos

Uma infraestrutura de software para o processamento de fluxos de dados em múltiplas camadas para cidades inteligentes

São Luís

2025

Vitor Alan de Lima Santos

Uma infraestrutura de software para o processamento de fluxos de dados em múltiplas camadas para cidades inteligentes

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Francisco José da Silva e Silva

São Luís

2025

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Santos, Vitor Alan de Lima.
Uma infraestrutura de software para o processamento de fluxos de dados em múltiplas camadas para cidades inteligentes / Vitor Alan de Lima Santos. - 2025.
55 p.

Orientador(a): Francisco José da Silva e Silva.
Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luis, Ma, 2025.

1. Processamento de Eventos Complexo. 2. Edge/fog/cloud Computing. 3. Blockchain. 4. Interscity. 5. Cidades Inteligentes. I. da Silva e Silva, Francisco José. II. Título.

Vitor Alan de Lima Santos

Uma infraestrutura de software para o processamento de fluxos de dados em múltiplas camadas para cidades inteligentes

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho _____ em São Luís, 18 de agosto de 2025:

Prof. Dr. Francisco José da Silva e Silva
Orientador

Prof. Dr. Antonio de Abreu Batista Júnior
Examinador

Profa. Dra. Simara Vieira da Rocha
Examinador

São Luís
2025

Agradecimentos

Em primeiro lugar, agradeço a minha família pelo amor, paciência e apoio incondicional que me motivaram a superar os desafios desta jornada. Agradeço também, ao meu orientador pela oportunidade de aprender com o desenvolvimento das pesquisas realizadas. Sou grato à Universidade Federal do Maranhão (UFMA) pelo suporte institucional e por oferecer um ambiente acadêmico propício à pesquisa. Registro meu agradecimento aos colegas e amigos do Laboratório de Sistemas Distribuídos Inteligentes (LSDi), pelas trocas de ideias e pelo constante suporte técnico. Por fim, agradeço aos amigos e demais colegas de curso pelo apoio durante essa fase.

"A tecnologia sozinha não basta —
é preciso dotá-la de propósito e significado."

Sherry Turkle em *Alone Together*

Resumo

Com o crescimento das cidades, crescem também os desafios de gerenciar os recursos disponíveis de forma eficiente. Assim, surge o conceito de Cidades Inteligentes, que integram a tecnologia com sua infraestrutura. Essa integração faz uso da Internet of Things (IoT) para coletar informação de sensores, gerando fluxos de dados que podem ser processados. O Context Data Process Orchestrator (CDPO) surge como resposta ao desafio de processar fluxos de dados gerados por Cidades Inteligentes de forma eficiente e segura. Estendendo a plataforma InterSCity com uma linguagem própria para definir *Event Process Networks* (EPNs) e com mecanismos de segurança baseados em *blockchain* para garantir autenticidade, autorização e rastreabilidade dos nós de processamento, o objetivo central do trabalho foi criar uma arquitetura capaz de implantar e atualizar dinamicamente regras de *Complex Event Processing* (CEP) nas múltiplas camadas *edge, fog e cloud*, mantendo comunicação segura e desempenho compatível com aplicações de Cidades Inteligentes. Para isso, o CDPO combina um roteamento sensível ao contexto, que direciona fluxos ao nó *fog* geograficamente mais próximo e uma linguagem declarativa de EPNs que permite reconfiguração, favorecendo evoluibilidade da solução proposta. A arquitetura foi desenvolvida utilizando protocolos seguros de comunicação e microsserviços, com o objetivo de permitir o processamento de eventos complexos com eficiência e segurança. A metodologia envolveu dois estudos de caso simulados no *software Simulator of Urban MObility* (SUMO): (i) gestão do transporte público, que modelou rotas de ônibus e atrasos no campus da UFMA, utilizando nós *edge, fog e cloud*; e (ii) monitoramento de vias urbanas, que avaliou a detecção da velocidade média de veículos em tempo real. O *middleware* registrou todas as detecções corretas de eventos, sem perdas de informação e alcançou acurácia próxima à simulação definida. O tempo entre ocorrência e detecção manteve-se em latência média de 1,816 s (máx. 4 s), enquanto a inserção dos mecanismos de segurança utilizando *blockchain* acrescentou apenas 147 ms na implantação de regras, demonstrando viabilidade de segurança sem sacrificar desempenho. As principais contribuições incluem: (i) linguagem declarativa de EPNs multicamadas; (ii) extensão que permite Complex Event Processing (CEP) à plataforma InterSCity; (iii) adoção de um modelo de gestão de identidades baseado em *blockchain*.

Palavras-chave: Processamento de Eventos Complexos; Edge/Fog/Cloud Computing; Blockchain; InterSCity; Cidades Inteligentes.

Abstract

With the growth of cities, the challenge of managing available resources efficiently arises. Thus, the concept of Smart Cities emerges, integrating technology with urban infrastructure. This integration leverages the *Internet of Things* (IoT) to collect information from sensors, generating data streams that can be processed. The Context Data Process Orchestrator (CDPO) appears as a response to the challenge of efficiently and securely processing the data streams generated by Smart Cities. It extends the InterSCity platform with its own language for defining *Event Process Networks* (EPNs) and with *blockchain*-based security mechanisms to guarantee authenticity, authorization, and traceability of processing nodes. The central objective of this work was to create an architecture capable of dynamically deploying and updating *Complex Event Processing* (CEP) rules across the *edge, fog, and cloud* layers, maintaining secure communication and performance compatible with Smart City applications. To achieve this, the CDPO combines context-aware routing—directing flows to the geographically nearest *fog* node—and a declarative EPN language that permits reconfiguration, fostering the evolvability of the proposed solution. The architecture was developed using secure communication protocols such as MQTT with TLS and HTTPS, and microservices, aiming to enable efficient and secure complex event processing. The methodology involved two case studies simulated in the *Simulator of Urban MObility* (SUMO): (i) public transportation management, which modeled bus routes and delays on the UFMA campus using *edge, fog, and cloud* nodes; and (ii) urban road monitoring, which evaluated real-time detection of vehicles' average speed. The *middleware* recorded all correct event detections without information loss and achieved accuracy close to the defined simulation. The time between occurrence and detection remained at an average latency of 1.816,s (max. 4,s), while the insertion of security mechanisms using *blockchain* added only 147,ms to rule deployment, demonstrating the feasibility of security without sacrificing performance. The main contributions include: (i) a multilayer declarative EPN language; (ii) an extension that enables Complex Event Processing (CEP) on the InterSCity platform; (iii) adoption of a *blockchain*-based identity management model; and (iv) experiments that evaluate performance, scalability, and reliability in real-time urban applications.

Keywords: Complex Event Processing; Edge/Fog/Cloud Computing; Blockchain; InterSCity; Smart Cities.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UFMA	*Universidade Federal do Maranhão*
CDPO	*Context Data Process Orchestrator*
CEP	*Complex Event Processing*
IOT	*Internet of Things*
MQTT	*Mosquitto*
HTTP	*Hypertext Transfer Protocol*
SUMO	*Simulation of Urban MObility*

Sumário

1 Introdução

O grande crescimento da população nas zonas urbanas traz desafios significativos para as cidades atuais, como a segurança pública, gerenciamento de recursos, conforto dos cidadãos e acessibilidade. Nesse contexto, as Cidades Inteligentes surgem como uma alternativa para melhorar a qualidade de vida dos habitantes, integrando soluções tecnológicas com a estrutura física da cidade para criar um ambiente urbano mais eficiente, sustentável e acessível.

As Cidades Inteligentes utilizam as Tecnologias da Informação e Comunicação (TICs) como parte da infraestrutura da cidade para desenvolver soluções para os desafios urbanos. Além disso, uma cidade é considerada inteligente quando os investimentos aplicados impulsionam um crescimento econômico sustentável e uma alta qualidade de vida, com uma gestão inteligente dos recursos da cidade (CARAGLIU; BO; NIJKAMP, 2011). Portanto, ao direcionar o uso de recursos para a utilização de TICs com o objetivo de resolver problemas urbanos, a população se beneficia com soluções inteligentes e sustentáveis, que facilitam a interação com recursos e serviços da cidade disponibilizados para todo cidadão.

A Internet das Coisas (Internet of Things - IoT) é uma das principais tecnologias utilizadas nas Cidades Inteligentes. O uso de IoT possibilita a criação de um ambiente em que diferentes aparelhos troquem informações entre si, permitindo a criação de novos serviços, programas e sistemas (MEDEIROS et al., 2018). Podemos considerar que IoT é a base para formalizar o conceito de Cidade Inteligente, pois ela possibilita que as informações de recursos urbanos sejam coletadas gerando fluxos de dados que devem ser processados para resolver problemas relacionados ao meio urbano. A tecnologia IoT pode ser aplicada em diversas áreas no contexto de Cidades Inteligentes, otimizando a mobilidade urbana com semáforos e estacionamentos inteligentes e modernizando a infraestrutura, como a iluminação pública.

Fluxos de dados referem-se à movimentação contínua de informações entre diferentes sistemas ou dispositivos, de forma dinâmica e em tempo real. Na era das Cidades Inteligentes, os fluxos de dados tornam-se especialmente importantes, pois permitem que informações provenientes de diversas fontes, como sensores, dispositivos móveis, redes sociais e outros sistemas, sejam coletadas, analisadas e transformadas em conhecimento útil para gestores públicos e cidadãos. O Esper [1], desenvolvido pela EsperTech, é uma das principais tecnologias para Processamento de Eventos Complexos (CEP), permitindo a análise de grandes volumes de eventos em tempo real. Utilizando a linguagem Event

[1] https://www.espertech.com/esper

Processing Language (EPL), baseada no SQL-92, o Esper facilita a identificação de padrões e anomalias em fluxos de dados contínuos, como os gerados por sensores urbanos em Cidades Inteligentes.

A plataforma InterSCity é uma solução de software projetada para o gerenciamento de sensores e recursos urbanos, como veículos de transporte público, semáforos e suas respectivas capacidades, incluindo, por exemplo, dados de geolocalização, como latitude e longitude. Desenvolvida de forma incremental e utilizando uma arquitetura baseada em microsserviços, a plataforma é de código aberto, o que facilita a pesquisa, o desenvolvimento e a realização de experimentos colaborativos no contexto das Cidades Inteligentes (ESPOSTE et al., 2017). Essa abordagem permite uma integração flexível e escalável, promovendo inovação contínua e colaboração entre diferentes partes interessadas na construção de soluções para ambientes urbanos inteligentes.

O trabalho desenvolvido apresenta o Context Data Process Orchestrator (CDPO), um middleware criado como uma extensão da plataforma InterSCity, que possibilita o processamento distribuído de fluxos de dados em múltiplas camadas, gerados pelas Cidades Inteligentes. Utilizando uma linguagem específica para definir Event Process Networks (EPNs), o CDPO orquestra a implementação de regras de processamento no motor Esper presente nas múltiplas camadas de forma eficiente e escalável. A solução foi desenvolvida com o objetivo de servir como ferramenta para que desenvolvedores consigam definir regras de processamento utilizando uma linguagem bem definida, a fim de obter informações relevantes dos fluxos de dados gerados por recursos da cidade.

Para garantir a seguridade da solução desenvolvida, foram incorporados mecanismos de segurança para assegurar a integridade e a confidencialidade das informações nos canais de comunicação, oferecendo uma infraestrutura confiável e protegida contra acessos não autorizados. Foi adotado um modelo de gestão de identidades, utilizado para garantir que os fluxos de dados sempre serão processados por nós considerados seguros, garantindo também a autenticidade dos serviços envolvidos na orquestração do processamento dos dados. Além disso, todos os canais de comunicação utilizam protocolos seguros de comunicação para garantir que os dados circulem de forma íntegra e confidencial entre camadas.

2 Objetivos

O objetivo geral deste trabalho é o desenvolvimento de extensões ao *middleware* InterSCity, que permitam o processamento de eventos complexos em múltiplas camadas (borda, nevoeiro e nuvem) de forma segura e que suportem aplicações criadas para o contexto de Cidades Inteligentes. Ademais, são objetivos específicos:

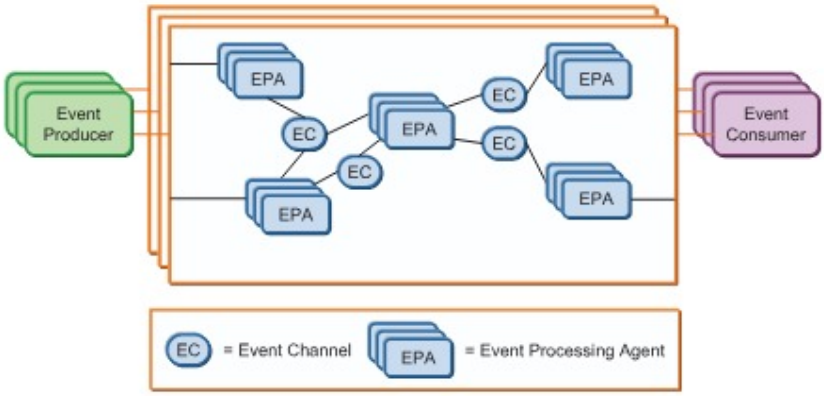
- Concepção de uma linguagem que permita a especificação de Redes de Processamento de Eventos Complexos a serem executadas em uma arquitetura de múltiplas camadas.
- Modelar e desenvolver uma arquitetura que lide com processamento de regras CEP em múltiplas camadas, permitindo o gerenciamento dinâmico de regras em tempo de execução.
- Estabelecimento de mecanismos para comunicação segura de forma a garantir autenticação e integridade dos dados entre todos os componentes de software envolvidos na arquitetura proposta.
- Implementação de mecanismos para o controle de acesso aos fluxos de dados sendo transmitidos e processados.
- Realizar experimentos para avaliar o desempenho do `software` a ser desenvolvido.

3 Fundamentação Teórica

3.0.1 Processamento de Eventos Complexos

O Processamento de Eventos Complexos ou *Complex Event Process* (CEP) é um paradigma de programação que tem como objetivo extrair informações a partir de eventos gerados em diferentes fontes. Com isso, é possível detectar padrões, anomalias no fluxo de dados e filtragem de informações. A Figura 1 apresenta um diagrama da disposição das entidades participantes no processamento de eventos complexos, assim como o fluxo dos dados.

Figura 1 – Rede de Processamento de Eventos



Fonte: (FLOURIS et al., 2017)

Os Produtores de Eventos são os componentes responsáveis pela geração e injeção de eventos no sistema de processamento. Eles atuam como a interface entre fontes de dados externas e a lógica de processamento, capturando ocorrências e as convertendo em um formato de evento estruturado. As fontes podem incluir sensores físicos, logs de aplicação, endpoints de API ou gatilhos de banco de dados, resultando em um fluxo contínuo de dados brutos para análise.

Os Processadores de Eventos, tecnicamente denominados Event Processing Agents (EPAs), constituem o núcleo computacional do sistema. Sua função é executar regras sobre os fluxos de eventos, realizando operações como filtragem, transformação, enriquecimento com dados contextuais, agregação em janelas e correlação para detecção de padrões. Um conjunto de EPAs interconectados forma uma Event Processing Network (EPN), uma topologia em grafo onde a saída de um agente serve como entrada para outro, permitindo a implementação de lógicas de processamento distribuídas.

Os Consumidores de Eventos são os componentes terminais do fluxo, responsáveis por receber e agir sobre os eventos resultantes. Eles se inscrevem para receber os eventos gerados pelos EPAs, e sua recepção tipicamente aciona uma ação externa, como a persistência de dados, a atualização de uma interface de usuário, o envio de notificações ou o acionamento de atuadores. Eles efetivamente materializam o resultado da análise de eventos, desacoplando a lógica de processamento da ação final.

O Esper [1] é uma tecnologia que tem como objetivo o processamento de eventos complexos com código aberto que fornece uma abordagem declarativa baseada em consultas para o processamento de eventos complexos. As consultas enviadas a um sistema CEP visam detectar eventos do fluxo de entrada com base em um dado padrão (ORTIZ et al., 2023). Com isso, esse mecanismo trabalha com 3 principais conceitos, sendo eles eventos, fluxos e consultas.

Evento é uma unidade de um fluxo de dados que ocorre em determinado momento, podendo ser um evento de localização composto por latitude e longitude, por exemplo. Fluxos são sequências de eventos, os quais a tecnologia Esper fornece mecanismos para o seu processamento utilizando a detecção de padrões. Isso é feito por meio das consultas, cuja estrutura deriva da sintaxe da linguagem SQL, utilizada para manipular bancos de dados.

3.0.2 Computação de Borda e Nevoeiro

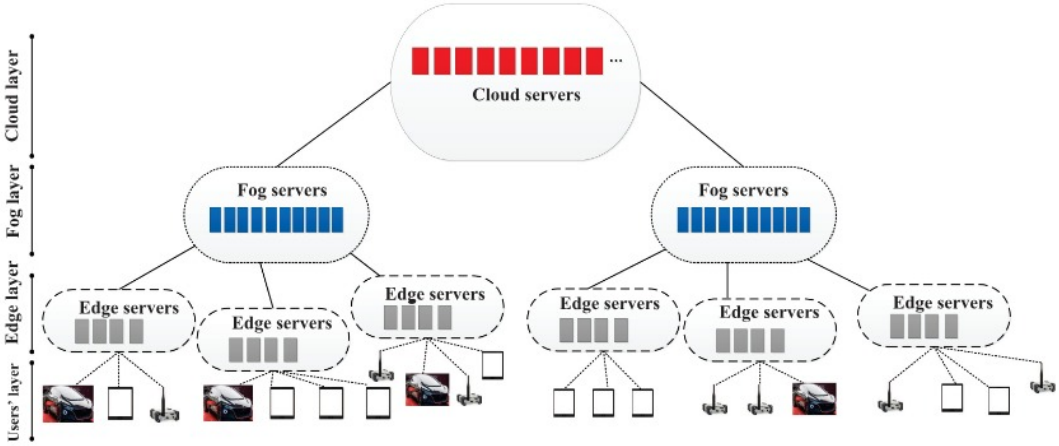
A Computação em Névoa, ou Fog Computing, é um paradigma de computação distribuída que introduz uma camada intermediária de processamento, armazenamento e rede entre os dispositivos de borda (edge devices) e a infraestrutura centralizada da nuvem (Cloud), O objetivo fundamental é executar tarefas de pré-processamento, filtragem, agregação e análise de dados localmente, mitigando a necessidade de transmitir fluxos de dados brutos e massivos para a nuvem. A Figura 2 ilustra uma arquitetura em camadas clássica que utiliza *fog/edge computing*.

Chamamos de *fog nodes* ou nós de névoa, os agentes de processamento da camada *fog*, e podem ser implantados em qualquer dispositivo com poder de computação e conexão a rede de internet. O objetivo primário da *fog computing* é fornecer latência baixa e previsível para aplicações IoT que necessitam baixo tempo de resposta (SHI et al., 2015). A baixa latência é atingida ao realizar o processamento em nós próximos aos dispositivos produtores de fluxos de dados, diminuindo a necessidade de utilizar os recursos computacionais da *cloud*, que geralmente fornece o resultado do processamento com maior tempo de resposta.

Edge computing ou computação de borda é um paradigma utilizado quando existe a possibilidade de realizar processamento em dispositivos produtores de eventos. Nesse

[1] https://www.espertech.com/

Figura 2 – Arquitetura Edge/Fog/Cloud Computing



Fonte: (OKEGBILE; MAHARAJ; ALFA, 2022)

paradigma, grande parte do processamento ocorre nos próprios dispositivos que geram os dados, tendo em vista que nos últimos anos o seu poder de processamento e armazenamento teve significativa evolução (ATLAM; WALTERS; WILLS, 2018). Sua ideia central é tornar a computação mais perto da fonte dos dados e é amplamente utilizada em sistemas IoT (CAO et al., 2020).

Edge e *fog computing* são amplamente utilizadas em conjunto, trazendo benefícios como a capacidade da aplicação escalar a medida em que novos produtores de dados participem da rede distribuída de processamento. Essa escalabilidade é alcançada através da natureza distribuída e hierárquica dos paradigmas, que se contrapõe ao modelo centralizado da nuvem. Em vez de concentrar toda a carga de processamento em uma única camada, a arquitetura escala horizontalmente ao adicionar novos nós de névoa ou de borda (*fog/edge nodes*) conforme a demanda por dispositivos e dados cresce. Cada nó possui capacidade computacional própria e processa a carga de sua vizinhança, resultando em uma arquitetura horizontalmente escalável.

3.0.3 InterSCity

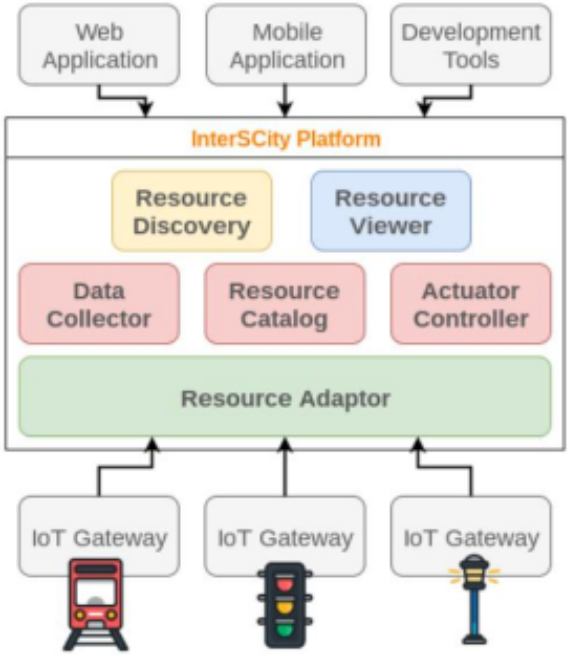
A plataforma InterSCity foi desenvolvida como uma plataforma baseada em microsserviços de código aberto para permitir pesquisa, desenvolvimento e experimentos colaborativos em Cidades Inteligentes (ESPOSTE et al., 2017). Sendo assim, um dos principais objetivos do projeto InterSCity é fornecer tecnologias e métodos para apoiar futuras Cidades Inteligentes. A plataforma atende alguns princípios de projeto que comprometem as soluções de iniciativas para Cidades Inteligentes, sendo as principais escalabilidade, flexibilidade e evoluibilidade.

A escalabilidade diz respeito ao suporte à estrutura da Cidade Inteligente, como a grande quantidade de sensores de IoT, conectividade de milhões de usuários, grande volume de dados e novos serviços que possam interagir com a plataforma.

Flexibilidade é a capacidade de adaptar-se às necessidades e requisitos que podem surgir ao decorrer do tempo. No contexto das Cidades Inteligentes, a heterogeneidade de sensores, atuadores e recursos da cidade devem ser levados em consideração ao se construir uma plataforma de software. Portanto, a flexibilização é um requisito primordial para que a plataforma InterSCity atue com eficiência no contexto de Cidades Inteligentes.

Evoluibilidade trata da capacidade que a plataforma deve ter de evoluir de acordo com a mudança na organização urbana. A evoluibilidade é a capacidade do sistema de evoluir ao longo do tempo, suportando modificações e melhorias rápidas com baixo custo e pequeno impacto arquitetônico, e é um elemento fundamental para o sucesso e valor econômico de software de longa duração (CARAGLIU; BO; NIJKAMP, 2011).

Figura 3 – Arquitetura Interscity



Fonte: (ESPOSTE et al., 2017)

A arquitetura da plataforma InterSCity é baseada em microsserviços, fornecendo *APIs RESTful* que utilizam o protocolo HTTP para comunicação do usuário com os serviços disponiblizados. A plataforma é composta por 6 microsserviços para gerenciamento de dados (`Resource Catalog`, `Data Collector` e `Actuator Controller`), integração de dispositivos de IoT (*Resource Adaptor*), descoberta de recursos da cidade por meio de dados contextuais (`Resource Discovery`) e visualização (`Resource Viewer`), todos utilizando o RabbitMQ [2] para comunicação interna.

[2] RabbitMQ: One broker to queue them all | RabbitMQ

O `Resource Adapter` funciona como *proxy* para serviços externos, permitindo que *Gateways IoT* registrem e atualizem recursos da cidade na plataforma que serão catalogados pelo `Resource Cataloguer`, responsável por manter um catálogo dos recursos, fornecendo um identificador universal para cada um. Um recurso da cidade é qualquer entidade física que faz parte do contexto urbano, como automóveis, edifícios, semáforos, latas de lixo e entre outros.

O microsserviço `Data Collector` é responsável por armazenar os dados coletados pelos sensores dos recursos da cidade. Os dados do sensor consistem em informações de contexto ou um evento vinculado a uma capacidade de recurso que é observada em um determinado momento (ESPOSTE et al., 2017). Todo recurso da cidade possui capacidades vinculadas, definidas como atributos referentes ao contexto do recurso, como a latitude e longitude no caso de recursos com mobilidade.

O `Actuator Controller` é o microsserviço que atua como um *middleware* padronizadas para as requisições direcionadas aos recursos da cidade que possuem atuadores. Além de gerenciar essas solicitações, ele também registra um histórico, permitindo auditoria e garantindo a rastreabilidade das ações executadas na plataforma. Para suportar picos de demanda em trabalhos baseados em eventos, o `Actuator Controller` é projetado para escalar através da adição de *workers*.

O `Resource Discovery` disponibiliza um serviço de busca sensível ao contexto, permitindo que aplicações clientes descubram os recursos disponíveis na cidade. Utilizando dados orquestrados do `Resource Catalog` e do `Data Collector`, ele oferece uma API de pesquisa com filtros que podem ser combinados para refinar os resultados, incluindo dados de localização e outros tipos de metadados. Para garantir leituras de baixa latência, este microsserviço armazena em cache os metadados estáticos dos recursos, uma vez que eles não mudam com frequência.

O `Resource Viewer` funciona como um microsserviço de visualização web, projetado para apresentar as informações da cidade de forma gráfica ao usuário. Ele consome os serviços `Resource Catalog` e `Data Collector` para exibir visualizações gerais e administrativas dos recursos, como sua localização em um mapa, dados de contexto em tempo real e gráficos representativos de dados históricos. O desenvolvimento futuro deste componente inclui o aprimoramento da visualização de dados.

3.0.4 Blockchain e Hyperledger

A tecnologia *blockchain* está atraindo enorme atenção e desencadeando vários projetos em diferentes indústrias (NOFER et al., 2017). *Blockchain* é basicamente um banco de dados compartilhado imutável, descentralizado e disponível ao público (XIE et

al., 2019). O primeiro exemplo de uso da tecnologia *Blockchain* foi a criptomoeda Bitcoin, criada por Satoshi Nakamoto, muito utilizada no mercado financeiro digital.

Blockchain consiste em conjuntos de dados que são organizados em blocos, onde cada bloco armazena múltiplas transações. Todas as transações são registradas e qualquer entidade que faz parte do sistema tem permissão para acessar, enviar e verificar essas transações (XIE et al., 2019). A *blockchain* é estendida por meio da adição de novos blocos à cadeia e, portanto, representa um registro completo do histórico de transações (NOFER et al., 2017).

Cada bloco possui um código *hash* que foi gerado utilizando as transações do bloco e o *hash* do bloco anterior, formando um elo entre eles na *blockchain*. Como todos os blocos guardam o *hash* do bloco anterior, qualquer entidade pode verificar se os dados não foram alterados, invalidando a possibilidade de alteração na cadeia de blocos (PIERRO, 2017).

Um mecanismo de consenso é o processo em que os nós validadores chegam a um acordo sobre o estado de um livro-razão (SWANSON, 2015). Se a maioria dos nós entrarem em acordo utilizando um mecanismo de consenso sobre a validade de transações em um bloco e sobre a validade do bloco, o bloco pode ser adicionado à cadeia (NOFER et al., 2017).

Uma das funcionalidades mais utilizadas da tecnologia *blockchain* são os contratos inteligentes (GAYVORONSKAYA; MEINEL, 2021). Essencialmente, um `contrato inteligente` executa automaticamente um conjunto de instruções quando condições pré-acordadas são cumpridas. Os termos do acordo entre as partes são escritos diretamente no código, e as transações são rastreáveis e irreversíveis, operando sem a necessidade de um intermediário central. Esta abordagem inovadora pode, por exemplo, substituir advogados e bancos que estiveram envolvidos em contratos de negociação de ativos dependendo de aspectos pré-definidos (FAIRFIELD, 2014).

Com o objetivo de oferecer infraestrutura robusta para aplicações baseadas em *blockchain*, a iniciativa Hyperledger [3] surge como uma plataforma de código aberto em crescimento rápido e com esforço colaborativo, criado para avançar tecnologias relacionadas a *blockchain* (AGGARWAL; KUMAR, 2021). Esta plataforma permite o desenvolvimento de sistemas transacionais que estabelecem confiança, responsabilidade e transparência em sua essência, ao mesmo tempo que automatiza e simplifica os processos de negócios.

Dentro do ecossistema Hyperledger, o Hyperledger Fabric [4] é entendido como uma plataforma *blockchain* aberta que foi lançada pela primeira vez em dezembro de 2015 pela Linux Foundation (YEWALE, 2018). Hyperledger Fabric fornece um design modular com uma demarcação de responsabilidades entre os nós em uma rede blockchain, implementação

[3] https://www.hyperledger.org/
[4] https://www.lfdecentralizedtrust.org/projects/fabric

de contratos inteligentes, consenso configurável e serviços aos participantes (LI; WONG; GUO, 2020).

4 Trabalhos Relacionados

Este capítulo oferece uma revisão abrangente das soluções tecnológicas mais recentes para cidades inteligentes, com foco em suas principais objetivos, características, requisitos e lacunas existentes.

4.1 Metodologia para Seleção dos Trabalhos Relacionados

Para realizar uma análise comparativa dos trabalhos relacionados, utilizou-se a qualificação de doutorado de D. Pereira (PEREIRA, 2025), a qual realizou um levantamento detalhado de requisitos de software para plataformas de Cidades inteligentes, extraindo apenas estudos que declarassem suporte explícito à "fluxos de dados ou eventos" e "processamento de dados". Esses requisitos constituem o núcleo funcional de qualquer plataforma que lida com Complex Event Processing (CEP).

Em seguida, restringiu-se o conjunto preliminar àqueles trabalhos que apresentassem implementação concreta, protótipo funcional, estudo de caso ou validação empírica. Propostas puramente conceituais ou limitadas a discussões arquiteturais foram descartadas, assegurando que a comparação se apoiasse em evidências mensuráveis de desempenho, escalabilidade e segurança.

Após isso, realizou-se uma busca complementar nas bases IEEE Xplore, Spring e Eslsevier com a *string* controlada `("Data Flow Processing") AND ("IoT" OR "Smart Cities") AND ("Edge Computing"OR "Fog Computing")`. O período de coleta concentrou-se entre 2019 e 2025, resultando na inclusão de trabalhos implementados como FUSERS Poredi et al. (2023), Aveiro Tech City Living Lab Rito et al. (2023) e VOICE Zaccarini et al. (2024), pois contemplavam os requisitos "fluxos de dados ou eventos" e "processamento de dados" e possuem implementação concreta.

4.2 Visão geral dos trabalhos selecionados

Thakkar et al. (2023) propõem uma plataforma baseada em *mobile edge computing* (MEC) organizada em três camadas (dispositivo móvel, borda (*edge*) e nuvem) para otimizar o processamento de dados em cidades inteligentes. No dispositivo móvel, um agente coleta e faz o pré-processamento básico (normalização e compressão); na borda, modelos de *Edge AI* filtram e agregam apenas os eventos mais relevantes; finalmente, só os resultados validados seguem para a nuvem, onde recebem análise mais aprofundada. Dessa

forma, a solução reduz tanto a latência quanto o congestionamento de rede, melhorando o tempo de resposta das aplicações clientes.

Pereira e Brayner (2023) apresentam a UFCity, uma proposta de plataforma baseada em *edge computing* construída sobre microsserviços containerizados. Sensores urbanos (tráfego, qualidade do ar, iluminação etc.) comunicam-se via protocolos leves (e.g. MQTT), levando dados brutos a *gateways* de borda, onde cada microsserviço, equipado com um modelo de IA especializado, realiza tarefas como previsão de congestionamento, detecção de anomalias e classificação de eventos críticos. A comunicação assíncrona entre serviços é feita por um barramento de mensagens, garantindo resiliência e escalabilidade. Por fim, apenas os resultados filtrados e enriquecidos seguem para a nuvem, reduzindo significativamente latência e volume de tráfego de rede.

Baucas e Spachos (2024) apresenta uma solução que utiliza a computação de borda (*edge computing*) como elemento central no processo de classificação de sons urbanos. A proposta consiste em uma plataforma distribuída de detecção e processamento de dados acústicos, onde todo o processamento é realizado localmente em dispositivos de borda, eliminando a necessidade de envio contínuo de dados brutos para a nuvem. A arquitetura implementada utiliza uma placa STM32 NUCLEO-64 acoplada a um microfone digital MEMS, conectada a um Raspberry Pi, responsável por executar uma rede neural do tipo MLP para a classificação dos sons capturados. Essa abordagem permite reduzir significativamente a latência na obtenção dos resultados, além de otimizar o consumo energético dos dispositivos.

Zaccarini et al. (2024) propõe uma plataforma baseada em *Edge-Fog-Cloud Computing* que explora o conceito de *Value of Information* (VoI), ou Valor da Informação, para otimizar o uso de recursos computacionais. A proposta consiste em um modelo de orquestração distribuída onde os microsserviços podem ser executados dinamicamente em diferentes camadas, ou seja, em dispositivos de borda (edge), nós intermediários (fog) ou *datacenters* na nuvem (cloud), de acordo com a criticidade da informação e os requisitos de desempenho. Dessa forma, o sistema prioriza de forma natural a alocação de recursos computacionais para os microsserviços que lidam com informações consideradas mais valiosas, promovendo uma utilização eficiente dos recursos disponíveis através do processamento de dados com maior relevância para a plataforma.

Zhang et al. (2023) enfatiza a importância da segurança da informação em sistemas para Cidades Inteligentes e propõe uma arquitetura que integra *blockchain* com computação em borda para assegurar a integridade e a confiabilidade dos dados coletados por sensores IoT. Na solução proposta, os dados gerados pelos dispositivos são armazenados diretamente em uma *blockchain* privada, distribuída entre os nós de borda da rede. A *blockchain* atua como um mecanismo confiável de armazenamento e auditoria, garantindo que os dados coletados por sensores permaneçam íntegros e verificáveis. Para reduzir a sobrecarga

computacional, a arquitetura realiza o processamento dos dados com algoritmos de inteligência artificial diretamente na camada de borda, preservando a privacidade dos dados sensíveis e otimizando o uso de recursos. Os resultados do processamento local são então encaminhados para aplicações clientes, garantindo segurança, baixa latência e escalabilidade.

Poredi et al. (2023) apresenta o FUSERS (*FUll-Spectrum, Environment-Resilient Surveillance*), um *framework* que utiliza o paradigma *Edge-Fog-Cloud Computing* para melhorar o monitoramento urbano feito por câmeras de vigilância. Na camada *edge*, os nós de borda que possuem dispositivos IoT (câmeras ópticas e sensores térmicos) realizam pré-processamento local antes de transmitir os fluxos de dados. A camada *fog*, composta por *gateways* próximos aos *edge nodes*, realiza o processamento dos dados transmitidos com o objetivo de disparar alertas imediatos para as equipes de segurança da cidade. Por fim, a camada *cloud* concentra dados históricos e poder computacional para treinar e atualizar o modelo *Hybrid-Pedestrian Detection* (HYPE), baseado em YOLOv5, criado para analisar padrões de comportamento através dos dados transmitidos pelos nós de borda.

Rito et al. (2023) apresenta o *Aveiro Tech City Living Lab (ATCLL)*, plataforma inteligente que cobre a cidade de Aveiro com 44 pontos de acesso *mmWave* integrados a nós *edge*. Cada terminal dispõe de um micro-servidor que pré-processa localmente os fluxos multimodais (câmeras, radares e sensores ambientais). Esse terminal ruídos, comprime dados, extrai métricas de mobilidade (p. ex., contagem de veículos/pessoas) e calcula indicadores ambientais antes de enviar apenas metadados à nuvem. Os experimentos relatados mostram que delegar essas tarefas à borda diminui a latência de entrega dos resultados em relação à estratégia de processamento apenas na nuvem, evidenciando um ganho considerável ao utilizar o paradigma que combina *edge* e *cloud computing*.

Em He et al. (2023), foi proposto um *middleware* desenvolvido com base em uma abordagem distribuída para Cidades Inteligenes. Esse projeto tem como objetivo viabilizar a programação rápida de serviços na nuvem e em dispositivos de borda, utilizando um modelo de programação baseado na *Next Generation Service Interface* (NGSI), uma arquietura composta por serviços distribuídos. Esse trabalho trouxe duas contribuições principais para o desenvolvimento de serviços voltados a Cidades Inteligentes: um modelo de programação baseado em padrões para computação em névoa (*fog computing*), que permite um desenvolvimento mais ágil de soluções, e o gerenciamento escalável de contexto, uma vez que utiliza uma arquitetura distribuída, possibilitando um desempenho superior em comparação com abordagens centralizadas.

Lymperis e Goumopoulos (2023) implementaram a plataforma SEDIA, voltada à integração de dados IoT semanticamente enriquecidos e ao desenvolvimento de aplicações para Cidades Inteligentes. A arquitetura distribui o processamento em três níveis

complementares: no *edge tier*, sensores e *gateways* próximos aos dispositivos coletam dados brutos e realizam o processamento inicial, reduzindo latência e tráfego de rede; no fog tier, brokers MQTT e serviços instalados em nós intermediários agregam e formatam os fluxos, oferecendo um tempo de resposta menor pois essa camada é implantada com proximidade à *edge tier*; Já o *cloud tier* realiza as atividades de maior complexidade, como raciocínio semântico, o armazenamento e a exposição de APIs que permitem, por exemplo, calcular rotas com melhor qualidade do ar.

4.3 Critérios de Comparação

Com base na metodologia utilizada para seleção dos trabalhos, requisitos funcionais e não funcionais foram definidos a partir das necessidades recorrentes observadas nesse conjunto (mobilidade, heterogeneidade, autenticação, privacidade, escalabilidade, dentre outras) que também estão presentes no levantamento sistemático de requisitos para plataformas de cidades inteligentes apresentado por Pereira (PEREIRA, 2025). Dessa convergência, foram definidos treze requisitos organizadas em seis categorias (1. Gestão de Recursos, 2. Segurança, 3. Eventos e Dados, 4. Processamento, 5. Aplicação e 6. Design), que servirão de parâmetros para a análise comparativa realizada nos próximos tópicos.

- **1. (RF1) Mobilidade:** A plataforma deve permitir que os dispositivos que acessam a plataforma sejam móveis, geralmente se comunicando via conexões sem fio. Além disso, os recursos da cidade também podem ser portáteis, fornecendo serviços de sensoriamento e atuação por meio de tecnologias sem fio.
- **1. (RF2) Repositório de recursos:** A plataforma deve manter um catálogo com informações sobre os recursos disponíveis na cidade, permitindo que os usuários pesquisem, inclusive com o uso de dados contextuais, para ativar recursos específicos.
- **1. (RNF1) Heterogeneidade de dispositivos:** A plataforma deve permitir a troca de dados entre diversos dispositivos de usuário e os vários recursos da cidade.
- **2. (RF3) Autenticação:** A plataforma deve permitir que usuários e sistemas de terceiros acessem os componentes da plataforma por meio de sistemas ou mecanismos de autenticação.
- **2. (RF4) Autorização:** A plataforma deve garantir que apenas usuários autorizados tenham acesso a todas ou partes das funcionalidades da plataforma, podendo adotar mecanismos de controle de acesso para garantir que os usuários tenham permissões adequadas para leitura, escrita e execução.
- **2. (RNF2) Privacidade de dados:** A plataforma deve garantir a privacidade dos dados dos usuários, incluindo aspectos legais, podendo usar técnicas para anonimizar

os dados dos indivíduos e mecanismos para acesso aos dados conforme critérios de privacidade.

- **3. (RF5) Fluxos de dados ou eventos:** A plataforma deve ser capaz de ler *streams* de dados ou eventos, permitindo a busca de padrões nesses *streams* para gerar novos conhecimentos.
- **4. (RF6) Processamento de dados:** A plataforma deve oferecer suporte a diversas funcionalidades de processamento de dados para atender às necessidades das aplicações, como funções matemáticas e lógicas, além de aplicar agrupamentos e filtros aos dados.
- **4. (RNF3) Processamento de dados em tempo real:** A plataforma deve permitir o processamento e análise de dados em tempo real ou quase em tempo real.
- **5. (RNF4) Aplicações de Cidades Inteligentes:** A plataforma deve permitir que aplicações de cidades inteligentes sejam executadas, internas ou externamente, oferecendo funcionalidades que permitam o acesso a dados, análise de dados, repositórios de recursos e serviços, entre outros itens pertencentes à plataforma.
- **6. (RNF5) Flexibilidade:** A plataforma deve ser adaptável o suficiente para diferentes tipos de cidades: grandes ou pequenas, com poucos ou muitos usuários, com ou sem necessidades específicas, como gerenciamento de tráfego urbano, eventos culturais, saúde pública ou novas leis. Também deve suportar a extensão de novas funcionalidades e serviços.
- **6. (RNF6) Dados contextuais:** A plataforma deve receber dados contextuais dos recursos e usuários da cidade, permitindo que as aplicações localizem os recursos da cidade por meio de consultas usando dados contextuais.
- **6. (RNF7) Escalabilidade:** A plataforma deve ser escalável em várias frentes, como atender a um grande número de usuários e ser capaz de expandir geograficamente e administrativamente.

4.4 Discussão

A partir da análise dos trabalhos relacionados apresentados na Tabela 2, comparando os trabalhos com base nos critérios listados na Tabela 1, observa-se que diferentes abordagens tem sido exploradas com o objetivo de atender às demandas de plataformas para Cidades Inteligentes. Essas abordagens abrangem amplamente os requisitos definidos, com destaque para o uso de paradigmas como *Edge Computing*, *Fog Computing* e *Cloud Computing*, além da incorporação de inteligência artificial e modelos semânticos para aprimorar a interpretação dos dados coletados.

Tabela 1 – Critérios de Comparação

Categoria	**Código**	**Descrição resumida**
Gestão de Recursos	RF1	Suporta mobilidade: dispositivos e recursos portáteis, conectividade sem fio.
	RF2	Suporta repositório de recursos: catálogo pesquisável, incluindo dados contextuais.
	RNF1	Suporta heterogeneidade de dispositivos: interoperabilidade entre múltiplos tipos de nós.
Segurança	RF3	Suporta autenticação: mecanismos para validar usuários e sistemas terceiros.
	RF4	Suporta autorização: controle de acesso (leitura, escrita, execução).
	RNF2	Suporta privacidade de dados: anonimização e políticas de acesso compatíveis com exigências legais.
Eventos e Dados	RF5	Suporta **fluxos de dados ou eventos**: ingestão contínua e detecção de padrões em *streams*.
Processamento	RF6	Suporta processamento de dados: operações matemáticas, lógicas, agregações e filtros.
	RNF3	Suporta processamento em tempo real: análise em tempo (quase) real.
Aplicação	RNF4	Suporta **aplicações de Cidades Inteligentes**: execução de apps internas ou externas, acesso a dados e serviços.
Design / Arquitetura	RNF5	Suporta flexibilidade: adaptação a diferentes portes de cidade e extensibilidade de funcionalidades.
	RNF6	Suporta dados contextuais: uso de contexto para descoberta e seleção de recursos.
	RNF7	Suporta escalabilidade: crescimento em número de usuários, dispositivos e abrangência geográfica/administrativa.

Tabela 2 – Tabela comparativa dos trabalhos relacionados

	1. Gestão de Recursos			**2. Segurança**			**3. Eventos e Dados**	**4. Processamento**		**5. Aplicação**	**6. Arquitetura**		
Trabalho	**RF1**	**RF2**	**RNF1**	**RF3**	**RF4**	**RNF2**	**RF5**	**RF6**	**RNF3**	**RNF4**	**RNF5**	**RNF6**	**RNF72**
(THAKKAR et al., 2023)	-	x	x	x	x	-	x	x	x	X	-	-	-
(PEREIRA; BRAYNER, 2023)	x	x	x	x	-	x	x	x	x	x	x	x	x
(BAUCAS; SPACHOS, 2024)	-	-	-	-	-	-	x	x	x	-	x	-	x
(ZACCARINI et al., 2024)	x	x	x	-	-	-	x	x	x	-	x	x	x
(ZHANG et al., 2023)	x	x	x	x	x	x	x	x	x	x	-	-	x
(POREDI et al., 2023)	-	-	-	-	-	x	x	x	x	-	x	-	x
(RITO et al., 2023)	x	x	x	x	x	-	x	x	x	x	-	-	x
(HE et al., 2023)	x	-	x	-	-	-	x	x	x	x	x	x	x
(LYMPERIS; GOUMOPOULOS, 2023)	x	x	x	x	x	x	x	x	x	x	x	-	x
CDPO + InterSCity	x	x	x	x	x	x	x	x	x	x	x	x	x

Os trabalhos analisados demonstram que a redução da latência depende fortemente de estratégias que levem o processamento para próximo dos dispositivos que coletam os dados . No Aveiro Tech City Living Lab Rito et al. (2023) comprovaram que manter apenas metadados na nuvem enquanto se executam tarefas em nós *edge* diminui significativamente o tempo de resposta. Resultados semelhantes foram observados no FUSERS Poredi et al. (2023), que organiza uma hierarquia *edge–fog–cloud* para processar fluxos de dados. Outros autores, como Thakkar et al. (2023), seguem a mesma direção, discutindo o uso da computação em borda como meio de garantir escalabilidade e *real-time processing*.

O Context Data Process Orchestrator (CDPO) alcançou uma baixa latência média na implantação de regras em múltiplas camadas (*edge, fog e cloud)*, e na entrega do resultado do processamento, sendo indicadores que evidenciam a aplicabilidade do CDPO para o contexto de Cidades Inteligentes.

A segurança, por sua vez, ainda aparece de forma fragmentada nos trabalhos correlatos. Zhang et al. (2023) utilizam *blockchain* apenas para auditoria, mantendo autenticação e autorização em componentes externos. Yu et al. (2022) apontam o risco dessa abordagem, pois expõem falhas de identidade em ecossistemas IoT densamente conectados. O CDPO enfrenta esse problema por meio de contratos inteligentes que executam autenticação, autorização e registro imutável dos nós de processamento. O baixo acréscimo na latência da implantação de regras de processamento medido nos experimentos realizados comprovam que é possível reforçar a segurança sem sacrificar desempenho.

Do ponto de vista arquitetural, existe um consenso de que plataformas para Cidades Inteligentes precisam ser evolutivas e extensíveis. Esposte et al. (2017) desenvolveram o InterSCity, plataforma de código aberto para gestão de recursos da cidade, adotando microsserviços para manter o *loose coupling* (diminuição de dependência entre componentes) e possibilitar implantações incrementais. Pereira e Brayner (2023) avançaram ao combinar microsserviços com inteligência artificial para criar um ecossistema de dados urbanos interoperável implementado na UFCity. VOICE explorou o conceito de Value of Information para orquestrar serviços ao longo do *continuum computacional*, priorizando nós que maximizam benefício sob restrições de QoS (ZACCARINI et al., 2024). O CDPO incorpora esses princípios, mas acrescenta duas contribuições: (i) um mecanismo de roteamento sensível ao contexto, que direciona fluxos ao nó *fog* geograficamente mais próximo—reduzindo latência e balanceando carga de forma dinâmica (ii) uma linguagem declarativa para redes de processamento de eventos (EPNs) que permite reconfiguração, facilitando a evolução contínua para arquiteturas de longa duração.

Com base nos trabalhos analisados, que definiram as bases conceituais para aplicações de Cidades Inteligentes e introduziram boas práticas, como microsserviços, mas costumam tratar desempenho, segurança e evoluibilidade como aspectos isolados. O CDPO demonstra, com evidência empírica, que é possível combinar esses três aspectos em uma plataforma única, capaz de processar fluxos de dados nas camadas *edge, fog e cloud* com baixa latência, de forma segura utilizando gestão de identidades e adaptar-se rapidamente a novos requisitos ao permitir a modificação das regras de processamento, assim, sendo uma solução totalmente aplicável no contexto de Cidades Inteligentes.

5 Resultados

Essa seção contém o conteúdo referente ao trabalho desenvolvido e resultados obtidos. Com isso, será apresentado o Context Data Proccess Orchestrator (CDPO), projetado e concebido como *middleware* para aplicações relacionadas a Cidades Inteligentes. Primeiramente serão descritos os requisitos necessários para conceber uma solução viável para o contexto de Cidades Inteligentes, que impactam diretamente nas decisões arquiteturais do projeto.

Por conseguinte, serão apresentados os diagramas referentes à arquitetura desenvolvida, dando ênfase nos microsserviços presentes nas múltiplas camadas, comunicação entre componentes e mecanismos de segurança implementados. Ademais, será apresentado um estudo de caso que aplica o uso da arquitetura para resolver um problema do mundo real. Por fim, conclusões serão relatadas a partir de experimentos para validar a arquitetura do modelo proposto, escalabilidade e acurácia no processamento de fluxos de dados.

5.1 Requisitos de Software

Para atingir o objetivo principal, planeja-se inicialmente a elicitação de requisitos funcionais e não funcionais. Nessa fase, o engenheiro de requisitos busca compreender o domínio do problema, captando as necessidades que o software deve atender (TURINE; MASIERO, 1996). Primeiramente, foram elicitados requisitos relacionados ao contexto da aplicação, com foco nas funcionalidades esperadas do *middleware* proposto. Em seguida, foi realizada uma análise de trabalhos relacionados, a fim de identificar necessidades recorrentes e desafios enfrentados por soluções semelhantes. Com base nessa elicitação, foram estabelecidos os requisitos que o *middleware* proposto opere de forma adequada:

- **(RF01) Implantação de Redes de Processamento de Eventos**. As *Event Process Network* (EPNs) especificam um conjunto de regras de processamento para executarem nas múltiplas camadas. Sendo assim, a plataforma deve receber uma Rede de Processamento de Eventos e executar o *deployment* da mesma em todos os níveis definidos na sua especificação.
- **(RF02) Desimplantação de Redes de Processamento de Eventos**. Além do *deployment*, é necessário desalocar as regras dos nós de processamento. Então, o *undeployment* de Redes de Processamento de Eventos é importante para que regras obsoletas sejam desimplantadas das múltiplas camadas.

- **(RNF01) Segurança na Implantação e Desimplanação**. O *deployment* e *undeployment* de redes de processamento de eventos deve ocorrer em um canal seguro de comunicação. Sendo assim, é necessário a utilização de um protocolo seguro de comunicação ao implantar regras de processamento nas múltiplas camadas.

- **(RNF02) Segurança na Comunicação entre Camadas**. Os eventos podem ser enviados entre camadas de processamento. Com isso, é necessário estabelecer um canal seguro de comunicação entre as camadas *edge*, *fog* e *cloud* utilizando protocolos seguros de comunicação.

- **(RNF03) Segurança na Entrega do Resultado do Processamento**. O envio do resultado do processamento aos possíveis destinos deve ocorrer de forma segura. Sendo assim, é necessário a implementação de um mecanismo de controle de acesso das redes de processamento de eventos no envio do resultado do processamento.

- **(RNF05) Evoluibilidade.** A capacidade de evolução é a capacidade do sistema para evoluir ao longo do tempo, suportando modificação rápida e aprimoramento com baixo custo e poucos impactos na arquitetura, e é um elemento fundamental para a sucesso e valor econômico de software de longa duração (BREIVOLD; CRNKOVIC; LARSSON, 2012). Desse modo, as plataformas relacionadas a Cidades Inteligentes devem ser adaptáveis, a fim de acompanharem o crescimento e novos requisitos do espaço urbano utilizando os recursos disponíveis de forma eficiente.

- **(RF03) Mobilidade:** A plataforma deve permitir que os dispositivos que acessam a plataforma sejam móveis, geralmente se comunicando via conexões sem fio. Além disso, os recursos da cidade também podem ser portáteis, fornecendo serviços de sensoriamento e atuação por meio de tecnologias sem fio.

- **(RF04) Repositório de recursos:** A plataforma deve manter um catálogo com informações sobre os recursos disponíveis na cidade, permitindo que os usuários pesquisem, inclusive com o uso de dados contextuais, para ativar recursos específicos.

- **(RNF06) Heterogeneidade de dispositivos:** A plataforma deve permitir a troca de dados entre diversos dispositivos de usuário e os vários recursos da cidade.

- **(RF03) Autenticação:** A plataforma deve permitir que usuários e sistemas de terceiros acessem os componentes da plataforma por meio de sistemas ou mecanismos de autenticação.

- **(RF04) Autorização:** A plataforma deve garantir que apenas usuários autorizados tenham acesso a todas ou partes das funcionalidades da plataforma, podendo adotar mecanismos de controle de acesso para garantir que os usuários tenham permissões adequadas para leitura, escrita e execução.

- **(RNF07) Privacidade de dados:** A plataforma deve garantir a privacidade dos dados dos usuários, incluindo aspectos legais, podendo usar técnicas para anonimizar os dados dos indivíduos e mecanismos para acesso aos dados conforme critérios de privacidade.
- **(RF05) Fluxos de dados ou eventos:** A plataforma deve ser capaz de ler *streams* de dados ou eventos, permitindo a busca de padrões nesses *streams* para gerar novos conhecimentos.
- **(RF06) Processamento de dados:** A plataforma deve oferecer suporte a diversas funcionalidades de processamento de dados para atender às necessidades das aplicações, como funções matemáticas e lógicas, além de aplicar agrupamentos e filtros aos dados.
- **(RNF08) Processamento de dados em tempo real:** A plataforma deve permitir o processamento e análise de dados em tempo real ou quase em tempo real.
- **(RNF09) Aplicações de Cidades Inteligentes:** A plataforma deve permitir que aplicações de cidades inteligentes sejam executadas, internas ou externamente, oferecendo funcionalidades que permitam o acesso a dados, análise de dados, repositórios de recursos e serviços, entre outros itens pertencentes à plataforma.
- **(RNF10) Flexibilidade:** A plataforma deve ser adaptável o suficiente para diferentes tipos de cidades: grandes ou pequenas, com poucos ou muitos usuários, com ou sem necessidades específicas, como gerenciamento de tráfego urbano, eventos culturais, saúde pública ou novas leis. Também deve suportar a extensão de novas funcionalidades e serviços.
- **(RNF11) Dados contextuais:** A plataforma deve receber dados contextuais dos recursos e usuários da cidade, permitindo que as aplicações localizem os recursos da cidade por meio de consultas usando dados contextuais.
- **(RNF12) Escalabilidade:** A plataforma deve ser escalável em várias frentes, como atender a um grande número de usuários e ser capaz de expandir geograficamente e administrativamente.

5.2 Especificação de Redes de Processamento de Eventos

As Redes de Processamento de Eventos, ou *Event Processing Networks* (EPNs), são definidas pelo usuário do Context Data Process Orchestrator (CDPO) com o objetivo de orquestrar a implantação das regras de processamento CEP nos níveis *edge*, *fog* e *cloud*. Uma EPN é composta por duas partes principais: a primeira corresponde à definição de seus atributos básicos, enquanto a segunda refere-se à especificação das regras de

processamento. As Tabelas 3 e 4 apresentam um resumo dos atributos necessários para a definição completa de uma rede de processamento de eventos.

Tabela 3 – Atributos para definição de rede de processamento de eventos

Atributo	**Tipo**	**Obrigatório**	**Descrição**
`enabled`	boolean	Não	Indica se a EPN está ativada/desativada.
`webhook_url`	string	Não	Webhook do resultado do processamento.
`rules`	array	Sim	Lista de regras de processamento.

Tabela 4 – Atributos da regra de processamento

Atributo	**Tipo**	**Obrigatório**	**Descrição**
`name`	string	Sim	Nome da regra de processamento.
`description`	string	Sim	Descrição da funcionalidade da regra.
`level`	string	Sim	Camada de execução da regra.
`target`	string	Sim	Destino do resultado do processamento
`tag_filter`	string	Sim	Filtro utilizando tags
`definition`	string	Sim	Consulta em EPL
`event_type`	string	Sim	Tipo do evento de entrada
`event_attributes`	map	Sim	Identificador único do veículo.
`output_event_type`	string	Sim	Tipo do evento gerado como saída.

O atributo *enabled* é utilizado para definir se a event process network irá iniciar habilitada, ou seja, processando fluxos de dados, ou em modo *stand by*, aguardando a ativação manual para realizar a implantação nos nós de destino. O *webhook_url* é definido pelo usuário para receber o resultado do processamento da EPN, sendo que qualquer camada (*edge*, *fog* e *cloud*), pode ser configuradas para o envio do resultado do processamento para o *webhook* escolhido.

As regras possuem a definição dos eventos que serão processados pelos nós. Sendo assim, toda regra possui um *name* e *description* que identifica a regra e descreve a sua função. Esses atributos são obrigatórios e auxiliam no entendimento da funcionalidade principal da EPN. Porém, não são utilizados para nenhuma funcionalidade específica, a não ser a identificação da regra implantada.

O *level* diz respeito ao nível que a regra irá executar, podendo ser *EDGE*, *FOG* ou *CLOUD*, e qualquer valor fornecido diferente dos estabelecidos irá ocasionar erro na implantação da rede de processamento de eventos. O *target* define o destino do resultado do processamento, podendo ser configurado como *EDGE*, *FOG*, *CLOUD*, *WEBHOOK* ou *INTERSCITY*. Quando configurado para o InterSCity, o nó de processamento atualiza as capacidades de recursos cadastrados previamente na plataforma InterSCity.

Os nós das aplicações recebem *tags* ou marcadores compostos por chave e valor. O microsserviço *tagger* possui a responsabilidade de atribuir esses marcadores, que podem ser recuperados por qualquer outro serviço. O atributo *tag_filter* recebe uma consulta,

que deve utilizar uma sintaxe especifica, desenvolvida para o microsserviço *tagger*, e com ela é possível definir qual conjunto de nós irá executar determinada regra, por exemplo as tags *group:ufma and type:carro* definem que o destino da regra devem ser nós que estejam no grupo "ufma"e sejam do tipo "carro".

O atributo *definition* recebe uma consulta na linguagem *esper* para ser instanciada no nó de processamento. Para definir o evento de entrada da consulta, é necessário descrevê-lo nos atributos *event_type*, que define o tipo do evento de entrada, *event_atributes* responsável por representar os atributos do evento com seus tipos primitivos e *output_event_type* que define o tipo do evento de saída.

5.3 Arquitetura do Context Data Process Orchestrator

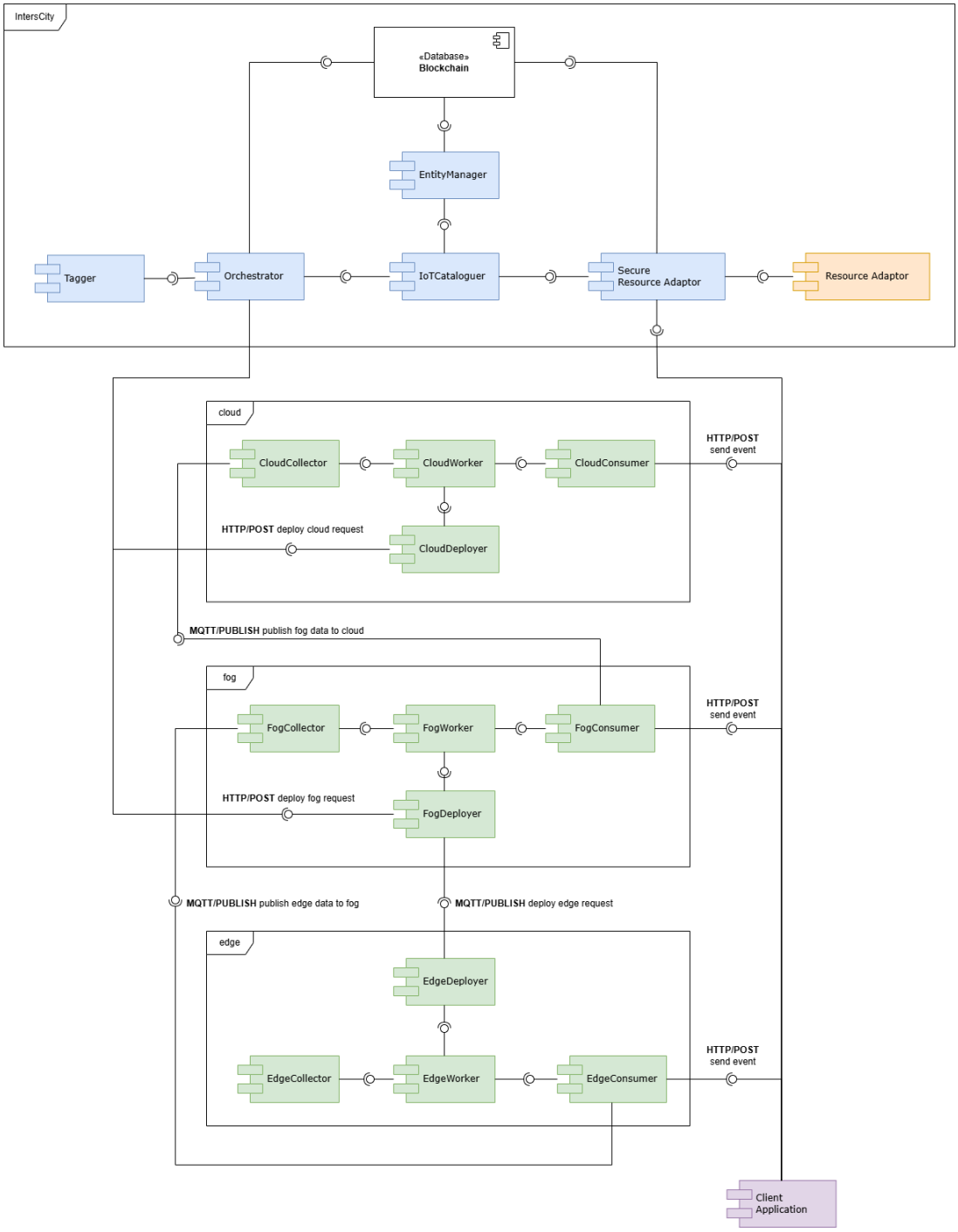
A arquitetura projetada para o Context Data Process Orchestrator (CDPO) foi construída levando em consideração os requisitos para a plataforma atuar como *middleware* de aplicações para o contexto das Cidades Inteligentes. Utilizando microsserviços agrupados em múltiplas camadas (*edge, fog e cloud*), o CDPO se integra a plataforma InterSCity sem modificar nenhum componente pré-existente, atuando como uma extensão direta. A Figura 4 ilustra a arquitetura do CDPO, que define microsserviços com responsabilidades específicas que se comunicam para permitir a implantação de regras de processamento em múltiplas camadas e o processamento de eventos complexos.

O `Tagger` é um serviço que possibilita a atribuição de *tags* dinâmicas à dispositivos IoT e nós de processamento. Por meio de *endpoints* que utilizam o protocolo HTTP disponibilizados pelo serviço, é possível atribuir *tags* e recuperar objetos por meio de uma linguagem específica que permite utilizar as principais operações lógicas, como AND, OR ou NOT. Por fim, o `Tagger` apenas armazena o registro das *tags* atribuídas aos objetos, mas o registro das entidades que receberam *tags* ficam registradas no `IotCataloguer`.

O componente `IotCataloguer` tem como responsabilidade manter o registro de dispositivos IoT e nós de processamento. Assim como o `Tagger`, o `IoTCataloguer` disponibiliza *endpoints* HTTP para recuperar dados armazenados, e permite o registro de novos dispositivos IoT e nós de processamento. Portanto, qualquer serviço que necessite consultar os dados das entidades participantes do processamento de eventos presentes na rede e recuperar suas informações, podem consultar o `IoTCataloguer`.

A *blockchain* foi utilizada na arquitetura para servir um modelo de gerenciamento de identidades. Responsável por armazenar as identidades de recursos da cidade, dispositivos IoT e nós de processamento, a *blockchain* fornece uma base para os mecanismos de segurança utilizados na solução proposta. Por meio dos contratos inteligentes, é possível realizar operações na *blockchain* de forma segura e auditável. Além disso, sua natureza

Figura 4 – Arquitetura do Context Data Process Orchestrator (CDPO)



distribuida e escalável viabiliza a sua aplicação na solução desenvolvida especialmente para as Cidades Inteligentes.

O componente `Entity Manager` tem com o objetivo de gerenciar as identidades presentes na *blockchain*. Por meio dos contratos inteligentes definidos para criação, consulta e revogação de identidades, o `EntityManager` gerencia as identidades presentes na *blockchain*. Através de *endpoints* HTTP, o `EntityManager` permite a criação de novas

identidades, que são necessárias para os mecanismos de segurança implementados na arquitetura proposta.

O microsserviço `Orchestrator` é responsável por receber redes de processamento de eventos como entrada e orquestrar a implantação/desimplantação nas múltiplas camadas (*edge*, *fog* e *cloud*). Para isso, o `Orchestrator` se comunica com o `Tagger` para realizar um filtro utilizando *tags* definidas na especificação da rede de processamento de eventos, com a finalidade de selecionar os nós de processamento que receberão regras CEP. Além disso, também se comunica com o `IoTCataloguer` para recuperar o registro dos nós de processamento que fazem parte da rede de múltiplas camadas.

Após selecionar os nós de processamento utilizando o `Tagger` e `IoTCataloguer`, o `Orchestrator` consulta a *blockchain* para validar a identidade dos nós escolhidos para o processamento. Essa consulta é importante para garantir que os nós escolhidos estão habilitados para o processamento dos fluxos de dados ou não estão com a sua identidade revogada. Por fim, ao enviar a rede de processamento para os nós de destino, é executado o protocolo *Challenge-Response Authentication*, apresentando um desafio que deve ser resolvido pela entidade que receberá a regra de processamento, com o objetivo de validar a sua identidade.

A arquitetura possui três camadas destinadas ao processamento de dados. A *edge* é a camada mais próxima do dispositivo IoT que coleta os dados, e é caraterizada por dispor de recursos limitados, porém com bom tempo de resposta. A camada *fog* deve ser implantada em proximidade a *edge*, com o objetivo de processar dados coletados pela borda mantendo um tempo de resposta consideravelmente menor que utilizar a computação em nuvem. A *cloud* é implantada em servidores remotos, geralmente com grande poder computacional, porém com maior tempo de respota em comparação com a *fog* e *edge*.

O microsserviço *Collector* é responsável por receber os fluxos de dados que serão destinados ao processamento. Na camada *edge*, o *Collector* possui uma interface direta com o dispositivo IoT que coleta os dados utilizando o protocolo MQTT [1]. Nas camadas *edge* e *fog*, o `Collector` é responsável apenas por receber os dados processados por camadas inferiores e direcionar os fluxos de dados para serem processados, também utilizando o protocolo MQTT para comunicação entre camadas.

O `Deployer` é o microsserviço que recebe as solicitações de implantação de regras de processamento. Nas camadas *fog* e *cloud*, o componente disponibiliza uma interface de comunicação utilizando o protocolo HTTP com SSL utilizada pelo `Orchestrator` para o envio das requisições de implantação. Porém na camada *edge*, o `Deployer` recebe requisições apenas da camada *fog* utilizando o protocolo MQTT, sendo uma solução que

[1] https://mqtt.org/

necessita menos poder computacional em relação à disponibilização de uma interface usando o protocolo HTTP.

O componente `Worker` é responsável por realizar o processamento das regras implantadas. Em todas as camadas, o `Deployer` envia as requisições de implantação de regras de processamento por meio do protocolo MQTT. Após isso, o `Worker`, realiza o *deployment* no motor Esper instanciado, além enviar os dados por meio de tópicos configurados pelo `Deployer` para receber dados do `Collector` e enviar o resultado do processamento.

O `Consumer` recebe o resultado de processamento realizado pelo `Worker` e direciona para o destino pré-configurado na definição da regra. Por meio do protocolo MQTT, o `Consumer` se inscreve nos tópicos configurados pelo `Deployer` no momento da implantação e direciona os dados processados para a *fog*, *cloud*, *webhook* ou InterSCity. Caso destino sejam as camadas de processamento, o consumer direciona os dados por meio do protocolo MQTT para a camada de destino. Porém, se o destino for um *webhook*, configurado na definição da rede de processamento de eventos, o consumer realiza uma requisição HTTP para o *endpoint* fornecido. Por fim, é possível também enviar o resultado ao InterSCity, por meio da interface HTTP disponibilizada pelo `Secure Resource Adaptor`.

Para atualizar as capacidades de recursos presentes no InterSCity, a arquitetura utiliza o `Secure Resource Adaptor`, um componente que atua como *middleware* entre o `Consumer` e `Resource Adaptor` fornecido pela plataforma InterSCity. Por meio de consultas a *blockchain*, o `Secure Resource Adaptor` valida a identidade dos nós de processamento que estão enviado dados ao InterSCity, garantindo que está recebendo dados apenas de entidades permitidas a participar da rede de processamento de eventos. Além de validar a identidade do `Consumer`, o próprio `Consumer` utiliza o protocolo *Challenge-Response Authentication* para confirmar a identidade do `Secure Resource Adaptor` que está recebendo o resultado do processamento.

5.4 Estudo de Caso (Gestão do Transporte Público)

O transporte público desempenha um papel fundamental no funcionamento das cidades modernas, sendo um elemento essencial para a mobilidade urbana sustentável e inclusiva. Ele oferece uma alternativa eficiente e acessível ao transporte individual motorizado, contribuindo para a redução do congestionamento nas vias, diminuição da emissão de poluentes e melhoria da qualidade do ar.

Sendo assim, a eficiência e a confiabilidade do transporte público impactam diretamente a qualidade de vida dos cidadãos e a produtividade das cidades. Atrasos, superlotação e falta de infraestrutura adequada afetam negativamente a experiência dos

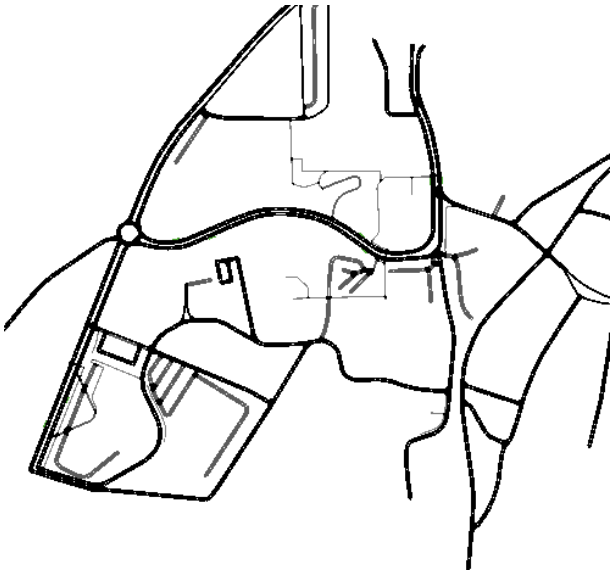
usuários e podem levar ao aumento do uso do transporte individual, agravando problemas urbanos como poluição e congestionamento.

Nesse contexto, esse estudo de caso tem como objetivo demonstrar como a solução proposta fornece uma infraestrutura capaz de coletar, processar e distribuir informações importantes sobre o tráfego do transporte público de uma cidade, a fim de que as autoridades competentes tomem decisões estratégicas para melhorar a eficiência dos transportes públicos, impactando positivamente a qualidade de vida dos cidadãos.

5.4.1 Metodologia

Para conduzir o estudo de caso foi utilizado uma simulação de tráfego urbano com o *software* SUMO (Simulation of Urban MObility). Através do simulador, foi realizado o mapeamento do campus Cidade Universitária da Universidade Federal do Maranhão (UFMA) conforme a Figura 5, registrando todas as vias e paradas de ônibus. Sendo assim, foram definidos 5 ônibus que trafegam pela simulação seguindo sempre a mesma rota. Outrossim, em determinados momentos da simulação foram definidos atrasos propositais de chegada às paradas de ônibus que acontecem de forma aleatória, além de um tráfego de carros definido como aleatório.

Figura 5 – Universidade Federal do Maranhão definida no simulador SUMO



Fonte: Autoria Própria

Desse modo, foi definido um *edge node* para cada ônibus da simulação. Cada *edge node* recebeu os dados coletados dos ônibus, que eram enviados a cada segundo, através do protocolo MQTT. Além disso, foi utilizado um *fognode* e um *cloudnode*, reponsáveis por realizar o processamento em camadas mais superiores caso seja necessário. Os dados processados foram enviados para um *webhook* de uma aplicação que registra os dados em uma base de dados para consultas posteriores. Sabendo disso, o JSON 5.1 ilustra um

fragmento importante para a definição da rede de processamento de eventos utilizada para esse estudo de caso.

Listing 5.1 – Definição da EPN para detecção de atrasos de ônibus

```
{
    "version":"1",
    "enabled":true,
    "webhook_url":"http://cdpo:8080/logger",
    "rules":[
         {
             "name": "SelectLateBus",
             "description": "Selecionando Atrasos",
             "level": "CLOUD",
             "target": "INTERSCITY",
             "tag_filter": "group:city",
             "definition": "select parada_com_atraso,
                timestamp from LateBus",
             "event_type": "LateBus",
             "event_attributes": {
                 "parada_com_atraso": "string",
                 "timestamp": "string"
             },
             "output_event_type": "SelectLateBus"
         },
         {
              "name":"LateBus",
              "description":"Detectar se o onibus esta
                 atrasado",
              "level":"FOG",
              "target":"CLOUD",
              "tag_filter":"block:ufma",
              "definition":"select a.bus_stop as
                 parada_com_atraso, current_timestamp.format
                 (\"yyyy.MM.dd HH:mm:ss\") as timestamp from
                 pattern[every a=BusArived -> (timer:interval(
                 10 min) and not BusArived(bus_stop=a.bus_stop
                 ))]",
              "event_type":"BusArived",
              "event_attributes":{
                 "bus_id":"int",
```

```
30                  "bus_stop":"string",
31                  "timestamp_arrived":"string",
32                  "timestamp_left":"string"
33              },
34              "output_event_type":"LateBus"
35          },
36          {
37              "name":"BusArivedPF",
38              "description":"Detectar a chegada do onibus na
                    parada do PF",
39              "level":"EDGE",
40              "target":"FOG",
41              "tag_filter":"type:bus",
42              "definition":"select a.lastOf().id as bus_id, '
                    PF' as bus_stop, a.firstOf().timestamp as
                    timestamp_arrived, a.lastOf().timestamp as
                    timestamp_left from pattern[every(a=Vehicle(
                    latitude = -2.55932 and longitude = -44.31212
                    ) until Vehicle(latitude != -2.55932 or
                    longitude != -44.31212))] where a is not null
                    ",
43              "event_type":"Vehicle",
44              "event_attributes":{
45                  "id":"string",
46                  "latitude":"double",
47                  "longitude":"double",
48                  "speed":"double",
49                  "timestamp":"string"
50              },
51              "output_event_type":"BusArived"
52            }
53          ]
54  }
```

Na camada *edge* foi definida uma regra de processamento que recebe eventos do tipo `BusLocation`, que representa eventos de localização dos ônibus da simulação, e realiza uma detecção de padrão que identifica a chegada de ônibus em determinado ponto geográfico. A saída dessa regra de processamento são eventos do tipo `BusArived`, que representa a chegada de um ônibus nas coordenadas pré-definidas, que são referentes a parada de ônibus

do Centro Pedagógico Paulo Freire, prédio de ensino localizado dentro da Universidade Federal do Maranhão.

Na camada *fog* foi implantada uma regra que utiliza os eventos do tipo `BusArived` recebidos da *edge* para realizar uma detecção de padrão que retorna um novo evento quando o intervalo entre os eventos de chegada dos ônibus atingem 10 minutos. Com isso, é possível detectar quando a parada de ônibus fica sem receber ônibus por pelo menos 10 minutos, gerando um novo evento nomeado como `LateBus`, que representa um atraso na parada analisada.

Na *cloud* foi definida uma regra simples que realiza a seleção dos eventos de atraso e os envia ao InterSCity. Quando o resultado do processamento é enviado ao InterSCity, são atualizadas as capacidades dos recursos vinculados aos nós de processamento. Para isso, foi necessário registrar previamente os recursos no InterSCity por meio do `Resource Cataloguer` e vincular as suas referências com os nós de processamento do tipo *edge* no `IoTCataloguer`.

Sendo assim, a rede de processamento implantada definiu uma regra de processamento na camada *edge* para cada parada de ônibus presente na Universidade Federal do Maranhão (UFMA), sendo que o que muda para cada regra é apenas a localização geográfica do ponto de ônibus. Além disso, toda a simulação, assim como a infraestrutura de camadas foi executada em uma rede local, onde atrasos por conta da latência de comunicação são mínimos.

Ademais, o resultado do processamento da camada *cloud* foi enviado para o InterSCity para que os resultados sejam posteriormente analisados. Portanto, a partir da implantação da infraestrutura de camadas e a definição da rede de processamento de eventos será possível realizar a detecção de atrasos em paradas de ônibus da cidade.

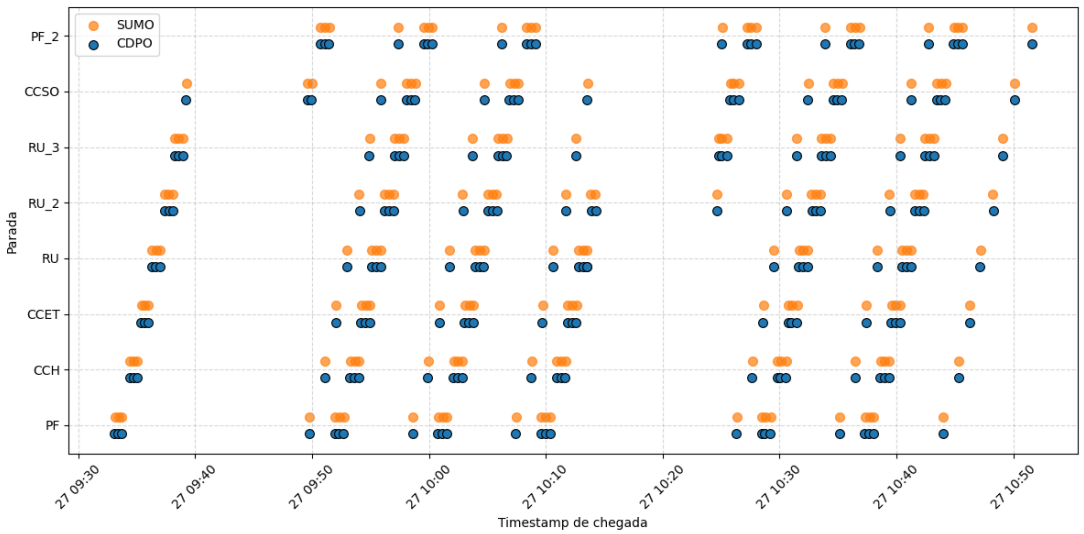
5.4.2 Resultados

Nessa seção descrevem-se os resultados obtidos a partir da simulação de tráfego no campus da UFMA e da infraestrutura de processamento de eventos em camadas (edge, fog e cloud). Serão analisadas métricas de desempenho do sistema, estatísticas de detecção de atrasos em paradas de ônibus e análises sobre a latência e a precisão das regras de processamento implementadas.

Para comparar os resultados obtidos com o processamento feito pela arquitetura desenvolvida foram utilizados os arquivos de *log* gerados pela própria simulação, os quais registram os eventos de localização dos ônibus e chegada nas paradas. Com isso, é possível analisar a precisão na detecção de atrasos do CDPO em comparação com os dados gerados pela simulação.

A Figura 6 ilustra um gráfico de dispersão temporal que compara a latência no processamento da chegada dos ônibus em pontos de parada com a simulação. Sabendo disso, é possível observar que todos os pontos referentes à chegada dos ônibus na simulação sobrepõem os pontos referentes ao processamento da chegada dos ônibus nas paradas pelo CDPO, demonstrando uma baixa latência na entrega do resultado do processamento em múltiplas camadas.

Figura 6 – Detecção dos eventos BusArived

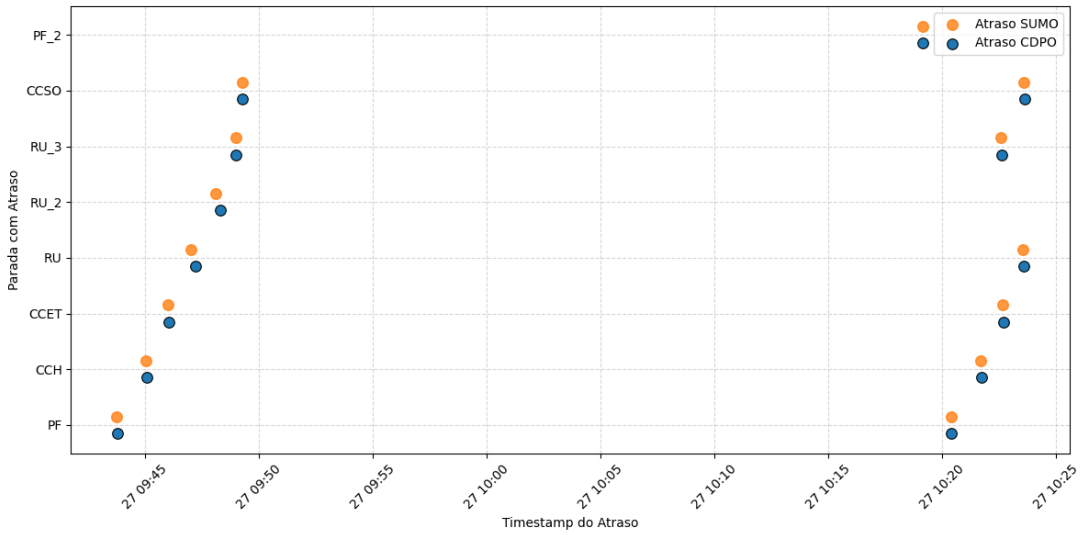


Fonte: Autoria Própria

Ademais, a Figura 7 ilustra um gráfico de dispersão que compara a detecção dos eventos de atraso (`LateBus`) com o atraso real gerado pela simulação. Com isso, é possível verificar que todos os eventos de atraso processados pelas múltiplas camadas possuíram baixa latência na entrega do resultado, se mostrando eficiente na detecção de padrões em tempo real.

Durante a execução do estudo de caso não houve nenhum falso positivo ou falso negativo. Ou seja, todos os atrasos de ônibus em paradas gerados pela simulação foram detectados pelo CDPO e nenhum atraso incoerente com a simulação foi detectado. Portanto, o processamento realizado pelas múltiplas camadas se mostrou preciso na entrega de resultados de acordo com a realidade simulada.

Figura 7 – Detecção dos eventos LateBus



Fonte: Autoria Própria

5.5 Estudo de Caso (Monitoramento de Vias Urbanas)

As vias urbanas destinadas à circulação de veículos, sejam transporte público ou particular, são utilizadas diariamente pela grande maioria da população residente nas cidades. O transito é um dos principais problemas urbanos, principalmente em cidades que tiveram um crescimento natural e não planejado, causando estresse, acidentes, atrasos e consequentemente piorando a qualidade de vida dos cidadãos.

Por isso, a análise do comportamento dos veículos que utilizam as vias públicas é necessária para obter conclusões acerca dos problemas de transito enfrentados nas cidades. Com essa análise, as autoridades responsáveis pela infraestrutura de transito conseguem tomar decisões estratégicas para melhorar o fluxo de veículos ou prevenir acidentes, como a criação de viadutos, elevados ou até mesmo a colocação de placas em trechos considerados perigosos.

Esse estudo de caso tem como objetivo mostrar que a solução desenvolvida é capaz de processar os eventos referentes a velocidade dos veículos em determinada região geográfica, e obter resultados que podem ser utilizados para tomadas de decisão estratégica por parte das autoridades responsáveis pela infraestrutura de transito da cidade.

5.5.1 Metodologia

Para conduzir o estudo de caso, foi utilizada uma simulação de tráfego urbano, assim como no estudo de caso anterior, com o SUMO (Simulation of Urban MObility). Na simulação também foi utilizado uma avenida do mapa da UFMA (Universidade Federal do

Maranhão) ilustrado na Figura 5 para análise da circulação de veículos. Foram definidos 10 veículos que percorrem a simulação em velocidades aleatoriamente variáveis.

Foi implantado um *edge node* para cada carro da simulação. Cada *edge node* recebia os dados coletados da simulação por meio do protocolo MQTT. Também foi definido um *webhook*, configurado na definição da rede de processamento de eventos para receber o resultado do processamento diretamente da camada *edge*, sem necessidade de utilizar *fog nodes* ou *cloud nodes* para obter as médias de velocidade dos veículos realizam o trajeto determinado. O JSON 5.2 ilustra a rede de processamento de eventos definida para detectar velocidade média dos veículos que realizaram o percurso definido.

Listing 5.2 – Definição da EPN para cálculo da velocidade média dos veículos

```
{
   "commit_id":"2",
   "version":"1",
   "enabled":true,
   "qos":"AT_MOST_ONCE",
   "atomic":true,
   "webhook_url":"http://cdpo:8080/webhook",
   "rules":[
      {
         "name":"CarMedia",
         "description":"Media de cada carro que passa pela
            avenida",
         "qos":"AT_MOST_ONCE",
         "level":"EDGE",
         "target":"WEBHOOK",
         "tag_filter":"type:vehicle",
         "definition":"select a.firstOf().id as car_id, a.
            average(i => i.speed) as media, current_timestamp
            .format(\"yyyy.MM.dd HH:mm:ss.SSS\") as timestamp
             from pattern[every (a=Vehicle(isLocationInArea(
            latitude,longitude,-2.557047,-44.309050,0.14) =
            True) until Vehicle(isLocationInArea(latitude,
            longitude,-2.557047,-44.309050,0.14) = False))]
            where a.firstOf().id is not null",
         "event_type":"Vehicle",
         "event_attributes":{
            "id":"string",
            "latitude":"double",
```

```
21              "longitude":"double",
22              "speed":"double"
23          },
24          "output_event_type":"CarSpeedAverage"
25      }
26  ]
27 }
```

Na camada *edge* foi implantada uma regra que recebe eventos do tipo `Vehicle`, composto pelos dados de latitude, logintude e velocidade dos veículos utilizados na simulação. Por meio da detecção de padrão, a regra acumula uma janela composta por todos os eventos `Vehicle` que acontecem em determinada área geográfica. Essa área é definida utilizando a função `isLocationInArea`, implementada no motor Esper do `Worker` com a linguagem java, para determinar se determinada localização está dentro de uma área circular composta pela latitude, longitude e um raio.

Após o primeiro evento do tipo `Vehicle` detectado pelo padrão definido na regra de processamento que não esteja dentro da área definida, a regra calcula a média de todas as velocidades instantâneas obtidas, resultado na velocidade média do veículo ao passar pela via escolhida da simulação. O evento resultado do processamento, do tipo `CarSpeedAverage`, é enviado diretamente para o *webhook* configurado na definição da rede de processamento.

5.5.2 Resultados

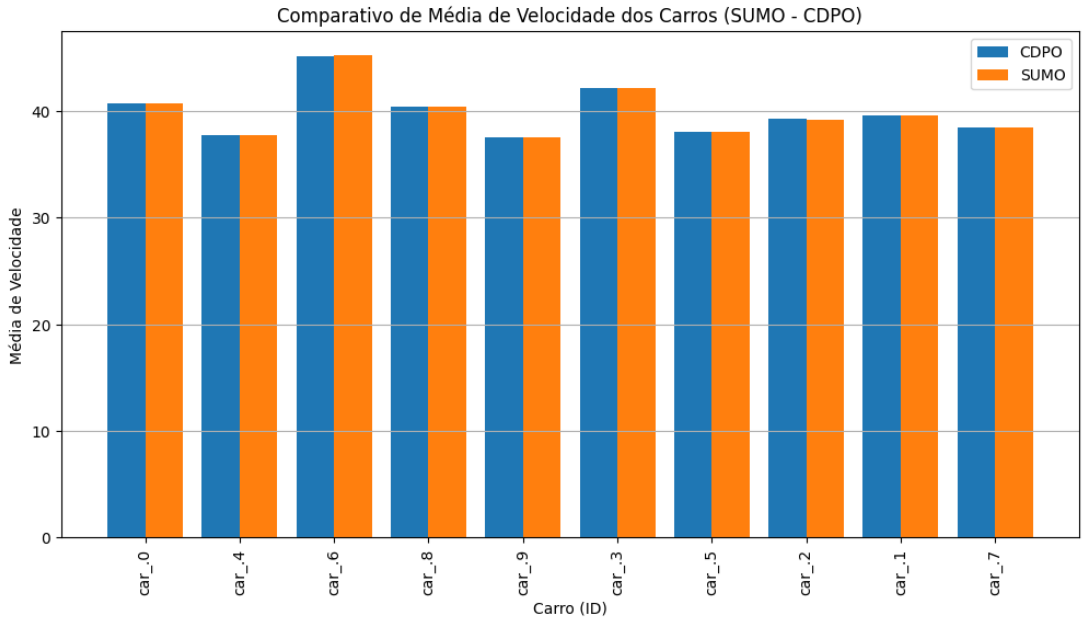
A execução da simulação gerou dois arquivos de log, um gerado localmente com os dados do próprio SUMO e outro gerado pelo Context Data Process Orchestrator (CDPO). A análise realizada consistiu em comparar os dois arquivos, com o objetivo de verificar a similaridade entre eles, e assim, definir o desempenho do *middleware* com base na acurácia, ou seja, se o CDPO conseguiu coletar os dados e realizar o cálculo correto das velocidades médias.

Primeiramente, houve a verificação de quantos registros de velocidade média cada sistema conseguiu detectar. A partir dessa análise, é possível verificar se houveram perdas de informações no processo de coleta do CDPO. Foi constatado que houveram 160 registros de velocidade média em ambos os arquivos, divididos igualmente entre os 10 veículos que compuseram a simulação, ou seja, 16 registros por id.

Percebe-se, logo, que o CDPO detectou todos os eventos ocorridos na simulação, não havendo perca de informações no momento da coleta. Contudo, é necessário que a EPN instanciada pelo middleware tenha tido uma boa taxa de acertos no cálculo das velocidades médias, isto é, consiga encontrar os mesmos valores calculados pelo SUMO. A

Figura 8 ilustra um gráfico de barras comparando a detecção de velocidades médias na simulação e no CDPO.

Figura 8 – Comparativo de Média de Velocidade dos Veículos



Fonte: Autoria Própria

A partir desses resultados, é possível observar que o CDPO obteve valores bem próximos em comparação aos valores calculados pelo SUMO, ou seja, o *middleware* conseguiu ter uma boa acurácia no cálculo da média das velocidades para todos os veículos da simulação. Para entender melhor as similaridades entre os resultados, a Tabela 5 demonstra alguns valores estatísticos de cada sistema.

Tabela 5 – Estatísticas de Velocidade

Estatística	**CDPO (km/h)**	**SUMO (km/h)**
Média de Velocidade	39.914	39.932
Média Máxima	45.123	45.207
Média Mínima	37.559	37.573
Desvio Padrão	2.226	2.237

Por fim, uma última análise realizada consistiu em avaliar o timestamp com a qual o CDPO registrou a velocidade média dos veículos. Essa análise é necessária para verificar se houve uma latência muito grande entre a ocorrência do evento no SUMO e a captação desse mesmo evento no CDPO. Para realizar essa análise, foi observado 18 o horário em que a velocidade média de cada veículo foi registrada nos arquivos de log do SUMO e do CDPO, e após isso, calculada a diferença entre o registro de cada volta. Desse modo, foi possível encontrar alguns dados estatísticos que relacionam os resultados, apresentados na Tabela 6

Tabela 6 – Latência da detecção de eventos

Estatística	**Valor (s)**
Latência Média	1,816
Latência Máxima	3.984
Latência Mínima	0.903

A partir dos dados da Tabela 6, é perceptível que não houve uma grande atraso entre a ocorrência do evento e sua detecção pelo CDPO, ficando a média em menos de dois segundos. É importante destacar que a comunicação via protocolo MQTT está sujeita às variações de rede, contudo, a maior diferença detectada não ultrapassou os 4 segundos. Dessa forma, nota-se que o CDPO detectou os eventos no momento de sua ocorrência, sem uma grande latência. Tal característica é essencial para que o middleware possa ser utilizado em contextos de aplicações que exigem baixo tempo de resposta.

5.6 Análise na Latência da Implantação de EPNs

Para que a solução desenvolvida para o contexto de Cidades Inteligentes desempenhe como esperado, é importante que os fluxos de eventos sejam processados com latência mínima. Por isso é necessário implantar e atualizar dinamicamente as regras de processamento através de múltiplas camadas de rede e computação sem introduzir atrasos significativos. A medição da latência de implantação é uma métrica essencial para validar o desempenho e a capacidade de resposta da arquitetura desenvolvida. Além disso, o baixo tempo de resposta é um requisito importante para aplicações clientes sensíveis ao tempo.

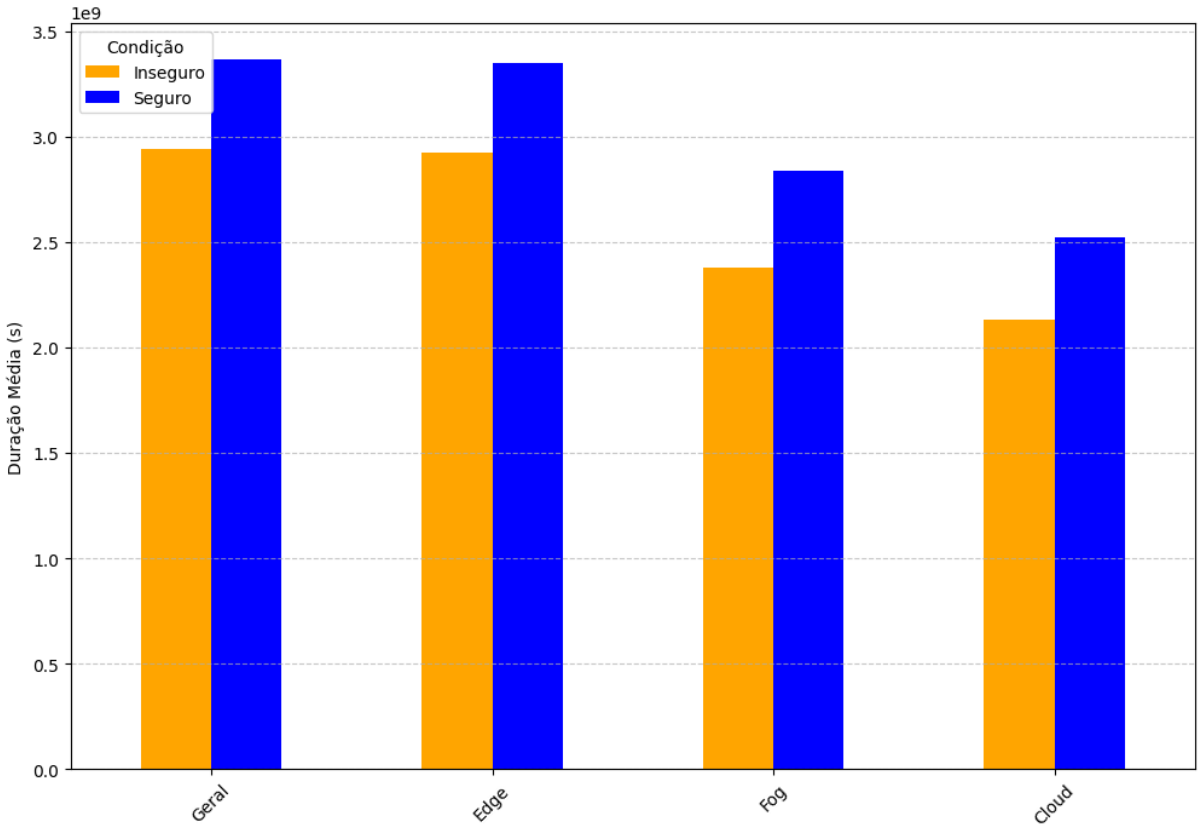
A partir do estudo de caso presente no Capítulo 5.4, foi possível realizar uma análise para avaliar o tempo de implantação da rede de processamento na arquitetura proposta, medindo a diferença na latência do *deployment* quando é deixado de utilizar os mecanismos de segurança baseados em *blockchain*. A Tabela 7 apresenta os resultados estatísticos de latência na implantação de 20 instâncias da rede de processamento. Com isso, é possível notar que a média da latência possui uma variação média de 147ms, o que consideramos ser um custo pequeno considerando os benefícios da implantação segura de redes de processamento no CDPO.

Estatística de Latência	**Arquitetura Insegura (s)**	**Arquitetura Segura (s)**
Média	2.943	3.090
Mediana	2.327	2.871
Máximo	3.416	3.950
Mínimo	2.664	2.698
Desvio Padrão	0.210s	0.316s

Tabela 7 – Comparação da Latência na Implantação

Também foi possível, a partir do estudo de caso presente no capítulo 5.4, avaliar a latência média nas camadas de processamento (*edge*, *fog* e *cloud*). Desse modo, a Figura 9 apresenta uma comparação entre o tempo médio de latência entre camadas de processamento levando em consideração o *deployment* seguro e inseguro. Com isso, é possível constatar que devido à baixa variação, de aproximadamente 139ms nas três camadas de processamento, a implementação dos mecanismos de segurança baseados em *blockchain* não afetou o desempenho da implantação de regras de processamento em múltiplas camadas de forma considerável.

Figura 9 – Latência Média do Deployment



Fonte: Autoria Própria

5.7 Discussão

Nessa seção serão discutidos os resultados obtidos apresentados anteriormente no capítulo de resultados. Abordando principalmente os resultados obtidos a partir da orquestração de estudos de caso. Também será discutido acerca da escalabilidade e evoluibilidade da arquitetura proposta para o contexto das Cidades Inteligentes.

5.7.1 Estudos de Caso

O estudo de caso da seção 5.5 demonstra o potencial do middleware CDPO para transformar a gestão de transporte público em Cidades Inteligentes. A capacidade de detectar atrasos de ônibus em tempo real e fornecer dados estruturados às autoridades representa um avanço significativo em relação aos métodos tradicionais. Com as informações processadas pelo CDPO, autoridades que administram os transportes públicos podem tomar decisões estratégicas para melhorar os serviços disponibilizados para a população, melhorando a qualidade de vida dos cidadãos.

Com a possibilidade de compor eventos ao definir redes de processamento de eventos, futuras extensões poderão incorporar análises preditivas, antecipando congestionamentos ou falhas na operação, possibilitando a extração de informações cada vez mais relevantes para grandes tomadas de decisão por parte da prefeitura das cidades, além de expandir o monitoramento a outras modalidades de transporte, consolidando a base para uma mobilidade urbana cada vez mais inteligente e responsiva.

Os resultados obtidos ao executar o estudo de caso demonstraram uma baixa latência na detecção de eventos de chegada dos ônibus aos pontos de parada (`BusArived`), assim como na detecção de eventos de atraso dos ônibus (`LateBus`). Além disso, não foi registrado nenhum evento incongruente com a simulação, demonstrando que todos os atrasos gerados foram detectados e nenhum atraso incoerente foi retornado como resultado, garantindo uma grande precisão no resultado do processamento feito pelo CDPO.

Ademais, por meio do estudo de caso presente na seção 5.4, foi demonstrado que o CDPO pode ser aplicado para a análise do comportamento dos veículos em determinada via pública. Com a capacidade de detectar a velocidade média de forma precisa, o CDPO consegue gerar resultados confiáveis para que as autoridades responsáveis pela infraestrutura de transito providenciem soluções para problemas cotidianos, como engarrafamentos ou altos índices de acidentes.

Os resultados obtidos por meio da análise de desempenho da execução da rede de processamento de eventos permitiram verificar que o *middleware* CDPO conseguiu detectar todos os eventos de média de velocidade gerados pela simulação. Além disso, o CDPO possuiu baixa latência na detecção de eventos de média de velocidade em comparação com os gerados pela simulação, com uma latência média de 1,816 segundos.

5.7.2 Escalabilidade

A escalabilidade da arquitetura proposta garante a sua viabilidade no contexto das Cidades Inteligentes, onde há uma crescente demanda por processamento de grandes volumes de dados gerados por sensores, dispositivos móveis, câmeras e demais recursos urbanos. A solução foi projetada para escalar de forma horizontal e distribuída, aproveitando

os paradigmas de *edge*, *fog* e *cloud computing*, que, em conjunto, possibilitam a descentralização do processamento e reduzem a sobrecarga em camadas superiores. Ao distribuir as tarefas de processamento entre diferentes nós conforme a proximidade com a fonte dos dados, a arquitetura consegue manter baixa latência e alto desempenho mesmo diante do aumento progressivo da carga de trabalho.

O uso de microsserviços independentes, que podem ser replicados ou substituídos de maneira isolada, contribui para a escalabilidade funcional da plataforma. À medida que novos dispositivos são integrados à rede urbana ou novos domínios de aplicação são incorporados, a plataforma consegue escalar de acordo com novas demandas sem comprometer os serviços já existentes. Cada camada da arquitetura pode ser escalada de forma autônoma, a *edge* pode ter novos sensores e dispositivos adicionados, a *fog* e *cloud* podem ser ampliadas com recursos computacionais adicionais, garantindo que o sistema responda adequadamente a diferentes níveis de complexidade e volume de dados.

Outro fator que reforça a escalabilidade é a utilização do protocolo MQTT para a comunicação entre os componentes, uma escolha estratégica por se tratar de um protocolo leve, eficiente e orientado a mensagens, ideal para sistemas distribuídos com restrições de largura de banda ou latência. Esse modelo de comunicação assíncrona desacopla os emissores dos receptores, permitindo que novos componentes sejam incorporados à rede sem causar interrupções ou impactos diretos na arquitetura existente. Dessa forma, a plataforma se mostra apta a acompanhar o crescimento orgânico e contínuo das cidades, mantendo sua eficiência operacional e oferecendo uma base sólida para a implementação de serviços urbanos inteligentes em larga escala.

5.7.3 Evoluibilidade

A utilização de microsserviços presente na organização interna dos nós de processamento distribuídos nas camadas *edge*, *fog* e *cloud* foi uma escolha arquitetural que reforça a capacidade de evolução do sistema ao permitir que cada nó possa ser estendido, modificado ou substituído por meio da adição de novos serviços independentes. Assim, novos requisitos funcionais ou não funcionais podem ser atendidos sem a necessidade de reconstruir o sistema como um todo, garantindo a continuidade do serviço mesmo durante atualizações ou mudanças estruturais. Essa modularidade promove a resiliência da arquitetura, pois falhas ou mudanças em um serviço específico não afetam os demais componentes.

Outro aspecto fundamental que fortalece a evoluibilidade da solução desenvolvida é a adoção de uma linguagem declarativa própria para a definição de redes de processamento de eventos. Essa linguagem permite aos desenvolvedores especificar regras de processamento que podem ser distribuídas dinamicamente entre os nós das diferentes camadas do sistema, sem necessidade de intervenção manual no código-fonte. Então, diante de novas demandas,

é possível gerenciar as regras de processamento em tempo real, adaptando o comportamento das aplicações clientes às novas condições. Essa flexibilidade reduz significativamente o tempo de resposta a alterações e minimiza os riscos associados a implantações frequentes.

A adoção de tecnologias consolidadas e amplamente utilizadas no ecossistema de desenvolvimento de software, como os protocolos HTTP e MQTT, o motor de processamento Esper e a tecnologia de *blockchain*, também representa um fator determinante para a evoluibilidade da arquitetura. Essas tecnologias são conhecidas por sua estabilidade, o que favorece a manutenção da compatibilidade com outras soluções e *framework* do mercado. Além disso, por serem baseadas em padrões abertos e bem documentados, facilitam a interoperabilidade com sistemas legados e novas plataformas que venham a surgir. Essa interoperabilidade é essencial em ambientes urbanos complexos e heterogêneos, nos quais diferentes sistemas precisam colaborar entre si.

6 Conclusão

Este trabalho teve como objetivo principal conceber, implementar e avaliar o *Context Data Process Orchestrator* (CDPO), uma extensão à plataforma InterSCity que orquestra regras de *Complex Event Processing* em múltiplas camadas (*edge, fog e cloud*) com mecanismos de segurança baseados em *blockchain*. Os resultados obtidos demonstram de forma clara que as metas estabelecidas foram alcançadas. Nos dois estudos de caso conduzidos, gestão do transporte público e monitoramento de vias urbanas, o CDPO obteve bons resultados ao detectar todos dos eventos relevantes, sem falsos positivos ou negativos, preservando a fidelidade do resultado do processamento.

Sobre desempenho, o *middleware* obteve latência média de 1,816s entre a ocorrência e a detecção dos eventos, com latência máxima de aproximadamente 4s. O acréscimo introduzido pelo modelo de gestão de identidade que permite autenticação/autorização utilizando *blockchain* não apresentou aumento significativo na latência da implantação de regras CEP, com variação média de 147ms quando comparada ao cenário sem segurança, evidenciando que é possível reforçar a confiabilidade sem comprometer o tempo de resposta.

No que se refere à escalabilidade, a arquitetura permite que novos nós de processamento sejam adicionados (ou removidos) sem interrupções, acomodando o crescimento orgânico de dispositivos e serviços nas cidades. Além disso, ao utilizar processamento em camadas próximas aos dispositivos IoT que coletam os dados (*edge e fog*), é possível diminuir a latência na entrega do resultado do processamento, que é uma grande necessidade para aplicações de tempo real. Ademais, a adoção de microsserviços independentes em cada camada reforça a evoluibilidade, pois facilita substituições e extensões pontuais sem impactos sistêmicos de grande porte.

Dentre as contribuições adicionadas pelo trabalho desenvolvido, destacam-se: (i) Propôs-se uma linguagem declarativa que descreve, de forma concisa, redes de processamento em múltiplas camadas (*edge, fog e cloud*); (ii) Desenvolveu-se uma extensão ao InterSCity que habilita o processamento de fluxos de dados, integrado ao ecossistema já consolidado da plataforma, ampliando seu escopo funcional para aplicações dependentes de eventos em tempo real. (iii) Incorporou-se um modelo de gestão de identidades baseado em *blockchain*, que provê autenticação, autorização e rastreabilidade de forma unificada, fortalecendo a segurança sem impacto significativo no desempenho.

Apesar disso, o trabalho desenvolvido apresenta algumas limitações: (i) O *deployment* do código de processamento está restrito a regras CEP, que apesar de bastante expressivas não contemplam computações como a execução de modelos de aprendizagem

de máquina; (ii) O registro dos eventos resultantes do processamento CEP na *blockchain* permitiria a auditoria e garantia de integridade dos mesmos.

Futuramente, existem objetivos importantes para serem atingidos: (i) validar o CDPO em ambientes urbanos reais; (ii) Explorar os usos do CDPO em outros domínios de aplicação, como gestão energética e saúde; (iii) Estender a implementação do CDPO com algoritmos de aprendizagem de máquina ou inteligência artificial para otimizar o uso dos recursos disponíveis.

Referências

AGGARWAL, S.; KUMAR, N. Hyperledger. In: *Advances in computers*. [S.l.]: Elsevier, 2021. v. 121, p. 323–343. Citado na página 20.

ATLAM, H. F.; WALTERS, R. J.; WILLS, G. B. Fog computing and the internet of things: A review. *big data and cognitive computing*, MDPI, v. 2, n. 2, p. 10, 2018. Citado na página 17.

BAUCAS, M. J.; SPACHOS, P. Edge-based data sensing and processing platform for urban noise classification. *IEEE Sensors Letters*, v. 8, n. 5, p. 1–4, 2024. Citado 2 vezes nas páginas 23 e 27.

BREIVOLD, H. P.; CRNKOVIC, I.; LARSSON, M. A systematic review of software architecture evolution research. *Information and Software Technology*, Elsevier, v. 54, n. 1, p. 16–40, 2012. Citado na página 30.

CAO, K.; LIU, Y.; MENG, G.; SUN, Q. An overview on edge computing research. *IEEE access*, IEEE, v. 8, p. 85714–85728, 2020. Citado na página 17.

CARAGLIU, A.; BO, C. D.; NIJKAMP, P. Smart cities in europe. *Journal of Urban Technology*, Routledge, v. 18, n. 2, p. 65–82, 2011. Disponível em: <https://doi.org/10.1080/10630732.2011.601117>. Citado 2 vezes nas páginas 12 e 18.

ESPOSTE, A. d. M. d.; KON, F.; COSTA, F. M.; LAGO, N. Interscity: A scalable microservice-based open source platform for smart cities. In: *Proceedings*. [S.l.: s.n.], 2017. Citado 5 vezes nas páginas 13, 17, 18, 19 e 28.

FAIRFIELD, J. A. Smart contracts, bitcoin bots, and consumer protection. *Wash. & Lee L. Rev. Online*, HeinOnline, v. 71, p. 35, 2014. Citado na página 20.

FLOURIS, I.; GIATRAKOS, N.; DELIGIANNAKIS, A.; GAROFALAKIS, M.; KAMP, M.; MOCK, M. Issues in complex event processing: Status and prospects in the big data era. *Journal of Systems and Software*, Elsevier, v. 127, p. 217–236, 2017. Citado na página 15.

GAYVORONSKAYA, T.; MEINEL, C. Blockchain. *Springer International Publishing*, Springer, v. 10, p. 978–3, 2021. Citado na página 20.

HE, S.; WANG, X.; LI, S.; PEI, K.; GUO, Z. High efficiency and fog flow direction-independent fog collector with pentagram bulges. *Surfaces and Interfaces*, Elsevier, v. 43, p. 103545, 2023. Citado 2 vezes nas páginas 24 e 27.

LI, D.; WONG, W. E.; GUO, J. A survey on blockchain for enterprise using hyperledger fabric and composer. In: IEEE. *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*. [S.l.], 2020. p. 71–80. Citado na página 21.

LYMPERIS, D.; GOUMOPOULOS, C. Sedia: A platform for semantically enriched iot data integration and development of smart city applications. *Future Internet*, MDPI, v. 15, n. 8, p. 276, 2023. Citado 2 vezes nas páginas 24 e 27.

MEDEIROS, F. S. B.; COLPO, I.; SCHNEIDER, V. A.; CARVALHO, P. S. de. Internet of things. *Revista Eletrônica de Administração e Turismo-ReAT*, v. 12, n. 7, p. 1652–1674, 2018. Citado na página 12.

NOFER, M.; GOMBER, P.; HINZ, O.; SCHIERECK, D. Blockchain. *Business & information systems engineering*, Springer, v. 59, p. 183–187, 2017. Citado 2 vezes nas páginas 19 e 20.

OKEGBILE, S. D.; MAHARAJ, B. T.; ALFA, A. S. A multi-user tasks offloading scheme for integrated edge-fog-cloud computing environments. *IEEE Transactions on Vehicular Technology*, IEEE, v. 71, n. 7, p. 7487–7502, 2022. Citado na página 17.

ORTIZ, G.; BAZAN-MUÑOZ, A.; LAMERSDORF, W.; PRADO, A. Garcia-de. Evaluating the integration of esper complex event processing engine and message brokers. *PeerJ Computer Science*, PeerJ Inc., v. 9, p. e1437, 2023. Citado na página 16.

PEREIRA, D.; BRAYNER, A. Ufcity: A software architecture to create data ecosystem in smart cities. In: *2023 Symposium on Internet of Things (SIoT)*. [S.l.: s.n.], 2023. p. 1–5. Citado 3 vezes nas páginas 23, 27 e 28.

PEREIRA, D. M. G. *A MICROSERVICES-AND ARTIFICIAL INTELLIGENCE-BASED SOFTWARE ARCHITECTURE FOR BUILDING FLEXIBLE SMARTCITY PLATFORMS*. Dissertação (Defesa de Doutorado) — UNIVERSIDADE FEDERALDOCEARÁ, Fortaleza, Ceará, mar. 2025. Defesa realizada em 11 de março de 2025. Citado 2 vezes nas páginas 22 e 25.

PIERRO, M. D. What is the blockchain? *Computing in Science & Engineering*, IEEE, v. 19, n. 5, p. 92–95, 2017. Citado na página 20.

POREDI, N.; CHEN, Y.; LI, X.; BLASCH, E. Enhance public safety surveillance in smart cities by fusing optical and thermal cameras. In: *2023 26th International Conference on Information Fusion (FUSION)*. [S.l.: s.n.], 2023. p. 1–7. Citado 3 vezes nas páginas 22, 24 e 27.

RITO, P.; ALMEIDA, A.; FIGUEIREDO, A.; GOMES, C.; TEIXEIRA, P.; ROSMANINHO, R.; LOPES, R.; DIAS, D.; VíTOR, G.; PERNA, G.; SILVA, M.; SENNA, C.; RAPOSO, D.; LUíS, M.; SARGENTO, S.; OLIVEIRA, A.; CARVALHO, N. B. de. Aveiro tech city living lab: A communication, sensing, and computing platform for city environments. *IEEE Internet of Things Journal*, v. 10, n. 15, p. 13489–13510, 2023. Citado 3 vezes nas páginas 22, 24 e 27.

SHI, Y.; DING, G.; WANG, H.; ROMAN, H. E.; LU, S. The fog computing service for healthcare. In: IEEE. *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech)*. [S.l.], 2015. p. 1–5. Citado na página 16.

SWANSON, T. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. *Report, available online*, v. 28, 2015. Citado na página 20.

THAKKAR, K.; PATEL, A.; PATEL, S.; PATEL, K.; NAYAK, A.; BUDHRANI, A. A complete virtual edge computing extrapolation architectural style, uses, and implications. In: *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*. [S.l.: s.n.], 2023. p. 63–70. Citado 2 vezes nas páginas 22 e 27.

TURINE, M. A. S.; MASIERO, P. C. Especificação de requisitos: uma introdução. 1996. Citado na página 29.

XIE, J.; TANG, H.; HUANG, T.; YU, F. R.; XIE, R.; LIU, J.; LIU, Y. A survey of blockchain technology applied to smart cities: Research issues and challenges. *IEEE communications surveys & tutorials*, IEEE, v. 21, n. 3, p. 2794–2830, 2019. Citado na página 20.

YEWALE, A. *Study of blockchain-as-a-service systems with a case study of hyperledger fabric implementation on Kubernetes*. Dissertação (Mestrado) — University of Nevada, Las Vegas, 2018. Citado na página 20.

YU, Z.; SONG, L.; JIANG, L.; SHARAFI, O. K. Systematic literature review on the security challenges of blockchain in iot-based smart cities. *Kybernetes*, Emerald Publishing Limited, v. 51, n. 1, p. 323–347, 2022. Citado na página 28.

ZACCARINI, M.; CANTELLI, B.; FAZIO, M.; FORNACIARI, W.; POLTRONIERI, F.; STEFANELLI, C.; TORTONESI, M. Voice: Value-of-information for compute continuum ecosystems. In: *2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN)*. [S.l.: s.n.], 2024. p. 73–80. Citado 4 vezes nas páginas 22, 23, 27 e 28.

ZHANG, X.; FENG, X.; ZHANG, H.; MI, B.; CHEN, K. Secure crowdsourced blockchain computation intelligence for iot systems. In: *2023 24th IEEE International Conference on Mobile Data Management (MDM)*. [S.l.: s.n.], 2023. p. 315–320. Citado 3 vezes nas páginas 23, 27 e 28.