Optimizing ETL Applications: A Refactoring and Containerization Approach

Luan Gustavo Oliveira Santana
Universidade Federal de Sergipe
Itabaiana – SE – Brasil
luan2103@academico.ufs.br

Luís Gabriel Costa Lima Ribeiro
Universidade Federal de Sergipe
Itabaiana – SE – Brasil
logxpbr@gmail.com

Methanias Colaço Júnior
Universidade Federal de Sergipe
Itabaiana – SE – Brasil
mjrse@hotmail.com

ABSTRACT

Objective: Apply refactoring and containerization techniques to the DUCA application, a social control platform focused on school management in the state of Sergipe, with emphasis on the ETL (Extract, Transform, Load) functionality. The goal is to improve the performance, maintainability, and scalability of the application.

Methodology/approach: The study involved restructuring the ETL functionality code to adapt the system to new formats of flat files, in addition to containerizing the application using Docker and Docker Compose. The evaluation was conducted through an *in vivo* case study, using metrics such as deployment time, structural code complexity, data load stability, and qualitative feedback from the development team.

Originality/relevance: The combination of code refactoring and containerization applied to a real legacy system is still scarcely explored in an integrated manner in the literature. This work provides a practical solution for modernizing ETL processes in public applications, with direct gains in scalability, standardization, and maintainability.

Main results: After the improvements, there was a reduction of approximately 99.1% in environment deployment time, stabilization of ETL routines, a decrease in error rates, and better control over execution time.

Theoretical/methodological contributions: The study demonstrates how modern software engineering practices can be applied to legacy systems, presenting a replicable methodology for environments that require high data control and development standardization. It highlights the use of Docker as a central tool for scalability and environment consistency.

RESUMO

Aplicar técnicas de refatoração e conteinerização na aplicação DUCA, uma plataforma de controle social voltada para a gestão escolar no estado de Sergipe, com foco na funcionalidade ETL (do inglês, *Extract, Transform, Load*). O objetivo é melhorar o desempenho, a manutenibilidade e a escalabilidade da aplicação. O estudo realizou a reestruturação do código da funcionalidade ETL para adaptar o sistema a novos formatos de arquivos *flat*, além da conteinerização da aplicação com Docker e Docker Compose. A avaliação foi conduzida por meio de um estudo de caso *in vivo*, com uso de métricas como tempo de implantação, complexidade estrutural do código, estabilidade da carga de dados e avaliação qualitativa da equipe técnica. A combinação de refatoração de código e conteinerização aplicada a um sistema legado real ainda é pouco abordada de forma integrada na literatura. Este trabalho oferece uma solução prática para a modernização de processos ETL em aplicações públicas, com ganhos diretos em escalabilidade, padronização e manutenção. Após as melhorias, observou-se uma redução de aproximadamente 99,1% no tempo de implantação do ambiente, estabilização das rotinas ETL, queda nas taxas de erro e melhor controle do tempo de execução. O trabalho demonstra como práticas modernas de engenharia de software podem ser aplicadas em sistemas legados, apresentando uma metodologia replicável para ambientes que exigem alto controle de dados e padronização no desenvolvimento. Destaca-se o uso do Docker como ferramenta central para escalabilidade e consistência de ambientes.

Keywords

ETL, Refactoring, Docker, Containerization, Laravel, Software Engineering

Palavras-Chave

ETL, Refatoração, Docker, Conteinerização, Engenharia de Software

1. INTRODUÇÃO

Nos últimos anos, a crescente complexidade dos sistemas de software tem exigido abordagens mais eficientes para o desenvolvimento e manutenção de aplicações. A refatoração de código, definida como o processo de melhorar a estrutura interna do software sem modificar seu comportamento externo, é fundamental para a melhoria contínua do software, permitindo não apenas a correção de problemas, mas também a

otimização da estrutura interna do código [1].

Neste contexto, a conteinerização surge como uma tecnologia poderosa para padronizar ambientes de desenvolvimento, garantindo que as aplicações funcionem de maneira consistente em diferentes ambientes e facilitando a transição entre programadores [4]. Docker, em particular, tem sido amplamente adotado devido à sua capacidade de criar contêineres que encapsulam uma aplicação juntamente com suas dependências, assegurando que o ambiente de execução seja idêntico em qualquer lugar que o contêiner seja implantado.

Com base nessas tecnologias modernas, este trabalho propõe sua aplicação sobre a DUCA, um sistema apresentado por PASSOS *et al.* (2019), que consiste em um aplicativo civil colaborativo voltado para a melhoria da gestão educacional. O estudo original destacou a importância da confiabilidade dos processos de manipulação de dados no contexto escolar, reforçando a necessidade de soluções eficazes para lidar com grandes volumes de informações. No entanto, não foram exploradas abordagens como refatoração e conteinerização, que poderiam otimizar significativamente o desempenho da aplicação.

Entretanto, apesar das vantagens oferecidas por essas técnicas, muitos sistemas legados continuam a enfrentar desafios significativos, como a complexidade na configuração dos ambientes de desenvolvimento e a dificuldade em manter a integridade e a eficiência dos processos de manipulação de dados. A aplicação DUCA, que atua em um contexto onde a gestão de dados é essencial, enfrenta desafios relacionados à sua funcionalidade de ETL (do inglês, *Extract, Transform, Load*), que requer alta performance e confiabilidade [6]. A Figura 1 ilustra a interface original da aplicação DUCA apresentada no estudo, que serviu como ponto de partida para as melhorias propostas neste estudo.

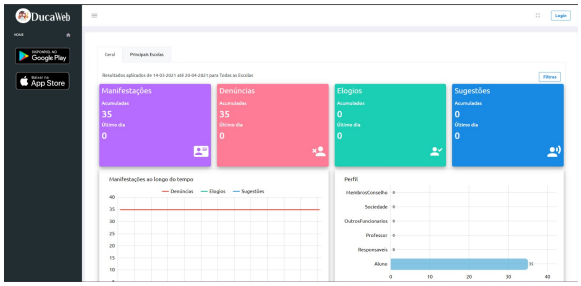

Figure 1: Tela inicial da aplicação

Com base nisso, o objetivo principal deste trabalho é realizar a refatoração e conteinerização da aplicação DUCA, com foco na funcionalidade de ETL, utilizando a tecnologia Docker. A proposta deste estudo é demonstrar como a aplicação dessas técnicas pode resolver os desafios enfrentados, melhorando o desempenho, a manutenção e a escalabilidade da aplicação.

A seguir, o trabalho está estruturado em seções que abordam a base conceitual, a revisão de trabalhos relacionados, a metodologia aplicada, um estudo de caso prático e, por fim, as conclusões obtidas a partir dos resultados.

2. BASE CONCEITUAL

A refatoração de código é uma prática fundamental no desenvolvimento de software, que visa melhorar a estrutura interna de um sistema sem modificar seu comportamento externo. Segundo FOWLER (2018), a refatoração é um processo disciplinado, realizado para aumentar a legibilidade, facilitar a manutenção e reduzir a propensão a erros no código. Ao melhorar o *design* do software após o código já ter sido escrito, a refatoração permite que o sistema continue evoluindo de maneira sustentável, mantendo sua funcionalidade original enquanto se adapta a novas exigências.

No contexto da gestão de dados, a funcionalidade de ETL (do inglês, *Extract, Transform, Load*) desempenha um papel crucial. Segundo COLAÇO JÚNIOR (2004), o processo de ETL envolve a extração de dados de diversas fontes operacionais, sua transformação para atender aos requisitos de qualidade e formato, e, finalmente, sua carga em um *Data Warehouse*, onde os dados são estruturados para facilitar a tomada de decisões. A refatoração de processos de ETL é particularmente importante, pois pode melhorar significativamente a eficiência e a robustez do sistema de processamento de dados, assegurando que as informações sejam manipuladas de maneira íntegra e precisa.

Para enfrentar os desafios relacionados à configuração do ambiente de desenvolvimento, que frequentemente incluem a gestão de componentes como bancos de dados, PHP, Laravel e Composer, a conteinerização tem se mostrado uma solução eficaz. Docker, em particular, oferece uma tecnologia que encapsula a aplicação, suas dependências e bibliotecas em um único contêiner. Isso garante que a aplicação funcione de maneira consistente, independentemente do ambiente em que está sendo executada. Além disso, Docker automatiza a execução e a implantação de aplicações em contêineres, facilitando a padronização dos ambientes de desenvolvimento e a colaboração entre diferentes programadores [3].

SCHLEGEL (2019) apresenta um estudo sobre a contêinerização de *pipelines* ETL utilizando Docker, demonstrando como essa tecnologia pode garantir consistência e portabilidade nos processos de extração e transformação de dados. Essa abordagem permite que os *pipelines* ETL sejam encapsulados de forma modular, garantindo que diferentes estágios do processo possam ser escalados conforme a necessidade, reduzindo a dependência de configurações específicas de ambiente e tornando a integração de novas fontes de dados mais ágil e eficiente.

Já KULAL (2024) detalha a construção de uma *pipeline* ETL automatizada usando Docker, Jenkins, Logstash, Elasticsearch e Kibana, destacando os benefícios da escalabilidade e automação. A integração dessas ferramentas possibilita a criação de fluxos de trabalho dinâmicos, permitindo monitoramento em tempo real e otimizações contínuas no processamento de dados. A utilização de ferramentas como Logstash e Elasticsearch dentro do contexto ETL demonstra como a conteinerização pode facilitar a implementação de arquiteturas orientadas a eventos, promovendo maior flexibilidade e robustez ao processo.

Além disso, PAYPRO GLOBAL (2024) discute a aplicação de contêineres Docker e Kubernetes, enfa-

tizando sua importância na padronização e eficiência do desenvolvimento de software. Kubernetes amplia os benefícios da conteinerização ao permitir a orquestração automática de múltiplos contêineres, garantindo alta disponibilidade, balanceamento de carga e escalabilidade sob demanda. Essa abordagem é essencial para sistemas que lidam com grandes volumes de dados, como *pipelines* ETL, garantindo que os serviços sejam executados de forma distribuída e resiliente.

Dessa forma, a combinação de refatoração de código e conteinerização demonstra-se uma abordagem promissora para a modernização e otimização de processos ETL. A adoção de tecnologias como Docker e Kubernetes permite que sistemas de processamento de dados sejam mais eficientes, escaláveis e resilientes a mudanças, tornando a manutenção e evolução das aplicações mais previsíveis e acessíveis.

3. TRABALHOS RELACIONADOS

A aplicação de técnicas de refatoração de código e conteinerização em sistemas complexos, especialmente em processos de ETL (Extract, Transform, Load), tem se destacado como um campo promissor de pesquisa. Apesar do potencial dessas abordagens para otimizar o desempenho, a escalabilidade e a manutenção de aplicações, a literatura recente ainda é relativamente limitada em estudos que exploram essas práticas de forma integrada.

ARUNA e PRADEEP (2020) propuseram melhorias de desempenho e escalabilidade utilizando tecnologias de contêineres em computação de borda baseada em IoT. O estudo enfatiza os benefícios da contêinerização, especialmente em termos de flexibilidade e capacidade de resposta em sistemas distribuídos. No entanto, ele não aborda diretamente a refatoração do código subjacente ou a aplicação dessas técnicas em sistemas de ETL.

ARUNA e PRADEEP (2020) discutiram os desafios de segurança e as melhores práticas recomendadas para o uso de contêineres Docker, com foco em ambientes de nuvem. Embora esse estudo seja essencial para a compreensão da segurança em contêineres, ele não se aprofunda na aplicação dessas práticas para otimização de sistemas ETL.

KAUR e KAUR (2016) realizaram um estudo prático de refatoração em uma aplicação de biblioteca desenvolvida em Java, utilizando ferramentas como Eclipse, JDeodorant e plugins de métricas para detectar "bad smells" e mensurar os impactos da refatoração. As autoras demonstraram, por meio de indicadores como Complexidade Ciclomática de McCabe, LCOM e WMC, que a refatoração é eficaz para reduzir a complexidade estrutural e melhorar a manutenibilidade do código. Embora o estudo não explore diretamente cenários envolvendo ETL ou conteinerização, seus achados corroboram a proposta deste trabalho ao evidenciar os benefícios mensuráveis da refatoração em sistemas existentes.

Diante disso, o presente trabalho propõe uma contribuição original ao integrar técnicas de refatoração de código e contêinerização com Docker, especificamente para otimizar a funcionalidade de ETL de uma aplicação existente, a DUCA. Diferente dos estudos mencionados, que abordam os temas de forma isolada ou em contextos específicos, este trabalho oferece uma abordagem prática e integrada, com uma análise detalhada das métricas de desempenho, escalabilidade e manutenção, antes e depois das melhorias implementadas.

4. ESTUDO DE CASO

Esta seção apresenta a aplicação prática da refatoração e conteinerização na plataforma DUCA, com foco na rotina de ETL. A iniciativa buscou solucionar problemas relacionados à manipulação de arquivos com diferentes estruturas, comprometendo a eficiência do sistema. As seções subsequentes detalham o objetivo específico da avaliação empírica, o planejamento das ações, as ferramentas e métodos utilizados, bem como as métricas e os critérios adotados para avaliar os resultados obtidos com a aplicação das melhorias implementadas.

4.1 Definição do Objetivo

Este estudo tem como objetivo principal melhorar o desempenho, a manutenção e a escalabilidade da aplicação DUCA por meio da refatoração do código e da conteinerização com a utilização de Docker. A refatoração é necessária, pois a aplicação, em sua forma atual, não consegue processar corretamente os arquivos *flat* em diversos formatos, tais como .csv, .json e .xlsx, fornecidos pelo *stakeholder*, devido a uma alteração na estrutura.

Especificamente, o estudo visa:

- **Melhorar o desempenho:** a refatoração do código foi iniciada para adaptar a funcionalidade de ETL à nova estrutura dos arquivos, otimizando o processamento dos dados. Isso inclui a reescrita das rotinas de extração e transformação para que consigam lidar com diferentes formatos de entrada de forma eficiente, diminuindo o tempo de execução e o consumo de recursos.

- **Facilitar a manutenção:** a conteinerização da aplicação, utilizando Docker, permitirá modularizar o sistema, facilitando o processo de manutenção. Dessa forma, será possível aplicar correções e atualizações relacionadas às mudanças nos arquivos de forma mais ágil e com menor impacto no ambiente geral da aplicação.

- **Aumentar a escalabilidade:** o uso de Docker também permitirá que a aplicação seja facilmente escalada, executando múltiplas instâncias da funcionalidade de ETL, conforme necessário. Isso garantirá que a aplicação possa lidar com volumes maiores de dados e uma diversidade de estruturas de entrada, sem comprometer o desempenho.

A implementação do Docker trará benefícios adicionais, como a consistência entre os ambientes de desenvolvimento e produção, garantindo maior confiabilidade no sistema. Além disso, destaca-se que a necessidade de utilização de arquivos .csv está atrelada a uma restrição tecnológica imposta pelo fornecedor dos dados. Espera-se que, ao final deste trabalho, a

aplicação DUCA esteja preparada para lidar com os novos arquivos e para operar de forma mais eficiente, com maior escalabilidade e facilidade de manutenção.

4.2 Planejamento

Nesta subseção, serão detalhadas as ferramentas e os métodos utilizados para implementar a refatoração e a conteinerização da aplicação DUCA. O processo foi conduzido para garantir que a aplicação se tornasse mais escalável e de fácil manutenção.

4.2.1 Instrumentação

Durante o processo de refatoração da aplicação, diversas ferramentas foram utilizadas para garantir eficiência, padronização e qualidade no desenvolvimento. GitLab e GitHub foram empregados para controle de versão, possibilitando a colaboração entre os desenvolvedores e o acompanhamento das modificações realizadas no código. Docker e Docker Compose foram implementados com o objetivo de permitir que cada componente da aplicação pudesse ser executado de forma isolada, facilitando a replicação e garantindo a consistência entre os ambientes de desenvolvimento, teste e produção. A refatoração do código foi conduzida dentro do framework Laravel, utilizando a versão 7.4 do PHP, aproveitando seus recursos para modularizar o sistema e garantir maior manutenibilidade. O Composer foi utilizado para o gerenciamento de dependências do projeto, otimizando o processo de instalação e atualização de bibliotecas. Como ambiente de desenvolvimento principal, foi adotado o Visual Studio Code, por oferecer integração ágil com as demais ferramentas utilizadas. Além disso, a ferramenta SonarQube foi empregada para análise estática do código, permitindo avaliar métricas de qualidade como complexidade, duplicação e coesão entre módulos.

Quanto aos métodos aplicados, destaca-se a refatoração modular, na qual o código foi reestruturado em componentes menores e mais coesos, o que facilitou a manutenção e o escalonamento do sistema. Outro ponto relevante foi a automação do processo de deploy, implementada por meio do Docker Compose, possibilitando a implantação da aplicação de forma padronizada e com mínima necessidade de ajustes manuais, contribuindo para maior consistência entre os ambientes.

4.2.2 DUCA Antes da Refatoração

A versão original da aplicação apresentava dificuldades na manipulação dos arquivos de entrada devido a mudanças estruturais nos dados fornecidos. Além disso, o ambiente de desenvolvimento exigia configurações manuais para garantir a execução do ETL, tornando o processo suscetível a erros e demandando mais tempo para ajustes e testes. A falta de padronização no carregamento dos dados resultava em inconsistências e impactava a eficiência do sistema, dificultando a escalabilidade e manutenção da aplicação.

4.2.3 Métricas

Para avaliar os efeitos da refatoração e da conteinerização da rotina de ETL na aplicação DUCA, serão adotadas métricas tanto quantitativas quanto qualitativas, com o objetivo de mensurar o impacto das mudanças na manutenção, estabilidade e desempenho do sistema.

Entre as métricas previstas está o tempo de implantação, utilizado para comparar a duração do processo de configuração do ambiente antes e depois da aplicação das melhorias, permitindo verificar o ganho em eficiência com a automação proposta.

A complexidade do código será analisada com base em critérios como complexidade ciclomática, duplicação de trechos e grau de modularidade, buscando estimar o nível de manutenibilidade e organização estrutural da aplicação após a refatoração.

Também serão realizados testes com diferentes cenários de carga de dados, incluindo arquivos com registros válidos e registros contendo erros estruturais. O objetivo é verificar a capacidade do sistema em identificar, tratar e isolar falhas durante o processo de ETL, contribuindo para a análise da robustez da solução.

Além das métricas técnicas, será conduzida uma avaliação qualitativa com os desenvolvedores envolvidos no projeto, visando coletar percepções sobre clareza estrutural, padronização do código e facilidade de manutenção.

Por fim, a documentação técnica será considerada como parte da avaliação, analisando a existência e qualidade de materiais produzidos para apoiar a replicação do ambiente e promover maior autonomia da equipe em futuras implantações.

4.2.4 Metodologia de Avaliação

A avaliação dos impactos da refatoração foi realizada com base na seguinte abordagem:

- **Teste A/B:** comparação do comportamento da aplicação antes e depois da refatoração e conteinerização, avaliando a estabilidade e eficiência do ETL. Para isso, foram isolados dois grupos distintos: um com a aplicação operando sem conteinerização e outro com a aplicação executada dentro de contêineres Docker. Essa separação permitiu a realização do estudo de caso *in vivo* de forma controlada, garantindo uma comparação precisa entre os dois ambientes.

4.3 Operação

A seção de Operação descreve a condução e a avaliação do estudo de caso relacionado à refatoração e conteinerização da aplicação DUCA. Este estudo se caracteriza como um estudo de caso *in vivo*, pois as análises foram realizadas em ambiente real, sem o rigor dos experimentos laboratoriais.

Para assegurar resultados confiáveis e replicáveis, foram adotados princípios da experimentação científica, mitigando vieses e aplicando técnicas validadas para coleta e análise de dados. No contexto da aplicação DUCA, que utiliza *PHP Laravel 7.4* e está conteinerizada com *Docker*, foram estabelecidos métodos específicos para avaliar a performance e o impacto das mudanças implementadas.

A operação da aplicação foi dividida em três etapas principais: *preparação, execução e coleta de métricas.*

4.3.1 Preparação

Consiste na configuração do ambiente de testes e preparação dos arquivos de entrada para os experi-

mentos. A configuração do ambiente foi padronizada utilizando *Docker*, garantindo a consistência entre os testes e o ambiente real de produção. Além disso, os arquivos *flat* (.csv, .json, .xlsx) foram organizados para assegurar a compatibilidade com a nova estrutura de ETL.

4.3.2 *Execução*

Durante essa fase, a aplicação processou os dados utilizando as melhorias implementadas. Para validar a estabilidade e eficiência da solução, foram analisados diferentes cenários de carga: um conjunto com dados válidos e bem estruturados, e outro contendo registros com violações propositalmente inseridas, como colunas ausentes, tipos de dados incorretos e registros duplicados.

Esses testes permitiram avaliar a resiliência do sistema frente a inconsistências, bem como sua capacidade de identificar e tratar erros de forma adequada. Foram realizadas múltiplas execuções do processo de ETL, monitorando os tempos de processamento, a taxa de sucesso nas importações e as falhas detectadas durante as cargas.

Coleta de Métricas

A coleta de métricas foi realizada com base nas seguintes métricas, utilizadas posteriormente na análise dos resultados:

- **Tempo de implantação:** tempo necessário para preparar o ambiente antes e depois da conteinerização, mensurado em minutos.
- **Complexidade do código:** análise da versão refatorada por meio do SonarQube, considerando aspectos como complexidade ciclomática e duplicação de código.
- **Estabilidade da carga de dados:** avaliação da robustez do processo de ETL a partir de execuções com dados válidos e com registros violados, observando a ocorrência e o tratamento de falhas.
- **Avaliação qualitativa dos desenvolvedores:** revisão técnica dos artefatos refatorados, considerando facilidade de manutenção, clareza estrutural e aderência às regras de negócio.

Essas métricas fornecerão uma visão abrangente do impacto da refatoração e conteinerização, permitindo identificar melhorias e potenciais ajustes para futuras otimizações.

5. RESULTADOS E DISCUSSÃO

Esta seção apresenta os principais resultados obtidos após a aplicação das melhorias na rotina de ETL da plataforma DUCA. São descritos os impactos da refatoração e da conteinerização no desempenho, na escalabilidade e na manutenção do sistema, com base nas métricas e critérios definidos anteriormente.

5.1 Tempo de Implantação

Com a introdução do Docker e do Docker Compose, o tempo médio de preparação do ambiente caiu de aproximadamente 40 horas (ou 2.400 minutos) para apenas 19,8 minutos, o que representa uma redução de aproximadamente 99,1 por cento no processo de implantação. Esse tempo inicial considerava um cenário com uma documentação mínima não formal; em sua ausência, a implantação poderia levar ainda mais tempo, de forma exponencial. A padronização automatizada do *setup* por meio da conteinerização contribuiu diretamente para essa melhoria. A Figura 2 exemplifica a execução da aplicação DUCA em *containers* Docker, evidenciando o ambiente modularizado com os serviços *app*, *db* e *ducaweb* operando de forma isolada e consistente. Já a Figura 3 apresenta visualmente a comparação entre os tempos de implantação antes e depois da conteinerização, reforçando o impacto da abordagem adotada.

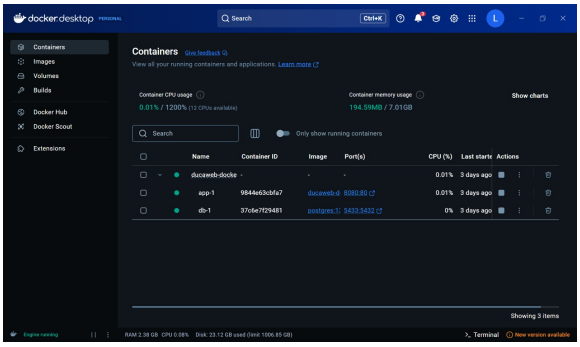

Figure 2: *Containers* Docker executando os serviços da aplicação DUCA.

Figure 3: Comparativo visual do tempo de implantação com e sem uso do Docker.

5.2 Manutenibilidade e Complexidade do Código

Foi observada uma melhora significativa na estrutura do código após a refatoração, com maior coesão e menor acoplamento entre os módulos. Embora não tenha sido possível comparar diretamente com a versão anterior — devido à ausência de um repositório versionado ou histórico rastreável —, as métricas coletadas via SonarQube sobre a versão refatorada indicam um cenário positivo.

A análise indicou uma complexidade ciclomática média de 2,9 por função, sem ocorrência de duplicações críticas e com estrutura de código bem organizada.

Esses indicadores refletem a adoção de boas práticas no processo de refatoração, contribuindo para um código mais legível, modular e de fácil manutenção.

5.3 Resultados Qualitativos

Antes da refatoração, a importação dos arquivos era frequentemente comprometida por inconsistências de estrutura, resultando em falhas recorrentes. Após a modularização e adaptação do código, a taxa de erros caiu drasticamente, pois o novo sistema passou a atender ao formato esperado dos arquivos *flat* e a tratar adequadamente diversas exceções.

Na prática, a carga de dados era conduzida de forma manual e despadronizada, exigindo que os desenvolvedores interviessem constantemente para ajustar colunas, corrigir tipos ou editar arquivos, a fim de viabilizar a leitura. Sem validações automáticas nem monitoramento, o processo se tornava lento, sujeito a falhas silenciosas e altamente dependente do conhecimento prévio da equipe técnica.

Como o sistema anterior não seguia um padrão estrutural e não oferecia retorno confiável, não foi possível mensurar uma taxa precisa de falhas. No entanto, diante da recorrência dos problemas, estima-se que a taxa de sucesso era praticamente nula, inviabilizando qualquer tentativa de automação consistente da carga de dados.

Além da manutenção corretiva, também foram aplicadas ações de manutenção perfectiva, com validações e exceções baseadas nas regras de negócio. Isso proporcionou maior robustez ao sistema, permitindo lidar com variações nos dados de entrada e garantindo a integridade das informações carregadas. A Figura 4 apresenta a nova interface de monitoramento das cargas ETL, permitindo o acompanhamento visual dos status de sucesso ou falha em tempo real.

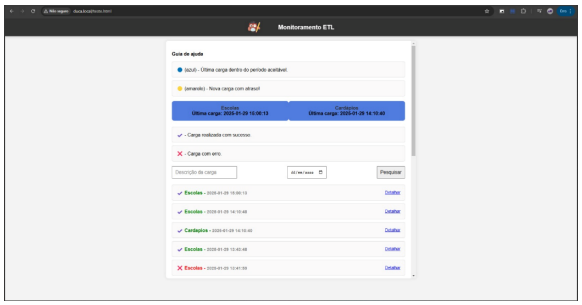

Figure 4: Interface de monitoramento das cargas ETL com indicação de status das execuções.

6. CONCLUSÃO

Este trabalho teve como objetivo principal aplicar técnicas de refatoração e conteinerização à funcionalidade ETL da aplicação DUCA, a fim de modernizar e otimizar sua performance frente às mudanças estruturais nos arquivos de dados utilizados pelo sistema. A necessidade dessa intervenção surgiu da dificuldade que o sistema apresentava para lidar com múltiplos formatos de entrada, exigindo uma solução que fosse ao mesmo tempo escalável, padronizada e de fácil manutenção.

Com a reestruturação do código, adotando uma abordagem modular e de responsabilidade única, foi possível aumentar significativamente a clareza e manutenibilidade do sistema. Além disso, a introdução da conteinerização com Docker e Docker Compose permitiu padronizar o ambiente de desenvolvimento e produção, reduzindo em 99,1 por cento o tempo de implantação e eliminando erros recorrentes relacionados à configuração manual do ambiente.

Através do estudo de caso *in vivo*, com testes realizados em arquivos reais fornecidos pelo *stakeholder*, foi possível estabelecer métricas consistentes para avaliar os impactos da solução. A redução da taxa de erros e o controle efetivo sobre o tempo de execução das rotinas ETL foram evidenciados ao longo do estudo. A aplicação, antes rígida e frágil, passou a ser flexível, adaptável e tecnicamente sustentável.

As contribuições deste trabalho vão além do contexto específico da aplicação DUCA. Ele serve como base metodológica para a modernização de outros sistemas legados que enfrentam desafios semelhantes, sobretudo em ambientes institucionais que demandam estabilidade, escalabilidade e integridade na manipulação de dados.

Como trabalhos futuros, recomenda-se a investigação de soluções baseadas em orquestração de *containers*, como o uso de Kubernetes para gerenciamento de múltiplas instâncias da funcionalidade ETL. Também é possível explorar técnicas de monitoramento contínuo e otimização dinâmica de rotinas de carga e transformação de dados, de modo a aprimorar ainda mais o desempenho do sistema em cenários de maior demanda.

7. REFERENCES

[1] FOWLER, Martin. *Refactoring: Improving the Design of Existing Code*. Boston: Addison-Wesley Professional, 2018.

[2] ARUNA, K.; PRADEEP, G. Performance and scalability improvement using IoT-based edge computing container technologies. *SN Computer Science*, v. 1, p. 91, 2020.

[3] MASDARI, M.; ZANGAKANI, M. Challenges and security issues in Docker-based cloud computing. *The Journal of Supercomputing*, 2020.

[4] POTDAR, A. M.; NARAYAN, D. G.; KENGOND, S.; MULLA, M. M. Performance evaluation of docker container and virtual machine. *Procedia Computer Science*, v. 171, p. 1419–1428, 2020.

[5] COLAÇO JÚNIOR, Methanias. *Projetando Sistemas de Apoio à Decisão Baseados em Data Warehouse*. Rio de Janeiro: Axcel Books do Brasil Editora Ltda., 2004.

[6] PASSOS, A.; RODRIGUES JÚNIOR, M. C.; et al. DUCA: um aplicativo civil colaborativo para alavancar a educação. In: SIMPOSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO (SBSI), 2019, Aracaju. Anais Estendidos do Simpósio Brasileiro de Sistemas de Informação. Porto Alegre: SBC, 2019. .

145-148. DOI:
https://doi.org/10.5753/sbsi.2019.7460.

[7] COLAÇO JÚNIOR, Methanias. IA Para A Galera Toda: Agentes e Inovação Experimental Sem Código. Edição independente, 2025.

[8] SCHLEGEL, Aaron. Containerizing ETL Data Pipelines with Docker. Medium, 2019. Disponível em: https://medium.com/@AaronSchlegel/containerizing-etl-data-pipelines-with-docker-9e30de90a313. Acesso em: 15 abr. 2025.

[9] KULAL, Akshatha. Building an Automated ETL Pipeline with Docker, Jenkins, Logstash, Elasticsearch, and Kibana. Medium, 2024. Disponível em: https://medium.com/@akshathakulal/building-an-automated-etl-pipeline-with-docker-jenkins-logstash-elasticsearch-and-kibana-6603a118816c. Acesso em: 20 mai. 2025.

[10] PAYPRO GLOBAL. Docker vs Kubernetes: Choosing the Right Container Strategy. 2024. Disponível em: https://payproglobal.com/blog/docker-vs-kubernetes/. Acesso em: 10 jun. 2025.

[11] KAUR, S.; KAUR, K. Analysis of the Impact of Refactoring on Software Quality: A Case Study. *International Journal of Software Engineering and Its Applications*, v. 10, n. 11, p. 1-14, 2016.