Sistema SCADA Web Escalável e Interoperável para Compartilhamento de Bicicletas Elétricas com Base na Norma IEC 61850

João L. Marinho [*] **Giovanna Siqueira** [**] **Nájila Martins** [**] **Yona Lopes** [**]

[*] *Faculdade de Ciência da Computação, Universidade Federal Fluminense, RJ, (e-mail: jglimamarinho@id.uff.br).*
[**] *Escola de Engenharia, Universidade Federal Fluminense, RJ (e-mail: gcsiqueira@id.uff.br, najilamartins@id.uff.br, yonalopes@id.uff.br)*

Abstract: This paper proposes a scalable monitoring system for shared electric bicycles, based on Web SCADA and integrated with the MMS protocol, in accordance with the IEC 61850 standard, aiming for interoperability and expansion. The solution employs a distributed architecture featuring an IoT controller, standardized communication, and a web interface for real-time supervision. Validation was conducted using a functional prototype to verify full compliance with the functionalities, and a virtualized testbed with calibrated load simulators for progressive stress testing to assess scalability. The results show that the solution efficiently organizes critical data, improves operational supervision, and supports system expansion while ensuring interoperability with other applications.

Resumo: Este trabalho propõe um sistema de monitoramento escalável para bicicletas elétricas compartilhadas, baseado em SCADA Web e integrado ao protocolo MMS, conforme a norma IEC 61850, visando interoperabilidade e expansão. A solução utiliza uma arquitetura distribuída com controlador IoT, comunicação padronizada e interface web para supervisão em tempo real. A validação foi realizada por meio de um protótipo funcional para verificar o atendimento completo às funcionalidades e por uma bancada de testes virtualizada com simuladores de carga calibrados para avaliação progressiva de escalabilidade. Os resultados mostram que a solução permite organizar dados críticos de forma eficiente, melhorar a supervisão operacional e suportar a expansão do sistema ao mesmo tempo que assegura a interoperabilidade com outras aplicações.

Keywords: Electric Bicycles, SCADA Systems, Internet of Things, IEC 61850, MMS
Palavras-chaves: Bicicletas Elétricas, Sistemas SCADA, Internet das Coisas, IEC 61850, MMS

1 INTRODUÇÃO

A micromobilidade refere-se a uma categoria de veículos compactos e de baixa velocidade voltados para deslocamentos de curta distância em áreas urbanas conforme Rathore et al. (2024). O crescimento dos sistemas de micromobilidade elétrica, especialmente bicicletas compartilhadas, tem intensificado a necessidade de soluções eficientes de monitoramento e gestão. Esses sistemas são caracterizados por uma arquitetura distribuída, composta por múltiplos dispositivos conectados, que geram continuamente dados operacionais segundo Corti et al. (2024). A correta coleta, processamento e visualização dessas informações são fundamentais para garantir a disponibilidade do serviço, a segurança dos usuários e a eficiência operacional. Entretanto, à medida que o sistema se expande, aumenta também a complexidade associada à gestão desses dados, tornando essencial o uso de arquiteturas capazes de lidar com grande volume de informações e permitir a integração entre dispositivos heterogêneos, de acordo com Saini et al. (2024).

Diante desse contexto, propõem-se o desenvolvimento de um sistema de monitoramento escalável e interoperável para bicicletas elétricas compartilhadas e orientado à experiência do usuário. O sistema é baseado em uma arquitetura *Supervisory Control and Data Acquisition* (SCADA) Web integrada ao protocolo *Manufacturing Message Specification* (MMS). Essa abordagem permite a padronização da comunicação, a interoperabilidade entre dispositivos e a expansão do sistema conforme Pitta et al. (2023). O sistema faz parte do projeto MoveUFF (Edital FAPERJ Nº 11/2021), que visa apoiar a mobilidade urbana sustentável ao implementar uma solução completa, do hardware ao sistema, para compartilhamento de bicicletas elétricas para discentes e docentes da Universidade Federal Fluminense (UFF).

O artigo está estruturado em mais cinco seções. A Seção 2 revisa a literatura pertinente e os fundamentos teóricos são resumidos na Seção 3. A Seção 3.2 detalha a proposta do projeto e a Seção 4 apresenta a implementação e os resultados obtidos. Por fim, a Seção 5 conclui o artigo e expõe perspectivas futuras.

2 TRABALHOS RELACIONADOS

No contexto de sistemas de micromobilidade e monitoramento distribuído, a literatura recente apresenta avanços relevantes em comunicação *Internet of Things* (IoT), transmissão de dados e gerenciamento de frotas, mas ainda com limitações quanto à padronização semântica das informações, à interoperabilidade entre componentes e à escalabilidade de arquiteturas de supervisão. Neste sentido Anandkumar and Kalidoss (2023) investigaram *gateways* IoT e protocolos de comunicação para o monitoramento em tempo real, destacando o papel dessas tecnologias na transmissão eficiente de dados. Embora o estudo forneça um arcabouço conceitual aplicável a sistemas de compartilhamento de bicicletas, ele não apresenta uma implementação concreta do sistema de monitoramento para frotas, tampouco aborda a padronização dos dados segundo modelos de informação industriais como a *International Electrotechnical Commission* (IEC) 61850. De mesmo modo, Gao et al. (2019) propuseram o *Probability-Driven Opportunistic Forwarding* (POF), uma estratégia de *store-carry-forward* para sistemas de compartilhamento de bicicletas, na qual as próprias bicicletas atuam como relés móveis, transportando e encaminhando dados sempre que possível até estações receptoras, porém sem padronização semântica dos dados. O sistema E-WheelShare de Salimbangon et al. (2025) valida a adoção de plataformas IoT para compartilhamento de bicicletas em campus universitário, contudo sem padronização industrial nem escalabilidade horizontal.

Em um contexto similar, porém mais próximo da adoção de padrões industriais, Ustun et al. (2019) investigaram a implementação de gerenciamento integrado de carga de veículos elétricos baseado na IEC 61850, identificando que a norma carece de ferramentas de projeto suficientes para soluções de recarga inteligente e que a ausência de capacidades abrangentes de integração web dificulta o desenvolvimento de soluções interoperáveis para o gerenciamento de frotas crescentes. O presente trabalho endereça essas lacunas ao propor uma arquitetura em três camadas: o controlador embarcado, a estação intermediária e o servidor central, que preserva a estrutura hierárquica da IEC 61850 desde a coleta no sensor até a persistência no servidor central e a traduz em estruturas *JavaScript Object Notation* (JSON) consumíveis por uma *Application Programming Interface* (API) REST, operando sobre uma camada de servidor em Kubernetes com escalabilidade horizontal.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Visão Geral da norma IEC 61850

O SCADA é um sistema de controle e monitoramento que permite aos operadores acompanhar e atuar sobre processos distribuídos geograficamente, por meio de ferramentas computadorizadas. Mercurio et al. (2009) demonstraram que sistemas SCADA podem ser realizados de forma aberta e interoperável por meio de serviços web, padronizados pelas normas IEC 61970 e IEC 61850.

A IEC 61850 padroniza a comunicação em sistemas de automação por meio de um modelo de informação orientado a objetos, organizado em uma estrutura hierárquica. Na TR 61850-90-8 a norma endereça mobilidade elétrica sendo uma referência em interoperabilidade para o sistema de energia, Pitta et al. (2023). O protocolo MMS, definido pela ISO 9506, é o mapeamento concreto dos serviços *Abstract Communication Service Interface* (ACSI) da IEC 61850-8-1 sobre TCP/IP, oferecendo serviços de autodescrição, leitura, escrita e subscrição de relatórios por meio do modelo de *Virtual Manufacturing Device* (VMD), como detalha a IEC 61850 (2003-2025).

3.2 SCADA Web interoperável e Escalável

A arquitetura desse sistema foi concebida em quatro subsistemas interdependentes, sendo o controlador da bicicleta, o controlador da estação, o servidor central e o sistema de monitoramento, organizados de modo que o dado percorra um caminho padronizado pela IEC 61850 desde a leitura do sensor embarcado até a sua exibição na interface do operador. A Figura 1 apresenta o fluxo de dados dos controladores embarcados na bicicleta e nas docas com o Servidor Central e o Sistema de Monitoramento, que são detalhados a seguir:

i) Controlador da Bicicleta: corresponde ao controlador embarcado na bicicleta, responsável pela aquisição de dados brutos dos sensores. O controlador é um servidor MMS instanciado localmente com a biblioteca LibIEC61850 em plataforma ARM (Orange Pi). Cada grandeza é mapeada nos atributos definidos na modelagem de dados da IEC 61850. Após a atualização dos atributos no servidor MMS, o controlador avalia a disponibilidade de conexão com o Servidor Central. Caso exista conectividade, ocorre a tentativa de envio direto dos relatórios. Em caso de falha, os dados são persistidos em um banco SQLite local, que atua como *buffer*, que é uma memória temporária embarcada. Esse mecanismo de *store-and-forward* garante que nenhuma leitura seja perdida, mesmo em áreas sem cobertura de sinal. O banco local é submetido a rotinas periódicas de limpeza, que removem registros já sincronizados.
ii) Controlador da Estação: atua como concentrador intermediário em uma Orange Pi, sincronizando-se com o servidor MMS da bicicleta na doca para transferir os dados do SQLite local para o seu próprio *buffer*. A abordagem preserva a estrutura da norma IEC 61850, assemelhando-se ao *store-carry-forward* de Gao et al. (2019). Em seguida, tenta o envio direto ao Servidor Central; se indisponível, enfileira os dados por prioridade (críticos primeiro) em uma *Redis Queue* (fila de mensagens) local até que a conexão seja restabelecida.
iii) Servidor Central e Camada de Tradução: concentra a telemetria da frota utilizando um Cliente MMS (LibIEC61850). A camada de tradução extrai valor, qualidade e tempo dos relatórios, convertendo a semântica IEC 61850 para estruturas JSON que preservam a hierarquia original. Códigos de qualidade (ex: `bom`, `inválido`) são adaptados para a interface e enviados via API REST. O servidor opera em *cluster* Kubernetes (k8s), garantindo escalabilidade horizontal automática de *pods* conforme a demanda, com os dados persistidos estruturalmente para supervisão e armazenamento.
iv) Sistema de Monitoramento: interface para supervisão em tempo real da frota e estações, com acesso restrito por uma camada de Autenticação. Sua arquitetura divide-

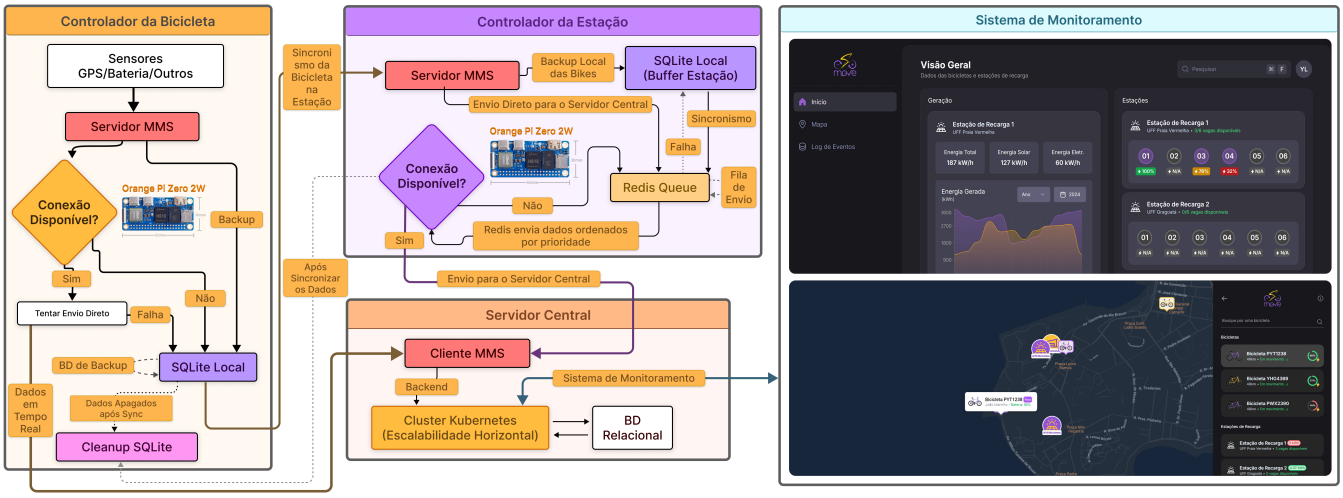

Figura 1. Arquitetura geral do sistema de monitoramento MoveUFF

se em duas partes: a Camada de Dados, que expõe os registros do PostgreSQL via API REST, WebSockets e balanceamento de carga (Traefik); e uma Interface do Usuário, desenvolvida em React/Next.js. Esta última separa componentes de cliente (focados em interface, cache e notificações) e de servidor (focados em análise, segurança e persistência de dados).

4 IMPLEMENTAÇÃO E RESULTADOS

A implementação foi validada em uma bancada de testes com dispositivos ARM que simulam controladores embarcados e enviam, em tempo real, dados já mapeados segundo a IEC 61850 ao Servidor Central. A avaliação concentrou-se na camada de orquestração executada em um nó de alta capacidade, com foco no comportamento elástico dos microsserviços.

A arquitetura adotada é apresentada na Figura 2, na qual o Serviço de Ingestão é o ponto de entrada do *pipeline*, responsável por receber, validar e classificar as mensagens provenientes da camada de tradução. Na arquitetura de produção, opera como cliente MMS da LibIEC61850. Para a validação de escalabilidade, o serviço recebe *buffers* ASN.1/BER pré-gerados pela LibIEC61850 e os submete à rotina de decodificação da própria biblioteca, reproduzindo o custo real do parser MMS sem depender da camada de transporte. Essa abordagem produz um tempo de serviço empírico com variabilidade realista associada a cache e predição de desvios, consistente com as medições de Hwang et al. (2020) em hardware ARM equivalente e com avaliações de desempenho de ponta a ponta em ambientes de teste instrumentados, conforme Demidov et al. (2023).

O item 2, *Redis Streams* (fila priorizada), é a camada de desacoplamento entre ingestão e processamento no Servidor Central, distinta do *Redis Queue* local da estação (§3.2), que atua como *buffer* de conectividade. As mensagens são organizadas em três filas de prioridade, inspiradas na abordagem de Akshatha et al. (2024), *Critical* (alertas de segurança), *Urgent* (variações operacionais significativas) e *Standard* (telemetria periódica). Sob saturação da fila, os consumidores despacham preferencialmente as mensagens de maior criticidade. Conforme demonstrado por Doshi et al. (2024), Redis, como camada de sincronização em arquiteturas distribuídas, garante persistência de mensagens em trânsito e tolerância a falhas.

Em seguida, o Serviço de Processamento consome as mensagens da fila Redis e realiza a tradução e o enriquecimento dos dados. A tradução converte as referências hierárquicas da IEC 61850 em estruturas JSON normalizadas, o enriquecimento interpreta os códigos de qualidade e calcula o índice de criticidade $C = w_1\,(1\text{–SoC}) + w_2\,(d/d_{\max}) + w_3\,q_{\mathrm{el}}$, em que State of Charge (SoC) é o estado de carga, d a distância à doca mais próxima (indicador indireto de risco de abandono) e q_{el} o desvio normalizado de tensão e corrente em relação aos valores nominais. Com pesos calibrados empiricamente $w_1 = 0{,}5$, $w_2 = 0{,}3$, $w_3 = 0{,}2$, os limiares $C > 0{,}7$ e $C > 0{,}3$ mapeiam as filas *Critical* e *Urgent*, respectivamente; demais mensagens seguem como padrão.

No item 4, Serviço de Persistência, são persistidas no PostgreSQL as estruturas JSON enriquecidas, mantendo o histórico completo de telemetria com indexação temporal e espacial, e executa rotinas de agregação periódica para séries temporais compactas.

Por fim, o serviço de Distribuição WebSocket entrega atualizações em tempo real ao Sistema de Monitoramento via sessões WebSocket persistentes, filtradas por região geográfica ou grupo de bicicletas. O *Horizontal Pod Autoscaler* (HPA) escala automaticamente com base no número de conexões ativas.

Essa decomposição permite que cada componente escale de forma independente. Conforme Ismail (2025), a modularidade inerente aos microsserviços promove escalabilidade independente em aplicações IoT. A adoção de Kubernetes na borda é corroborada por Saini et al. (2024), que demonstram que plataformas como KubeEdge permitem processamento localizado com latência reduzida.

4.1 Ambiente de Validação Experimental

Para validar a escalabilidade, foi construída uma bancada de testes laboratorial composta por dispositivos ARM e in-

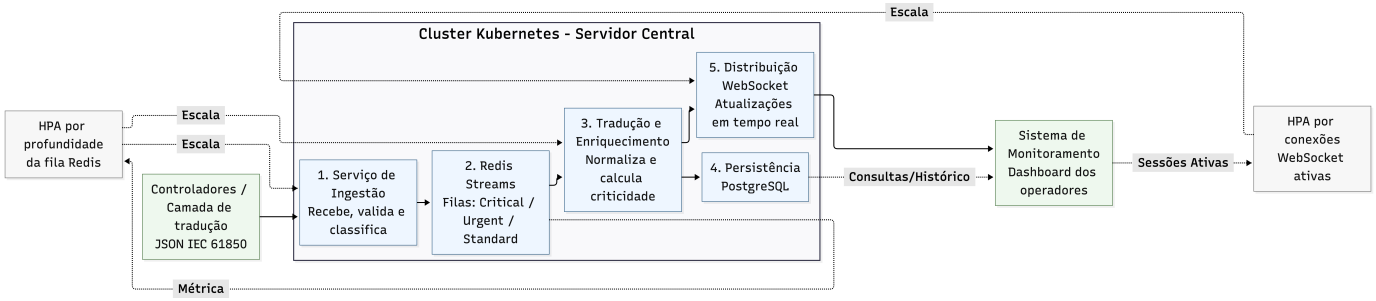

Figura 2. Arquitetura de microsserviços do Servidor Central com escalonamento horizontal automático baseado na profundidade da fila Redis.

fraestrutura virtualizada. A abordagem foi inspirada pelo *framework* proposto por Pereira et al. (2023), que utiliza contêineres Docker para emular redes IoT heterogêneas em larga escala.

Dispositivos embarcados físicos: duas placas Orange Pi Zero 2W, atuando como controlador de bicicleta e controlador de estação (Servidores MMS), foram utilizadas para validação funcional do protocolo na porta 102 com a LibIEC61850 compilada para ARM segundo Lee and Hwang (2025). Para o teste de escalabilidade, a camada de transporte MMS é substituída por simuladores de carga que preservam a semântica da IEC 61850 no nível de aplicação, com custo computacional de decodificação equivalente calibrado.

Simuladores de carga: contêineres Docker que emulam o comportamento completo de múltiplos IEDs, com simulação contínua de posição GPS, degradação de bateria e correlação bateria–esforço com ruído gaussiano. As cargas úteis (*payloads*), preservam a hierarquia completa da norma IEC 61850 com formato `mag/q/t` ou `stVal/q/t`, incluindo distribuição realista de códigos de qualidade (95% `good`, 3% `questionable`, 2% `invalid`). Os *payloads* são transmitidos ao Serviço de Ingestão em intervalos de 5 segundos por bicicleta, com variação de atraso temporal configurável. Essa abordagem é análoga ao método de Pereira et al. (2023), e sua validade é corroborada por Xue et al. (2024).

Cluster Kubernetes: implantado via k3d em uma estação Apple com o chip M4 (ARM64, 32 GB RAM, SSD NVMe), com os quatro microsserviços da arquitetura, o agente de mensagens *Redis Streams* e o banco PostgreSQL. O HPA foi configurado com métricas customizadas baseadas na profundidade da fila Redis, conforme Galande et al. (2024). A escolha do Apple M4 como plataforma de testes é conservadora em relação ao ambiente de produção previsto, no qual o Servidor Central operará em infraestrutura de um centro de dados com maior capacidade computacional.

Ferramentas de monitoramento: Prometheus e Grafana para métricas de infraestrutura, um serviço coletor de latência dedicado que calcula estatísticas fim-a-fim (média, P95, P99) e taxa de entrega e *scripts* customizados para coleta de capturas de estados via PromQL.

4.2 Protocolo de Teste de Estresse Progressivo

O teste de estresse foi conduzido em cinco fases progressivas, cada uma incrementando o número de bicicletas simuladas enquanto mantinha fixo o número de estações (4 docas) e o intervalo de transmissão (5 segundos por bicicleta). O protocolo foi projetado seguindo a metodologia de Dwiyanto et al. (2025), que demonstrou estabilidade de um sistema de API REST para monitoramento veicular com até 100 dispositivos concorrentes.

Cada fase foi executada por 15 minutos após estabilização, com métricas coletadas nos últimos 10 minutos para eliminar efeitos transitórios. Os valores reportados correspondem à média de cinco execuções independentes por fase.

Previamente às fases de carga, o protocolo MMS sobre LibIEC61850 foi validado fim-a-fim em bancada com os Orange Pi físicos atuando simultaneamente como Servidor MMS na bicicleta e Cliente MMS no Servidor Central, integrados à infraestrutura completa de fila, persistência e distribuição, conforme Lee and Hwang (2025). Nessa validação preliminar, a conformidade dos *payloads* com o modelo de dados da IEC 61850 foi verificada por inspeção das mensagens, confirmando a preservação integral da hierarquia de dados da norma. Durante as Fases 1–5 do teste de estresse, o transporte MMS físico é substituído pelos simuladores de carga calibrados descritos em §5.1, o que constitui uma limitação metodológica assumida.

Na Fase 1 (10 bicicletas, 2 msgs/s), o sistema operou com uma réplica de cada serviço, apresentando latência média de 47 ms e taxa de entrega de 100%, estabelecendo a linha de base de desempenho do *pipeline* de microsserviços.

Na Fase 2 (50 bicicletas, 10 msgs/s), o sistema manteve-se estável com uma réplica de cada serviço, com latência média de 58 ms.

A Fase 3 (200 bicicletas, 40 msgs/s) representou o primeiro teste significativo do autoescalonamento. A profundidade da fila Redis ultrapassou o limiar configurado, disparando o HPA que instanciou uma segunda réplica do Serviço de Ingestão e do Serviço de Processamento, com tempo de escalonamento de 22 segundos [1] . Após estabilização, a latência média situou-se em 82 ms com taxa de entrega de 100%.

Na Fase 4 (500 bicicletas, 100 msgs/s), o sistema demonstrou comportamento elástico consistente: o Serviço de Ingestão escalou para 4 réplicas, o de Processamento

[1] Os tempos de escalonamento referem-se ao intervalo entre o cruzamento do limiar e a criação efetiva do *pod*. A Fase 3 é mais lenta por partida a frio do servidor de métricas, as seguintes operam próximo ao período padrão do HPA (15 s).

para 3 e o WebSocket para 2. A latência média elevou-se para 134 ms, com taxa de entrega de 99,97% - aproximadamente 18 mensagens descartadas em cerca de 60.000, todas do tipo *Standard* e ocorridas durante a transição do autoescalonamento. Sob esta condição de saturação, o mecanismo de priorização mostrou-se efetivo: mensagens da fila *Critical* apresentaram tempo médio de espera 34% menor que mensagens *Standard*. Com 500 marcadores no mapa, a camada de interface do usuário empregou agrupamento geográfico (Figura 3), reduzindo os elementos DOM de 500 para aproximadamente 80 *clusters*.

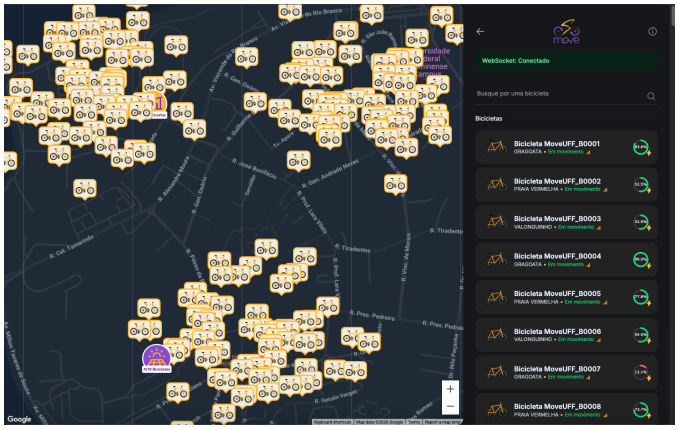

Figura 3. Interface web de monitoramento em tempo real durante a Fase 4 dos testes na bancada de testes.

Na Fase 5 (1.000 bicicletas, 200 msgs/s), o Serviço de Ingestão escalou para 7 réplicas, o de Processamento para 5 e o WebSocket para 3, totalizando 15 *pods* ativos. A latência média atingiu 218 ms (P95 em 347 ms) - valores compatíveis com operação SCADA, onde a resolução temporal de segundos é adequada. A taxa de entrega de 99,91% representa a perda de aproximadamente 108 mensagens em 120.000, concentradas em momentos de escalonamento. Nenhuma mensagem *Critical* ou *Urgent* foi descartada, validando a eficácia do mecanismo de priorização.

4.3 Teste de Resiliência a Falhas

Durante a Fase 5, um *pod* do Serviço de Ingestão foi deliberadamente encerrado (`kubectl delete pod`) para simular uma falha de infraestrutura. O Kubernetes detectou a indisponibilidade em 4 segundos e criou um *pod* substituto operacional em 11 segundos. Durante esse intervalo de 15 segundos, as mensagens foram retidas na fila Redis sem perda, confirmando a resiliência do enfileiramento persistente, consistente com as propriedades demonstradas por Doshi et al. (2024). Conforme destacado por Ergun et al. (2023), técnicas de gerenciamento dinâmico de confiabilidade em sistemas IoT de borda são fundamentais para garantir qualidade de serviço em cenários em que falhas individuais não devem comprometer a operação global.

4.4 Análise dos Resultados

A Tabela 1 consolida as métricas coletadas nas cinco fases do teste de estresse progressivo, acompanhadas em tempo real via painel Grafana (Figura 4).

Os resultados demonstram que a arquitetura suporta crescimento linear da frota com degradação sublinear de desempenho: enquanto o número de *bikes* (bicicletas) cresceu 100 vezes (de 10 para 1.000), a latência média cresceu apenas 4,6 vezes (de 47 para 218 ms), e a taxa de entrega manteve-se acima de 99,9%. Esse comportamento é atribuído à eficácia do autoescalonamento horizontal e ao desacoplamento proporcionado pelo Redis. Registre-se que esses resultados refletem a escalabilidade computacional em nó único de alta capacidade, a validação entre múltiplos nós físicos permanece como trabalho futuro.

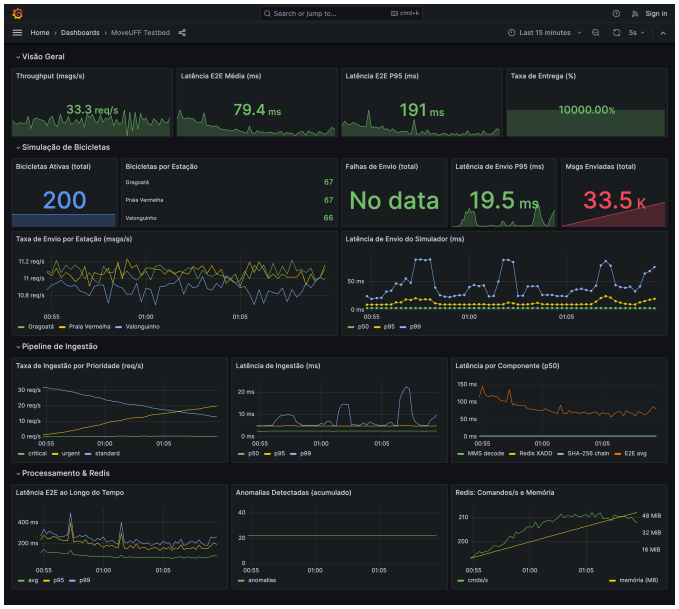

Figura 4. Painel Grafana mostrando métricas durante o teste de estresse.

Tabela 1. Teste de estresse progressivo do sistema de monitoramento MoveUFF

Métrica	**Fase 1** **10 *bikes***	**Fase 2** **50 *bikes***	**Fase 3** **200 *bikes***	**Fase 4** **500 *bikes***	**Fase 5** **1000 *bikes***
Ingestão (msgs/segundo)	2	10	40	100	200
Latência média fim-a-fim (ms)	47	58	82	134	218
Latência P95 (ms)	63	89	127	198	347
Taxa de entrega (%)	100,0	100,0	100,0	99,97	99,91
Pods - Ingestão	1	1	2	4	7
Pods - Processamento	1	1	2	3	5
Pods - WebSocket	1	1	1	2	3
CPU média cluster (%)	8	14	31	52	74
Memória média cluster (%)	12	18	29	41	58
Profundidade máx. fila Redis	3	12	48	127	312
Tempo de escalonamento HPA (s)	-	-	22	18	15

A comparação com trabalhos correlatos reforça a adequação dos resultados. Dwiyanto et al. (2025) reportam tempos de resposta entre 3,93 e 9,19 ms para até 100 dispositivos. O presente sistema apresenta latência superior por incorporar camadas de enfileiramento priorizado e tradução semântica, porém escala para um número significativamente maior de dispositivos. A arquitetura de

Syanie et al. (2025) reporta latência abaixo de abaixo de 300 ms para até 100 clientes. O MoveUFF opera em patamar semelhante sob carga de 200 msgs/s na Fase 5, correspondendo a uma frota de 1.000 bicicletas. Uma comparação direta é limitada pela diferença de semântica de cliente entre os trabalhos, mas os resultados sugerem que o custo adicional do enfileiramento priorizado e da tradução semântica da IEC 61850 é absorvido pelo autoescalonamento sem penalidade significativa de latência.

5 CONCLUSÃO

Este trabalho detalhou a implementação de um sistema de monitoramento escalável para bicicletas elétricas compartilhadas, baseado em uma arquitetura SCADA Web integrada ao protocolo MMS, conforme a norma IEC 61850. A arquitetura proposta foi implementada em um protótipo funcional de estação inteligente e validada em um ambiente de testes progressivo, sustentando crescimento de 100× da frota (10 a 1.000 bicicletas) com aumento sublinear da latência (47 ms a 218 ms) e taxa de entrega superior a 99,9%, sem descarte de mensagens *Critical* ou *Urgent*.

Os resultados experimentais demonstraram a viabilidade da solução. A interface SCADA Web possibilitou a visualização em tempo real das variáveis operacionais, a comunicação baseada em MMS garantiu a interoperabilidade entre os dispositivos, e o mecanismo de priorização por filas Redis associado ao HPA assegurou comportamento elástico consistente, reforçado pela resiliência demonstrada no teste de falha de *pod*. A análise do comportamento do sistema indica que a arquitetura suporta a expansão para múltiplas estações e dispositivos sem reestruturações significativas, graças à modelagem modular adotada.

Como trabalhos futuros, destacam-se: a adoção de escalonamento automático preditivo para reduzir o tempo de reação do HPA (15–22 s), a extensão da orquestração para nós de borda via KubeEdge Saini et al. (2024), a validação com *hardware* real em escala e em *cluster* distribuído entre múltiplos nós físicos e otimizações de renderização WebGL para frotas superiores a 10.000 unidades.

6 AGRADECIMENTOS

Os autores gostariam de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) e ao apoio fornecido pela Rede Temática RIBIERSE-CYTED (723RT0150).

REFERÊNCIAS

Akshatha, P.S. et al. (2024). Priority-enabled MQTT: a robust approach to emergency event messaging. *Journal of Engineering and Applied Science*, 71, 67.

Anandkumar, R. et al. (2023). Design and implementation of IoT gateway with MQTT and IEC 61850 MMS protocol. *International Journal of Membrane Science and Technology*, 10(1), 330–342.

Corti, F. et al. (2024). A comprehensive review of charging infrastructure for electric micromobility vehicles: Technologies and challenges. *Energy Reports*, 12, 545–567.

Demidov, I. et al. (2023). IEC-61850 performance evaluation in a 5G cellular network: UDP and TCP analysis. In M. Fathi, E. Zio, and P.M. Pardalos (eds.), *Handbook of Smart Energy Systems*, 2729–2761. Springer, Cham.

Doshi, R. et al. (2024). Distributed MQTT broker: A load-balanced Redis-based architecture. In *ESCI*. IEEE.

Dwiyanto, R.A. et al. (2025). Performance analysis of REST API in a real-time IoT-based vehicle monitoring system. *IJRES*, 14(3), 766–784.

Ergun, K. et al. (2023). Dynamic reliability management of multigateway IoT edge computing systems. *Internet of Things Journal*, 10(5), 3864–3889.

Galande, B. et al. (2024). Event-driven real-time autoscaling mechanisms in Kubernetes. *IJSREM*, 8(5).

Gao, Y. et al. (2019). POF: Probability-driven opportunistic forwarding for bike-sharing system. *Access*, 7, 52214–52225.

Hwang, S.H. et al. (2020). IEC 61850 traffic delay measurement using Raspberry Pi, libiec61850 and IEEE 1588. *TEST Engineering & Management*, 83, 12182–12187.

IEC 61850 (2003-2025). IEC 61850 - communication networks and systems in substations. Technical report, IEC.

Ismail, I. (2025). Microservices and IoT architectures: Convergence and application in smart agriculture. *Preprints*.

Lee, S.H. et al. (2025). Iec 61850-based data modeling and prototype implementation for hydrogen refueling stations using libiec61850 and raspberry pi. *Architecture Image Studies*, 6(4), 125–136.

Mercurio, A. et al. (2009). Open-source implementation of monitoring and controlling services for EMS/SCADA systems by means of web services—IEC 61850 and IEC 61970 standards. *Transactions on Power Delivery*, 24(3), 1148–1153.

Pereira, W.T. et al. (2023). A virtualized testbed for IoT: Scalability for swarm application. In *CCNC*, 1074–1079. IEEE.

Pitta, I. et al. (2023). Shared electric vehicle systems based on IEC 61850-90-8: Implementation proposal and proof of concept. In *WCNPS*, 1–7. IEEE.

Rathore, P. et al. (2024). Optimizing micromobility infrastructure: A comprehensive study of docking, locking, and charging solutions in metro cities. *ShodhKosh: Journal of Visual and Performing Arts*, 5(ICoMABE), 106–115.

Saini, L.M. et al. (2024). KubeEdge for scalable IoT applications. In *IC3I*, 899–905. IEEE.

Salimbangon, J. et al. (2025). E-wheelshare: An iot-based bicycle sharing system prototype for university of cebu. *Journal of Computer Science and Technology Studies*, 7(3), 1018–2021.

Syanie, M.H.P. et al. (2025). Rbac-enabled secure data transmission architecture for electric vehicles in the iov context. In *IES*, 467–472. IEEE.

Ustun, T.S. et al. (2019). Implementation of IEC 61850 based integrated EV charging management in smart grids. In *VPPC*. IEEE.

Xue, Z. et al. (2024). A design for an IEC61850 digital simulation environment. In *ICPST*. IEEE.