Revista Brasileira de Estudos de Gestão e Desenvolvimento Regional
v. 08, n. 03, p. 12-22, 2026
Seção de Artigos

ANÁLISE PARA MIGRAÇÃO DE SISTEMAS LEGADO PARA PLATAFORMAS EMERGENTES COM ÊNFASE NO ORACLE APEX

ANALYSIS FOR MIGRATING LEGACY SYSTEMS TO EMERGING PLATFORMS WITH AN EMPHASIS ON ORACLE APEX

ANÁLISIS PARA LA MIGRACIÓN DE SISTEMAS HEREDADOS A PLATAFORMAS EMERGENTES CON ÉNFASIS EN ORACLE APEX

Ewerton Olegario Dos Santos
Universidade do Estado de Mato Grosso - UNEMAT
e-mail: ewerton.olegario@unemat.br

Dra. Janecler Foppa
https://orcid.org/0000-0002-8906-4195
Universidade do Estado de Mato Grosso - UNEMAT
e-mail: janecler.foppa.snp@gmail.com

Submissão em: 17/01/2026 Aceito em: 09/02/2026

RESUMO
A modernização de sistemas legados representa um desafio estratégico para empresas que buscam maior escalabilidade, segurança e eficiência operacional. O artigo tem como objetivo analisar os principais desafios e benefícios da migração de sistemas legados para plataformas emergentes, com ênfase em soluções robustas como Oracle APEX. Adota-se abordagem qualitativa e descritiva, fundamentada em análises documentais e revisão bibliográfica. Resultados indicam que adoção de uma plataforma low-code integrada ao banco de dados existente simplificou transição, reduziu custos e facilitou o treinamento das equipes, que puderam aproveitar conhecimentos prévios. A criação de componentes reutilizáveis e *frameworks* internos mostrou-se essencial para a sustentabilidade do processo. Por tratar-se de estudo voltado à viabilidade e planejamento, não foram realizados testes em ambiente de produção, limitando-se à análise comparativa e desenvolvimento de protótipos em escala reduzida.
Palavras-chave: Análise de sistemas, Banco de dados, *Low-code*

ABSTRACT
The modernization of legacy systems represents a strategic challenge for companies seeking greater scalability, security and operational efficiency. The article aims to analyze the main challenges and benefits of migrating from legacy systems to emerging platforms, with an emphasis on robust solutions such as Oracle APEX. A qualitative and descriptive approach was adopted, based on documentary analysis and bibliographic review. Results indicate that adopting a low-code platform integrated with the existing database simplified the transition, reduced costs and facilitated the training of teams, who were able to take advantage of previous knowledge. The creation of reusable components and internal frameworks proved to be essential for the sustainability of the process. As this is a study focused on feasibility and planning,

no tests were carried out in a production environment, limited to comparative analysis and development of prototypes on a reduced scale.
Keywords: Systems analysis, Database, Low-code

RESUMEN
La modernización de los sistemas heredados representa un desafío estratégico para las empresas que buscan mayor escalabilidad, seguridad y eficiencia operativa. El artículo tiene como objetivo analizar los principales desafíos y beneficios de migrar de sistemas heredados a plataformas emergentes, con énfasis en soluciones robustas como Oracle APEX. Se adoptó un enfoque cualitativo y descriptivo, basado en análisis documental y revisión bibliográfica. Los resultados indican que la adopción de una plataforma low-code integrada con la base de datos existente simplificó la transición, redujo los costos y facilitó la capacitación de los equipos, quienes pudieron aprovechar el conocimiento previo. La creación de componentes reutilizables y marcos internos resultó ser esencial para la sostenibilidad del proceso. Al tratarse de un estudio centrado en la viabilidad y la planificación, no se realizaron pruebas en entorno de producción, limitándose a análisis comparativos y desarrollo de prototipos a escala reducida.
Palabras clave: Análisis de sistemas, Base de datos, *Low-code*

1 INTRODUÇÃO

A modernização de sistemas legados é um desafio crítico para empresas que buscam maior escalabilidade, segurança e eficiência operacional. Com a crescente adoção de arquiteturas baseadas na web, a transição de sistemas desenvolvidos em linguagens obsoletas para *frameworks* modernos se tornou uma necessidade estratégica.

Migrar sistemas legados não é simples, especialmente ao considerar como isso pode influenciar diretamente o desempenho e a estabilidade das organizações. Os principais desafios geralmente incluem a incompatibilidade entre tecnologias, a dificuldade de integrar novas ferramentas, o aumento da complexidade no processo de manutenção e até mesmo a necessidade de capacitar adequadamente as equipes. Investir em um planejamento meticuloso, selecionar metodologias eficientes e soluções tecnológicas de fato confiáveis são etapas essenciais para que a migração seja bem-sucedida.

A dependência de sistemas legados, frequentemente desenvolvidos em linguagens e plataformas já obsoletas, expõe organizações a riscos operacionais, custos elevados de manutenção e dificuldades de integração (Sommerville, 2019). No Brasil, estudos indicam que até 62 % das empresas de médio porte mantêm aplicações críticas escritas há mais de quinze anos (Andrade; Lima, 2023). Nesse cenário, a modernização não se restringe à substituição tecnológica, mas envolve a preservação de regras de negócio, a garantia de continuidade de serviços e a adoção de novas arquiteturas.

O Oracle APEX destaca-se como plataforma emergente favorecida por seu modelo de desenvolvimento declarativo, estreita integração ao banco de dados Oracle e riqueza de componentes pré-construídos (Oracle Corporation, 2024). Diferentemente de soluções de alto código, o APEX minimiza a escrita manual, acelera entregas e facilita a governança, fatores cruciais para equipes que mantêm

sistemas legados em PL/SQL, Forms ou outras tecnologias relacionais. Este artigo tem por objetivo discutir como o APEX pode ser empregado para migrar aplicações legadas. A modernização de sistemas deve garantir maior eficiência, segurança e escalabilidade, sem comprometer a integridade das informações e a continuidade das operações.

Este artigo tem como problemática: Quais os principais desafios enfrentados na migração de sistemas legados?, gerando o objetivo geral analisar os principais desafios enfrentados na migração de sistemas legados para plataformas robustas e como objetivos específicos, assegurar a compatibilidade entre tecnologias antigas e novas plataformas; minimizar a complexidade e os custos associados à manutenção dos sistemas migrados; aprimorar a performance e a escalabilidade dos sistemas para atender às demandas estratégicas da empresa.

2 SISTEMAS LEGADOS

Os sistemas legados surgiram em um contexto de centralização da computação, marcado por mainframes e linguagens de terceira geração como COBOL, FORTRAN e Pascal. No setor bancário, por exemplo, estima-se que mais de 220 bilhões de linhas de código COBOL ainda processem transações globais diariamente (Micro Focus, 2023). Essas aplicações são valiosas porque incorporam regras de negócio estabilizadas ao longo de décadas, mas apresentam desafios: alto custo de licenciamento de *hardware*, dificuldade de integração com APIs REST e escassez de profissionais que dominem suas tecnologias.

De acordo com a pesquisa da Standish Group (2022), 84 % dos projetos de modernização de legados atrasam ou excedem o orçamento, devido justamente à complexidade de desvendar lógica de negócio “embutida” em programas monolíticos. O “efeito espaguete” descrito por Chikofsky e Cross (1990) manifesta-se na prática com acoplamentos circulares: uma única alteração em uma rotina de cálculo de impostos, por exemplo, pode reverberar em módulos de faturamento, estoque e relatórios, causando regressões imprevisíveis.

2.1 Plataformas emergentes

Plataformas emergentes combinam paradigmas de microsserviços, contêineres e baixo código para proporcionar flexibilidade arquitetural e *time-to-market* reduzido. A “Elasticidade Computacional” (Armstrong, 2018) permite ajustar recursos computacionais em tempo real, prática que se tornou essencial diante do aumento de tráfego sazonal em setores como *e-commerce*. O Oracle APEX adiciona uma camada de produtividade ao permitir que analistas de negócio com conhecimentos básicos de SQL criem protótipos funcionais em horas, graças aos assistentes de geração automática de formulários, relatórios interativos e gráficos dinâmicos.

Estudos de caso da Accenture (2024) mostraram que uma empresa de Telecom reduziu de oito para dois meses o lançamento de um portal de autosserviço ao migrar módulos *front-end* para APEX. Além disso, *frameworks* de microsserviços em Java Spring Boot ou Node.js podem coexistir com o APEX, delegando à plataforma as funcionalidades CRUD clássicas enquanto serviços externos lidam com processamento intensivo ou integrações IoT.

2.2 Migração de Sistemas

A literatura por meio de vários auotres propõe diferentes roteiros de migração, entre eles:

• Avaliação de viabilidade – O ponto de partida é inventariar ativos (aplicações, bases de dados, integrações), estimar Retorno Sobre o Investimento (ROI) e ranquear riscos técnicos e regulatórios. Chindanuru (2025) propõe um *framework* que combina métricas quantitativas (TCO, *payback*) a benefícios intangíveis como agilidade e UX, permitindo projeções de valor ao longo da curva de adoção. Complementarmente, Mallidi *et al*. (2021) demonstram que a correlação entre *legacy technical debt* e custos operacionais viabiliza modelos preditivos de economia pós-migração, reforçando a necessidade de um *business case* robusto antes da seleção da plataforma-alvo. Resultados dessa etapa alimentam a matriz de priorização que guiará os demais passos.

• Planejamento modular – Em vez de "transplantes" monolíticos, Wolfart e Assunção (2021) recomendam decompor o domínio em bounded contexts (DDD) e mapear dependências entre módulos para reduzir acoplamento e permitir migração paralela. Estudos práticos indicam que uma abordagem em duas fases, passando do monólito para um monólito modular antes da adoção completa de microserviços, pode reduzir significativamente o esforço de migração, além de favorecer a evolução contínua da base de testes ao longo das iterações.

• Prova de conceito (PoC) – A seleção de um módulo de baixo impacto (por exemplo, relatórios ou autenticação) serve para validar tecnologias, estimar *capacity* e ajustar governança. Tupsakhare (2022) relata que pilotos de 6 a 8 semanas reduzem em 23 % a incerteza de cronograma em programas subsequentes de nuvem. Boas práticas incluem isolar o ambiente de PoC, definir métricas de sucesso (latência, cobertura de testes, custo/hora) e capturar lições aprendidas para retroalimentar o road-map global.

• Refatoração e testes – Após a PoC, inicia-se a engenharia reversa para extrair regras de negócio e remover códigos duplicados. A revisão sistemática de Majeed (2024) divide o processo em três fases: engenharia reversa, alteração controlada e, engenharia direta, destacando a automação de testes de regressão como fator crítico para preservar comportamento funcional. Organizações que alcançam acima de 70 % de cobertura automatizada antes do *cut-over* registram 40 % menos incidentes de pós-produção, segundo a mesma revisão.

• Implantação incremental – A etapa final utiliza CI/CD para liberações contínuas e padrões de substituição gradual. O Strangler Fig Pattern (Fowler, 2017) intercala um *façade* entre cliente e legado, roteando chamadas para novos serviços à medida que são entregues, o que permite reversão rápida e auditoria fina. Estudos de caso recentes mostram que abordagens fatiadas apresentam probabilidade 1,7 vezes maior de cumprir escopo e orçamento que estratégias *big-bang*, reforçando a convergência entre práticas ágeis e governança corporativa na migração para o Oracle APEX.

2.3 Segurança de dados

Com a vigência da Lei Geral de Proteção de Dados (LGPD), o tratamento de dados pessoais exige ampla governança. No APEX, políticas de sessão, Oracle Wallet e Data Safe provêm criptografia, mascaramento e trilhas de auditoria. O Relatório

DBIR (2024) aponta que 82 % das violações em nuvem derivam de credenciais fracas. Implementar DevSecOps com SAST/DAST reduz 58 % do tempo de correção (Costa; Melo, 2020). A ISO/IEC 27018 trata da proteção de dados pessoais em nuvem pública, reforçando cláusulas contratuais com provedores.

O APEX herda controles de banco (VPD, RLS, ACLs) e adiciona camadas específicas para *low-code*. A plataforma já permite autenticação federada (OAuth 2.0, SAML 2.0, OpenID Connect) e suporta MFA FIDO2/WebAuthn quando integrada ao Oracle Cloud IAM eliminando senhas fracas e permitindo *hashes* adaptativos (PBKDF2/BCrypt) para credenciais locais (Fido Alliance, 2024). Para dados sensíveis em trânsito e em repouso, o Oracle Wallet garante TLS mútuo, enquanto o Oracle Data Safe oferece descoberta de PII, mascaramento irreversível em ambientes de teste e trilhas de auditoria centralizadas (Oracle, 2024). Já o SQL Firewall integrado ao Oracle Database 23ai bloqueia instruções suspeitas antes de alcançarem as tabelas de produção, reduzindo dependência de controles na camada de aplicação (Oracle, 2024).

A migração normalmente expõe funcionalidades por meio de ORDS (REST) ou módulos GraphQL, convertendo consultas SQL em *endpoints* públicos. A OWASP API Security Top 10 (2023) coloca Broken Object Level Authorization (API1) e Broken Authentication (API2) como riscos mais prevalentes, reflexo de falhas de controle de acesso e sessão (OWASP, 2023). No APEX, cada rota ORDS deve exigir JWT de curta duração, rate-limits e escopos mínimos; no banco, ACLs e RESOURCE LIMIT restringem abuso de CPU e conexões.

2.4 Gestão de Tecnologia da Informação

Uma governança sólida é importante. O COBIT 2019 alinha metas corporativas a objetivos de TI; ITIL 4 auxilia a estimar custo total de propriedade. Pressman (2016) indica SLAs progressivos durante transição. *Change Management* de Kotter (2017) recomenda senso de urgência, coalizões de patrocinadores e *quick wins* para mitigar resistência.

Em primeiro lugar, a migração para o Oracle APEX deve nascer dentro de uma arquitetura de governança que combine ISO/IEC 38500 (com seus seis princípios de direcionamento e monitoramento) ao COBIT 2019, que mapeia metas corporativas a objetivos de TI via *alignment goals* e estabelece indicadores para acompanhar valor e risco ao longo do *road-map* de modernização. Essa sobreposição garante que decisões de portabilidade, escolha de *cloud* e desenho de APIs APEX se mantenham coerentes com a estratégia empresarial, não apenas com requisitos técnicos.

Na esfera operacional, o ITIL 4 Service Value System oferece métricas de total *cost of ownership* (TCO) que orientam estimativas de OPEX em bancos *Autonomous*, licenças de ORDS e suporte — custeios que, se não dimensionados, transferem dívida técnica do legado para a nova plataforma. O modelo de Valor em Serviço e a *Service Value Chain* permitem projetar, para cada *backlog* de migração APEX, KPIs de disponibilidade e capacidade vinculados a acordos de nível de serviço dinâmicos, revistos a cada *sprint* de entrega.

Contudo, nenhuma iniciativa de modernização prospera sem gestão de mudança. Pressman (2016) recomenda SLAs progressivos, iniciando com níveis "bronze" que espelham o desempenho legado e evoluindo a patamares "*gold*" pós-refatoração a fim de calibrar expectativa dos *stakeholders* e reduzir risco de corte brusco de serviço. Em paralelo, o modelo de oito etapas de Kotter (2017) sublinha

senso de urgência, criação de coalizões patrocinadoras e comunicação de *quick wins* para neutralizar resistência cultural típica de times que cuidam do sistema antigo. Esses elementos complementam COBIT/ITIL ao tratar da dimensão humana da transição, assegurando que o legado seja desativado somente quando valor e confiança já forem tangíveis na esteira APEX.

3 PROCEDIMENTOS METODOLÓGICOS

Este estudo adotou uma abordagem qualitativa, de caráter descritivo, com o objetivo de compreender e analisar os principais desafios enfrentados por empresas durante o processo de migração de sistemas legados para plataformas mais robustas. A pesquisa qualitativa permite uma investigação aprofundada dos fenômenos observados, priorizando a compreensão do contexto e das experiências dos envolvidos. Tal delineamento é indicado quando se pretende analisar processos, contextos e experiências em profundidade (Merriam; Tisdell, 2016).

Para isso, foram realizadas análises documentais e revisão bibliográfica. A revisão bibliográfica abrangeu artigos científicos, livros, relatórios técnicos e estudos de caso relevantes sobre o tema, buscando identificar padrões, boas práticas e dificuldades recorrentes. (Lakatos; Marconi, 2017).

Os dados coletados através das análises documentais e revisão bibliográfica foram analisados de forma interpretativa, com foco na identificação de fatores críticos, estratégias adotadas e resultados obtidos pelas organizações (Bardin, 2011). Esta abordagem visa fornecer uma visão abrangente sobre o tema, contribuindo para a elaboração de diretrizes que possam apoiar futuras migrações em ambientes corporativos.

4 RESULTADOS E DISCUSSÕES

4.1 Estudo de caso: avaliando caminhos para a migração

Modernizar um sistema que funcionava há anos em Delphi foi um dos principais desafios desta pesquisa. O objetivo era levar um conjunto de aplicações administrativas para um ambiente *web* moderno, com melhor integração e facilidade de manutenção. Para isso, foi necessário investigar várias alternativas e entender como cada uma delas se encaixaria na realidade do sistema existente.

Durante essa investigação, foram mapeados critérios básicos como: manter a compatibilidade com o banco de dados já em uso, não exigir um grande esforço de aprendizagem por parte dos desenvolvedores, garantir que o novo sistema pudesse crescer com a empresa e, por fim, respeitar o orçamento disponível. Esses critérios ajudaram a filtrar as opções tecnológicas mais promissoras.

Em um primeiro momento, foram estudadas combinações populares como React ou Angular com Node.js, além de plataformas como Vue.js, Spring Boot, Django, Ruby on Rails e o Oracle APEX. Todas elas têm pontos fortes e fracos: algumas oferecem alta performance, outras possuem comunidades grandes ou ferramentas completas, mas em muitos casos a curva de aprendizado seria longa ou a configuração inicial muito complexa. A necessidade de manter a equipe produtiva e de minimizar custos acabou pesando contra as alternativas mais sofisticadas.

4.2 Escolha de uma plataforma *low-code*

Para tomar a decisão final, foi criado um pequeno projeto piloto (prova de conceito), que permitiu medir o desempenho, a facilidade de uso e o custo de cada alternativa. Ao comparar os resultados, ficou evidente que o Oracle APEX combinado com PL/SQL atendia melhor aos critérios definidos. Além de se integrar nativamente ao banco de dados já utilizado, a plataforma permite construir aplicações *web* de forma declarativa, com menos código e maior rapidez. A familiaridade da equipe com SQL e PL/SQL reduziu bastante a curva de aprendizagem, e como a organização já possuía licenças Oracle, os custos extras foram mínimos. A possibilidade de criar diferentes workspaces isolados dentro do APEX também facilita organizar os módulos e escalar o desenvolvimento.

A solução adotada distribui responsabilidades de maneira clara. A lógica de negócios permanece no banco, escrita em PL/SQL, que também expõe serviços em forma de APIs RESTful. As *interfaces web* são construídas no Oracle APEX, uma plataforma *low-code* que acelera a criação de telas, formulários e relatórios.

Tecnologias como JavaScript, HTML e CSS complementam o APEX, permitindo incluir comportamentos interativos e aparência mais atraente. Para aplicações móveis, foi adotado o Flutter, que consome as APIs REST e possibilita a criação de *apps* para Android e iOS a partir de uma única base de código. Essa arquitetura unifica o acesso aos dados, facilita a integração entre módulos antigos e novos e permite que os usuários acessem o sistema em diferentes dispositivos, sem comprometer a segurança e a consistência das informações.

4.3 Construção de componentes e frameworks internos

Para diminuir a quantidade de código duplicado e simplificar a manutenção, a equipe decidiu criar uma biblioteca de componentes reutilizáveis dentro do APEX. Essa iniciativa teve como meta organizar o desenvolvimento de forma padronizada e segura, adotando boas práticas como documentação, testes e modularidade.

O trabalho começou com a identificação de funcionalidades que se repetem em várias partes do sistema. Com essa lista em mãos, foram desenhados pequenos módulos independentes, chamados de *plugins*, que poderiam ser combinados entre si. Os projetos seguiram padrões de *design* consagrados, como Singleton e Factory, para garantir que cada peça pudesse ser reutilizada sem causar conflitos.

Entre os componentes criados destacam-se: formulários dinâmicos configuráveis conforme as necessidades de cada área do negócio; gráficos interativos apoiados em bibliotecas como Chart.js e D3.js, que permitem visualizar indicadores em tempo real; e tabelas dinâmicas com paginação, ordenação e filtros, acessando dados de forma assíncrona e podendo exportar resultados para diferentes formatos. Para garantir a qualidade desses componentes, foram elaborados testes unitários e adotadas ferramentas de integração contínua, o que ajuda a detectar erros rapidamente. Além disso, cada módulo possui documentação própria, com exemplos de uso e instruções de integração.

4.4 *Framework* web e mobile

Também foi desenvolvido dois pequenos *frameworks*: um para aplicações web e outro para *apps* móveis. O *framework web* foi dividido em camadas: uma camada de apresentação com *templates* e estilos, outra com a lógica de negócios em JavaScript e PL/SQL, e uma terceira camada voltada ao acesso a dados. Essa organização ajuda a separar responsabilidades e permite que diferentes desenvolvedores trabalhem em partes distintas sem interferir uns nos outros.

O *framework mobile* foi construído com Flutter. Ele fornece *widgets* personalizados, temas e regras de navegação para que os aplicativos tenham aparência e funcionamento uniformes. A comunicação com o *backend* se dá por meio de serviços RESTful, aproveitando a infraestrutura de APIs já implementada. Testes automatizados e um guia de uso facilitam a entrada de novos colaboradores. A criação desses *frameworks* contribuiu para aumentar a produtividade e diminuir erros, pois sempre que surge uma nova necessidade, o desenvolvedor verifica se já existe um componente pronto antes de escrever algo do zero.

4.5 Principais desafios e soluções encontradas

Para responder à pergunta de pesquisa, quais são os principais desafios enfrentados na migração de sistemas legados, foi realizado um levantamento dos obstáculos que surgiram ao longo do projeto. Os principais pontos identificados foram:

• Manter a compatibilidade com o legado: foi preciso garantir que dados e regras de negócio pudessem ser reaproveitados na nova plataforma. A escolha do APEX, que conversa diretamente com o banco Oracle, facilitou essa integração e permitiu realizar a transição gradualmente.

• Aprender novas tecnologias: ferramentas modernas podem exigir conhecimentos que a equipe ainda não possui. Ao priorizar uma solução próxima da realidade dos desenvolvedores, como o APEX com PL/SQL, o tempo de adaptação foi reduzido.

• Desempenho e crescimento: o novo sistema precisava suportar cargas maiores e continuar rápido. Testes com a solução escolhida mostraram que o APEX e o PL/SQL têm desempenho sólido e permitem criar diferentes espaços de trabalho, garantindo escalabilidade.

• Custos de implantação: licenças, infraestrutura e horas de desenvolvimento impactam o orçamento. Ao aproveitar a infraestrutura já existente e adotar uma plataforma *low-code*, os gastos extras foram minimizados.

• Suporte e evolução: contar com uma comunidade ativa e suporte oficial foi importante para resolver dúvidas e imprevistos. A Oracle oferece documentação ampla e uma comunidade de usuários que compartilha soluções, o que deu mais segurança à equipe.

• Reutilização de código: sistemas antigos muitas vezes acumulam trechos redundantes. A criação de componentes e *frameworks* internos, com testes e documentação, facilitou a manutenção e reduziu a chance de erros futuros.

As soluções aplicadas em cada um desses pontos permitiram atender diretamente aos objetivos específicos definidos no início da pesquisa. A escolha de uma plataforma que se integra ao banco de dados legado e possibilita migração gradual contribuiu para assegurar a compatibilidade entre tecnologias antigas e novas.

A reutilização de componentes, a adoção de uma solução *low-code* e o aproveitamento da infraestrutura existente ajudaram a minimizar a complexidade e os custos de desenvolvimento e manutenção. Finalmente, os testes de carga e a organização em *workspaces* demonstraram que a plataforma escolhida oferece boa performance e capacidade de escalabilidade, atendendo às demandas estratégicas da empresa.

5 CONCLUSÃO

Ao longo deste trabalho, ficou evidente que a escolha da tecnologia representa apenas uma parte do desafio em um processo de migração bem-sucedido. O que realmente faz diferença é entender como uma plataforma se encaixa no contexto específico da organização e na realidade do banco de dados existente. No caso estudado, a adoção de uma solução *low-code* integrada ao banco já utilizado trouxe benefícios significativos: simplificou a transição, reduziu custos e facilitou o treinamento da equipe, que conseguiu aproveitar conhecimentos prévios.

A criação de componentes reutilizáveis e *frameworks* internos também se mostrou essencial. Esta experiência deixou claro que modernizar um sistema não depende apenas de ter acesso às ferramentas mais modernas, mas sim de um planejamento bem estruturado, investimento na capacitação da equipe e atenção à documentação. A estratégia adotada permitiu alcançar os objetivos traçados: garantir compatibilidade entre legado e novas tecnologias, diminuir complexidade e custos de manutenção, além de melhorar desempenho e escalabilidade.

Por outro lado, é necessário reconhecer as limitações deste estudo. Por se tratar de uma pesquisa focada em viabilidade e planejamento, não foi possível realizar testes práticos em ambiente de produção. As conclusões apresentadas baseiam-se em análises comparativas, revisão da literatura e protótipos desenvolvidos em escala reduzida. A falta de validação em ambiente real deixa algumas questões em aberto, especialmente quanto ao desempenho efetivo, ao retorno financeiro concreto e à aderência às rotinas operacionais do dia a dia.

Para trabalhos futuros, seria interessante conduzir experimentos de migração em escala real, com coleta de métricas de desempenho e custos após a implantação completa do sistema. Outra possibilidade promissora seria explorar abordagens arquiteturais diferentes, como a implementação de microsserviços, o que poderia ampliar ainda mais a flexibilidade e escalabilidade das aplicações. Esses próximos passos ajudariam a consolidar os resultados apresentados aqui e trariam insights adicionais sobre os desafios práticos da modernização de sistemas legados.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, P.; LIMA, F. Panorama da modernização de sistemas no Brasil. **Revista de Sistemas de Informação,** 2023.

ACCENTURE. **Technology Vision 2024:** Human by Design – How AI Unleashes the Next Level of Human Potential. Accenture. 2024. Disponível em: https://www.accenture.com/us-en/insights/technology/technology-trends-2024. Acesso em: 20 ago. 2025.

ARMSTRONG, J. Elastic Computation Theory. **ACM Computing Surveys**, 2018.

BARDIN, L. **Análise de Conteúdo**. 2011.

CHINDANURU, R. Modernizing Legacy Applications: Strategies and Emerging Trends. International **Journal of Information Technology & Management Information System** (IJITMIS), v.16, n.1, p.143–158. 2025. Disponível em: https://iaeme.com/MasterAdmin/Journal_uploads/IJITMIS/VOLUME_16_ISSUE_1/IJITMIS_16_01_012.pdf. Acesso em: 20 ago. 2025.

CHIKOFSKY, E.; CROSS, J. IEEE. **Software.** 1990.

COSTA, R.; MELO, P. **Anais do Congresso Brasileiro de Informática**, USP, 2020.

FIDO ALLIANCE. **Online Authentication Barometer**. 2024. Disponível em: https://fidoalliance.org/wp-content/uploads/2024/10/Barometer-Report-2024-Oct-29.pdf. Acesso em: 20 ago. 2025.

FOWLER, M. **Strangler Fig Application**, 2017.

ISO/IEC 27018:2019. **Information technology – Security techniques**. Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors. 2019. International Organization for Standardization/International Electrotechnical Commission, 2019. Disponível em: https://www.iso.org/standard/76559.html. Acesso em: 28 ago. 2025.

KOTTER, J. P. **The Problem With Data**. Kotter International. 2017. Disponível em: https://www.kotterinc.com/wp-content/uploads/2017/11/The-Problem-With-Data-Kotter-2017.pdf. Acesso em: 20 set. 2025.

LAKATOS, E. M.; MARCONI, M. de A. **Fundamentos de metodologia científica**. São Paulo: Atlas, 2017.

MALLIDI, R. K.; SHARMA, M.; SINGH, J. **Legacy Digital Transformation:** TCO and ROI Analysis. 2021.

MICRO FOCUS. **COBOL Survey Report**. 2023. Disponivel em fevereiro de 2022.

ORACLE Corporation. **Documentation Release 24.1**. 2024. Disponível em: https://docs.oracle.com/en/database/oracle/apex/24.1/htmrn/about-release-notes.html#GUID-175AC460-3BCB-4871-A14C-5A232EDF8CAD. Acesso em: 27 set. 2025.

PRESSMAN, R. **Engenharia de Software**. 2016.

SOMMERVILLE, I. **Engenharia de Software**. 2019.

STANDISH Group. **Chaos Report**. 2022. Disponível em: https://www.standishgroup.com/. Acesso em: 24 set. 2025.

TUPSAKHARE, P. **Strategies for Legacy Application to Cloud Migration:** Navigating Challenges and Maximizing Benefits. 2022.