

Integração entre middleware IoT e aplicação web para auditoria patrimonial com RFID UHF

Eduardo R. Fortes¹, Tiago Antônio Rizzetti¹

¹Colégio Técnico Industrial de Santa Maria (CTISM)
Universidade Federal de Santa Maria (UFSM) - Santa Maria - RS - Brasil

eduardo.fortes@redes.ufsm.br, rizzetti@ctism.ufsm.br

Abstract. *This paper describes the integration between a legacy IoT middleware, extended with UHF RFID reading modules still under physical testing, and a web application for asset auditing in a public educational institution. The application consumes the middleware through a REST API for history and a WebSocket for real-time events, requiring no further change to it. We analyse the relay pattern adopted by the application backend, the separation between service account and per-user identity, the reconnection strategy applied along the chain and the message contracts. The middleware-to-application hop was measured end to end with simulated readers on the production MQTT broker and the institutional asset registry.*

Resumo. *Este artigo descreve a integração entre um middleware IoT legado, expandido com módulos de leitura RFID UHF ainda em fase de testes físicos, e uma aplicação web de auditoria patrimonial em uma instituição pública de ensino. A aplicação consome o middleware por uma API REST para o histórico e por um WebSocket para os eventos em tempo real, sem exigir novas alterações nele. Analisam-se o padrão de retransmissão adotado no backend da aplicação, a separação entre conta de serviço e identidade por pessoa, a estratégia de reconexão aplicada ao longo da cadeia e os contratos de mensagem. O salto entre o middleware e a aplicação foi medido ponta a ponta com leitores simulados no broker MQTT de produção e com o cadastro patrimonial institucional.*

1. Introdução

O controle dos bens patrimoniais em instituições públicas de ensino é uma exigência legal estabelecida pela Lei n.º 4.320/64 [Viecelli and Markoski 2013], mas na prática permanece majoritariamente manual: uma comissão percorre cada ambiente, identifica os bens e registra as informações, em um processo lento e suscetível a falhas. No Colégio Técnico Industrial de Santa Maria (CTISM), etiquetas EPC-Gen2 já estão associadas aos bens e leitores portáteis permitem levantamentos pontuais, mas essa abordagem não atende à auditoria contínua de bens sujeitos a movimentação entre ambientes.

A instituição conta ainda com uma infraestrutura de Internet das Coisas (IoT) em implantação, voltada ao rastreamento de pessoas por beacons *Bluetooth Low Energy* (BLE) e composta por nós ESP32, broker MQTT, plataforma Home Assistant e um middleware em Python responsável pelo controle de acesso baseado em papéis (RBAC), pela persistência e pela exposição de uma API REST [Vieira 2026]. O middleware já opera

em produção, embora o uso da tecnologia em ambiente real ainda esteja em fase de implementação. Trabalho anterior dos autores expandiu essa infraestrutura com módulos de leitura RFID UHF, em placas próprias que integram o leitor R200 ao ESP32, firmware com comunicação MQTT sobre TLS, uma Controladora em C++ e a adaptação do middleware para receber, persistir e redistribuir as leituras. Os módulos já foram construídos e os testes com antenas em ambiente físico estão em andamento.

Ao final dessa expansão, porém, o dado disponível para consumo ainda é cru: o middleware registra a leitura de determinada etiqueta por um leitor em determinado instante, mas não identifica que bem é aquele, nem se a passagem é esperada. Atribuir esse significado cabe a uma camada de aplicação, e construí-la sobre um middleware que já opera em produção e que atende também ao fluxo BLE preexistente impõe quatro desafios técnicos: consumir o histórico e o fluxo em tempo real sem novas alterações no middleware nem replicação do seu modelo de dados; autenticar pessoas sem criar um segundo cadastro de identidades e sem expor a credencial de serviço; preservar a continuidade do fluxo diante de interrupções em qualquer salto da cadeia; e atribuir semântica a uma leitura que chega sem direção de movimento e com um identificador de portal que nenhuma camada anterior valida. Este artigo descreve como cada desafio foi tratado na integração entre esse middleware e a aplicação web de gestão e auditoria patrimonial construída sobre ele, e mede o custo da solução no salto entre os dois.

2. Trabalhos Relacionados

[Dresch Jr. et al. 2008] propuseram um sistema de controle patrimonial automatizado baseado em RFID para ambientes institucionais, concentrado em levantamentos pontuais com leitores portáteis. [Resende et al. 2021] avançaram com um sistema de monitoramento patrimonial em tempo real e de baixo custo, baseado em ESP32 e RFID, sem abordar a integração com infraestruturas IoT preexistentes nem a segurança da comunicação entre a borda e os serviços de retaguarda. [Vieira 2026] estabeleceu a infraestrutura de rastreamento de presença sobre a qual este trabalho se apoia e apontou, como trabalho futuro, a integração com tecnologias de identificação como o RFID. O presente trabalho diferencia-se por tratar explicitamente da fronteira entre a infraestrutura de coleta e a aplicação que a consome, em vez de conceber uma solução isolada.

3. Arquitetura Geral

O sistema é uma cadeia de camadas que converge no middleware, conforme a Figura 1. Na borda, o ESP32 acoplado ao leitor UHF R200, em placa própria, lê etiquetas EPC-Gen2 a cada 200 ms, deduplica localmente em uma janela de 5 s e publica, via MQTT sobre TLS com autenticação mútua por certificados, um *payload* JSON com o código EPC, o identificador do portal, o instante da leitura e o RSSI. A Controladora, em C++, subscreve o broker Mosquitto, aplica um segundo *cooldown* de 5 s por EPC e encaminha a leitura ao middleware por uma conexão WebSocket persistente, com fila de envio para os períodos de indisponibilidade. O middleware, em Python com FastAPI e SQLAlchemy, persiste a leitura, republica o evento em um tópico MQTT de saída e o retransmite a todos os clientes WebSocket conectados.

A aplicação web situa-se abaixo dessa fronteira. Seu backend, em Node.js com Express e o ORM Prisma, é o único componente que se comunica com o middleware; o

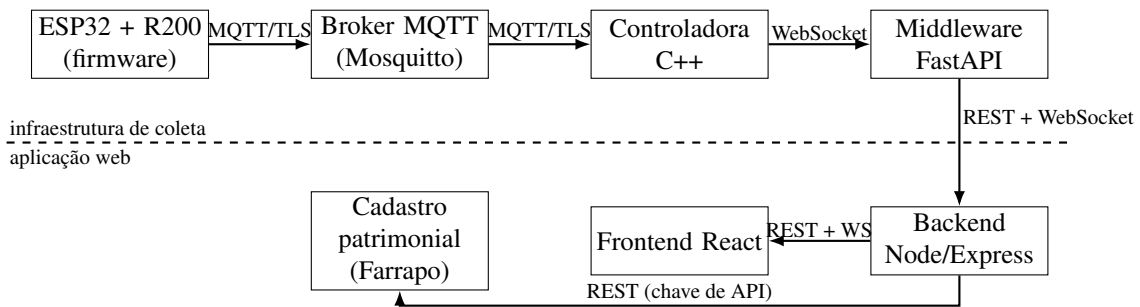


Figura 1. Cadeia de camadas; a linha tracejada marca a fronteira entre coleta e aplicação.

frontend, desenvolvido em React, comunica-se exclusivamente com o backend. A aplicação mantém banco próprio, com a auditoria das leituras, os perfis dos portais, as liberações de saída e os alertas de divergência, e obtém a descrição de cada bem do cadastro patrimonial institucional (Farrapo), por uma API REST autenticada por chave.

4. Comunicação entre o Middleware e a Aplicação

4.1. Canais de consumo e contratos de mensagem

O middleware expõe dois canais complementares. O histórico paginado, com filtros por EPC, portal e intervalo de datas, vem do *endpoint* REST `GET /rfid/leituras`, que exige a permissão `rfid:read` do sistema RBAC. Pelo *endpoint* WebSocket `/rfid/ws`, a Controladora publica as leituras e qualquer cliente conectado as recebe em tempo real por *broadcast*. A aplicação usa o REST para carregar o histórico ao abrir uma tela e o WebSocket para o fluxo contínuo de eventos, sem consultas repetidas [Fette and Melnikov 2011].

A coexistência de MQTT e WebSocket atende a requisitos distintos e não é redundante. O MQTT serve à borda: opera sobre TLS com autenticação mútua por certificado, desacopla produtores de consumidores e permite entregar a mesma leitura a mais de um assinante, o que o middleware explora ao republicar os eventos em um tópico de saída. O WebSocket serve ao consumo pela aplicação e pelo navegador, que não interpreta MQTT nativamente; expor o broker à camada web exigiria distribuir credenciais e contornar o middleware, onde residem o controle de acesso e a persistência. Apenas MQTT deixaria o navegador sem canal de tempo real; apenas WebSocket exigiria uma conexão persistente por dispositivo de borda, sem broker intermediário e sem entrega a múltiplos consumidores. A escolha é também, em parte, herdada: o middleware é preexistente e define essas duas interfaces.

A mensagem que atravessa a cadeia é um objeto JSON com os campos `tipo`, `epc`, `portal`, `timestamp` e `rsssi`. O campo `tipo`, com valor `leitura_rfid`, permite que cada consumidor ignore mensagens que não lhe dizem respeito. A Controladora, que subscreve todos os tópicos do broker, descarta as republicações do próprio middleware por um *guard* simples: o `timestamp` do ESP32 é numérico, enquanto o republicado está em ISO 8601. Ao retransmitir ao navegador, o backend preserva os campos originais e acrescenta `descricao`, `numPatrimonio` e `alerta`.

4.2. Retransmissão e enriquecimento

O backend da aplicação mantém uma única conexão WebSocket com o middleware, em um módulo de retransmissão (*relay*), e expõe um WebSocket próprio aos navegadores. Cada evento recebido percorre um caminho único: a leitura é enriquecida com a descrição e o número de patrimônio do bem, avaliada pelas regras de divergência, gravada na auditoria e só então retransmitida a todos os navegadores, já com o alerta que eventualmente gerou.

Três razões justificam o navegador nunca se conectar diretamente ao middleware: a credencial de leitura não é exposta ao cliente; o enriquecimento e as regras rodam uma vez por leitura, e não uma vez por navegador aberto; e o frontend recebe um evento que já carrega significado, mantendo a lógica de negócio fora da interface.

4.3. Identidade e autenticação

A aplicação mantém duas integrações distintas com a autenticação do middleware. A conta de serviço é uma credencial fixa do backend, com a permissão `rfid:read`; seu *token* é obtido em `POST /token`, mantido em cache e renovado em resposta 401. A identidade por pessoa é validada a cada login: a aplicação submete a credencial do usuário ao mesmo `POST /token`, lê o papel em `GET /users/me` e emite um *token* JWT próprio, válido por 8 horas, que governa a sessão; o *token* do middleware não é reutilizado. Nenhuma das duas rotas de identificação exige permissão no RBAC, o que permitiu adotar o middleware como provedor de identidade sem solicitar alterações nele. Como o papel é relido a cada entrada, um rebaixamento no middleware se reflete no login seguinte, e a aplicação não armazena a senha de quem se autentica pelo middleware. As duas integrações não compartilham código nem cache, para que uma pessoa nunca herde o acesso da conta de serviço. Com o middleware indisponível, a autenticação falha de forma fechada: contas locais de privilégio mínimo continuam entrando, mas contas institucionais são recusadas.

4.4. Resiliência

A mesma estratégia de reconexão é aplicada em cada salto da cadeia: a Controladora tenta restabelecer a conexão com o middleware a cada 3 s, o *relay* do backend faz o mesmo com o middleware e o navegador faz o mesmo com o backend. As chamadas HTTP ao middleware e ao cadastro patrimonial têm tempo limite de 5 s, para que um serviço no ar mas sem responder não deixe uma requisição pendurada. O backend anuncia aos navegadores, por uma mensagem `estado_middleware`, cada mudança na sua própria conexão com o middleware, e o estado atual quando um navegador se conecta: o indicador da tela reflete o estado do *relay*, e um middleware fora do ar é percebido pelo operador mesmo com o backend em funcionamento.

4.5. Semântica acrescentada pela aplicação

A aplicação supre as lacunas semânticas apontadas na introdução. Os portais são descobertos a partir dos identificadores presentes nas leituras e recebem um perfil com nome, posição no mapa e tipo, interno ou saída, que é o que torna a leitura interpretável como movimento. A saída legítima é representada por uma liberação explícita, com autor, motivo e validade, e duas regras de divergência, leitor desconhecido e saída não autorizada, são avaliadas na chegada de cada leitura, sempre pelo mesmo caminho de processamento.

5. Validação e Resultados

Validação funcional. Como a instalação dos portais ainda se encontra em fase de testes físicos, a cadeia foi validada em ambiente de software, com o ESP32 substituído por um simulador que publica no broker MQTT de produção, com TLS mútuo, o mesmo *payload* do firmware. O que é simulado é a passagem pelo leitor; as etiquetas e os bens são reais, correspondendo a 19 itens do patrimônio do CTISM. Os cenários funcionais foram confirmados: leitura em portal interno e leitura em portal de saída com liberação vigente, ambas sem alerta; a mesma leitura após o cancelamento da liberação, com alerta crítico; e leitura de leitor não cadastrado, com alerta de leitor desconhecido. Na consulta ao cadastro patrimonial, 18 dos 19 EPCs retornaram descrição e número de registro coincidentes com os dados históricos, e um retornou não encontrado, por ter sido desvinculado do cadastro.

Tabela 1. Medições do salto entre o middleware e a aplicação, em três sessões

Medida	Condição	n	Mediana	p25–p75	p95
Latência do salto (ms)	leitura isolada	54	10,9	10,1–17,9	22,4
Entrega ao navegador (ms)	1 cliente	81	8,7	8,2–9,5	11,3
	5 clientes	405	9,0	8,1–9,8	11,5
	10 clientes	810	9,3	8,6–10,5	13,4
	20 clientes	1.620	10,8	9,8–11,9	13,3
	50 clientes	4.050	12,1	11,5–13,1	14,8

Avaliação de desempenho. As medições foram feitas em três sessões independentes, com todos os componentes em um mesmo servidor (Ubuntu 24.04, 4 vCPU, 3 GB de RAM) e o cadastro patrimonial real, com a ordem das configurações sorteada em cada sessão e as três primeiras leituras de cada uma descartadas como aquecimento. A Tabela 1 resume os resultados. Não houve perda: as 6.966 entregas esperadas foram observadas, e as trinta leituras publicadas durante quedas provocadas do middleware chegaram à aplicação, retidas na fila de envio da Controladora; em dez repetições, o *relay* detectou as quedas em 0,02 s e reconectou em 2,0 s de mediana.

A latência do salto é o intervalo entre a gravação da leitura no middleware e a gravação na auditoria da aplicação, e cobre o enriquecimento e a avaliação das regras. Com as leituras chegando isoladas, a mediana é de 10,9 ms. Quando a camada anterior as entrega em rajadas, o backend as processa em série e a latência cresce com a posição na fila, chegando a cerca de 320 ms na última leitura de uma rajada. A primeira leitura de cada sessão, que paga também o estabelecimento da conexão com o cadastro patrimonial, custou 393, 492 e 402 ms; é esse custo que o cache evita nas leituras seguintes. Na consulta de histórico com 50 registros, o enriquecimento acrescenta 3,4 ms de mediana sobre a consulta crua ao middleware, de 5,9 para 9,3 ms em 60 chamadas de cada.

O custo do fan-out é pequeno e mensurável. A mediana da entrega sobe de 8,7 ms com um cliente para 12,1 ms com cinquenta, e a dispersão entre os clientes de uma mesma leitura cresce de 0,1 ms com cinco clientes para 2,7 ms com cinquenta, com o último a conectar recebendo cerca de 2 ms depois do primeiro, o que é compatível com um envio em laço sobre a lista de conexões. Uma medição anterior, de uma sessão única e dez leituras por configuração, sugerira o contrário, com atraso maior para cinco clientes do que para vinte; com três sessões e oitenta e uma leituras por configuração esse comportamento não

se reproduziu. O backend conta ainda com 49 testes automatizados, entre eles um teste estrutural que impede consultas a dados patrimoniais fora do ponto único de filtragem.

Limitações. Os dois conjuntos de resultados têm alcances diferentes. A validação funcional demonstra que a cadeia opera ponta a ponta e que as regras de divergência respondem como especificado, com etiquetas e bens reais. A avaliação de desempenho caracteriza apenas o salto entre o middleware e a aplicação, com os componentes em um mesmo servidor e a borda emulada por um simulador, de modo que os números dizem respeito ao custo do software de integração e não ao desempenho em campo: taxa de detecção, alcance de leitura e efeitos de interferência dependem dos portais instalados e permanecem em aberto. A cadeia foi desde então distribuída em máquinas virtuais distintas, uma para o middleware e outra para a aplicação, o que passa a impor pela rede a fronteira entre as camadas; repetir as medições nessa topologia é o passo seguinte.

6. Conclusão e Trabalhos Futuros

Este artigo descreveu como uma aplicação web de auditoria patrimonial se integra a um middleware IoT legado pelas interfaces que ele já expõe, sem exigir novas alterações nele, por canais REST e WebSocket, um padrão de retransmissão que concentra o enriquecimento e as regras no backend, uma separação estrita entre conta de serviço e identidade por pessoa e uma estratégia de resiliência repetida em cada salto. As medições confirmaram que o dado cru da infraestrutura ganha significado na camada de aplicação a um custo de dezenas de milissegundos, sem acoplamento além dos contratos de API.

Como trabalhos futuros destacam-se a conclusão dos testes físicos dos módulos de leitura e a instalação de um piloto em uma porta do CTISM, com coleta de taxa de detecção e latência; a repetição das medições na topologia distribuída já implantada; a ativação do filtro de dados por nível de acesso, já preparado no backend e agora alimentado pelo grau de criticidade que o cadastro patrimonial passou a expor; e a substituição da chave estática do WebSocket da Controladora por autenticação JWT. O código-fonte não é público por conter informações sensíveis da instituição.

Referências

- Dresch Jr., A., Efrom, D. R., and Grumovski, D. (2008). Sistema de controle de patrimônio via RFID. *E-Tech: Atualidades Tecnológicas para Competitividade Industrial*, 1(1):47–57.
- Fette, I. and Melnikov, A. (2011). The WebSocket protocol. RFC 6455, Internet Engineering Task Force.
- Resende, W. J. F. et al. (2021). Desenvolvimento de um sistema para monitoramento e controle patrimonial, utilizando RFID e dispositivos IoT. *Brazilian Journal of Development*, 7(7):67357–67368.
- Viecelli, M. E. and Markoski, A. (2013). A importância do controle patrimonial para as entidades públicas: um estudo de caso no centro de educação superior do norte do rio grande do sul (cesnors). *Revista de Administração*, 11(20):9–28.
- Vieira, E. d. F. (2026). Um sistema para rastreamento e adaptação de recursos em ambientes de educação profissional e tecnológica (EPT). Master's thesis, Universidade Federal de Santa Maria, Santa Maria, RS.