Pró-reitoria de Pesquisa e Pós-Graduação
INSTITUTO FEDERAL São Paulo

16º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2025

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES PELO USO DE FERRAMENTA LOW-CODE E A APLICAÇÃO DE FRAMEWORKS

ARTHUR VILAR ALVES[1], EROS DE ASSIS BRITO[2], OSVANDRE A. MARTINS[3]

[1] Aluno do curso de Bac. em Sistemas de Informação, IFSP, Campus Votuporanga, arthur.alves@aluno.ifsp.edu.br.
[2] Aluno do curso de Bac. em Sistemas de Informação, IFSP, Campus Votuporanga, eros.assis@aluno.ifsp.edu.br.
[3] Professor do Ensino Básico, Técnico e Tecnológico, IFSP, Campus Votuporanga, osvandre@ifsp.edu.br.

Área de conhecimento (Tabela CNPq): 1.03.03.00-6 - Metodologias e Técnicas da Computação.

RESUMO: No desenvolvimento de software, a produtividade e a qualidade das soluções representam fatores críticos de sucesso. Deste fato emerge o desafio de elevar a produtividade sem comprometer a qualidade e possíveis formas de enfrentamento são representadas pela adoção do paradigma RAD (*Rapid Application Development*), de *frameworks*, de ferramentas *Low-Code* e *No-Code*. Neste contexto, encontra-se em andamento um trabalho que visa explicitar a relação entre esses elementos, ou seja, como o emprego de um implica no emprego do outro no desenvolvimento de software. Para tanto, realiza-se uma revisão teórica sobre RAD, ferramentas *Low-Code* e *frameworks* Java, seguida de um estudo de caso de desenvolvimento de software para a *Web*, buscando conhecer os benefícios e as limitações da adoção dessa abordagem e oferecer subsídios para possíveis aplicações práticas em casos reais de projeto e construção de software. Apresentam-se, portanto, os resultados parciais e os próximos passos planejados ao desenvolvimento deste trabalho no campo de Sistemas de Informação.

PALAVRAS-CHAVE: desenvolvimento de software; RAD; ferramentas *low-code*; aplicação de *frameworks*.

RAPID APPLICATION DEVELOPMENT BY APPLYING LOW-CODE TOOLS AND FRAMEWORKS

ABSTRACT: In software development, productivity and solution quality represent critical success factors. This raises the challenge of increasing productivity without compromising quality. Possible approaches to addressing this challenge include adopting the RAD (Rapid Application Development) paradigm, frameworks, and low-code and no-code tools. In this context, work is underway to clarify the relationship between these elements, that is, how the use of one implies, the use of the other in software development. To this end, a theoretical review of RAD, low-code tools, and Java frameworks is conducted, followed by a case study of web software development. This study seeks to understand the benefits and limitations of adopting this approach and provide insights for possible practical applications in real-world software design and development. Therefore, the partial results and the next steps planned for the development of this work in the field of Information Systems are presented.

KEYWORDS: software development; RAD; low-code tools; frameworks application.

INTRODUÇÃO

O desenvolvimento de sistemas de software tem se consolidado como elemento central para a transformação digital e a inovação nas organizações. A crescente complexidade das demandas e a pressão por prazos cada vez mais curtos impulsionam a adoção de metodologias que privilegiem flexibilidade e rapidez nas entregas. Modelos tradicionais, como o cascata, ainda são utilizados, mas apresentam limitações significativas diante de contextos de constante mudança (Noleto, 2020). Nesse cenário, o paradigma *Rapid Application Development* (RAD) surge como alternativa, oferecendo um processo iterativo, com prototipação rápida e envolvimento direto dos usuários finais (Beynon-Davies *et al.*, 1999). Ao encurtar ciclos de desenvolvimento e permitir ajustes contínuos, o RAD contribui para reduzir riscos e aumentar as chances de sucesso em projetos de software.

Em suporte à aplicação do paradigma RAD, principalmente no tocante a questões de implementação, surgem as ferramentas *Low-Code* (pouca codificação). Segundo Hurlburt (2021), elas têm conquistado espaço por possibilitar que aplicações grandes e complexas sejam desenvolvidas exigindo cada vez menos esforços de codificação manual. Neste sentido, Rokis e Kirikova (2022) complementam que elas oferecem interfaces gráficas e componentes reutilizáveis aos desenvolvedores, favorecendo o envolvimento de não-programadores no processo de desenvolvimento de software. A redução da necessidade de codificação manual veio avançando ao longo dos anos até chegar ao ponto de surgirem ferramentas classificadas como *No-Code* (nenhuma codificação).

Essas ferramentas agem, de certa forma, como intérpretes para uma linguagem de programação como Java, por exemplo, amplamente adotada no meio corporativo devido a um histórico de indícios de estabilidade, segurança e escalabilidade. Ainda segundo Hurlburt (2021), ao empregar uma ferramenta *Low-Code* ou *No-Code* o desenvolvedor expressa suas necessidades e vontades não por comandos complexos, mas de forma mais simples e intuitiva, arrastando e soltando componentes visuais, enquanto a ferramenta se encarrega de traduzir essas expressões conforme a sintaxe e estruturação de códigos exigida pela linguagem de programação. No final das contas se obtém códigos para compilar e executar, mas, no caso, a sua construção foi favorecida por automações.

Ainda sobre recursos viabilizadores de RAD notam-se os *frameworks*, citados por Schmidt, Gokhale e Natarajan (2004) como conjuntos integrados de artefatos de software que colaboram para prover uma arquitetura reutilizável por famílias de aplicativos. Sua estrutura busca favorecer a produtividade no desenvolvimento de software, pois já traz consigo uma série de soluções para problemas conhecidos pela comunidade, proporcionando reuso de desenho, implementação e validação.

Nota-se, portanto, uma relação estrita entre ferramentas *Low-Code* ou *No-Code*, *frameworks* e o paradigma RAD, mas como essa relação acontece e vem se intensificando na prática? Para esclarecer esse fato, encontra-se em andamento um trabalho que visa demonstrar o emprego de *frameworks* por meio do uso de uma ferramenta *Low-Code*, elucidando como esses conceitos e tecnologias se relacionam no intuito de acelerar o processo de criação de aplicações e aumentar a produtividade dos desenvolvedores. De maneira específica, este trabalho busca reunir conhecimentos acerca de conceitos e tecnologias, e por meio de um estudo de caso, explorar a aplicação prática deles, para verificar como essa combinação efetivamente se mostra capaz de acelerar o desenvolvimento, promovendo também a qualidade.

MATERIAIS E MÉTODOS

Com base em definições apontadas por Gil (2008) e por Lakatos e Marconi (2003), o trabalho em andamento se classifica como de pesquisa aplicada, qualitativa, exploratória e descritiva. No tocante a procedimentos, este lança mão do desenvolvimento de um estudo de caso.

Assim, o processo metodológico estabelecido envolve pesquisa bibliográfica, estudo e exploração de conceitos, métodos e tecnologias por meio da realização de um estudo de caso de desenvolvimento de software.

Para tanto, considera-se o modelo iterativo e incremental citado por Pressman (2011) e Beynon-Davies *et al.* (1999), bem como as fases do processo RAD, aplicadas de forma adaptada em respeito a limitações de tempo e de escopo relativas a um Trabalho de Conclusão de Curso.

A realização do estudo de caso de desenvolvimento de software, representa uma forma de se ter contato com conceitos, métodos, técnicas e tecnologias associadas a RAD, *frameworks* e ferramenta *Low-Code*. Neste sentido, cita-se o emprego dos seguintes recursos tecnológicos:

- Plataforma RAD *Low-Code* - ferramenta visual para o desenvolvimento rápido de aplicações, com recursos de modelagem, geração automática de código, gerenciamento de entidades e construção de interfaces;
- *Frameworks* baseados na linguagem Java que forneçam estrutura para implementação das porções *back end* e *front end* de aplicativos corporativos;
- SGBD (Sistema Gerenciador de Banco de Dados) relacional como parte da porção *back end* responsável pela persistência de dados;
- ferramentas de implementação de repositório de artefatos versionados (documentos, códigos-fonte e outros recursos de implementação), provendo controle de versões, histórico de mudanças e documentação de progresso.

Representantes desses recursos são escolhidos com base no levantamento bibliográfico e em critérios como compatibilidade, produtividade, documentação disponível e aderência aos princípios de modularidade, reuso e desenvolvimento incremental.

Estrategicamente, definiu-se o tipo de software para o estudo de caso, com base nos resultados de um trabalho realizado por Fanelli (2023). Trata-se de um software cuja especificação, análise, desenho e construção resultaram em um aplicativo em versão desktop que exige instalação na estação de trabalho do usuário. A missão para o referido estudo de caso se refere a produzir uma versão mais robusta deste software, capaz de ser acessada pela *Web*. Durante esta produção, coletam-se e registram-se informações relativas ao desempenho e à produtividade, conforme a percepção dos desenvolvedores, viabilizando a realização de análises e elaboração de relatos da experiência.

RESULTADOS E DISCUSSÃO

A revisão bibliográfica realizada proporcionou a obtenção de conhecimentos fundamentais sobre conceitos associados: a agilidade no desenvolvimento de software; ao paradigma RAD, o ciclo de vida de projeto de software neste modelo e as limitações dessa abordagem; às ferramentas *Low-Code*; e *Frameworks* populares na linguagem Java.

Destaca-se que a pesquisa e o estudo acerca das ferramentas *Low-Code* e *Frameworks* se mostraram cruciais para a seleção de tecnologias aplicáveis e viabilizadoras do estudo de caso planejado. Cita-se a seleção de uma ferramenta denominada *JMIX Studio* (JMIX.IO, 2025) que promete acelerar o desenvolvimento de sistemas para a *Web*, primando pelo emprego de *Frameworks* para atender a requisitos de segurança, robustez e escalabilidade. Neste sentido, constatou-se que ela aplica, exigindo pouco domínio do programador, um *Framework* próprio, constituído a partir do uso integrado de outros dois produzidos por terceiros e que são amplamente aceitos pela comunidade de desenvolvedores Java. Tratam-se do *Spring Boot* (Spring.io, 2025) e do *Vaadin* (Vaadin, 2025).

Constatou-se, a partir de relatos de Boaglio (2017), que o *Spring Boot* se posiciona além de um simples *Framework*, pois sua estrutura facilita questões de acesso a banco de dados, implementação de filas de execução, serviços *Web* REST (*Representational State Transfer*) consumíveis por tecnologias de *front end* (interfaces de

usuário), entre outros. Por falar em tecnologias de *front end*, o *Vaadin* possibilita a criação de interfaces de usuário ricas e interativas, a partir de componentes reutilizáveis para a digitação de datas, textos, caixas de seleção e de combinação, tabelas para dados, gráficos e outros comumente empregados pelos desenvolvedores.

Completando a seleção de tecnologias do estudo de caso, citam-se um Sistema Gerenciador de Banco de Dados relacional (*PostgreSQL*), o sistema de controle de versão distribuído *Git* e a plataforma de hospedagem de repositórios *GitHub*, utilizada para gerenciar e compartilhar os artefatos de desenvolvimento.

A ferramenta *JMIX Studio* é disponibilizada como uma extensão para o IDE (*Integrated Development Environment*) IntelliJ IDEA (JetBrains, 2025), possuindo licença de uso gratuita (*Community*) que, apesar de limitada em algumas funções, apresentou indícios de suporte adequado à realização do estudo de caso. Constatam-se também outros tipos de licença como a *Academic*, gratuita para instituições de ensino, e algumas específicas para profissionais autônomos e empresas.

Uma vez obtidos os conhecimentos básicos e fundamentais em RAD e selecionados os recursos tecnológicos necessários, na forma de ferramenta *Low-Code* e *Frameworks*, deu-se início à realização do estudo de caso, produzindo artefatos de especificação e de análise de sistemas. Com base nas versões produzidas por Fanelli (2023), modelos em UML (*Unified Modeling Language*) (Booch; Rumbaugh; Jacobson, 2005) foram trabalhados para se obter o desenho de um projeto de banco de dados e os desenhos de interfaces gráficas de usuário para a Web.

De posse desses modelos, ferramenta *Low-Code* foi explorada no sentido de dar suporte aos aspectos de prototipagem em RAD. Constataram-se então indícios de eficiência da ferramenta no tocante a agilizar o desenvolvimento de sistemas de uso corporativo com reuso prático e fácil de funcionalidades associadas à autenticação, autorização, implementação de menu de opções e, principalmente, operações CRUD (*Create, Read, Update, and Delete* com persistência em banco de dados. No caso, acredita-se que o projeto e a implementação de tais funcionalidades certamente demandariam muito mais esforço sem a aplicação dessa tecnologia associada à RAD.

Outro ganho percebido foi a integração simplificada entre as camadas de persistência e de interfaces de usuário. Recursos como *Data Stores*, entidades e *View Models* permitiram criar rapidamente formulários e tabelas dinâmicas sem a necessidade de escrever mapeamentos complexos. Tal fato trouxe a percepção imediata de agilidade no desenvolvimento e, consequentemente, mais confiança para se avançar no uso da ferramenta e no desenvolvimento.

Por outro lado, nem tudo tem se mostrado simples. A curva de aprendizagem da ferramenta *Low-Code* JMIX, especialmente para personalizações mais avançadas, mostrou-se consideravelmente lenta, dado o volume de recursos e funcionalidades disponíveis. Além disso, como a ferramenta dispõe de recursos e padrões próprios, acredita-se na possibilidade de haver limitações na capacidade de atender requisitos de projetos que exijam um nível muito alto de personalização ou integrações muito específicas. Mesmo assim, os benefícios percebidos até o momento superam as impressões negativas e serão devidamente pontuados na documentação final dos trabalhos.

A experiência realizada até então, proporcionou a obtenção de fortes indícios de que a relação entre RAD, ferramentas *Low-Code* e *frameworks* se mostra evidentemente próxima, não apenas conceitualmente, mas também em termos de aplicação prática, indicando que a estratégia se mostra promissora para o desenvolvimento de sistemas corporativos com mais agilidade e qualidade.

Salienta-se que, no período de produção deste relato, o desenvolvimento do estudo de caso se encontra em torno de 40%. Ainda se faz necessário concluir a implementação do protótipo pretendido e submetê-lo a testes de integração seguidos da análise dos artefatos produzidos automaticamente pela ferramenta *Low-Code* em relação aos *frameworks* envolvidos. Contudo, notam-se indícios de que essa combinação aparentemente

consegue equilibrar rapidez de entrega e boas práticas de engenharia de software, algo que é cada vez mais valorizado no universo de desenvolvimento de sistemas de software.

CONCLUSÕES

Este documento apresentou informações acerca de uma ação associada à pesquisa aplicada e desenvolvimento de tecnologia, focada na demonstração prática da relação entre o paradigma RAD, o emprego de *frameworks* e o uso de *Low-Code*. Trata-se de um conjunto de conceitos, métodos, técnicas e tecnologias que vem revolucionando a área de desenvolvimento de software, promovendo a criação, cada vez mais eficaz, de soluções em software, principalmente para o ambiente corporativo.

As percepções iniciais, mesmo que ainda obtidas de forma empírica, indicam que a estratégia de apoiar o desenvolvimento de soluções nos pilares do paradigma e nas tecnologias em RAD podem favorecer a obtenção de ganhos em tempo e produtividade, sem comprometer a qualidade e a escalabilidade do sistema.

CONTRIBUIÇÕES DOS AUTORES

O.A.M. contribuiu com a definição do tema e do escopo, bem como com a orientação do trabalho. A.V.A. e E.A.B. se encarregaram da realização das atividades de pesquisa, estudo, exploração, registros, análises e produção de relatos associados. Todos os autores contribuíram com a revisão do trabalho e aprovaram a versão submetida.

REFERÊNCIAS

BEYNON-DAVIES, P. *et al.* Rapid application development (rad): an empirical review. **European Journal of Information Systems**, Taylor & Francis, v. 8, n. 3, p. 211–223, 1999. Disponível em: <https://doi.org/10.1057/palgrave.ejis.3000325>. Acesso em: 15 jul. 2025.

BOAGLIO, F. **Spring Boot: acelere o desenvolvimento de microsserviços**. [S.l.]: Casa do Código, 2017. 222 p.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML - Guia do Usuário**. 2. ed. [S.l.]: Campus, 2005. 176 p.

FANELLI, G. M. **Desenvolvimento de um Sistema RFID para Gestão de Bens Patrimoniais**. Trabalho de Conclusão de Curso (Graduação em Bel. em Eng. Elétrica) - IFSP, Votuporanga, 2023.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 6. ed. [S.l.]: Atlas, 2008. 220 p.

HURLBURT, G. F. Low-code, no-code, what's under the hood? **IT Professional**, v. 23, n. 6, p. 4–7, 2021.

JETBRAINS. **IntelliJ IDEA - The Leading IDE for Professional Development in Java and Kotlin**. 2025. Disponível em: <https://www.jetbrains.com/idea/>. Acesso em: 10 jul. 2025.

JMIX.IO. **Rapid Web App Dev Platform for Java Developers**. 2025. Disponível em: <https://www.jmix.io>. Acesso em: 12 jul. 2025.

LAKATOS, E. M.; MARCONI, M. de A. **Fundamentos de Metodologia Científica**. 5. ed. [S.l.]: Atlas, 2003. 310 p.

NOLETO, C. Modelo cascata: o que é e por que está ultrapassado? **Blog Be Trybe**, betrybe, 2020. Disponível em: <https://blog.betrybe.com/tecnologia/modelo-cascata/>. Acesso em: 20 jul. 2025.

PRESSMAN, R. S. **Engenharia de software: Uma Abordagem Profissional**. [S.l.]: Grupo A - AMGH, 2011.

ROKIS, K.; KIRIKOVA, M. Challenges of low-code/no-code software development: A literature review. In: NAZARUKA, Ē.; SANDKUHL, K.; SEIGERROTH, U. (Ed.). **Perspectives in Business Informatics Research**. Cham: Springer International Publishing, 2022. p. 3–17. ISBN 978-3-031-16947-2.

SCHMIDT, D. C.; GOKHALE, A.; NATARAJAN, B. Leveraging application frameworks: Why frameworks are important and how to apply them effectively. **Queue**, Association for Computing Machinery, New York, NY, USA, v. 2, n. 5, p. 66–75, jul. 2004. ISSN 1542-7730. Disponível em: <https://doi.org/10.1145/1016998.1017005>. Acesso em: 11 jul. 2025.

SPRING.IO. Spring data2025.0.0. **SpringData**, 2025. Disponível em: <https://spring.io/projects/spring-data>. Acesso em: 17 jul. 2025.

VAADIN. Vaadin docs. **Vaadin**, 2025. Disponível em: <https://vaadin.com/docs/latest/>. Acesso em: 15 jul. 2025.