AUTOMAÇÃO DE PROCESSOS EMPRESARIAIS COM APIs RESTful CONTEINERIZADAS EM DOCKER

Pablo Boaz Carvalho Gomes[1], Vivian Thaís Varela Oliveira[1], Victor Carvalho Galvão de Freitas[1], Diego da Silva Pereira[1]

[1] Laboratório de Pesquisa em Redes e Sistemas Computacionais (NOCS LAB), Instituto Federal do Rio Grande do Norte, Parnamirim, Brasil
(pablo.boaz@academico.ifrn.edu.br)

Resumo: Este trabalho propõe o desenvolvimento de uma arquitetura de automação de processos empresariais baseada em APIs RESTful conteinerizadas com Docker. O sistema expõe endpoints organizados por regras de negócio configuráveis, permitindo o roteamento e processamento automatizado de solicitações sem intervenção manual constante. A solução é projetada para ser reprodutível, escalável e acessível a equipes de TI de pequeno e médio porte, utilizando ferramentas amplamente adotadas na indústria .

Palavras-chave: Docker; APIs RESTful; Automação de Processos; Conteinerização; FastAPI

INTRODUÇÃO

A transformação digital força empresas de todos os portes a adotarem tecnologias que otimizem rotinas, reduzam custos e acelerem as respostas ao mercado. Nesse contexto, a automação de processos é vital para pequenas e médias empresas (PMEs) que precisam de escalabilidade operacional, mas não possuem grandes infraestruturas de TI .

A comunicação entre sistemas via APIs (*Application Programming Interfaces*) do tipo RESTful é a base da integração tecnológica moderna. Seguindo os princípios de Fielding (2000), estas APIs permitem que aplicações distintas troquem dados de forma padronizada, *stateless* e escalável. Na prática, isso viabiliza a execução automática de tarefas repetitivas, reduzindo erros humanos e conectando softwares heterogêneos com facilidade .

Paralelamente, a conteinerização com Docker tornou-se o padrão de mercado para empacotar e distribuir aplicações, garantindo portabilidade entre ambientes, isolamento de dependências e facilidade de escalonamento (Merkel, 2014). A união de APIs RESTful com containers Docker resulta em uma arquitetura robusta e reprodutível para serviços de automação, funcionando de forma independente do hardware utilizado, o que ajuda equipes de TI com recursos limitados .

Embora o interesse por estas tecnologias seja crescente, a literatura ainda carece de exemplos práticos de pipelines completos que integrem regras de negócio flexíveis, exposição de endpoints e implantação conteinerizada acessível para quem não domina DevOps. A maioria dos trabalhos aborda estes componentes isoladamente (Dumas et al., 2018). Diante disso, este trabalho propõe e implementa uma arquitetura baseada em APIs RESTful conteinerizadas com Docker para automação comercial, avaliando sua viabilidade técnica, desempenho sob carga e impacto prático em cenários que simulam PMEs.

MATERIAL E MÉTODOS

Esta pesquisa aplicada utiliza uma abordagem experimental e quantitativa. O percurso de desenvolvimento foi estruturado em quatro etapas principais: (i) mapeamento e modelagem dos fluxos de trabalho a automatizar; (ii) definição das regras de negócio e estrutura dos endpoints da API; (iii) desenvolvimento e documentação da API RESTful; e (iv) parametrização dos containers e deploy da solução com Docker .

Os cenários criados simulam rotinas de PMEs, como a triagem, validação e o encaminhamento automático de chamados de suporte, pedidos de compras e fluxos de aprovação. As regras de negócio foram programadas como módulos Python independentes, o que facilita manutenções e expansões futuras do código .

A API RESTful foi construída com o framework FastAPI (Python 3.11), escolhido pela sua alta velocidade de resposta, escrita moderna e suporte nativo à documentação automática via OpenAPI/Swagger. Os endpoints foram organizados por domínio (solicitações, aprovações e notificações), com validação de dados de entrada feita pela biblioteca Pydantic. O isolamento da aplicação foi realizado via Docker (versão 24.0), utilizando a imagem base python:3.11-slim para reduzir o tamanho final do arquivo. A orquestração dos serviços (API, base de dados PostgreSQL e serviço de filas) foi centralizada no Docker Compose, permitindo iniciar toda a infraestrutura com um único comando no terminal. Os testes foram executados localmente em uma máquina com processador AMD Ryzen 7 3700U, 8 GB de RAM e Windows 11 Pro (via Docker Desktop e WSL 2), replicando o hardware modesto comum em PMEs.

RESULTADOS E DISCUSSÃO

A aplicação desenvolvida expõe 14 endpoints divididos em três módulos: solicitações, validações e alertas. Com o Swagger UI, todas as rotas foram documentadas automaticamente, tornando os testes de integração intuitivos para equipes técnicas e gestores em cada cenário.

Os testes de carga realizados com a ferramenta Locust demonstraram que a API conteinerizada respondeu de forma estável a até 200 requisições simultâneas sem degradação significativa de desempenho, com latência média abaixo de 50 ms. A imagem Docker gerada totalizou 320 MB, tamanho adequado para distribuição em ambientes com largura de banda limitada.

Em termos operacionais, o sistema automatizado reduziu o tempo médio de processamento de cada requisição em cerca de 82% em comparação com o fluxo manual simulado. O tempo de execução caiu de 4,5 minutos para menos de 1 minuto, limitando a intervenção humana apenas ao tratamento de exceções. Estes dados alinham-se com a literatura, que posiciona as APIs como elemento central na modernização de processos (Dumas et al., 2018.

A arquitetura também se provou altamente reprodutível: o comando *docker-compose up* faz todo o provisionamento do ambiente, eliminando configurações manuais no servidor. Isto facilita a adoção por equipes sem experiência prévia em DevOps, um diferencial importante para pequenas empresas.

Este trabalho demonstrou que a integração de módulos lógicos programáveis , APIs RESTful e Docker é uma abordagem viável e eficiente para modernizar fluxos organizacionais. O ecossistema alcançou alta eficiência no roteamento estruturado de chamados, baixa latência e redução clara de tempo de processamento. Como trabalhos futuros, pretende-se validar a solução com dados reais de empresas parceiras, incluir rotinas de auditoria e monitoramento interno para aumentar a transparência dos fluxos de processos, e estudar o comportamento da arquitetura sob orquestração em Kubernetes para ambientes em larga escala.

CONCLUSÃO

A aplicação deste projeto demonstrou que APIs RESTful conteinerizadas com Docker, integradas a módulos lógicos programáveis, constituem uma solução viável e eficaz para a automação de processos empresariais em PMEs. O sistema desenvolvido, com uma latência média inferior a 50 ms, processou até 200 requisições simultâneas, reduzindo em 82% o tempo de processamento de solicitações em relação ao fluxo manual. Foi gerada uma imagem Docker de 320 MB, adequada para cenários empresariais com infraestrutura limitada. Com um único comando, foi possível confirmar a reprodutibilidade da arquitetura, tornando-a acessível às equipes sem expertise avançada em DevOps. Como trabalhos futuros, o aprimoramento dos mecanismos de auditoria é crucial para o entendimento pleno e validação da arquitetura em empresas parceiras, adotando um processo de autenticação utilizando JWT para controle de papéis (RBAC) e a exploração da orquestração utilizando Kubernetes para ambientes cuja produção é de larga escala .

AGRADECIMENTOS

Os autores agradecem ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Campus Parnamirim, pelo suporte estrutural e acadêmico durante o desenvolvimento desta pesquisa.

REFERÊNCIAS

Dumas, M.; La Rosa, M.; Mendling, J.; Reijers, H. A. Fundamentals of Business Process Management. 2. ed. Springer, Berlin, 2018.

Merkel, D. Docker: Lightweight Linux Containers

for Consistent Development and Deployment. Linux Journal, vol. 2014, n. 239, 2014.

Ramírez, S. FastAPI: modern, fast (high-performance), web framework for building APIs with Python. Disponível em: https://fastapi.tiangolo.com. Acesso em: 2025.

Fielding, R. T. Architectural Styles and the Design of Network-based Software Architectures. Tese de Doutorado. University of California, Irvine, 2000.