User Interface for Cyber-Physical System - UICPS

Rafael Ifanger Ribeiro[1]

rafael.ifanger.ribeiro@gmail.com

riribeiro@cti.gov.br

Antonio Pestana Neto[1]

antonio.pestana@cti.gov.br

[1]**Divisão de Sistemas Ciber Físicos (DISCF)**

Resumo: O presente documento tem por objetivo apresentar a arquitetura da interface de usuário para o sistema ciber físico do projeto Energia Inteligente. A interface está dividida em três partes principais - Scheduler, Websocket e página web - e possui integração com os módulos funcionais de telemetria.

Introdução

O desenvolvimento de sistemas ciberfísicos voltados ao monitoramento e análise da qualidade de energia requer uma integração consistente entre camadas de aquisição de dados, processamento e transmissão. A confiabilidade dos resultados obtidos depende tanto da precisão dos módulos de medição quanto da eficiência do fluxo de dados entre os componentes da plataforma. Nesse contexto, torna-se essencial projetar uma arquitetura capaz de gerenciar, processar e disponibilizar informações de forma segura, escalável e em tempo real, seguindo arquiteturas distribuídas e sistemas ciberfísicos [2] .

A plataforma proposta neste trabalho é composta por três módulos principais: a API (*Application Programming Interface*) REST (*REpresentational State Transfer*) desenvolvida em Flask, responsável por gerenciar autenticação, configuração e controle de dispositivos; o canal de comunicação em tempo real via WebSocket (Flask-SocketIO), dedicado à transmissão contínua de dados provenientes dos sensores; e o sistema de execução assíncrona com Celery e Redis (Scheduler), empregado no agendamento e processamento de tarefas periódicas. Essa divisão permite isolar responsabilidades, melhorar a manutenção do código e otimizar o uso de recursos computacionais, em arquitetura de microserviços [3], garantindo que as diferentes funcionalidades do sistema operem de forma coordenada.

Arquitetura da Plataforma

A arquitetura do serviço implementado em Flask tem papel central na integração entre a plataforma de monitoramento de sensores e o ThingsBoard, que atua como sistema de gerenciamento e também como broker de dados. O Flask foi utilizado para desenvolver uma API REST responsável tanto pela configuração dos sensores e usuários quanto pela disponibilização de rotas de integração externa. Essa API funciona como camada intermediária, abstraindo a complexidade da comunicação direta com o ThingsBoard, e permitindo que aplicações clientes interajam com a plataforma de forma padronizada e desacoplada.

O ThingsBoard, por sua vez, exerce dois papéis: além de fornecer mecanismos de visualização e organização de telemetria, ele opera como broker de mensagens, suportando protocolos como MQTT, HTTP e CoAP. Isso possibilita que os dados provenientes dos sensores sejam transmitidos em tempo real e armazenados de forma estruturada, com suporte a regras de processamento e notificações. Na prática, a API do Flask se comunica com a API do ThingsBoard para criar, atualizar e manipular dispositivos, ativos e suas relações, bem como enviar telemetria no formato JSON. Essa integração garante que o usuário final acesse dados validados e organizados sem precisar lidar diretamente com a complexidade do broker, assegurando escalabilidade e interoperabilidade com diferentes módulos do sistema.

Com isso, a camada web construída em Flask não apenas fornece uma interface de controle e autenticação (via JWT), mas também garante confiabilidade no fluxo de dados entre sensores, banco de dados local (PostgreSQL) e ThingsBoard, servindo como ponto de orquestração para a plataforma.

A arquitetura de comunicação do serviço de Websocket da plataforma foi projetada para suportar tanto transmissões contínuas de dados em tempo real quanto consultas históricas otimizadas, garantindo flexibilidade na análise e na interação com os sensores. Para isso, emprega-se um canal bidirecional baseado no protocolo WebSocket [4] (implementado com Flask-SocketIO), que estabelece uma conexão persistente sobre TCP para transporte confiável das informações. No entanto, considerando a necessidade de baixa latência na aquisição de dados críticos, o sistema também integra, em determinados fluxos, a utilização de transporte orientado a mensagens leves via UDP. Embora o UDP não possua mecanismos nativos de confiabilidade ou ordenação de pacotes, sua menor sobrecarga o torna adequado para transmissões em que a entrega imediata prevalece sobre a confirmação de recebimento, especialmente em cenários de monitoramento contínuo.

No contexto do WebSocket, o sistema mantém conexões ativas com autenticação baseada em JWT (JSON Web Token - consiste em um token textual dividido em três partes: header (cabeçalho); payload (conteúdo); e signature (assinatura), codificadas em Base64URL e enviada na requisição do cliente de modo que o servidor consiga validar se o acesso é válido). O usuário se autentica com credenciais válidas, o servidor gera e retorna o JWT, e em cada requisição subsequente o cliente envia esse token no cabeçalho HTTP, possibilitando a validação segura de cada sessão e o gerenciamento de permissões de acesso.

A transmissão de dados via WebSocket permite que as leituras dos sensores sejam entregues ao cliente praticamente no instante em que são capturadas, viabilizando o streaming em tempo real de variáveis elétricas e ambientais. Além disso, essa mesma aplicação expõe endpoints REST para operações que não demandam comunicação contínua, como a exportação em lote (batch) de dados. Essa funcionalidade permite que o usuário selecione períodos e variáveis específicas para extração, retornando um conjunto formatado em extensão .csv para análise posterior ou integração com outras ferramentas.

A coexistência dessas duas abordagens - streaming contínuo via WebSocket e exportação batch via REST - confere à plataforma a capacidade de atender tanto requisitos operacionais imediatos quanto demandas analíticas históricas. Essa integração é essencial para aplicações de qualidade de energia e condicionamento ambiental, nas quais é necessário observar eventos transientes em tempo real e, ao mesmo tempo, dispor de registros consolidados para estudos comparativos e diagnósticos.

O diagrama da Figura 1 descreve o fluxo de autenticação e manutenção da sessão no WebSocket integrado à API REST. O processo começa com o usuário enviando credenciais (user:password) para o endpoint /token por meio de Basic Auth, recebendo em resposta um token de acesso e um token de atualização (refresh). Com isso, é possível iniciar a conexão WebSocket, a partir da qual são feitas as requisições de data streaming, que ativam o fluxo contínuo de telemetria dos sensores.

Durante todo o ciclo, o sistema valida do token de acesso: caso ainda esteja válido, o fluxo segue normalmente; se expirar, entram em ação os mecanismos de renovação. Essa atualização pode ocorrer tanto por requisição HTTP ao endpoint de refresh quanto por um evento interno no próprio WebSocket (WS Atualizar_Token). Dessa forma, a sessão de dados em tempo real não é interrompida, pois sempre que um token expira, ele é renovado antes de comprometer a continuidade da comunicação. O sistema garante segurança (via tokens JWT) e confiabilidade (renovação automática) em uma arquitetura que combina API REST para controle e exportação e WebSocket para transmissão contínua e bidirecional dos dados.

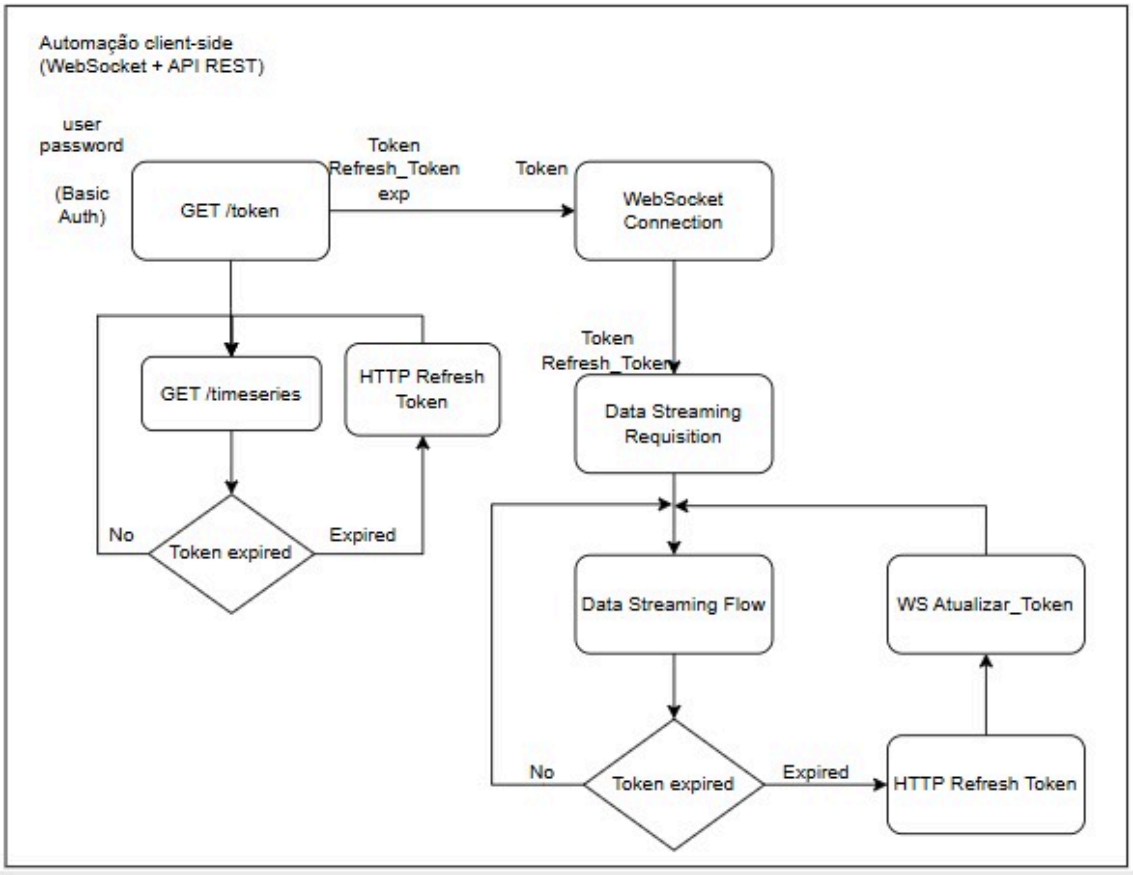

Figura 1 – Fluxo de autenticação e manutenção da sessão WebSocket integrado à API REST, destacando os processos de obtenção, validação e renovação de tokens para garantir transmissão contínua e segura de dados.

O fluxo de extração e análise foi automatizado para fechar o ciclo de controle do ar-condicionado, de modo que assim se obtenha a eficiência energética. Os módulos [1] enviam os dados para o broker ao qual salva os dados no banco e a plataforma com controle de acesso disponibiliza o serviço de requisição em streaming ou batch para que alguma thread instanciada no serviço de Scheduler possa adquirir os dados; analisar por algoritmo; e retornar por controle RPC os novos estados do ar condicionado (caminho ao ponto de menor custo energético).

Além do controle, tem-se também a possibilidade de instanciar uma *thread* utilizando o serviço de Scheduler criado com o Celery para registro de análise periódica, seja para relatório da qualidade do sistema ou alguma necessidade de alarme. O Celery é um framework sistema distribuído de gerenciamento de filas de tarefas assíncronas, projetado para delegar a execução de processos demorados ou recorrentes a *workers* independentes, garantindo que a aplicação principal não seja bloqueada. Ele utiliza um broker de mensagens - no caso deste trabalho, o Redis - para intermediar a comunicação entre a aplicação que envia as tarefas e os *workers* que as executam.

Além disso, o Celery possui o Celery Beat, responsável por agendar tarefas periódicas, como notificações, alarmes ou coletas programadas de dados. Sua arquitetura permite paralelismo e escalabilidade, pois múltiplos *workers* podem processar tarefas simultaneamente, garantindo eficiência mesmo sob alta carga.

Sendo assim, o conjunto como um todo desenvolvido até agora a possibilidade dos novos integrantes do projeto enfocarem diretamente em controle e análise sem preocupação de construção de arquitetura de serviço para adquirir e fornecer medições e gerência dos dispositivos (e seus parâmetros) presentes no projeto.

A Figura 2 apresenta a arquitetura geral da plataforma desenvolvida (UICPS), destacando os principais módulos de interface, comunicação e integração com serviços de armazenamento e processamento de dados

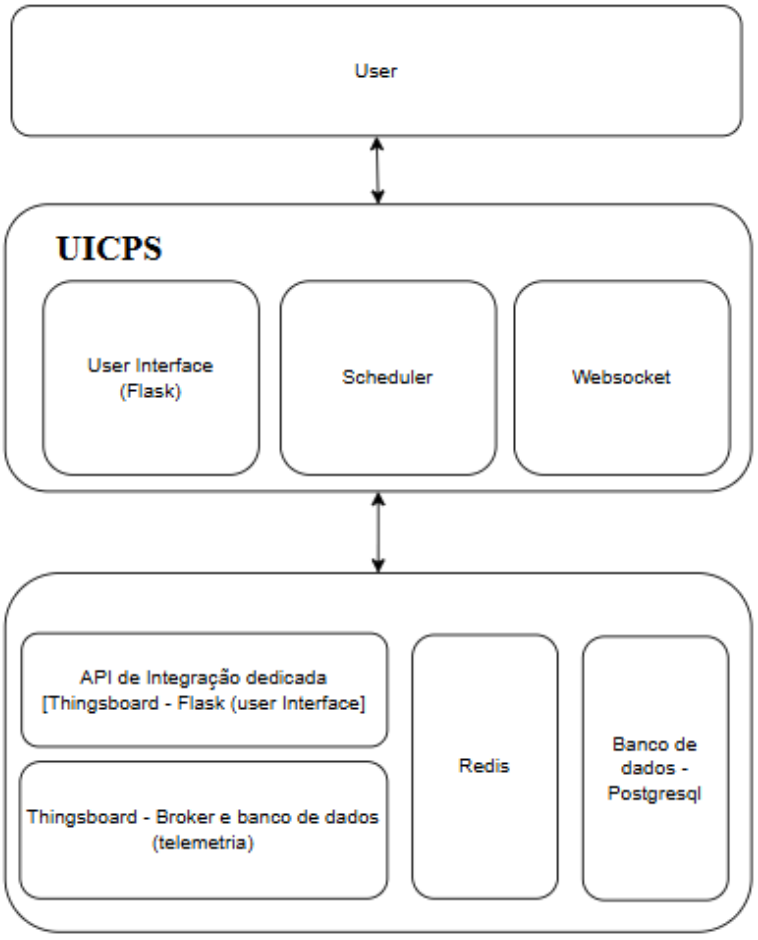

Figura 2 – Arquitetura geral da plataforma UICPS e seus principais componentes.

Resultado

Os resultados obtidos a partir da implementação da plataforma desenvolvida confirmam a viabilidade e eficiência da arquitetura proposta, tanto no que diz respeito ao fluxo contínuo de aquisição de dados quanto às funcionalidades de exportação histórica e controle automatizado. O uso combinado de API REST e WebSocket mostrou-se fundamental para atender diferentes perfis de utilização: enquanto o WebSocket assegurou transmissão em tempo real de leituras críticas dos sensores, a API REST permitiu consultas e exportações em lote, ampliando as possibilidades de análise e integração com outras aplicações. A adoção de autenticação via JWT, aliada ao mecanismo de renovação de tokens descrito na Figura 1.

O emprego do Celery, em conjunto com o Redis, demonstrou eficiência no gerenciamento de tarefas assíncronas, permitindo o agendamento de rotinas periódicas e o disparo de processos de análise ou controle sem comprometer a responsividade da aplicação principal. Isso possibilitou, por exemplo, que módulos de controle energético sejam instanciados diretamente pelo Scheduler, permitindo decisões automatizadas - como o ajuste dinâmico de sistemas de climatização - tomadas com base nos dados recebidos em tempo real e nas condições registradas em histórico.

A integração com o ThingsBoard consolidou a arquitetura como uma solução escalável e interoperável. Atuando simultaneamente como broker de mensagens e plataforma de armazenamento e visualização de telemetria, o ThingsBoard viabilizou a organização dos dados em estruturas padronizadas, facilitando tanto a análise contínua quanto a elaboração de relatórios periódicos. Essa integração assegurou que os usuários finais pudessem acessar informações confiáveis sem a necessidade de interagir diretamente com camadas técnicas de baixo nível.

De forma geral, a arquitetura proposta mostrou-se robusta para atender aos requisitos de aplicações em sistemas ciberfísicos de qualidade de energia e ambiente. As Figuras 1 e 2 sintetizam os principais aspectos da solução, desde o fluxo de autenticação e manutenção da sessão WebSocket até a visão integrada dos módulos que compõem a plataforma UICPS. Os resultados apontam que a solução permite não apenas o monitoramento eficiente das variáveis medidas, mas também a criação de rotinas automatizadas de análise e controle, potencializando a utilização dos dados para fins de eficiência energética, confiabilidade operacional e suporte a pesquisas futuras.

Embora os resultados apresentados confirmem o funcionamento da arquitetura desenvolvida, alguns pontos de atenção e limitações foram identificados, servindo como base para futuros aprimoramentos. Um dos aspectos críticos está relacionado à latência e confiabilidade no uso combinado de WebSocket e UDP. Apesar de o UDP oferecer baixa sobrecarga e maior velocidade na transmissão, sua ausência de mecanismos nativos de confirmação pode comprometer a entrega em cenários de alta instabilidade de rede. Assim, embora o protocolo seja adequado para variáveis que exigem baixa latência, a necessidade de estratégias de verificação ou redundância deve ser considerada para aplicações que demandam confiabilidade absoluta.

Outro ponto diz respeito à escalabilidade da solução. A utilização de Celery e Redis garantiu paralelismo e distribuição de tarefas de forma eficiente nos testes realizados, mas em cenários de grande volume de sensores ou alta frequência de aquisição, é possível que surjam

gargalos tanto na camada de armazenamento (PostgreSQL e Redis) quanto na orquestração de *workers*. Nesse sentido, torna-se relevante avaliar o emprego de balanceadores de carga, partição de filas de tarefas e até integração com *brokers* mais robustos, como RabbitMQ ou Kafka, dependendo da criticidade do cenário.

A segurança foi assegurada com o uso de JWT e renovação de tokens, porém, em aplicações de larga escala e em ambientes críticos, faz-se necessário considerar mecanismos complementares, como autenticação multifatorial, criptografia ponta a ponta nos canais WebSocket e monitoramento ativo de anomalias. Além disso, a dependência do ThingsBoard como broker e plataforma de visualização introduz um ponto centralizado que, embora facilite a integração, também pode se tornar um gargalo de desempenho ou um ponto único de falha caso não haja redundância.

Por fim, no que se refere à aplicabilidade futura, a arquitetura proposta abre caminho para a incorporação de algoritmos de aprendizado de máquina e análise preditiva, que podem utilizar tanto os dados em tempo real quanto os registros históricos para antecipar falhas, identificar padrões de consumo e propor estratégias de otimização energética mais sofisticadas. Esse potencial amplia a relevância da plataforma não apenas como um sistema de monitoramento, mas como uma ferramenta de suporte à decisão em tempo real.

Os resultados alcançados demonstram que a solução é viável e funcional, mas também evidenciam oportunidades claras de evolução, especialmente no que tange à escalabilidade. Essas perspectivas constituem os próximos passos do trabalho, consolidando a plataforma como suporte para estudos e aplicações em sistemas ciberfísicos de monitoramento e controle.

Conclusão

O desenvolvimento da plataforma proposta demonstrou a viabilidade de uma solução distribuída, segura e escalável para o monitoramento e análise da qualidade de energia, integrando diferentes camadas de aquisição, processamento e transmissão de dados em uma arquitetura modular. A combinação entre API REST (Flask), canal de comunicação em tempo real via WebSocket e sistema de agendamento de tarefas assíncronas com Celery e Redis permitiu não apenas a entrega contínua e confiável de informações, mas também a criação de rotinas automatizadas de análise e controle, fundamentais para aplicações em sistemas ciberfísicos.

A integração com o ThingsBoard, que atua simultaneamente como broker de mensagens e plataforma de armazenamento e visualização de telemetria, assegurou interoperabilidade e organização dos dados, reduzindo a complexidade para o usuário final e garantindo maior flexibilidade na expansão da solução. Além disso, o uso de mecanismos de autenticação e renovação de tokens JWT reforçou a segurança do sistema.

Os resultados obtidos evidenciam que a plataforma cumpre os objetivos propostos, oferecendo uma infraestrutura confiável e adaptável para aplicações que demandam tanto o acompanhamento em tempo real quanto a análise histórica de dados. Contudo, o estudo também destacou pontos de melhoria relacionados à escalabilidade, à resiliência frente a falhas e à possibilidade de incorporar técnicas de aprendizado de máquina para aprimorar o processo de tomada de decisão.

A arquitetura desenvolvida, por conseguinte, representa um avanço significativo para projetos de monitoramento e controle da qualidade de energia, constituindo-se como uma

base para evoluções futuras e aplicações em larga escala, especialmente em contextos que exigem eficiência energética. Também, no projeto, a plataforma apresenta um avanço no fluxo de dados, encerrando a questão de aquisição e busca (broker e endpoints de Websocket e upload de dados em lote). Em consoante, é entregue uma estrutura pronta para o trabalho de análise de dados, havendo apenas o serviço para os novos bolsistas de tratamento e estatísticas, bem como ênfase no controle dos atuadores para eficiência energética, finalizando, então, a proposta total do Condicionamento Ambiental e Eficiência Energética.

Referências

[1] RIBEIRO, Rafael. Sensoriamento para controle de gasto mínimo energético no sistema ciber físico de eficiência energética e condicionamento ambiental. XXIV Jornada de Iniciação Científica do Centro de Tecnologia da Informação Renato Archer - JICC´2024, [*s. l.*], 2024.

[2] NEWMAN, Sam. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 2015.

[3] FOWLER, Martin; LEWIS, James. Microservices: a definition of this new architectural term. 2014.

[4] FETTE, Ian; MELNIKOV, Alexey. The WebSocket Protocol. RFC 6455, IETF, 2011.