INTERFACE TECNOLÓGICA

MODERNIZAÇÃO DE UMA APLICAÇÃO LEGADA: um estudo de caso com contêineres

MODERNIZING A LEGACY APPLICATION: a case study with containers

Ronaldo Rodrigues Martins – ronaldo.martins@fatec.sp.gov.br
Faculdade de Tecnologia de Catanduva – Catanduva – SP – Brasil

Gustavo da Silva de Oliveira – social.gustavosilva@gmail.com
Faculdade de Tecnologia de Catanduva – Catanduva – SP – Brasil

Pedro Marino Messias dos Santos – pedro.marino23@gmail.com
Faculdade de Tecnologia de Catanduva – Catanduva – SP – Brasil

DOI: 10.31510/infa.v22i2.2374
Data de submissão: 26/09/2025
Data do aceite: 02/12/2025
Data da publicação: 20/12/2025

RESUMO

Este trabalho apresenta a modernização de um sistema legado por meio da conteinerização, utilizando como estudo de caso o sistema de gestão de eventos acadêmicos da FATEC Catanduva. O objetivo é demonstrar a viabilidade técnica da adoção de contêineres para aumentar a portabilidade, escalabilidade e facilidade de manutenção do sistema, superando limitações observadas na arquitetura original. O estudo foi conduzido com abordagem qualitativa e natureza aplicada, envolvendo análise da arquitetura existente, construção de uma solução baseada em Docker e avaliação dos resultados obtidos após a implantação em ambiente de nuvem. Os resultados indicam ganhos relevantes em isolamento, reprodutibilidade, organização arquitetural e facilidade de implantação, evidenciando que a contêinerização constitui uma alternativa eficaz para modernização de aplicações legadas.

Palavras-chave: Modernização de Software. Contêineres. Docker. Sistemas Legados.

ABSTRACT

This work presents the modernization of a legacy system through containerization, using as a case study the academic event management system of FATEC Catanduva. The objective is to demonstrate the technical feasibility of adopting containers to increase the system's portability, scalability, and maintainability, overcoming limitations observed in the original architecture. The study was conducted using a qualitative and applied approach, involving analysis of the existing architecture, construction of a Docker-based solution, and evaluation of the results obtained after deployment in a cloud environment. The findings indicate relevant gains in isolation, reproducibility, architectural organization, and ease of deployment, showing that containerization constitutes an effective alternative for modernizing legacy applications.

Keywords: Modernization of Software. Containers. Docker. Legacy Systems.

1 INTRODUÇÃO

A conteinerização emergiu como uma solução tecnológica estratégica para a modernização de sistemas legados, permitindo que aplicações desenvolvidas em tecnologias obsoletas ou sem documentação adequada sejam executadas em ambientes contemporâneos com modificações mínimas. Os contêineres encapsulam todas as dependências e configurações necessárias de *software*, garantindo portabilidade entre diferentes infraestruturas e facilitando a migração para ambientes em nuvem (Bass, Weber e Zhu, 2015).

Nesse contexto, a Faculdade de Tecnologia de Catanduva (FATEC Catanduva) enfrentava um desafio significativo: apresentava limitações técnicas de escalabilidade e dependia de uma infraestrutura difícil de manter. Além disso, a ausência dos desenvolvedores originais e de documentação adequada tornou a manutenção inviável, reforçando a necessidade de uma solução que preservasse as funcionalidades essenciais ao mesmo tempo em que modernizasse a infraestrutura.

Diante desse cenário, este trabalho tem como objetivo demonstrar a viabilidade técnica da conteinerização como estratégia de modernização para sistemas legados, utilizando como estudo de caso o sistema de gerenciamento de simpósio da FATEC Catanduva. De forma específica, propõe-se: (1) analisar a arquitetura atual do sistema e identificar pontos críticos; (2) implementar uma solução baseada em contêineres utilizando Docker; e (3) avaliar os benefícios obtidos em termos de escalabilidade, portabilidade e manutenção.

As contribuições deste trabalho buscam demonstrar como a conteinerização pode ser aplicada de forma eficaz na modernização de sistemas legados, oferecendo evidências alcançados em escalabilidade, portabilidade e manutenção. Além disso, o trabalho reforça o potencial da abordagem como referência para projetos semelhantes em instituições de ensino ou em outros contextos que demandem soluções de modernização tecnológica.

2 FUNDAMENTAÇÃO TEÓRICA

A evolução da infraestrutura de TI tem sido fortemente impulsionada pelas tecnologias de virtualização e conteinerização, especialmente no contexto da computação em nuvem. Este capítulo aborda: (1) os fundamentos da virtualização como base para otimização de recursos e suporte à nuvem; (2) a conteinerização como virtualização em nível de aplicação, destacando sua eficiência e portabilidade no ciclo de desenvolvimento e implantação de *software*; e (3) os

conceitos de sistemas legados, seus desafios de manutenção e escalabilidade, bem como estratégias para análise e mitigação de riscos.

2.1 Computação em Nuvem

A computação em nuvem representa uma transformação significativa na forma como os recursos de tecnologia da informação são disponibilizados e consumidos. Ela permite acesso remoto e sob demanda a recursos computacionais, promovendo inovação, agilidade e competitividade em diversos setores, como saúde, educação, finanças e governo (Marston *et al*., 2011). Mais do que uma tecnologia, a nuvem reflete uma mudança no modelo de negócios, migrando de um paradigma centrado em ativos físicos para um ecossistema baseado em serviços, impulsionado pela virtualização e por economias de escala (Vaquero *et al*., 2009; Marston *et al*., 2011), o que reforça seu papel estratégico nas organizações contemporâneas.

Em termos conceituais, a computação em nuvem consiste em um modelo que disponibiliza recursos compartilhados, como servidores, armazenamento, redes e aplicações, de forma virtualizada e acessível de qualquer lugar e a qualquer momento (Mell e Grance, 2011). Entre seus princípios centrais estão o autoatendimento sob demanda, que permite provisionamento de recursos sem interação direta com o provedor (Armbrust *et al*., 2010), o *pooling* de recursos, que possibilita compartilhamento eficiente entre múltiplos clientes (Buyya *et al*., 2009), a elasticidade para escalonamento automático conforme a demanda (Zhang, Cheng e Boutaba, 2010) e o serviço medido, que assegura transparência e controle do consumo (Marston *et al*., 2011). Esses elementos consolidam a nuvem como um marco na evolução da TI, oferecendo otimização de custos e maior flexibilidade operacional (ISO/IEC 17788, 2014).

Para atender às diferentes necessidades das organizações, a computação em nuvem organiza-se em modelos de serviço que equilibram controle técnico e simplicidade operacional. A Infraestrutura como Serviço (do inglês, *Infrastructure as a Service* - IaaS) fornece recursos básicos de computação, como máquinas virtuais, armazenamento e redes, permitindo que os usuários gerenciem sistemas e aplicações enquanto o provedor mantém a infraestrutura (Buyya *et al*., 2009). A Plataforma como Serviço (do inglês, *Platform as a Service* - PaaS) disponibiliza ambientes pré-configurados, abstraindo a infraestrutura e permitindo que os desenvolvedores concentrem-se na lógica de negócio (Zhang, Cheng e Boutaba, 2010), enquanto o *Software* como Serviço (do inglês, *Software as a Service* - SaaS) oferece aplicativos completos via navegador, com manutenção e atualizações gerenciadas

pelo provedor (Armbrust *et al*., 2010).

Essas características trazem vantagens expressivas, incluindo escalabilidade dinâmica, redução de custos iniciais, acessibilidade remota, segurança avançada e resiliência operacional (Buyya *et al*., 2009; Vaquero *et al*., 2009; Zhang, Cheng e Boutaba, 2010). Por outro lado, a adoção da nuvem exige atenção a desafios como dependência de conectividade com a internet, risco de *vendor lock-in*, latência em aplicações sensíveis e complexidade regulatória, especialmente em relação ao armazenamento de dados em jurisdições estrangeiras. A escolha adequada depende, portanto, de uma avaliação criteriosa que equilibre os benefícios operacionais com as limitações potenciais (Armbrust *et al*., 2010).

2.2 Contêineres

Os contêineres representam uma forma de virtualização leve utilizada para empacotar todos os componentes essenciais de uma aplicação. Diferentemente das máquinas virtuais tradicionais, que dependem de hipervisores para emular o *hardware* completo, os contêineres compartilham o *kernel* do sistema operacional da máquina hospedeira, criando ambientes isolados e consistentes que podem ser executados em diferentes sistemas ou plataformas computacionais. Essa característica assegura que a aplicação funcione de maneira idêntica, independentemente do ambiente em que é implantada (Microsoft Azure, s.d.).

Essa consistência se dá porque os contêineres encapsulam não apenas o código da aplicação, mas também os arquivos de configuração, bibliotecas e dependências necessárias, formando um pacote portátil e autossuficiente. Com isso, é possível evitar conflitos de dependências e garantir que desenvolvedores e operadores utilizem o mesmo ambiente, simplificando o desenvolvimento, teste, implantação e escalabilidade de aplicações em ambientes distribuídos (Docker, s.d.; Monteiro, Xavier e Valente, 2017).

O Docker surge como uma plataforma consolidada para implementar contêineres, fornecendo uma estrutura padronizada para empacotar, transportar e executar aplicações de forma isolada. Cada contêiner funciona como uma “caixa virtual” leve, executando sua aplicação de forma independente, mas compartilhando o sistema operacional do hospedeiro. Essa padronização resolve problemas comuns de incompatibilidade entre ambientes e garante portabilidade entre diferentes sistemas, facilitando a distribuição de *software* em larga escala (Microsoft Azure, s.d.; Docker, s.d.).

Em aplicações multicontêiner, o Docker Compose simplifica ainda mais o processo, permitindo definir serviços, redes e volumes em um único arquivo de configuração,

facilitando a replicação de ambientes de desenvolvimento, teste e produção (Monteiro, Xavier e Valente, 2017). Dessa forma, a orquestração assegura consistência operacional, otimiza recursos e mantém a eficiência em aplicações modernas, especialmente em contextos de desenvolvimento colaborativo e implantação em produção (Santos, 2020, *apud* Rosa e Dos Reis Mota, 2021).

2.3 Softwares Legados

Os *softwares* legados constituem sistemas antigos que sustentam operações essenciais nas organizações, embora frequentemente utilizem tecnologias desatualizadas e possuam código de difícil compreensão. Esses sistemas permanecem em funcionamento devido à sua importância para o negócio, mas sua evolução e modernização são limitadas por restrições técnicas, organizacionais e de conhecimento da equipe atual (Pressman, 2016).

Esses sistemas apresentam características marcantes que os diferenciam de aplicações mais recentes. Geralmente, possuem projetos extensos, documentação incompleta ou inexistente, dependências complexas e arquitetura rígida, o que aumenta a dificuldade de manutenção e refatoração. A resistência à modificação, seja por falta de expertise ou pelo risco de impactar funções críticas, é uma característica central dos *softwares* legados, tornando-os sistemas estáveis, mas tecnologicamente limitados (Pressman, 2016).

A manutenção desses sistemas enfrenta desafios significativos. A obsolescência tecnológica, a escassez de profissionais qualificados, a integração complexa com sistemas modernos, os riscos de segurança e os altos custos de sustentação tornam a gestão de *softwares* legados uma tarefa crítica (IEEE, 2014). Além disso, a resistência organizacional à mudança e a dependência de infraestruturas desatualizadas perpetuam a vulnerabilidade desses sistemas, dificultando a inovação e aumentando o risco de falhas operacionais (Armbrust *et al*., 2010).

Diante desses obstáculos, diversas estratégias podem ser adotadas para equilibrar a preservação das funcionalidades críticas e a modernização tecnológica. A reengenharia e a refatoração gradual permitem evoluir sistemas sem descontinuá-los, enquanto a virtualização por contêineres viabiliza a execução de aplicações legadas em ambientes modernos sem alterações profundas (Bass, Weber e Zhu, 2015). A substituição total ou parcial por soluções *cloud-native* ou comerciais deve ser avaliada com base em custo-benefício e impacto nos fluxos de trabalho, e a gestão do conhecimento, com documentação detalhada e treinamento, é essencial para preservar expertise em tecnologias legadas (ISO/IEC 14764, 2022).

3 PROCEDIMENTOS METODOLÓGICOS

O objetivo deste trabalho é analisar a viabilidade técnica da conteinerização como estratégia de modernização de sistemas legados, utilizando como estudo de caso o sistema de gestão do Simpósio Acadêmico-Científico da FATEC Catanduva. O estudo de caso foi escolhido por se tratar de um sistema legado crítico, pouco documentado, e por permitir observar, de forma aplicada, os benefícios da conteinerização em seu contexto real. Conforme Yin (2015), o estudo de caso é um método adequado para investigar fenômenos complexos em seus ambientes naturais, especialmente quando não há limites claros entre o objeto de estudo e seu contexto, permitindo análise detalhada de processos técnicos e operacionais.

Para detalhar a metodologia adotada, esta seção apresenta o procedimento do estudo estruturado em três fases principais: (i) análise da arquitetura atual do sistema e identificação de pontos críticos; (ii) implementação de uma solução baseada em contêineres utilizando Docker; e (iii) documentação e versionamento do sistema. Cada uma dessas etapas será descrita a seguir, evidenciando o fluxo de trabalho e a integração entre as fases do estudo.

- **Análise da arquitetura e identificação de pontos críticos**: A primeira etapa da metodologia envolveu o mapeamento detalhado da arquitetura atual do sistema, incluindo módulos, fluxos de dados entre aplicações *frontend* e *backend*, dependências de bibliotecas e *frameworks*, bem como os componentes de lógica de negócio relacionados ao gerenciamento de inscrições, submissões e emissão de certificados. A compreensão das limitações técnicas e operacionais foi obtida por meio de observação direta, entrevistas com usuários e análise do código-fonte, em função da escassez de documentação formal. Esta fase possibilitou identificar os pontos críticos que precisariam ser tratados para garantir a continuidade e modernização do sistema.
- **Modernização baseada em contêineres**: Com base nos dados coletados na etapa anterior, implementou-se a solução de conteinerização. Foram criados e otimizados contêineres específicos para *backend* e *frontend* da aplicação existe utilizando a ferramenta Docker. Para orquestração e replicação do ambiente, foi utilizada a ferramenta Docker Compose, permitindo a execução integrada e consistente dos serviços durante o desenvolvimento, testes e homologação. A implantação final

ocorreu em uma máquina virtual na plataforma Microsoft Azure[1], configurada para reproduzir o ambiente de desenvolvimento, assegurando consistência e reprodutibilidade. Esta etapa garantiu que o sistema pudesse operar de forma isolada, portátil e escalável.

- **Documentação e versionamento**: Todos os arquivos de configuração, scripts e documentação do sistema foram organizados em um repositório público no GitHub[2], garantindo reprodutibilidade, transparência e flexibilidade operacional. Essa abordagem permite que a implementação e os procedimentos adotados possam ser replicados em outros contextos de modernização de sistemas legados, assegurando continuidade e facilidade de manutenção.

4 DESENVOLVIMENTO

O desenvolvimento do trabalho seguiu uma sequência estruturada de etapas, cada uma voltada a compreender, modernizar e documentar o sistema legado. Nesta seção, são apresentadas as principais atividades realizadas, desde a análise da arquitetura existente, passando pela implementação da modernização com contêineres, até a organização de documentação e versionamento do projeto, destacando as estratégias adotadas para garantir reprodutibilidade, escalabilidade e manutenção eficiente da aplicação.

4.1 Análise da Arquitetura

A etapa inicial concentrou-se na compreensão do sistema legado, fundamental para viabilizar sua modernização e conteinerização. Na ausência de documentação formal, as informações foram obtidas por meio de conversas com antigos desenvolvedores e usuários. Esse processo permitiu mapear as principais funcionalidades:

- **Inscrição de participantes**: registro de participantes em diferentes eventos.
- **Gerenciamento de eventos**: cadastro e organização de palestras e demais atividades.
- **Geração de certificados**: emissão de certificados digitais aos participantes.

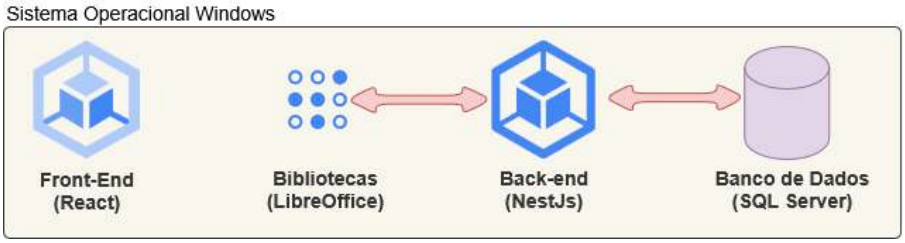
Além disso, foram identificados pontos críticos relacionados a dificuldades técnicas, instabilidades e limitações decorrentes de tecnologias obsoletas.

A Figura 1 ilustra a arquitetura atual da aplicação, que foi analisada para mapear sua

[1] https://azure.microsoft.com/pt-br/
[2] https://github.com/

estrutura e dependências. O sistema é hospedado em servidor de nuvem, exigindo a alocação de uma máquina virtual com sistema operacional Windows Server, uma vez que variáveis de ambiente e configurações foram originalmente projetadas para esse sistema operacional. A aplicação é organizada em duas unidades: *backend* em NestJS[3], que estrutura o código de forma modular, e *frontend* em React[4], utilizado para criação de interfaces dinâmicas. O banco de dados mantido é o Microsoft SQL Server[5], garantindo persistência e gerenciamento das informações. Além disso, o sistema utiliza bibliotecas LibreOffice[6] para geração automática de certificados. Esse mapeamento técnico foi essencial para compreender o ambiente existente e orientar as etapas seguintes do projeto.

Figura 1 – Arquitetura Original



Fonte: Produção própria (2025)

4.2 Projeto de Contêineres e Alocação em Nuvem

A Figura 2 ilustra a organização dos contêineres do sistema, mostrando como *frontend* e *backend* foram estruturados para manter a consistência, escalabilidade e facilidade de manutenção. Após o mapeamento inicial do sistema, procedeu-se ao projeto dos contêineres, projetados para executar em sistema operacional Linux, etapa essencial para assegurar a escalabilidade e manutenção da aplicação. Optou-se pela criação de dois contêineres distintos: um para o *backend* e outro para o *frontend*, permitindo atualização e implantação independentes de cada camada.

O contêiner do *backend* concentra toda a lógica de negócio do sistema, incluindo módulos de processamento de inscrições, cadastro de eventos e geração de certificados. Para sua execução, é incluído o Node.js, necessário para rodar aplicações NestJS, além do código

[3] https://nestjs.com/
[4] https://react.dev/
[5] https://www.microsoft.com/pt-br/sql-server
[6] https://pt-br.libreoffice.org/

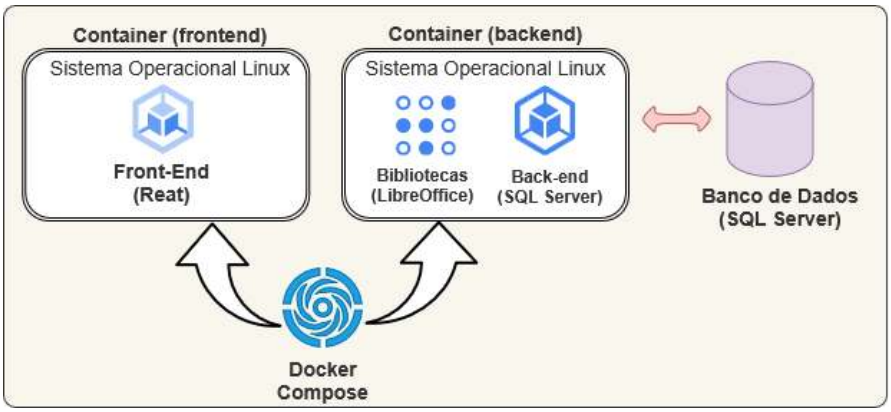
compilado, arquivos de configuração e o LibreOffice, que permite gerar certificados em PDF. Essa configuração garante que o *backend* opere de forma isolada, segura e reprodutível.

O contêiner do *frontend* serve a interface web compilada em React. Para a execução em produção, não é necessário Node.js, pois os arquivos compilados são servidos de forma estática por um servidor dentro do contêiner, garantindo portabilidade e independência em relação ao ambiente.

A comunicação entre os containers do *frontend* e do *backend* é organizada pelo Docker Compose, uma ferramenta que permite gerenciar múltiplos containers de forma coordenada. No projeto, o Docker Compose define dois serviços principais: o *frontend* e o *backend*, cada um rodando em seu próprio container isolado.

Cabe ressaltar que o *backend* compartilha arquivos críticos, como certificados e documentos, por meio de volumes que conectam pastas da máquina hospedeira ao container. Dessa forma, os dados gerados e processados pelo *backend* ficam persistentes e acessíveis conforme necessário, mesmo que o container seja reiniciado.

Figura 2 **- Projeto de Contêineres**



Fonte: Produção própria (2025)

Com a aplicação organizada em contêineres, a etapa seguinte consistiu na definição e configuração do ambiente de implantação em nuvem. Optou-se pelo uso da plataforma Microsoft Azure, pois a instituição pode usufruir de créditos acadêmicos gratuitos por meio do programa *Azure for Students*, viabilizando o desenvolvimento sem custos adicionais.

O sistema de eventos possui operação sazonal, funcionando aproximadamente um mês em duas ocasiões ao ano. Dessa forma, a máquina virtual é alocada sob demanda, garantindo eficiência no uso de recursos e redução de custos operacionais. Para a execução do sistema,

foi criada uma máquina virtual com sistema operacional Ubuntu Server, versão estável e leve, configurada com 2 vCPUs e 4 GB de memória RAM, suficiente para suportar a carga de trabalho prevista, equilibrando desempenho e custo.

4.3 Versionamento do Sistema

O controle de versão e a documentação do sistema desempenharam papel central no desenvolvimento, garantindo rastreabilidade das alterações e organização das funcionalidades implementadas. Para isso, utilizou-se o GitHub, reunindo todos os arquivos do projeto, incluindo código-fonte, arquivos de configuração do Docker e Docker Compose, e *scripts* auxiliares. Esse ambiente permite gestão colaborativa, histórico detalhado das modificações e manutenção da integridade do projeto em diferentes estágios de desenvolvimento.

Paralelamente, foi elaborada documentação das principais funcionalidades do sistema, como inscrições, coleta de presença e emissão de certificados, organizada no próprio repositório para suprir a ausência de registros da versão original e orientar uso e manutenção. A combinação do GitHub para versionamento com a documentação integrada aumentou a reprodutibilidade, sustentabilidade e continuidade do projeto, facilitando futuras atualizações.

5 RESULTADOS E DISCUSSÃO

A modernização da aplicação legada mostrou-se tecnicamente viável e alinhada aos objetivos propostos, com avanços evidentes em portabilidade, escalabilidade e manutenção. A implantação em ambiente de nuvem possibilitou uma análise qualitativa consistente dos benefícios da conteinerização, demonstrando sua eficácia como estratégia para superar limitações do sistema anterior. Os resultados obtidos reforçam a aderência da solução às práticas consolidadas de modernização baseadas em virtualização leve.

No eixo da portabilidade, verificou-se que a aplicação passou a operar de forma uniforme em diferentes ambientes, incluindo máquinas de desenvolvimento e servidores Linux. A conteinerização eliminou conflitos recorrentes da versão legada ao encapsular todas as dependências necessárias à execução, dispensando ajustes manuais durante migrações entre provedores. Esse comportamento confirma as propriedades de isolamento descritas por Bass, Weber e Zhu (2015), evidenciadas pela reprodutibilidade do ambiente e pela capacidade de padronizar a implantação, aspecto especialmente relevante para um sistema utilizado sazonalmente.

Quanto à escalabilidade, a separação do sistema em serviços independentes ampliou a flexibilidade na gestão de recursos. A possibilidade de replicação horizontal dos contêineres do backend permitiu atender aumentos de demanda sem alterações estruturais no código ou no ambiente, em conformidade com as recomendações de modularização e isolamento discutidas por Monteiro, Xavier e Valente (2017). A utilização do Docker Compose simplificou a orquestração dos serviços e proporcionou maior previsibilidade no tratamento de variações de carga, reduzindo riscos de inconsistências e melhorando a eficiência operacional.

No âmbito da manutenção, os resultados foram igualmente expressivos. A divisão entre frontend e backend possibilitou atualizações independentes, menor risco de regressões e maior facilidade de depuração. A reconstrução completa do ambiente com um único comando reforçou a reprodutibilidade da implantação e eliminou a necessidade de instalar dependências manualmente. O uso de um repositório versionado no GitHub ampliou a rastreabilidade das modificações e mitigou a dependência de conhecimento tácito — um problema recorrente em sistemas legados segundo Pressman (2016).

Ainda que tenham sido identificadas limitações, como a curva de aprendizado associada à tecnologia de contêineres e o leve aumento no consumo de recursos, os benefícios observados superam amplamente tais restrições. Desse modo, os resultados reforçam os estudos de Bass, Weber e Zhu (2015) e de Monteiro, Xavier e Valente (2017), evidenciando que a conteinerização é uma estratégia sólida e recomendável para modernização de aplicações legadas em contextos com infraestrutura limitada e heterogênea.

6 CONSIDERAÇÕES FINAIS

O estudo demonstrou que a conteinerização é uma estratégia tecnicamente viável para modernizar aplicações legadas, permitindo isolar serviços, simplificar implantações e garantir maior portabilidade e reprodutibilidade. A separação entre *frontend* e *backend* e o uso de Docker Compose contribuíram para uma arquitetura mais organizada e de fácil manutenção.

A análise mostrou que a solução reduziu inconsistências entre ambientes, facilitou a operação em nuvem e permitiu implantações sazonais com menor esforço operacional. Entretanto, foram identificadas limitações relacionadas à curva de aprendizado da equipe e ao consumo adicional de recursos para execução dos contêineres.

De forma geral, os resultados obtidos indicam que a conteinerização pode ser utilizada como uma abordagem recomendável para modernizar sistemas legados em instituições com restrições técnicas, orçamentárias ou de pessoal. Como trabalhos futuros, sugere-se explorar

automação da implantação com ferramentas de CI/CD, avaliação comparativa de desempenho entre ambientes conteinerizados e nativos e evolução da arquitetura para um modelo *cloud-native*.

REFERÊNCIAS

ARMBRUST, M. *et al*. A View of Cloud Computing. **Communications of the ACM**, v. 53, n. 4, p. 50–58, 1 abr. 2010.

BASS, L.; WEBER, I.; ZHU, L. **DevOps: A Software Architect's Perspective**. Boston, MA, USA: Addison-Wesley Educational, 2015.

BUYYA, R. *et al*. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. **Future generations computer systems: FGCS**, v. 25, n. 6, p. 599–616, 2009.

DOCKER. **What is Docker?**. [s.d.]. Disponível em: https://docs.docker.com/get-started/docker-overview/. Acesso em: 15 maio 2025.

IEEE. **Guide to the Software Engineering Body of Knowledge (SWEBOK): version 3.0.** [S.l.]: IEEE Computer Society Press, 2014.

ISO/IEC 14764. **Software Engineering - Software Life Cycle Processes - Maintenance**. 2022.

ISO/IEC 17788. **Information technology — Cloud computing — Overview and vocabulary**. 2014.

MARSTON, S. *et al*. **Cloud computing – The business perspective**. Decision Support Systems, v. 51, n. 1, p. 176–189, 2011.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. 2011.

MICROSOFT AZURE. **What is a container?**. [s.d.]. Disponível em: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-container. Acesso em: 15 maio 2025.

MONTEIRO, A.; XAVIER, P.; VALENTE, D. **Uma Caracterização em Larga Escala da Arquitetura de Sistemas Docker**. 2017. Disponível em: https://homepages.dcc.ufmg.br/~mtov/pub/2017-vem-lucas.pdf. Acesso em: 23 set 2025.

PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. Porto Alegre: AMGH, 2016.

ROSA, J. A. C.; DOS REIS MOTA, J. **Utilização e orquestração de containers em aplicações web**. *Revista do Fórum Gerencial*, v. 1, n. 2, p. 126-139, 2021.

VAQUERO, L. M. *et al*. **A break in the clouds: towards a cloud definition**. *ACM SIGCOMM Computer Communication Review*, v. 39, n. 1, p. 50–55, 2009.

YIN, Robert K. **Estudo de Caso: Planejamento e métodos**. Porto Alegre: Bookman Editora, 2015.

ZHANG, Q.; CHENG, L.; BOUTABA, R. **Cloud computing: state-of-the-art and research challenges**. *Journal of Internet Services and Applications*, v. 1, p. 7–18, 2010.