XII CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 19, 20 e 21 de maio de 2026
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

Desenvolvimento de plataforma web conteinerizada para criação de UDFBs no padrão O-PAS

Caio Caetano Carnier[1] - ITI/IFSP[1]
Prof. Dr. Eduardo André Mossin - IFSP[2]
Prof. Dr. Rodrigo Palucci Pantoni - IFSP[3]

Introdução

A automação industrial moderna exige ferramentas que combinem flexibilidade e interoperabilidade. Em resposta a essa demanda, o padrão O-PAS™ (*Open Process Automation Standard*) propõe uma arquitetura aberta, segura e modular (Qamsane et al., 2022). Central nessa estrutura estão os Blocos Funcionais Definidos pelo Usuário (*User Defined Function Blocks* - UDFBs), que encapsulam lógicas de controle personalizadas através de arquivos padronizados, como o *NodeSet* e o *SourceCode AddData* (*The Open Group*, 2023). Para desenvolver esses blocos, foi criada uma ferramenta denominada FBBuilder Desktop (Pantoni et al., 2024). A ferramenta recebe como entrada um texto estruturado em IEC 61131-3 (*Structured Text*), que já contém a lógica do bloco funcional em desenvolvimento. A partir disso, ela oferece mecanismos para transformar essa lógica em uma aplicação executável no padrão O-PAS. Na prática, o usuário define e organiza parâmetros (entradas, saídas e configurações), e a ferramenta faz o mapeamento automático desses elementos para uma estrutura compatível com OPC UA, permitindo que o bloco funcione dentro de uma arquitetura moderna de automação. Com isso, ela reduz o trabalho manual, garante padronização e facilita a integração da lógica desenvolvida com sistemas industriais reais. Embora funcional, essa versão do FBBuilder operava de forma centralizada, apresentando forte dependência do sistema operacional local, o que dificultava a manutenção, a escalabilidade e a colaboração entre múltiplos usuários. A dependência de bibliotecas instaladas na máquina limitava a portabilidade da solução e impunha barreiras à integração com fluxos de trabalho modernos de engenharia de *software* (Bernstein, 2014). Nesse contexto, a evolução para uma plataforma web, desenvolvida em C# com o framework .NET, representa um avanço significativo ao alinhar a ferramenta aos paradigmas atuais de aplicações distribuídas. A nova arquitetura desacopla a interface de usuário, o banco de dados MySQL e os serviços de engenharia, sendo todos executados em contêineres independentes, com o *Nginx* atuando como *gateway* de comunicação. Essa abordagem não apenas elimina dependências do sistema operacional, mas também favorece a escalabilidade, a portabilidade e a integração com ambientes modernos de desenvolvimento e implantação.

[1]Estudante do Curso de Engenharia Elétrica, IFSP - Sertãozinho. caio.carnier@aluno.ifsp.edu.br. Bolsa Captada Externamente (BCE) - APPDI - PD&I Nº 07/2024 - IFSP e Nova Smar S.A. https://orcid.org/0009-0003-7166-9860
[2] Doutor. IFSP - Sertãozinho. emossin@ifsp.edu.br. https://orcid.org/0000-0001-5144-518X
[3] Doutor. IFSP - Sertãozinho. rpantoni@ifsp.edu.br. https://orcid.org/0000-0001-8644-71181

XII CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 19, 20 e 21 de maio de 2026
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

Objetivo

Desenvolver e avaliar a NovaFBB, migrando a ferramenta *desktop* legada para uma arquitetura *web* distribuída no ecossistema .NET, visando a criação escalável de UDFBs no padrão O-PAS. Para isso, definiram-se as seguintes etapas metodológicas:

- Arquitetura de *Backend*: Projetar o *backend* em *ASP.NET* Core utilizando arquitetura em camadas (*Controllers*, *Services* e *Repositories*) para reaproveitar e modularizar as lógicas do código legado.
- Comunicação via *APIs*: Desenvolver *APIs REST* para garantir uma comunicação assíncrona e desacoplada entre *frontend* e *backend*.
- Interface Interativa: Construir o *frontend* com *Razor*, *HTML*, *CSS* e *JavaScript*, permitindo a manipulação fluida de dados e estruturas complexas pelo usuário.
- Persistência de Dados: Implementar o acesso a dados de forma performática utilizando o *micro-ORM Dapper* integrado ao banco de dados *MySQL*.
- Processamento Dinâmico: Integrar a leitura e o processamento de arquivos JSON em memória para renderização dinâmica das estruturas de configuração na interface visual.
- Conteinerização da Solução: Após o desenvolvimento, aplicar a conteinerização de toda a infraestrutura (aplicação *web*, banco de dados *MySQL* e *proxy Nginx*) utilizando *Docker*, garantindo isolamento, portabilidade e escalabilidade.
- Validação: Avaliar a arquitetura proposta em termos de desempenho de requisições, eficiência da distribuição em contêineres e facilidade de manutenção.

Metodologia

O desenvolvimento da ferramenta web, denominada NovaFBB, foi conduzido com base na migração de uma aplicação desktop para uma arquitetura web distribuída no ecossistema .NET, seguindo princípios de engenharia de software e separação de responsabilidades. A lógica de criação dos *UDFBs* segue os fundamentos estabelecidos em Spagiari, Mossin e Pantoni (2025), sendo neste trabalho abstraída pela plataforma desenvolvida. O processo de desenvolvimento da ferramenta foi estruturado nas seguintes etapas:

(1) Definição da Arquitetura de *Backend*: O *backend* foi desenvolvido em *ASP.NET Core* (C#), adotando uma arquitetura em camadas baseada em *Controllers, Services e Repositories*. Essa organização permitiu modularizar as regras de negócio, facilitar a manutenção e reaproveitar componentes do sistema legado.

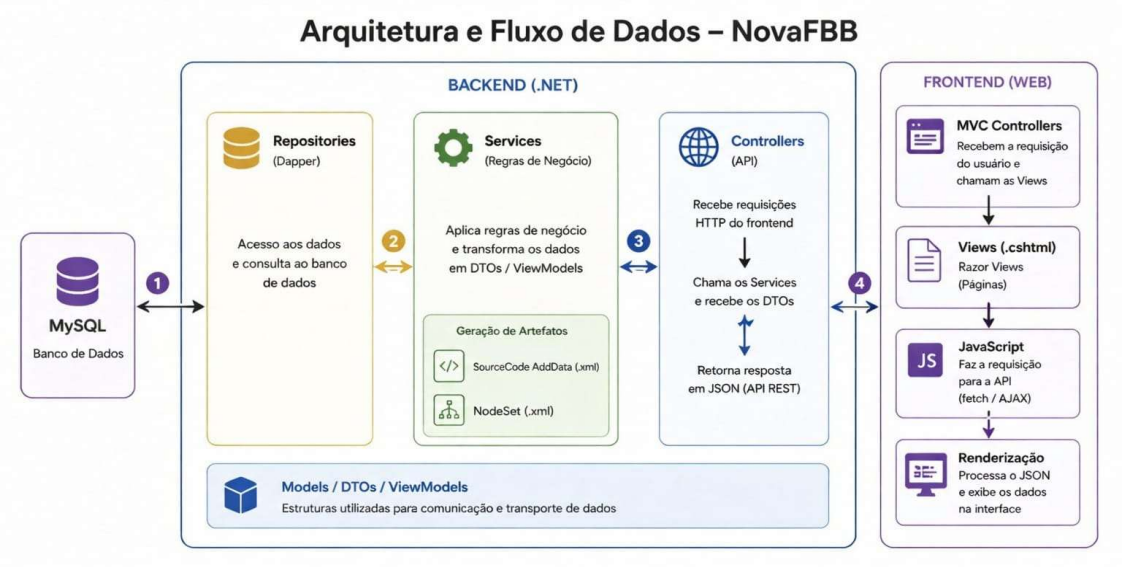
(2) Implementação de *APIs REST*: Foram desenvolvidas *APIs REST* para viabilizar a comunicação entre *frontend* e *backend* de forma assíncrona e desacoplada, permitindo maior escalabilidade e flexibilidade na integração entre os componentes do sistema.

(3) Desenvolvimento da Interface Web: O frontend foi construído utilizando *Razor*, *HTML*, *CSS* e *JavaScript*, possibilitando a criação de uma interface interativa e responsiva. Essa abordagem permite que o usuário manipule estruturas complexas diretamente no navegador.

(4) Estruturação e Processamento de Dados: O sistema foi projetado com o uso de *Models* e *DTOs* para organização dos dados, além da definição de estruturas auxiliares, como *VariableTypes*. A aplicação realiza o processamento dinâmico de arquivos *JSON* em memória, permitindo a renderização de componentes de forma flexível e adaptável na interface.

(5) Conteinerização e Processamento sob Demanda: Após o desenvolvimento, a aplicação foi conteinerizada utilizando *Docker*, segmentando o sistema em serviços independentes, como o *backend*, o banco de dados *MySQL* e o servidor *Nginx*. Além disso, foi implementado um mecanismo de execução dinâmica em contêineres temporários para tarefas específicas, permitindo que operações mais custosas sejam realizadas de forma isolada, sem impactar o desempenho do sistema principal. Essa abordagem garante isolamento, melhor gerenciamento de recursos e maior escalabilidade da aplicação.
(6) Validação da Arquitetura: A solução foi avaliada qualitativamente quanto à sua organização, escalabilidade e facilidade de manutenção, verificando sua aderência aos princípios de aplicações distribuídas e engenharia de *software* moderna. A figura 1 apresenta um esquema da organização geral do *backend* do sistema.

Figura 1 - Arquitetura do *backend*



Fonte: Autoria própria

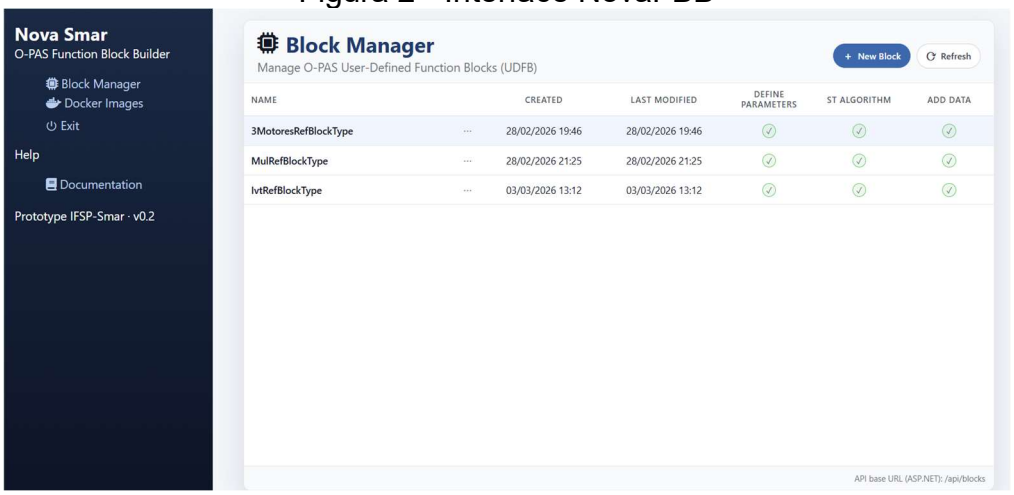
Resultados

A implementação da NovaFBB demonstrou ganhos significativos na agilidade do fluxo de engenharia e na modernização da interface de usuário. A migração para uma plataforma *web* em *.NET*, aliada à conteinerização, resultou em um ambiente mais responsivo, escalável e independente do sistema operacional. Em contraste com a versão desktop, a nova solução centraliza a lógica no *backend* e distribui os serviços em contêineres, garantindo uma interface fluida e um fluxo de trabalho mais dinâmico.
Do ponto de vista de engenharia de software, a aplicação foi estruturada em ASP.NET Core com arquitetura em camadas (*Controllers, Services* e *Repositories*), promovendo separação de responsabilidades e facilitando a manutenção e evolução do sistema. A comunicação via *APIs REST* assíncronas reduziu o acoplamento entre *frontend* e *backend*, enquanto o uso do *Dapper* integrado ao *MySQL* assegurou operações eficientes de acesso a dados.
A adoção de contêineres proporcionou isolamento entre os componentes, aumentando a portabilidade, a segurança e a reprodutibilidade do ambiente. O uso do *Nginx* como

gateway garantiu a comunicação segura, enquanto a separação dos serviços contribuiu para a escalabilidade da aplicação.

A interface, desenvolvida com *Razor*, *HTML*, *CSS* e *JavaScript*, permitiu a criação de componentes dinâmicos e responsivos. O uso de processamento de JSON em memória no backend possibilitou a construção eficiente de estruturas hierárquicas consumidas pela interface via *API*. A figura 2 apresenta a interface principal da NovaFBB.

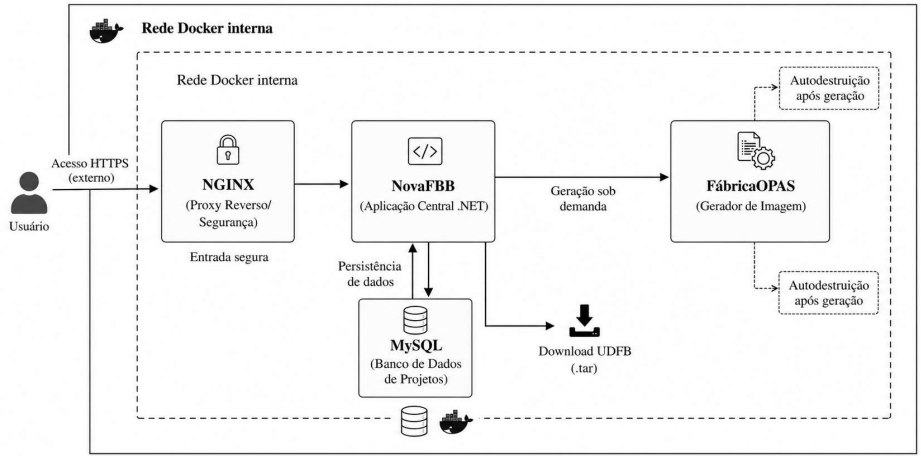
Figura 2 - Interface NovaFBB



Fonte: Autoria própria

Adicionalmente, a adoção de uma arquitetura baseada em serviços e *APIs* favorece a escalabilidade horizontal da solução, permitindo a futura integração com outros sistemas e a evolução para ambientes distribuídos mais complexos, mantendo consistência e desempenho. A figura 3 mostra uma visão geral da organização dos containers.

Figura 3 - Diagrama de blocos da arquitetura dos containers.



Fonte: Autoria própria

Conclusão

A modernização da ferramenta de engenharia por meio da NovaFBB demonstrou que a migração para uma abordagem distribuída baseada em contêineres e desenvolvida em *.NET* constitui uma solução robusta e escalável para a automação industrial. A eliminação de dependências do sistema operacional local, aliada a uma interface moderna, ágil e responsiva, reforça a viabilidade de plataformas de engenharia *web* que abstraem a complexidade da infraestrutura do usuário final. Conclui-se que a orquestração dinâmica dos serviços em contêineres proporciona um fluxo de trabalho mais eficiente, seguro e desacoplado, atendendo às demandas de escalabilidade e flexibilidade exigidas em ambientes industriais.

Referências

BERNSTEIN, D. Containers and cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing, v. 1, n. 3, p. 81-84, 2014. DOI: 10.1109/MCC.2014.51.

PANTONI, R. P. et al. Design and implementation of o-pas user-defined function blocks. Journal of Electrical Systems and Information Technology, v. 11, p. 55, 2024. Disponível em: <https://link.springer.com/article/10.1186/s43067-024-00183-9>.

QAMSANE, Y. et al. Open process automation- and digital twin-based performance monitoring of a process manufacturing system. IEEE Access, v. 10, p. 60823–60835, 2022.

SPAGIARI, V. O.; MOSSIN, E. A.; PANTONI, R. P. Projeto e implementação do mecanismo de execução de blocos funcionais definidos pelo usuário conforme a especificação O-PAS. In: XI Congresso de Iniciação Científica do IFSP Itapetininga, Itapetininga, 27, 28 e 29 de maio de 2025. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus Itapetininga, 2025.

THE OPEN GROUP. O-PAS™ Standard, Version 2.1. Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, Reino Unido, 2023.