Aplicação de atividades práticas para o ensino de conceitos de Engenharia de Software com o uso de ferramentas empregadas na indústria

Jean Francisco da Silva[1], **Hiran Nonato M. Ferreira**[1], **Wedson G. da Silveira Júnior**[1], **Vinícius Alves Silva**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais (IFSULDEMINAS) - Passos/MG, Brasil

jean.francisco@alunos.ifsuldeminas.edu.br
{hiran.ferreira, wedson.junior, vinicius.silva}@ifsuldeminas.edu.br

Abstract. *Software Engineering courses often face a mismatch between classroom content and industry practices, resulting in a skill gap that impacts students' professional preparedness. This paper reports on the application of a sequence of practical activities in a Software Engineering course of a Computer Science undergraduate program. The intervention, conducted by the instructor with the support of a teaching assistant, comprised: (i) the development of a simple application with Create, Read, Update, Delete (CRUD) operations; (ii) the implementation of automated tests using different frameworks; and (iii) the configuration of Continuous Integration (CI) pipelines with GitHub Actions. Voluntary online questionnaires were applied at the end of each activity to capture students' perceptions. Results suggest that the proposal helped bridge theoretical content and practical software development scenarios, while also revealing difficulties in using DevOps tools, especially GitHub Actions. The paper discusses these findings, limitations, and lessons learned, providing insights for instructors interested in similar initiatives.*

Resumo. *Cursos de Engenharia de Software ainda enfrentam um descompasso entre os conteúdos trabalhados em sala de aula e as práticas adotadas na indústria, configurando um skill gap que afeta a formação dos estudantes. Este artigo apresenta um relato de experiência sobre a aplicação de uma sequência de atividades práticas em uma disciplina de Engenharia de Software de um curso de Bacharelado em Ciência da Computação. A intervenção, conduzida pelo professor com apoio de um monitor, envolveu: (i) o desenvolvimento de uma aplicação simples com operações Create, Read, Update, Delete (CRUD); (ii) a implementação de testes automatizados com diferentes frameworks; e (iii) a configuração de pipelines de Integração Contínua com GitHub Actions. Questionários on-line, de resposta voluntária, foram aplicados ao final de cada atividade para registrar a percepção dos estudantes. Os resultados indicam que a proposta aproximou conteúdos teóricos de situações práticas de desenvolvimento, mas também evidenciou dificuldades no uso de ferramentas de DevOps, especialmente GitHub Actions. O artigo discute essas evidências, limitações e lições aprendidas, oferecendo subsídios para docentes interessados em iniciativas semelhantes.*

1. Introdução

O crescimento acelerado do mercado de Tecnologia da Informação (TI) tem ampliado a demanda por profissionais qualificados e impulsionado a oferta de cursos superiores na área de Computação. Diferentes estudos e relatórios de mercado apontam que a procura por talentos em TI segue em alta, impulsionada pela transformação digital de diversos setores da economia e pela necessidade de modernização de processos organizacionais [Moreira 2024, Valença et al. 2023]. Ao mesmo tempo, autores têm questionado a capacidade dos modelos tradicionais de educação superior em TI de responder adequadamente a esse cenário, indicando sinais de desalinhamento entre formação acadêmica e demandas profissionais [Goulart 2019].

No contexto específico da Engenharia de Software, esse desalinhamento costuma ser descrito como um *skill gap* entre as competências desenvolvidas na universidade e aquelas esperadas pela indústria. Estudos de mapeamento e revisões sistemáticas têm apontado que, embora os currículos de Engenharia de Software abordem tópicos centrais da área, muitas vezes os estudantes concluem a graduação com pouca experiência prática em atividades que se aproximem da realidade do mercado [Cico et al. 2021, Diniz et al. 2024, Assyne et al. 2022, Akdur 2021]. Entre os fatores associados a esse quadro, destacam-se a rápida evolução das tecnologias, a dificuldade de atualização contínua das instituições de ensino e a predominância de abordagens expositivas centradas no professor.

A área de Engenharia de Software, por sua vez, ocupa posição estratégica na formação em Computação ao tratar de princípios, processos e práticas para o desenvolvimento sistemático de software [Valente 2020]. Abrange desde a modelagem e especificação de sistemas até atividades de implementação, teste, implantação e manutenção, articulando diferentes técnicas e ferramentas ao longo do ciclo de vida do software. Não por acaso, o ensino de Engenharia de Software no nível superior tem sido objeto de mapeamentos e discussões específicas, que apontam tanto avanços quanto desafios na forma como seus conteúdos são trabalhados em sala de aula [dos Santos et al. 2021].

Diversas iniciativas têm buscado aproximar o ensino de Engenharia de Software do contexto profissional por meio da adoção de metodologias ativas, do uso de projetos de código aberto, de abordagens baseadas em exemplos e da exploração sistemática de ferramentas utilizadas na indústria. Exemplos incluem o uso de projetos reais para aproximar estudantes de comunidades de software livre [Nascimento 2017], o emprego de estratégias de *example-based learning* em disciplinas de Engenharia de Software [Bonetti et al. 2025], a adoção de sala de aula invertida em conteúdos de processos e práticas de desenvolvimento [Vilela 2023] e o uso de metodologias ativas para o ensino de testes de software [Elgrably and Oliveira 2023]. Em um contexto mais micro, trabalhos também têm destacado o papel de objetos de aprendizagem e de ferramentas computacionais no apoio ao ensino de conteúdos específicos [Cadaval 2022, Nascimento Sassano et al. 2021], bem como propostas de produtos educacionais com foco na articulação entre teoria e prática em Engenharia de Software [Silva 2022].

Entre as práticas contemporâneas que têm ganhado espaço tanto na indústria quanto em currículos de Computação, destacam-se os testes automatizados e as práticas

de DevOps, incluindo Integração Contínua (CI) e Entrega Contínua (CD). Essas práticas combinam princípios ágeis, automação de pipelines e uso intensivo de ferramentas como Git, GitHub e plataformas de CI/CD, contribuindo para a melhoria da qualidade do software e para a redução do tempo entre desenvolvimento e entrega [Ahnert 2024, Enemosah 2025]. Incorporar tais práticas ao ensino de Engenharia de Software, de forma alinhada aos objetivos pedagógicos da disciplina, constitui um desafio relevante e, ao mesmo tempo, uma oportunidade de reduzir a distância entre sala de aula e mercado de trabalho.

Neste contexto, este artigo apresenta um relato de experiência sobre a aplicação de um conjunto de atividades práticas no ensino de conceitos de Engenharia de Software em um curso de Bacharelado em Ciência da Computação. A intervenção foi planejada pelo professor responsável pela disciplina e aplicada em conjunto com um monitor de graduação, articulando três atividades principais: (i) desenvolvimento de uma aplicação simples com operações *Create, Read, Update, Delete* (CRUD) em diferentes tecnologias; (ii) implementação de testes automatizados com o uso de frameworks amplamente empregados na indústria; e (iii) configuração de pipelines de Integração Contínua utilizando GitHub Actions. Ao final de cada atividade, foram aplicados questionários on-line, de resposta voluntária, com o objetivo de registrar a percepção dos estudantes sobre dificuldades, contribuições e possibilidades de melhoria.

O objetivo deste trabalho é descrever o contexto e a intervenção realizada de forma a fornecer subsídios para que outros docentes possam adaptar ou reutilizar a sequência de atividades.

2. Fundamentação teórica

Nesta seção são apresentados conceitos e resultados de pesquisas que embasam o relato de experiência descrito neste trabalho. São discutidos, de forma sintética, aspectos gerais da Engenharia de Software e das competências demandadas pela indústria, contribuições sobre o ensino de Engenharia de Software e abordagens pedagógicas na área, bem como trabalhos que exploram objetos de aprendizagem, atividades práticas, uso de ferramentas computacionais e práticas contemporâneas relacionadas a testes automatizados e DevOps.

2.1. Engenharia de Software e competências demandadas pela indústria

A Engenharia de Software pode ser entendida como um campo da Computação dedicado à aplicação de princípios de engenharia ao desenvolvimento de software, com o objetivo de produzir sistemas de qualidade de forma sistemática e previsível [Valente 2020]. Entre suas atividades centrais, destacam-se o levantamento de requisitos, a modelagem, a implementação, os testes e a manutenção de software ao longo de seu ciclo de vida.

Nas últimas décadas, diferentes estudos têm apontado um descompasso entre aquilo que é desenvolvido nas universidades e as competências esperadas pela indústria de software. Mapas e revisões sistemáticas sobre tendências em Engenharia de Software indicam que tecnologias, práticas e papéis profissionais evoluem em ritmo acelerado, exigindo dos profissionais uma combinação de conhecimentos técnicos, habilidades de colaboração e capacidade de adaptação [Cico et al. 2021, Assyne et al. 2022]. Estudos empíricos com profissionais apontam lacunas em competências específicas, abrangendo tanto fundamentos técnicos quanto habilidades de comunicação e trabalho em equipe

[Akdur 2021]. No contexto brasileiro, pesquisas recentes reforçam a existência de um *skill gap* entre a formação em Engenharia de Software e as demandas da indústria, destacando a necessidade de maior aproximação entre academia e mercado por meio de parcerias, projetos conjuntos e revisão de currículos [Diniz et al. 2024].

2.2. Ensino de Engenharia de Software e abordagens pedagógicas

O ensino de Engenharia de Software no nível superior tem sido objeto de diferentes investigações, que apontam avanços na inclusão de conteúdos relacionados a processos, qualidade e métodos ágeis, mas também desafios na integração entre teoria e prática e na adoção de estratégias pedagógicas que favoreçam o protagonismo dos estudantes [dos Santos et al. 2021]. Nesse cenário, diversas abordagens têm sido exploradas para tornar o ensino mais alinhado ao contexto profissional.

A adoção de projetos de código aberto como plataforma para atividades de ensino permite que estudantes interajam com artefatos reais, práticas de colaboração distribuída e fluxos de desenvolvimento próximos aos encontrados na indústria [Nascimento 2017]. Abordagens baseadas em exemplos (*example-based learning*) apoiam a compreensão de conceitos por meio de casos concretos que podem ser analisados e estendidos pelos estudantes [Bonetti et al. 2025]. Metodologias ativas, como sala de aula invertida e aprendizagem baseada em projetos, também têm sido aplicadas em disciplinas de Engenharia de Software, com relatos de ganhos na participação e na apropriação de conceitos abstratos [Vilela 2023, Elgrably and Oliveira 2023].

2.3. Objetos de aprendizagem, atividades práticas e ferramentas computacionais

No âmbito da Educação em Computação, diversos trabalhos destacam o papel de objetos de aprendizagem e de ferramentas computacionais no apoio ao ensino de conteúdos específicos, atuando em um nível mais pontual do processo de ensino-aprendizagem. Cadaval [Cadaval 2022] investiga o desenvolvimento e a avaliação de objetos de aprendizagem para o ensino de lógica matemática, enquanto Sassano et al. discutem o uso de ferramentas computacionais para apoiar o ensino de matemática na modalidade a distância [Nascimento Sassano et al. 2021]. Nesses casos, o foco recai sobre o desenho de recursos e atividades práticas concretas que podem ser empregados dentro de diferentes abordagens pedagógicas.

De forma complementar, propostas de produtos educacionais voltados ao ensino de Engenharia de Software buscam articular teoria e prática por meio de sequências de atividades estruturadas, explorando o uso de ferramentas e cenários próximos à realidade do desenvolvimento de software. Silva [Silva 2022], por exemplo, organiza conteúdos de Engenharia de Software em abordagens práticas que se aproximam de situações encontradas em ambientes profissionais. O uso de ferramentas amplamente empregadas na indústria — como Trello [Trello 2023], Git [Git 2025] e GitHub [GitHub 2025] — tem se mostrado uma estratégia relevante para aproximar o ambiente de sala de aula de contextos reais de desenvolvimento, apoiando a gestão de tarefas, o versionamento de artefatos e o trabalho colaborativo entre estudantes.

2.4. Testes automatizados, DevOps e ferramentas da indústria

Entre as práticas contemporâneas que têm ganhado espaço tanto na indústria quanto em currículos de Computação, destacam-se os testes automatizados e as práticas de DevOps,

incluindo Integração Contínua (CI) e Entrega Contínua (CD). Essas práticas combinam princípios ágeis, automação de *pipelines* e uso intensivo de ferramentas como Git, GitHub e plataformas de CI/CD, contribuindo para a melhoria da qualidade do software e para a redução do tempo entre desenvolvimento e entrega [Ahnert 2024, Enemosah 2025].

No contexto educacional, incorporar testes automatizados e práticas de DevOps ao ensino de Engenharia de Software, de forma alinhada aos objetivos pedagógicos da disciplina, constitui um desafio e, ao mesmo tempo, uma oportunidade de reduzir a distância entre sala de aula e mercado. Atividades que envolvem a escrita de testes com *frameworks* amplamente difundidos e a configuração de *pipelines* de CI em plataformas reais permitem oferecer experiências próximas das práticas adotadas em equipes profissionais, ao mesmo tempo em que exploram conceitos fundamentais de qualidade, automação e manutenção de sistemas. Nesta perspectiva, o trabalho aqui relatado propõe uma sequência de atividades práticas que integra desenvolvimento de aplicações, testes automatizados e configuração de *pipelines* de Integração Contínua, utilizando ferramentas efetivamente empregadas na indústria.

3. Materiais e Métodos

3.1. Contexto da disciplina e participantes

A intervenção foi realizada na disciplina de Engenharia de Software de um curso de Bacharelado em Ciência da Computação de uma instituição pública federal de ensino. A disciplina, ofertada para estudantes do quinto e sexto períodos, aborda, entre outros tópicos, modelagem de software, processos de desenvolvimento, testes e aspectos introdutórios de DevOps.

As atividades práticas foram planejadas pelo professor responsável pela disciplina e aplicadas em sala de aula com o apoio de um monitor de graduação, ao longo de um semestre letivo. A proposta integrou-se à estrutura regular da disciplina, complementando as aulas expositivas com experiências práticas envolvendo ferramentas amplamente utilizadas na indústria, tais como MySQL, Java, PHP, JavaScript/Node.js, *frameworks* de testes (JUnit, Jest, Selenium e Cypress) e GitHub Actions para Integração Contínua.

Ao final de cada atividade, foi disponibilizado um questionário on-line, construído na ferramenta Google Forms, com questões fechadas e abertas sobre a percepção dos estudantes quanto à clareza das explicações, ao nível de dificuldade, às contribuições para a aprendizagem e às sugestões de melhoria. A participação foi voluntária, não havendo associação direta entre o preenchimento das respostas e a avaliação formal da disciplina.

3.2. Desenho da intervenção e atividades práticas

A metodologia adotada organiza-se em três atividades principais, aplicadas de forma sequencial ao longo do semestre:

- **Atividade 1 – Modelagem e desenvolvimento de software (CRUD)**;
- **Atividade 2 – Testes de software**;
- **Atividade 3 – DevOps e Integração Contínua**.

Cada atividade foi planejada para reforçar conteúdos discutidos previamente em aula expositiva, por meio de exemplos e exercícios práticos conduzidos em laboratório.

3.2.1. Atividade 1: Modelagem e desenvolvimento de software

A primeira atividade teve como foco a modelagem e o desenvolvimento de uma aplicação simples com operações CRUD, servindo como base para as etapas posteriores. Inicialmente, foram apresentados exemplos de entidades de negócio e histórias de usuário para um sistema de gerenciamento de funcionários e produtos. Em seguida, o professor e o monitor demonstraram duas aplicações de exemplo (em Java com *JFrame* e em PHP com *frontend* em HTML, CSS e JavaScript), ambas com CRUD de funcionários e persistência em MySQL.

A partir desses exemplos, cada grupo implementou um CRUD de produtos, utilizando a mesma estrutura de banco de dados e podendo escolher entre Java, PHP ou Node.js com JavaScript. Durante o desenvolvimento, professor e monitor apoiaram os estudantes na compreensão da estrutura do código e em boas práticas de organização.

3.2.2. Atividade 2: Testes de software

A segunda atividade ampliou os conceitos de testes de software. Foram apresentados exemplos de testes unitários em Java e JavaScript, utilizando JUnit e Jest, e de testes funcionais com Selenium e Cypress. Na prática, os estudantes implementaram testes unitários para funcionalidades simples e testes funcionais para o sistema CRUD desenvolvido na Atividade 1, verificando o comportamento da aplicação a partir da perspectiva do usuário final. O monitor auxiliou na configuração dos ambientes e na interpretação de mensagens de erro, enquanto o professor reforçava conceitos de tipos de teste e papel do QA no processo de desenvolvimento.

3.2.3. Atividade 3: DevOps e Integração Contínua

A terceira atividade integrou os testes automatizados à prática de Integração Contínua, por meio da configuração de *workflows* com GitHub Actions. Foi disponibilizado um repositório no GitHub contendo o código do sistema CRUD e arquivos de configuração de testes. Cada estudante (ou grupo) realizou um *fork* do repositório e um *clone* para a máquina local. Um *workflow* de CI previamente configurado executava automaticamente os testes a cada *push*.

Os estudantes completaram métodos faltantes e observaram o comportamento do *pipeline* diante de testes bem-sucedidos ou falhos. O monitor apoiou a resolução de problemas relacionados ao uso de Git e GitHub e à leitura dos registros do GitHub Actions, enquanto o professor discutia conceitos de CI, feedback rápido e automação.

3.3. Procedimentos de coleta e análise de dados

Os dados apresentados neste relato foram obtidos por meio dos questionários aplicados ao final de cada atividade. Os instrumentos, construídos no Google Forms, continham questões fechadas em escala Likert e questões abertas para comentários e sugestões. A participação foi voluntária, informando-se aos estudantes que as respostas seriam utilizadas para fins de avaliação pedagógica das atividades e, de forma agregada, para fins acadêmicos.

Os dados foram analisados de maneira descritiva, sem identificação individual dos participantes. Na análise apresentada neste artigo, são utilizadas porcentagens e frequências simples para sumarizar as respostas às questões fechadas, bem como uma síntese qualitativa das respostas abertas, destacando comentários recorrentes sobre aspectos positivos, dificuldades e sugestões de aprimoramento.

4. Resultados e Discussão

Nesta seção são apresentados e discutidos os resultados obtidos com a aplicação das três atividades práticas descritas na Seção 3. Em cada atividade, são analisadas questões selecionadas dos questionários de percepção respondidos pelos estudantes, com base em 25 respostas na Atividade 1, 17 respostas na Atividade 2 e 13 respostas na Atividade 3.

4.1. Atividade 1: Modelagem e desenvolvimento de software

A Atividade 1 teve como objetivo apoiar a compreensão do ciclo de desenvolvimento de software a partir da implementação de uma aplicação CRUD em diferentes tecnologias, tomando como base exemplos desenvolvidos em Java (com *JFrame*) e em PHP com *frontend* em HTML, CSS e JavaScript. Ao final da atividade, os estudantes responderam a um questionário com seis questões fechadas e abertas.

A Figura 1 apresenta os resultados da primeira pergunta, que investigou a experiência prévia dos participantes com modelagem e desenvolvimento de sistemas.

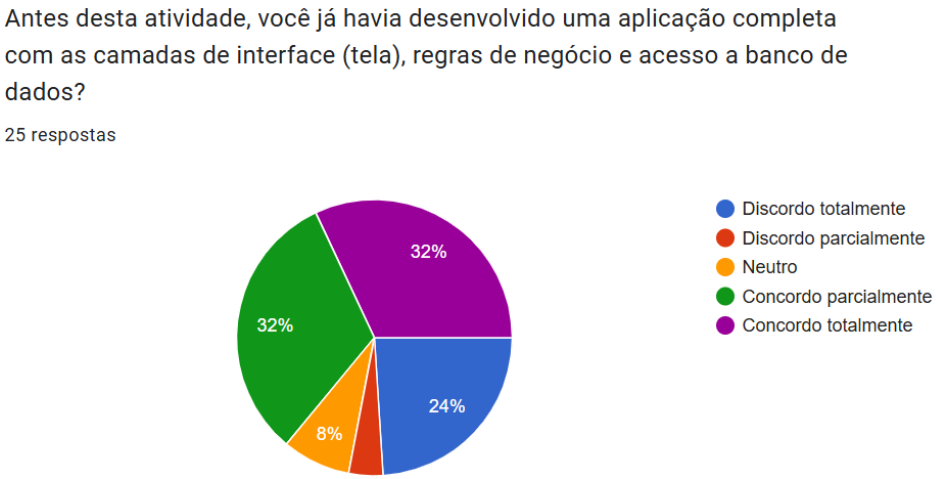

Figura 1. Experiência prévia com modelagem e desenvolvimento de sistemas.
Fonte: Autoria própria (2025).

Conforme ilustrado na Figura 1, 14 participantes relataram já ter desenvolvido algum tipo de sistema (englobando front-end, back-end e banco de dados), enquanto cerca de 10 declararam não possuir experiência anterior nessa atividade. Esse resultado reforça a heterogeneidade da turma em termos de vivência prática com desenvolvimento de software, indicando que a proposta da Atividade 1 deveria, ao mesmo tempo, consolidar conhecimentos para estudantes com mais experiência e oferecer uma primeira vivência estruturada de implementação para aqueles com pouca ou nenhuma prática.

A Figura 2 apresenta os resultados da pergunta que buscou avaliar em que medida a Atividade 1 contribuiu para a compreensão do ciclo de desenvolvimento de software, desde as etapas iniciais até os ajustes finais na aplicação.

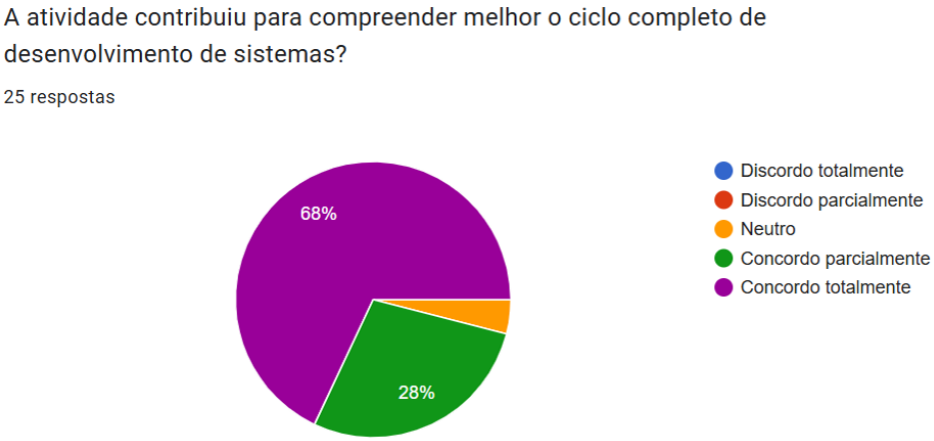

Figura 2. Contribuição da atividade para compreender o ciclo de desenvolvimento de software.

Fonte: Autoria própria (2025).

Os dados da Figura 2 indicam que 96% dos participantes concordaram parcial ou totalmente que a atividade ajudou a compreender melhor o ciclo de desenvolvimento, sugerindo que a combinação entre análise de exemplos prontos e implementação de um CRUD de produtos foi efetiva para articular teoria e prática. Nas respostas abertas, os estudantes destacaram como pontos positivos a possibilidade de observar uma mesma entidade de negócio implementada em diferentes tecnologias e a compreensão mais clara das etapas envolvidas no desenvolvimento.

Esses resultados dialogam com trabalhos que enfatizam o potencial de abordagens baseadas em exemplos e em atividades práticas para apoiar a aprendizagem de Engenharia de Software [Bonetti et al. 2025, Silva 2022], indicando que a exposição a múltiplas tecnologias, ancorada em um mesmo cenário de negócio, pode contribuir para ampliar a compreensão dos estudantes sobre o processo de desenvolvimento.

4.2. Atividade 2: Testes de software

A Atividade 2 teve como foco a ampliação dos conceitos de testes de software a partir da implementação de testes unitários e funcionais com diferentes *frameworks* (JUnit e Jest para testes unitários; Selenium e Cypress para testes funcionais). Os estudantes responderam a um questionário com questões buscando captar sua percepção sobre a contribuição da atividade e sua familiaridade prévia com testes automatizados.

A Figura 3 apresenta os resultados da primeira questão, que investigou em que medida a atividade ajudou a compreender a utilização de testes de software em um contexto não apenas acadêmico, mas também de mercado.

Conforme ilustrado na Figura 3, 88,2% dos estudantes avaliaram que a atividade teve impacto positivo na compreensão da utilização de testes de software, indicando que a prática contribuiu para conectar os conceitos discutidos em aula a situações mais próximas da realidade profissional. Apenas uma minoria indicou que a atividade teve impacto neutro ou reduzido, o que pode estar associado a estudantes que já possuíam experiência prévia mais consolidada na área.

A Figura 4 apresenta os resultados da pergunta que buscou avaliar em que medida

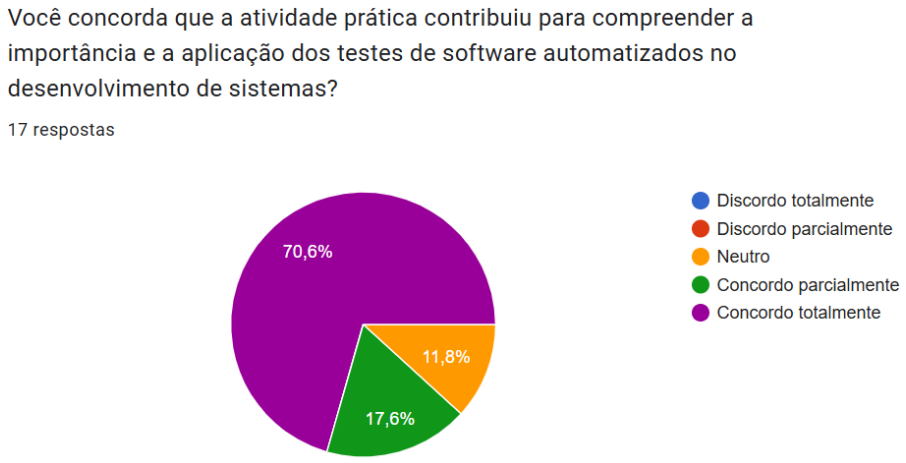

Figura 3. Contribuição da atividade para compreender a utilização de testes de software.

Fonte: Autoria própria (2025).

as ferramentas utilizadas na atividade (JUnit, Jest, Selenium e Cypress) se conectaram às aulas expositivas de testes de software.

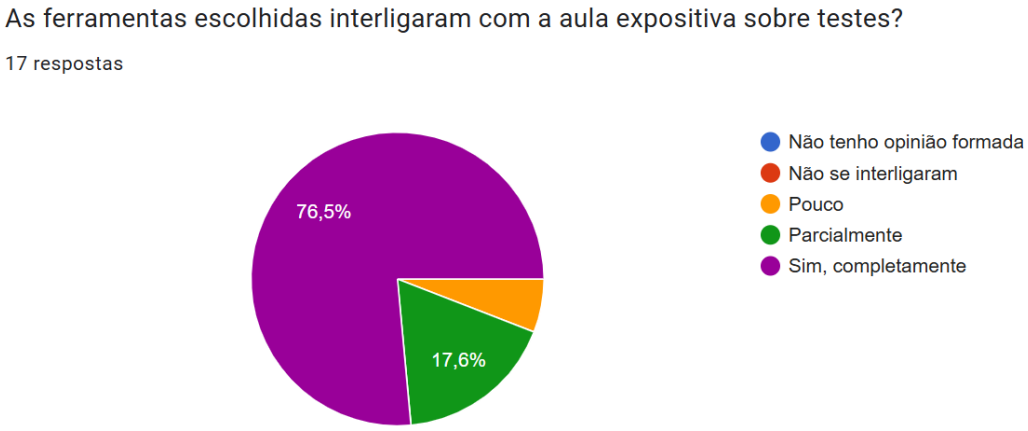

Figura 4. Relação entre as ferramentas utilizadas e as aulas expositivas.

Fonte: Autoria própria (2025).

Os resultados da Figura 4 mostram que 94,1% dos participantes consideraram que as ferramentas apresentadas tiveram boa conectividade com o conteúdo teórico, reforçando a percepção de que a atividade funcionou como um elo entre o que foi discutido conceitualmente e a prática de escrita e execução de testes automatizados. Nas respostas abertas, os estudantes relataram que a atividade ajudou a entender melhor a importância dos testes automatizados e o papel do QA (*Quality Assurance*) no processo de desenvolvimento, além de sugerirem a inclusão de mais exemplos e tempo de prática. Esses achados se aproximam de resultados de Elgrably e Oliveira [Elgrably and Oliveira 2023], que destacam o potencial de metodologias ativas e atividades práticas para fortalecer a compreensão de testes de software no ensino superior.

4.3. Atividade 3: DevOps e Integração Contínua

A Atividade 3 buscou integrar os testes automatizados da Atividade 2 com a prática de Integração Contínua, utilizando GitHub Actions para configurar *workflows* de execução automática de testes a cada alteração no repositório. O questionário investigou em que medida a atividade contribuiu para o entendimento de Integração Contínua e para a articulação desse conceito com o conteúdo teórico de DevOps. Os resultados estão representados na Figura 5.

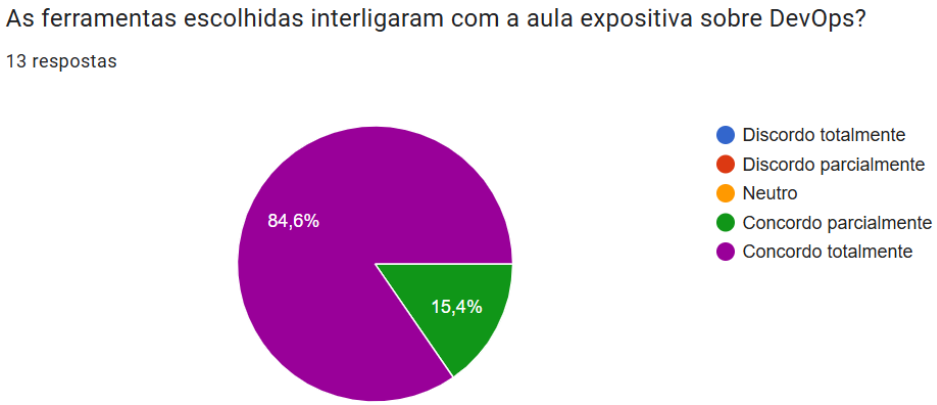

Figura 5. Contribuição da atividade e do GitHub Actions para compreender Integração Contínua e DevOps.
Fonte: Autoria própria (2025).

Como mostrado na Figura 5, todos os participantes avaliaram a atividade de forma positiva: 84,6% concordaram totalmente e 15,4% concordaram parcialmente que o uso do GitHub Actions contribuiu para complementar a aula expositiva de DevOps e para o entendimento de Integração Contínua. Esses resultados indicam boa aceitação da ferramenta como recurso de apoio ao ensino, reforçando a pertinência de incorporar práticas de CI em disciplinas de Engenharia de Software.

Por outro lado, a percepção dos estudantes quanto ao nível de dificuldade da atividade foi mais distribuída, conforme apresentado na Figura 6.

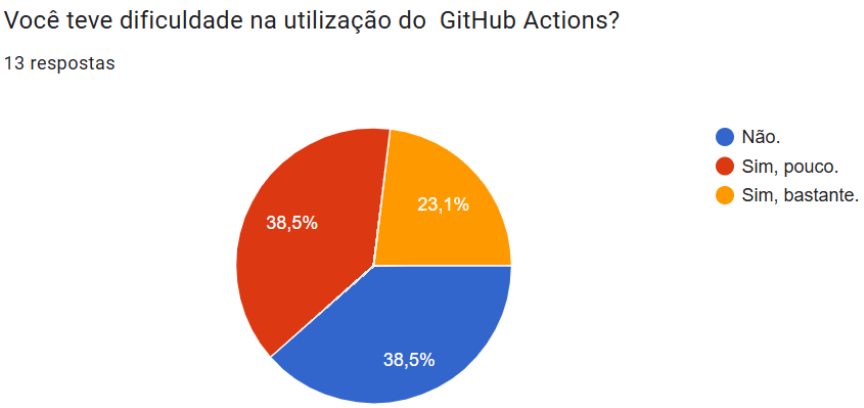

Figura 6. Nível de dificuldade percebido no uso do GitHub Actions.
Fonte: Autoria própria (2025).

De acordo com a Figura 6, 38,5% dos estudantes relataram não ter tido dificuldade no uso do GitHub Actions, enquanto 38,5% indicaram ter tido "um pouco de dificuldade" e 23,1% relataram "bastante dificuldade". Uma possível explicação para esse resultado está relacionada à necessidade de conhecimentos prévios de Git e GitHub, bem como à compreensão de conceitos específicos de DevOps, como *workflows* e arquivos de configuração, que podem representar uma curva de aprendizagem significativa para estudantes com pouca familiaridade com essas práticas.

Nas respostas abertas, os participantes mencionaram que a atividade foi importante para consolidar conceitos de Integração Contínua e para compreender, na prática, o papel da automação em pipelines de desenvolvimento. Ao mesmo tempo, sugeriram a inclusão de momentos adicionais de revisão de Git e GitHub antes da configuração dos *workflows* e a oferta de exemplos ainda mais simples como ponto de partida.

4.4. Síntese dos resultados

De forma geral, os resultados indicam que a sequência de atividades práticas contribuiu para aproximar o ensino de Engenharia de Software de práticas e ferramentas presentes no mercado de trabalho. A Atividade 1 favoreceu a compreensão do ciclo de desenvolvimento de software a partir da implementação de um CRUD em diferentes tecnologias, ainda que partindo de uma turma com níveis distintos de experiência prévia. A Atividade 2 ampliou a visão dos estudantes sobre testes automatizados, combinando testes unitários e funcionais com diferentes *frameworks* e reforçando a conexão entre teoria e prática. Já a Atividade 3 possibilitou um primeiro contato estruturado com Integração Contínua e DevOps, evidenciando tanto o potencial formativo do uso de GitHub Actions quanto as dificuldades associadas à adoção de práticas de DevOps em contextos educacionais.

A presença de um monitor ao longo das atividades mostrou-se relevante para lidar com a complexidade técnica envolvida, especialmente na Atividade 3, permitindo que dúvidas pontuais fossem tratadas de forma mais ágil.

5. Conclusão

Este artigo apresentou um relato de experiência sobre a aplicação de atividades práticas no ensino de Engenharia de Software, articulando o uso de ferramentas amplamente empregadas na indústria com os conteúdos de uma disciplina de graduação. A intervenção, conduzida pelo professor e por um monitor de graduação, envolveu o desenvolvimento de um CRUD em diferentes tecnologias, a implementação de testes automatizados e a configuração de *pipelines* de Integração Contínua com GitHub Actions.

Os resultados dos questionários de percepção indicam que as atividades contribuíram para aproximar o conteúdo teórico da prática profissional, fortalecer a compreensão de conceitos de Engenharia de Software e proporcionar aos estudantes um primeiro contato estruturado com ferramentas utilizadas no mercado. Ao mesmo tempo, foram identificadas dificuldades, especialmente relacionadas ao uso de ferramentas de DevOps e à necessidade de consolidar previamente conceitos de Git e GitHub.

Este trabalho apresenta algumas limitações relevantes. Em primeiro lugar, trata-se de um estudo de caso único, realizado em uma disciplina específica, em um curso e instituição particulares, o que limita a generalização dos resultados para outros contextos. Em segundo lugar, a avaliação baseia-se predominantemente na percepção dos estudantes

que optaram por responder aos questionários, sem controle sobre possíveis vieses decorrentes da participação voluntária. Além disso, não foram coletadas medidas objetivas de desempenho antes e depois da intervenção, o que impede a análise de ganhos de aprendizagem em termos quantitativos. Do ponto de vista operacional, a escolha das ferramentas esteve condicionada às condições dos laboratórios disponíveis, incluindo sistemas operacionais, permissões de instalação e configuração de rede, o que impôs restrições à adoção de determinadas tecnologias.

Como trabalhos futuros, pretende-se reaplicar e ajustar a sequência de atividades em novas turmas, incorporando as sugestões dos estudantes e as lições aprendidas; expandir a metodologia para outras disciplinas de Engenharia de Software e áreas correlatas que exijam forte componente prático; complementar os dados de percepção com evidências quantitativas de aprendizagem; e aperfeiçoar a preparação dos estudantes para o uso de ferramentas de DevOps, incluindo momentos específicos para revisão de Git e GitHub antes da configuração dos *pipelines*.

Espera-se que a descrição detalhada do contexto, da intervenção e das lições aprendidas possa apoiar outros docentes interessados em incorporar atividades semelhantes em suas disciplinas, contribuindo para reduzir a lacuna entre a formação acadêmica e as demandas da indústria de software.

Uso de Inteligência Artificial

Este artigo contou com o apoio pontual de ferramentas de Inteligência Artificial generativa para auxiliar na organização do texto e na revisão linguística. Todas as decisões de conteúdo, estrutura, análise e interpretação dos resultados foram tomadas pelos autores, que se responsabilizam integralmente pelo texto apresentado.

Referências

Ahnert, S. (2024). Devops no desenvolvimento de software - itix tecnologia.

Akdur, D. (2021). Skills gaps in the industry: Opinions of embedded software practitioners. *ACM Trans. Embed. Comput. Syst.*, 20(5):1–39.

Assyne, N., Ghanbari, H., and Pulkkinen, M. (2022). The state of research on software engineering competencies: A systematic mapping study. *Journal of Systems and Software*, 185(1):111183.

Bonetti, T. P., Silva, W., and Colanzi, T. E. (2025). Example-based learning in software engineering education: A systematic mapping study. *arXiv*, abs/2503.18080(1):37.

Cadaval, L. S. (2022). Desenvolvimento e avaliação de objetos de aprendizagem para o ensino de lógica matemática. Master's thesis, Universidade Federal do Pampa, Alegrete.

Cico, O., Jaccheri, L., Nguyen-Duc, A., and Zhang, H. (2021). Exploring the intersection between software industry and software engineering education-a systematic mapping of software engineering trends. *Journal of Systems and Software*, 172(1):110736.

Diniz, W., Valença, M., França, C., Santos, A., and Pincovsky, M. (2024). The skill gap in software industry: A mapping study. In *ANAIS DO SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES)*, pages 192–200, Porto Alegre.

dos Santos, M. E. S., Rocha, T. S., and Perkusich, M. B. (2021). O ensino de engenharia de software no nível superior: um mapeamento sistemático. *Revista Principia*, 1(56):116–125.

Elgrably, I. S. and Oliveira, S. R. B. (2023). Uma abordagem para o ensino de testes de software utilizando metodologias ativas em cursos superiores de computação. In *ANAIS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCACÃO (CBIE)*, pages 13–25, Porto Alegre.

Enemosah, A. (2025). Enhancing devops efficiency through ai-driven predictive models for continuous integration and deployment pipelines. *International Journal of Research Publication and Reviews*, 6(1):871–887.

Git (2025). Git - gittutorial documentation.

GitHub (2025). Sobre o github e o git - github docs.

Goulart, A. R. (2019). A falência do atual modelo de educação superior em tecnologia da informação (ti). Dissertação (mestrado em administração de organizações), Universidade de São Paulo, Ribeirão Preto.

Moreira, A. (2024). Demanda por talentos em ti cresce e impulsiona o mercado - guia da faculdade 2024.

Nascimento, D. M. C. (2017). *Educação em Engenharia de Software com a adoção de projetos de código aberto: uma análise detalhada*. Tese de doutorado, Universidade Federal da Bahia, Salvador.

Nascimento Sassano, F. C., Lopes Guerra Neto, H., Hilário Barizão, A., Ribeiro Sencio, R., Onaya, , and Welter, N. (2021). O uso de ferramentas computacionais no processo de ensino e aprendizagem de matemática na ead. *Revista Brasileira de Aprendizagem Aberta e a Distância*, 20(1):21.

Silva, F. B. d. (2022). Projeto de um produto educacional para o processo de ensino-aprendizagem de conteúdos de engenharia de software através de abordagens práticas. Trabalho de conclusão de curso (tcc), Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – CAMPUS PASSOS, Passos.

Trello (2023). Reúna, organize e resolva suas tarefas de qualquer lugar — trello.

Valença, M., Diniz, W., Pincovsky, M., França, C., and Cabral, G. (2023). Mercado de trabalho em tecnologia da comunicação e informação (ti): análise de um experimento de aproximação entre academia e indústria no porto digital. In *ANAIS DO WORKSHOP SOBRE ASPECTOS SOCIAIS, HUMANOS E ECONÔMICOS DE SOFTWARE (WASHES)*, pages 1–10, Porto Alegre.

Valente, M. T. (2020). *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Independente, Belo Horizonte.

Vilela, P. R. S. (2023). Ensino de engenharia de software utilizando sala de aula invertida. In *ANAIS DA ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE (ERES)*, pages 21–30, Porto Alegre.