
Ensinar algoritmos na era da inteligência artificial generativa: um relato de experiência docente no ensino superior tecnológico
Teaching algorithms in the age of generative artificial intelligence: a teaching experience report in higher technological education

William Corrêa Mendes

Doutorando em Ciências Contábeis e Administração, FUCAPE, Brasil.
Docente, Faculdade EDUFOR (EDUFOR), São Luís, MA, Brasil.
Lattes: <https://lattes.cnpq.br/7726054867638395>
ORCID: <https://orcid.org/0009-0006-8272-8549>
E-mail: william.mendes@edufor.edu.br

Marcos Christian Silva Matos

Mestre em Direção Estratégica em Tecnologias da Informação, FUNIBER, Brasil.
Docente, Faculdade EDUFOR (EDUFOR), São Luís, MA, Brasil.
Lattes: <http://lattes.cnpq.br/8568026725518533>
ORCID: <https://orcid.org/0009-0008-3435-8445>
E-mail: marcos.matos@edufor.edu.br

Revista Interlocus, volume 2, número 5, 2026

DOI: 10.5281/zenodo.21285770

Editor: Gláuber Tulio Fonseca Coelho

Artigo recebido: 29/06/2026

Artigo Aceito: 07/07/2026

Artigo Publicado: 09/07/2026

RESUMO: Este artigo apresenta um relato de experiência docente na disciplina de Algoritmos Computacionais do curso de Análise e Desenvolvimento de Sistemas da Faculdade EDUFOR, em São Luís (MA), construído a partir das perspectivas complementares do professor Marcos, que inaugurou a disciplina em 2024.1 e a conduziu ao longo de 2024 e 2025, e do professor William, que a assumiu em 2026.1 com uma abordagem deliberadamente orientada à integração da inteligência artificial generativa (IAG). O problema central que o relato enfrenta é a seguinte tensão: como ensinar raciocínio lógico-computacional quando ferramentas como ChatGPT, GitHub Copilot e Google Gemini são capazes de resolver os exercícios da disciplina em segundos? O argumento desenvolvido é que a IAG não elimina a necessidade de aprender algoritmos — ela desloca o foco da resposta certa para o processo de raciocínio. As decisões pedagógicas descritas incluem a integração da IAG em vez de sua proibição, a exigência de registro de prompts como prática de responsabilização e visibilidade do processo, o uso de soluções geradas por IA como objeto de análise coletiva em sala e a articulação com o projeto de extensão curricularizada Portal de Ideias. A fundamentação teórica ancora essas decisões nos conceitos de pensamento computacional, no framework TPACK e nas competências humanas que a engenharia de software requer além da geração de código. O relato conclui com uma análise do fenômeno do *vibe coding* como horizonte que radicaliza a tensão original, reforçando o argumento de que a formação do raciocínio computacional permanece indispensável porque é ela que confere ao desenvolvedor agência sobre as ferramentas que o cercam.

Palavras-chave: Algoritmos computacionais; Inteligência artificial generativa; Pensamento computacional; Mediação pedagógica; Ensino superior tecnológico.

ABSTRACT: This article presents a teaching experience report from the Computational Algorithms course in the Systems Analysis and Development program at Faculdade EDUFOR, São Luís (Maranhão, Brazil), built from the complementary perspectives of Professor Marcos, who inaugurated the course in 2024.1 and taught

it through 2024 and 2025, and Professor William, who took over in 2026.¹ with a deliberately integrative approach to generative artificial intelligence (GenAI). The central problem addressed is the following tension: how to teach logical-computational reasoning when tools such as ChatGPT, GitHub Copilot, and Google Gemini can solve course exercises in seconds? The argument developed is that GenAI does not eliminate the need to learn algorithms — it shifts the focus from the correct answer to the reasoning process. The pedagogical decisions described include integrating GenAI rather than banning it, requiring prompt registration to support process accountability and visibility, using AI-generated solutions as objects of collective classroom analysis, and aligning the course with the curricularized extension project Portal de Ideias. The theoretical foundation anchors these decisions in computational thinking, the TPACK framework, and the human competencies software engineering requires beyond code generation. The report concludes with an analysis of the vibe coding phenomenon as a horizon that radicalizes the original tension, reinforcing the argument that computational reasoning education remains indispensable because it is what gives developers agency over the tools that surround them.

Keywords: Computational algorithms; Generative artificial intelligence; Computational thinking; Pedagogical mediation; Higher technological education.



1. INTRODUÇÃO

A inteligência artificial generativa (IAG) transformou, em poucos anos, a relação entre estudantes de computação e o processo de aprendizagem de programação. Ferramentas como ChatGPT, GitHub Copilot e Google Gemini são capazes de gerar soluções completas para exercícios de algoritmos em questão de segundos, a partir de um simples enunciado em linguagem natural (Cambaz; Zhang, 2024; Prather et al., 2024). Esse fenômeno coloca o docente de cursos tecnológicos diante de uma tensão inédita: como ensinar a pensar algorítmicamente quando a máquina já pensa por você?

Essa pergunta não admite resposta simples. Proibir o uso de IAG em sala de aula é, além de operacionalmente inviável, pedagogicamente questionável em um curso cujo egresso atuará em mercados onde essas ferramentas serão onipresentes (Lau; Guo, 2023). Por outro lado, permitir o uso irrestrito sem mediação pedagógica deliberada pode comprometer o desenvolvimento das competências cognitivas que fundamentam a formação do engenheiro de *software*: a capacidade de decompor problemas, modelar soluções, abstrair e comunicar-se com precisão técnica com clientes e equipes (Cambaz; Zhang, 2024; Kasneci et al., 2023).

Este artigo apresenta um relato de experiência docente na disciplina de Algoritmos Computacionais do curso de Análise e Desenvolvimento de Sistemas (ADS) da Faculdade EDUFOR, em São Luís (MA). O relato é construído a partir de duas perspectivas complementares: a do professor Marcos, que inaugurou a disciplina no curso em 2024.1 e a conduziu ao longo de 2024 e 2025, vivendo a chegada da IAG no processo de construção de sua prática docente, e a do professor William, que assumiu a disciplina em 2026.1, já com a IAG plenamente incorporada ao cotidiano dos estudantes e com uma abordagem pedagógica deliberadamente orientada para a integração dessas ferramentas. Essa complementaridade de olhares permite não apenas narrar o presente, mas identificar como a maturidade pedagógica na relação com a IAG se desenvolve ao longo de ciclos letivos e o que, em todo esse processo, permanece essencial.

O objetivo não é oferecer um modelo definitivo de como ensinar algoritmos com IA, mas compartilhar decisões pedagógicas reais, suas motivações e as reflexões que delas emergiram, contribuindo para o debate crescente sobre o papel da IAG na educação em computação no Brasil.

2. CONTEXTUALIZAÇÃO: A DISCIPLINA, O CURSO E O MOMENTO

2.1 O contexto institucional

A Faculdade EDUFOR, localizada em São Luís do Maranhão, oferta o curso de Análise e Desenvolvimento de Sistemas (ADS) com foco na formação de profissionais aptos a atuar no desenvolvimento, implantação e manutenção de sistemas de informação. A disciplina de Algoritmos Computacionais ocupa posição estratégica na matriz curricular: é, para a maioria dos estudantes, o primeiro contato formal com raciocínio lógico-computacional, linguagens de programação e resolução estruturada de problemas.

O perfil predominante dos ingressantes é de jovens sem experiência prévia com programação, muitos oriundos de contextos com acesso limitado a tecnologia avançada. Para esse público, a curva de aprendizagem já era desafiadora antes da IAG. A chegada das ferramentas de IA generativa adicionou uma camada nova de complexidade ao processo de ensino-aprendizagem — não por torná-lo mais difícil, mas por tornar mais visível uma questão que sempre esteve presente: o estudante está aprendendo a resolver problemas, ou está aprendendo a obter respostas?

2.2 Duas perspectivas, um mesmo desafio

O professor Marcos foi o primeiro docente a lecionar Algoritmos Computacionais na EDUFOR, em 2024.1 (semestre de inauguração do curso de ADS na instituição). Sua experiência foi, portanto, dupla: construir a disciplina do zero em um curso novo e, simultaneamente, lidar com a presença crescente da IAG nas turmas. O boom do ChatGPT havia ocorrido em fins de 2022 e, ao longo de 2023, o tema se consolidou mundialmente — de modo que já na primeira turma da EDUFOR a ferramenta era conhecida e utilizada, ainda que de formas variadas e nem sempre transparentes.

A avaliação naquele momento era conduzida principalmente por exercícios práticos, atividades em sala, participação e pequenas entregas, com provas quando necessário. O foco, desde o início, era perceber se o estudante conseguia aplicar o conteúdo, não apenas reproduzi-lo. Com a presença crescente da IAG, especialmente após a consolidação do ChatGPT em 2023, essa orientação avaliativa precisou se reconfigurar. O professor Marcos passou a valorizar mais atividades contextualizadas, projetos e resolução de problemas reais, além de observar a capacidade do estudante de explicar o próprio raciocínio. A IA, nesse contexto, não eliminou a necessidade de aprender — mas mudou o tipo de aprendizado esperado. O estudante passou a precisar saber perguntar melhor, interpretar respostas, validar informações e usar a ferramenta com senso crítico. Em suas palavras: “a principal mudança foi sair de uma avaliação muito centrada na resposta final e passar a observar mais o processo: como o estudante chegou à resposta, como usou os recursos disponíveis e se realmente compreendeu o que estava fazendo”.

O professor William assumiu a disciplina em 2026.1, em um cenário onde a IAG já estava plenamente incorporada ao cotidiano dos estudantes e onde havia dois anos de prática institucional acumulada na própria EDUFOR como referência. Diferentemente de uma postura reativa, de proibir ou ignorar, a abordagem adotada foi de integração deliberada: reconhecer que a IA é uma realidade do mercado para o qual se está formando o estudante e construir práticas pedagógicas que convivam com ela sem abrir mão do desenvolvimento do pensamento lógico.

O que aproxima as duas experiências não é o momento histórico, mas o deslocamento pedagógico que ambas produziram: da resposta certa como objetivo para o processo de raciocínio como objeto de formação. A IAG tornou esse deslocamento urgente e, ao mesmo tempo, pedagogicamente possível, ao oferecer novas formas de tornar o processo visível e analisável.

3. FUNDAMENTAÇÃO TEÓRICA

A dificuldade de ensinar programação a estudantes iniciantes não é uma novidade pedagógica recente. Desde os primeiros anos da educação em computação, pesquisadores identificaram que a aprendizagem de algoritmos impõe desafios distintos dos que caracterizam outras disciplinas das ciências exatas. Robins, Rountree e Rountree (2003), em uma revisão abrangente da literatura sobre ensino e aprendizagem de programação, destacam que os novatos enfrentam dificuldades sobretudo na fase de planejamento e modelagem da solução, e não na compreensão de sintaxe isolada. A distinção entre novatos eficazes e ineficazes, conforme os autores, reside menos na quantidade de conhecimento acumulado e mais nas estratégias que mobilizam diante de problemas novos: os mais bem-sucedidos desenvolvem esquemas de resolução que integram conhecimento declarativo e procedimental de forma coordenada, enquanto os demais recorrem a tentativas sem ancoragem conceitual.

Gomes e Mendes (2007) reforçam esse diagnóstico ao apontarem que a principal causa do insucesso na aprendizagem de programação introdutória não é a dificuldade com a linguagem em si, mas a ausência de competências de resolução de problemas. Os autores identificam um conjunto de fatores convergentes: metodologias de ensino inadequadas, métodos de estudo pouco eficazes adotados pelos próprios estudantes e o alto grau de abstração exigido desde as primeiras semanas de contato com a disciplina. O estudante que chega à aula de algoritmos sem habilidade prévia para decompor um problema em partes menores, identificar padrões e construir soluções incrementais tende a fracassar mesmo com instrução de qualidade.

Essa constatação tem implicações diretas para a forma como se compreende o papel da IAG nesse cenário. A inteligência artificial generativa não criou as dificuldades históricas do ensino de algoritmos; elas antecederam em décadas a chegada de qualquer ferramenta de código automático. O que a IAG fez foi tornar mais aguda e visível uma tensão que sempre existiu: a possibilidade de o estudante obter o produto correto (o código que funciona) sem desenvolver o processo que a disciplina pretende cultivar, que é a capacidade de pensar sobre o problema. Compreender essa distinção é pré-requisito para qualquer abordagem pedagógica responsável em cursos de computação na era atual.

Essa tensão entre produto e processo remete diretamente ao que a literatura denomina pensamento computacional. Wing (2006) propõe que o pensamento computacional constitui uma competência fundamental para todas as pessoas, não apenas para profissionais de computação, do mesmo modo que leitura, escrita e aritmética integram a formação básica de qualquer cidadão. A autora o define como um modo

de pensar que envolve resolver problemas, projetar sistemas e compreender o comportamento humano a partir dos conceitos fundantes da ciência da computação. Entre esses conceitos, destacam-se a abstração, a decomposição de problemas complexos, o reconhecimento de padrões e o raciocínio sobre algoritmos. O argumento central de Wing é preciso: pensar computacionalmente não equivale a programar um computador. É pensar em múltiplos níveis de abstração, formular problemas de modo que soluções computacionais se tornem possíveis e saber distinguir o que é computável do que não é (Wing, 2006).

Esses elementos — abstração, decomposição, reconhecimento de padrões e raciocínio sobre algoritmos — permitem que o pensamento computacional seja trabalhado como objetivo pedagógico explícito, não como subproduto implícito do ensino de programação. No Brasil, o pensamento computacional ganhou reconhecimento formal com a Base Nacional Comum Curricular (BNCC) em 2018, mas Pontual Falcão e França (2021) mostram que a formalização do mandato não se traduziu em mudança consistente nas escolas: o conhecimento dos docentes sobre o tema permanece incipiente e a formação de professores para integrar o pensamento computacional em suas disciplinas segue sendo o principal gargalo. O cenário que essa literatura descreve no nível da educação básica reproduz-se, com outras feições, no ensino superior tecnológico: a automação do produto não elimina a necessidade de formação do processo. Uma ferramenta de IA generativa pode gerar código correto para um enunciado específico sem que o estudante percorra nenhuma das etapas cognitivas que a disciplina visa cultivar. O pensamento computacional, enquanto objetivo pedagógico, permanece irreduzível: não é delegável a uma ferramenta.

A chegada das ferramentas de IA generativa ao ambiente educacional produziu, nos últimos anos, uma literatura crescente sobre seus usos, riscos e possibilidades. Kasneci et al. (2023) oferecem um panorama sistemático das oportunidades e desafios que os modelos de linguagem de grande escala representam para a educação. Entre as oportunidades, identificam-se o potencial para personalização do processo de aprendizagem, criação de conteúdo adaptado, geração de *feedback* imediato e ampliação do acesso a suporte pedagógico. Entre os desafios, apontam a dependência excessiva dos estudantes de respostas automáticas, riscos associados a vieses nos dados de treinamento, questões de direitos autorais e ameaças à integridade acadêmica. Os autores são categóricos quanto a um ponto: os LLMs têm papel assistivo, não substitutivo, e não são capazes de substituir o docente na gestão das relações pedagógicas e na avaliação do aprendizado real (Kasneci et al., 2023).

No campo específico da educação em programação, Prather et al. (2024) conduziram um dos primeiros estudos com estudantes iniciantes utilizando o GitHub Copilot em disciplinas introdutórias. Os resultados revelam tanto as atrações quanto as armadilhas da ferramenta para esse público: os estudantes relataram surpresa com a capacidade do assistente de antecipar o que queriam escrever, mas enfrentaram dificuldades cognitivas e metacognitivas expressivas, incluindo incapacidade de avaliar a correção das sugestões recebidas e tendência a aceitar código sem compreendê-lo. O risco de dependência, portanto, não é apenas comportamental; é epistêmico: o estudante pode construir uma ilusão de competência que só se revela problemática quando privado da ferramenta.

Evidências empíricas complementam esse diagnóstico. Jošt, Taneski e Karakatič (2024), em experimento com trinta e dois estudantes de graduação ao longo de dez semanas, encontraram correlação negativa significativa entre o uso de LLMs para geração de código e para depuração e as notas finais. O uso das mesmas ferramentas para buscar explicações adicionais, contudo, não produziu correlação negativa de igual magnitude. A distinção tem implicações diretas para a prática docente: LLMs como suporte à compreensão podem ter papel legítimo na aprendizagem; LLMs como substitutos do raciocínio computacional tendem a comprometer o desenvolvimento das competências que a disciplina visa formar (Jošt; Taneski; Karakatič, 2024).

No Brasil, Silva et al. (2024) realizaram um mapeamento sistemático de estudos sobre IAG no ensino de programação, analisando quinze trabalhos empíricos. O ChatGPT aparece como a ferramenta mais utilizada em aproximadamente 80% dos casos. Os estudos identificam melhorias na resolução de problemas, na motivação e na qualidade do código produzido. Contudo, os mesmos estudos registram desafios consistentes: inconsistências nas respostas geradas, risco de dependência que inibe o desenvolvimento de autonomia e preocupações com integridade acadêmica (Silva et al., 2024). Azevedo e Wainer (2024), a partir de experiência prática na UNICAMP, defendem a integração em vez da proibição, mas documentam o desafio

específico que a IAG representa para os exercícios tradicionais de programação: grande parte deles já pode ser resolvida automaticamente, o que exige reconfiguração das práticas avaliativas e o deslocamento do foco para tarefas que exijam compreensão contextualizada.

A constatação de que a mediação pedagógica é o fator diferenciador entre uso produtivo e uso evasivo da IA convida a uma reflexão sobre o papel do docente na era das ferramentas generativas. Mishra e Koehler (2006) propõem o *framework* TPACK (Technological Pedagogical Content Knowledge) como modelo para compreender o conhecimento docente necessário à integração eficaz de tecnologia ao ensino. O argumento central dos autores é que tecnologia, pedagogia e conteúdo formam três domínios de conhecimento cujas intersecções geram formas distintas de saber docente, e que nenhuma boa prática pedagógica emerge da adição mecânica de tecnologia a um currículo preexistente. O docente precisa compreender as relações entre os três domínios e tomar decisões que levem em conta simultaneamente o que ensinar, como ensinar e com qual suporte tecnológico, sem jamais presumir que existe uma solução tecnológica universalmente válida (Mishra; Koehler, 2006).

Aplicado ao ensino de algoritmos com IAG, o *framework* implica que o docente não pode simplesmente tolerar ou proibir a ferramenta: precisa tomar decisões pedagógicas fundamentadas sobre como a tecnologia transforma o que é possível ensinar e aprender. O professor que compreende o conteúdo (algoritmos), a pedagogia (como se desenvolve o raciocínio lógico em iniciantes) e a tecnologia (o que a IAG faz e deixa de fazer) está em condições de construir práticas que aproveitam as potencialidades da ferramenta sem abrir mão dos objetivos formativos da disciplina. Azambuja e Ferreira da Silva (2024) sustentam que as universidades precisam repensar suas estratégias pedagógicas diante de ferramentas que assumem, em larga medida, a transmissão de conhecimento explícito, tarefa que historicamente cabia ao docente. O professor, nessa perspectiva, passa a ter como função primordial desenvolver o pensamento crítico, a criatividade e a capacidade de tomada de decisão ética dos estudantes, orientando-os a formular os problemas corretos e não apenas a encontrar as respostas automáticas que a IA pode fornecer (Azambuja; Ferreira Da Silva, 2024).

O horizonte formativo que emerge dessa perspectiva não se resolve apenas pela mediação tecnológica do docente; exige também que o estudante desenvolva competências que as ferramentas de IA não são capazes de substituir. Pressman e Maxim (2016), em obra de referência sobre engenharia de *software*, definem o desenvolvimento de sistemas como uma atividade fundamentalmente humana, cujos desafios mais complexos não estão na geração do código, mas nas etapas que a precedem: a comunicação com o cliente, a eliciação de requisitos em contextos de ambiguidade, as decisões de arquitetura com restrições reais e a manutenção de sistemas que evoluem ao longo do tempo. Nenhuma dessas etapas admite automação completa, porque todas envolvem julgamento contextualizado e responsabilidade por decisões que têm consequências concretas no mundo real.

Cambaz e Zhang (2024), em revisão sistemática da literatura sobre ferramentas de geração de código assistida por IA no ensino de programação, identificam que esses recursos podem ser produtivos tanto para estudantes quanto para docentes, desde que os riscos associados sejam ativamente gerenciados. Entre os riscos documentados, destaca-se a dependência excessiva nas saídas geradas, que inibe o desenvolvimento de competências de resolução de problemas, precisamente as que a formação em computação visa cultivar. A revisão aponta ainda que o conhecimento de programação permanece condição necessária para o uso eficaz dessas ferramentas: sem essa base, o desenvolvedor não consegue formular pedidos precisos, avaliar o que foi gerado nem identificar onde a solução proposta falha. Formar um desenvolvedor de *software* é, portanto, construir essa base antes de delegar à ferramenta; e isso exige tempo, prática deliberada e mediação pedagógica que nenhum assistente automático, até o momento, substitui (Cambaz; Zhang, 2024).

O fenômeno denominado *vibe coding* representa a radicalização mais recente da tensão que este relato discute. O termo foi cunhado pelo cientista da computação Andrej Karpathy em fevereiro de 2025 para descrever uma prática em que o desenvolvedor descreve o que quer em linguagem natural, delega a geração do código inteiramente à IA e aceita os resultados sem necessariamente lê-los ou compreendê-los (Sarkar; Drosos, 2025). Meske et al. (2025) definem o *vibe coding* como um paradigma de desenvolvimento em que humanos e IA engajam em um fluxo colaborativo para co-criar artefatos de *software* por meio de diálogo em linguagem natural, deslocando a mediação da intenção do desenvolvedor da instrução determinística para a

inferência probabilística. Em outras palavras: onde antes o desenvolvedor precisava traduzir sua intenção em sintaxe formal e precisa, agora ele a expressa em linguagem ordinária e a IA assume a responsabilidade de interpretá-la e convertê-la em código executável.

Esse deslocamento não elimina a necessidade de competência em programação — ele a redistribui. Sarkar e Drosos (2025), no primeiro estudo empírico sobre a prática, analisaram mais de oito horas de sessões gravadas de *vibe coding* com reflexões em voz alta e concluíram que a expertise não é substituída, mas reorientada: da escrita direta de código para a avaliação de código gerado, a gestão de contexto e as decisões sobre quando intervir manualmente. Li et al. (2025), em estudo com vinte e dois profissionais de equipes de produto, identificaram uma tensão análoga entre eficiência (produzir o artefato rapidamente) e reflexão (garantir que o artefato produzido corresponde à intenção real). Essa tensão entre produto e processo, que este relato situa no ensino de algoritmos, reaparece nas equipes reais que já operam com *vibe coding* como prática cotidiana.

O risco educacional que emerge desse cenário é preciso. Meske et al. (2025) documentam que, à medida que o desenvolvimento migra para a engenharia de prompts, canais tradicionais de transmissão de conhecimento tácito, como mentoria, programação em pares e revisão de código, são enfraquecidos, com impacto negativo particular sobre desenvolvedores menos experientes. Sarkar e Drosos (2025) reforçam: a capacidade de avaliar código gerado por IA, identificar onde ele falha e decidir quando intervir manualmente depende de uma base técnica que o *vibe coder* sem formação não possui, e que nenhuma ferramenta de geração automática é capaz de construir por ele. Formar o raciocínio computacional, nesse horizonte, não se torna menos necessário: torna-se a competência que distingue o desenvolvedor que tem agência sobre a ferramenta daquele que tem subserviência a ela.

4. A EXPERIÊNCIA: DECISÕES, TENSÕES E ADAPTAÇÕES

As experiências narradas nesta seção envolveram ao todo quatro turmas: duas conduzidas pelo professor Marcos (37 estudantes em 2024.1 e 32 em 2025.1) e uma pelo professor William em 2026.1, composta por 114 estudantes de dois períodos distintos, 97 do primeiro e 17 do segundo, que cursam a disciplina conjuntamente. Os depoimentos de estudantes citados ao longo da seção foram obtidos de forma voluntária: os estudantes se dispuseram espontaneamente a compartilhar suas percepções e consentiram com o uso de suas falas, respondendo por mensagem de texto. Não se trata de amostragem sistemática: as falas foram selecionadas por sua capacidade de ilustrar tensões e percepções recorrentes observadas pelos docentes ao longo do processo.

4.1 A experiência do professor Marcos: construir a disciplina enquanto a IAG chegava

O professor Marcos viveu um desafio duplo: inaugurar a disciplina de Algoritmos Computacionais na EDUFOR em 2024.1 e, simultaneamente, responder à presença crescente da IAG nas entregas dos estudantes. Sua trajetória entre 2024 e 2025 é, em si mesma, um processo de adaptação pedagógica em tempo real.

A primeira mudança perceptível veio nas entregas. Atividades que antes geravam textos e códigos heterogêneos, com marcas de sintaxe, de vocabulário, de erros e de raciocínio de cada estudante, passaram a apresentar linguagem homogênea, sofisticada demais para alguns perfis e desconectada das discussões feitas em sala. O diagnóstico foi rápido: o problema não era apenas desonestidade acadêmica, mas o risco de terceirização do processo de aprendizagem.

A resposta foi reformular as avaliações para tornar o processo visível e exigível. Um exemplo concreto: onde antes se pedia um texto sobre impactos da tecnologia na sociedade, passou-se a exigir também que o estudante relacionasse o tema a uma situação discutida em aula, gravasse uma defesa oral curta ou apresentasse exemplos vinculados ao seu contexto local ou profissional. Em alguns casos, as atividades passaram a ter etapas progressivas: rascunho, discussão, revisão e versão final, dificultando a substituição do raciocínio por uma resposta gerada automaticamente.

Uma tentativa que não funcionou foi a proibição. Vedar o uso de IAG nas atividades gerou mais dificuldade de controle do que aprendizado: alguns estudantes escondiam o uso, outros ficavam inseguros até para recorrer a ferramentas legítimas de apoio. O foco, percebeu-se, não deveria ser vigiar o uso da

ferramenta, mas desenvolver critérios para que o estudante a utilizasse de forma crítica e transparente, deixando claro quando o uso era permitido, em que condições, e o que precisava ser autoria inequívoca do estudante: a análise, a reflexão e o posicionamento.

O que o professor Marcos preservou ao longo desse processo foi o que já orientava sua prática antes da IAG: a convicção de que decorar conteúdo tem pouco valor se o estudante não consegue interpretar, relacionar e argumentar. Os momentos de diálogo em sala, os debates e as atividades colaborativas permaneceram — porque a IAG alterou a forma de produzir informação, mas não substituiu a construção coletiva nem a capacidade de defender ideias diante de outras pessoas.

O episódio que cristalizou essa percepção foi preciso. Um estudante entregou uma atividade tecnicamente impecável: texto bem estruturado, referências organizadas, linguagem cuidadosa. Durante a apresentação, porém, não conseguia explicar conceitos básicos que constavam do próprio trabalho. Diante de perguntas sobre exemplos práticos ligados ao contexto da disciplina, travou completamente. Foi ali que o desafio pedagógico se tornou inequívoco: não se tratava de combater a tecnologia, mas de impedir que ela substituísse o esforço cognitivo necessário para aprender.

4.2 A postura de partida: não proibir, integrar

Se a experiência do professor Marcos foi construída na tensão entre a chegada da IAG e a necessidade de responder a ela em tempo real, o professor William assumiu a disciplina em 2026.1 já com um diagnóstico claro e uma postura definida: não proibir, integrar.

Quando o professor William assumiu a disciplina de Algoritmos Computacionais, a pergunta sobre o uso de IA pelos estudantes já estava dada, não como possibilidade futura, mas como realidade presente. A decisão pedagógica de não proibir o uso de ferramentas de IAG não decorreu de omissão ou permissividade, mas de uma convicção explícita: o mercado de trabalho para o qual se está formando esses estudantes já opera com essas ferramentas, e prepará-los para ignorá-las seria uma desonestidade formativa.

Essa postura foi comunicada aos estudantes desde as primeiras aulas. A IA generativa foi apresentada não como ameaça ao processo de aprendizagem, nem como solução mágica, mas como ferramenta de produtividade — a mesma lógica que orienta, por exemplo, o uso de um depurador (*debugger*) ou de uma IDE avançada. Assim como nenhum professor de programação proíbe o uso de IDEs com autocomplete, proibir a IAG seria anacrônico.

4.3 O que a IA revela sobre como os estudantes aprendem

Uma das percepções mais significativas emergidas da experiência diz respeito ao que o uso (e o mau uso) da IAG revela sobre os processos de aprendizagem dos estudantes. Em disciplinas anteriores, como Programação para Web, o professor William identificou respostas que carregavam marcas inequívocas de geração automática: linguagem homogênea, estrutura padronizada, ausência de erros que seriam naturais em estudantes iniciantes e, sobretudo, ausência da compreensão do que estava sendo entregue.

O que a IAG muda não é a natureza do problema, pois a evitação da aprendizagem antecede em décadas qualquer ferramenta de geração automática, mas a escala e a sofisticação do produto gerado, tornando mais urgente a pergunta sobre o que o estudante está de fato aprendendo.

A resposta pedagógica adotada não foi punitiva, mas deslocadora: em vez de avaliar apenas o produto (o algoritmo correto), passou-se a valorizar o processo — a capacidade do estudante de explicar, defender e modificar a solução apresentada. Um estudante que entrega um algoritmo correto gerado por IAG, mas não consegue explicar por que uma estrutura de repetição foi escolhida em detrimento de outra não desenvolveu a competência que a disciplina se propõe a desenvolver. Um estudante do primeiro período, em depoimento sobre o próprio processo, chegou a essa distinção de forma direta: “Gerar um código não necessariamente está ligado ao aprendizado daquele código.”

4.4 IA como instrumento pedagógico em sala

Além de tolerar o uso de IAG pelos estudantes, o professor William passou a utilizá-la ativamente como recurso didático. Em sala de aula, soluções geradas por IA são apresentadas e analisadas coletivamente: o que o algoritmo faz? Por que essa abordagem? Onde ele poderia falhar? Existe uma solução mais eficiente?

Esse movimento transforma a IAG de objeto de suspeita em objeto de estudo. O estudante aprende não apenas a usar a ferramenta, mas a questionar sua saída — competência que o mercado de trabalho exigirá de qualquer desenvolvedor que utilize assistentes de código. Não basta saber pedir à IA; é preciso saber avaliar o que ela entrega.

Da mesma forma, a preparação das aulas passou a contar com apoio de IAG: roteiros, exemplos, exercícios e slides são gerados com assistência da ferramenta e validados pelo docente antes de serem utilizados. Isso não representa perda de autoria intelectual — representa ganho de produtividade, liberando tempo e energia para o que a IA não faz: a gestão das relações pedagógicas, a adaptação em tempo real às dificuldades da turma e o julgamento sobre o que é relevante ensinar.

Essas escolhas expressam, na prática, uma das intersecções que Mishra e Koehler (2006) descrevem no *framework* TPACK: o conhecimento pedagógico tecnológico (TPK), que emerge não da adição mecânica da ferramenta ao currículo, mas de decisões situadas sobre como tecnologia, conteúdo e pedagogia se articulam.

4.5 A extensão como espaço de convivência produtiva com a IA

Simultaneamente à disciplina de Algoritmos, o professor William conduz com a mesma turma um projeto de extensão curricularizada — o Portal de Ideias — no qual os estudantes são desafiados a desenvolver soluções digitais para problemas reais da comunidade, utilizando IA generativa como ferramenta de apoio ao processo criativo e técnico.

Como prática de responsabilização pelo processo, os estudantes são orientados a registrar os prompts utilizados e submetê-los junto ao relatório e ao código-fonte gerado, diretamente no AVA. Essa exigência cumpre duas funções pedagógicas simultaneamente: torna o uso da IA visível e analisável e desloca o foco avaliativo do produto para o processo, tornando o uso da IA exigível e não apenas permitido.

A articulação entre as duas disciplinas é pedagogicamente intencional. Se em Algoritmos o foco é o desenvolvimento do raciocínio lógico-computacional, no projeto de extensão o estudante precisa mobilizar esse raciocínio em contexto real — interagindo com usuários, levantando requisitos, tomando decisões de *design* e comunicando soluções para não especialistas. Nesse espaço, a IAG aparece em seu papel mais adequado: como acelerador do processo criativo e técnico, não como substituto do julgamento humano. Essa perspectiva encontra respaldo teórico no *framework* GCED (Gestão do Conhecimento na Extensão Curricularizada Mediada por Tecnologias Digitais), proposto pelo primeiro autor em trabalho paralelo¹, que posiciona a IAG como amplificador nas etapas de externalização e combinação do conhecimento, preservando a centralidade das interações humanas nos processos de socialização e internalização (Corrêa Mendes; Bezerra de Melo; Do Rosario Miranda, 2026).

A experiência reforça uma percepção central: a IA generativa é capaz de propor soluções, gerar código e estruturar argumentos. O que ela não faz — e o estudante precisa aprender a fazer — é compreender o problema do usuário, negociar expectativas, lidar com ambiguidade e tomar decisões que envolvem contexto, responsabilidade e ética. Essas são competências humanas e continuam sendo o núcleo da formação em ADS.

4.6 O que permanece insubstituível

Há uma pergunta que o professor William traz explicitamente para os estudantes: *se a IA já resolve o problema, por que você ainda precisa aprender a resolver?*

A resposta que ele oferece não é retórica. O desenvolvedor de *software* não é aquele que resolve algoritmos no vácuo: é aquele que conversa com o cliente, entende o problema que o cliente nem sempre

¹ O primeiro autor, William Corrêa Mendes, é autor principal do *framework* GCED citado neste relato.

sabe descrever, traduz ambiguidade em especificação técnica e decide, com responsabilidade, qual solução implementar. A IA pode gerar o código. Ela não pode fazer a pergunta certa para o cliente errado.

Essa distinção encontra confirmação na trajetória de um estudante que cursou Algoritmos no primeiro período, com o professor Marcos, e chegou ao Desenvolvimento Web no quinto. Em depoimento sobre o percurso, ele identificou o ponto central de sua aprendizagem: “O mais importante dessa fase foi aprender a quebrar problemas grandes em partes menores. Quando fui para o Desenvolvimento Web, isso me ajudou muito: eu já conseguia entender melhor como funcionava a lógica por trás das funcionalidades.”

Pensar algoritmicamente é, portanto, menos sobre saber codificar e mais sobre saber decompor problemas, identificar padrões, questionar premissas e comunicar raciocínio. São essas capacidades que a disciplina de Algoritmos precisa desenvolver e que nenhuma ferramenta de IAG ainda tornou obsoletas. Como aponta o *framework* GCED, desenvolvido pelo primeiro autor em trabalho paralelo, a inteligência artificial generativa atua como amplificador de processos cognitivos já constituídos, mas não os substitui: a socialização e a internalização do conhecimento permanecem ancoradas nas interações humanas (Corrêa Mendes; Bezerra de Melo; Do Rosario Miranda, 2026).

5. REFLEXÕES E IMPLICAÇÕES

A distinção entre uso produtivo e uso evasivo da IAG, central às reflexões desta seção, pode ser sintetizada conforme o Quadro 1.

Quadro 1 — Uso evasivo e uso produtivo da IAG no ensino de algoritmos

Dimensão	Uso evasivo	Uso produtivo
Relação com a ferramenta	IA como substituto do raciocínio	IA como interlocutora do processo
Foco da atividade	Produto (código correto)	Processo (raciocínio desenvolvido)
Papel do estudante	Passivo — recebe e entrega	Ativo — formula, questiona e avalia
Competência desenvolvida	Nenhuma	Metacognição e pensamento computacional
Sinal observável	Não consegue explicar a solução	Consegue defender, modificar e questionar

Fonte: elaborado pelos autores.

A experiência narrada neste relato não constitui um modelo replicável por definição: é uma resposta situada, construída por dois docentes em uma instituição específica, com uma turma específica, em um momento histórico cujas características tecnológicas mudam rapidamente. Reconhecer essa limitação não diminui o que a experiência revela; ao contrário, é o que permite extrair dela implicações que outros docentes possam examinar criticamente em seus próprios contextos.

A primeira implicação diz respeito à escolha entre proibir e integrar. Lau e Guo (2023), em estudo com vinte docentes de ciência da computação de nove países diferentes, documentaram que a polarização entre proibição e adoção irrestrita é a postura mais comum na primeira reação dos professores à IAG, e que nenhuma das duas extremidades tende a produzir resultados formativos satisfatórios. A experiência do professor William sinaliza um terceiro caminho: a integração deliberada, construída sobre decisões pedagógicas explícitas. Isso não significa aceitar qualquer uso da IA sem questionamento. Significa reconhecer que o uso existirá de qualquer maneira e que o papel do docente é criar condições para que ele seja produtivo, orientado ao desenvolvimento da competência, não à sua evasão.

A distinção entre uso produtivo e uso evasivo da IAG não é óbvia para o estudante e raramente emerge por si só. O uso evasivo ocorre quando a IA funciona como atalho que contorna o processo de aprendizagem: o estudante submete o enunciado, recebe o código, entrega a atividade e não percorre nenhuma etapa do raciocínio que a atividade pretendia desenvolver. O uso produtivo ocorre quando a IA é mobilizada como

interlocutora do processo: o estudante formula hipóteses, consulta a ferramenta, analisa as respostas recebidas, questiona inconsistências e constrói sua solução a partir de um diálogo ativo com a ferramenta. A diferença está no engajamento metacognitivo; o docente pode criar condições para o segundo por meio de práticas avaliativas que tornam o processo visível e exigível. Um estudante do primeiro período, em depoimento sobre o próprio uso das ferramentas, formulou essa fronteira com clareza: “Não funciona pedir algo sem o conhecimento prévio de como deve ser a entrega. IA ajuda muito, porém, sem saber o que se faz, perde-se o controle e o domínio do resultado final.” O que o estudante descreve é, precisamente, a diferença entre usar a ferramenta como amplificador de uma competência já constituída e usá-la como substituto de uma competência que ainda não existe.

A exigência de registro de prompts, adotada pelo professor William, é uma dessas práticas. Ela funciona como mecanismo de responsabilização que desloca o foco avaliativo do produto para o processo: o que o estudante perguntou à IA, em que sequência, como refinou as perguntas e de que forma incorporou as respostas ao próprio raciocínio. Esse registro é, ele mesmo, uma forma de metacognição e uma competência transferível para outros contextos profissionais. O desenvolvedor de *software* que trabalha com ferramentas de IA em equipes reais precisará articular o que pediu à ferramenta, por que descartou determinadas sugestões e como validou o que foi aceito. A prática de registrar prompts não é apenas uma estratégia de controle pedagógico: é antecipação de uma habilidade profissional concreta. Silveira Nascimento et al. (2026), em pesquisa com quarenta docentes da educação básica em Manaus, identificaram que o engajamento dos estudantes com IA generativa tende a ser mais produtivo quando acompanhado de mediação crítica e intencional por parte do professor, o que reforça a pertinência de práticas pedagógicas que não apenas permitem o uso da ferramenta, mas o estruturam deliberadamente.

A articulação entre a disciplina de Algoritmos e o projeto de extensão Portal de Ideias representa outro princípio que merece atenção por parte de docentes em situações formativas similares. A extensão curricularizada não é acessório: é o espaço onde o raciocínio desenvolvido em Algoritmos encontra o contexto real que lhe dá sentido. Um estudante que aprende a decompor um problema em sala pode não perceber, sem esse contraponto, que a competência mais difícil não é escrever o algoritmo correto, mas compreender o problema que o usuário real tem, muitas vezes antes do usuário ele mesmo conseguir descrevê-lo. A articulação intencional entre componentes curriculares, com a IAG presente nos dois espaços, mas com papéis diferentes em cada um, constitui uma estratégia de formação integral que o ensino modular isolado não alcança.

Por fim, é necessário nomear o que este relato não é. Não há grupo de controle, nem medidas padronizadas de desempenho, nem generalização estatística possível a partir de uma experiência com uma turma em uma instituição. As percepções dos docentes são subjetivas e podem conter vieses de confirmação. O que o relato oferece é densidade de contexto e honestidade sobre as decisões tomadas, o que, na tradição do relato de experiência docente, tem valor epistêmico próprio. Outros docentes em contextos comparáveis poderão reconhecer situações análogas, avaliar as abordagens descritas criticamente e, se for o caso, adaptá-las ou refutá-las. É esse o convite que o relato estende.

6. CONSIDERAÇÕES FINAIS

A pergunta que abre este artigo (“como ensinar algoritmos quando a IA já resolve por você?”) não encontra uma resposta definitiva nas experiências aqui narradas. O que as experiências do professor Marcos e do professor William revelam é que essa pergunta contém uma premissa que merece exame: a de que o ensino de algoritmos tem como objetivo principal a produção de código correto. Se o objetivo fosse esse, a IAG já o tornaria obsoleto. Mas o objetivo nunca foi apenas esse. O ensino de algoritmos forma a capacidade de decompor problemas, abstrair padrões, construir modelos mentais e comunicar raciocínio com precisão técnica, competências que nenhuma ferramenta atual substitui e que todo desenvolvedor de *software* precisará mobilizar, com ou sem IA ao lado.

A postura pedagógica descrita neste relato pode ser sintetizada em duas escolhas complementares: integrar em vez de proibir e mediar em vez de ignorar. Integrar significa reconhecer a IAG como realidade do mercado para o qual se está formando o estudante e construir práticas que convivam com ela sem abdicar dos objetivos formativos. Mediar significa criar as condições pedagógicas que distinguem o uso produtivo da

ferramenta de seu uso evasivo: o registro de prompts, o questionamento coletivo das saídas da IA, a articulação com a extensão, a avaliação do processo e não apenas do produto. Nenhuma dessas escolhas é isenta de dificuldade. Ambas exigem do docente um esforço contínuo de diagnóstico e adaptação que a mera vigilância sobre o uso da IA não demanda.

O horizonte que se desenha nos próximos anos radicaliza a tensão que este artigo discute. Como mostram Sarkar e Drosos (2025) e Meske et al. (2025), a formação do raciocínio computacional não se torna menos necessária nesse cenário — torna-se a competência de supervisão, avaliação e decisão que dá ao desenvolvedor agência sobre a ferramenta, em vez de subserviência a ela. Zhang et al. (2026), em análise de mais de três mil e oitocentos bugs produzidos por agentes de codificação autônomos, documentam que a maioria das falhas ocorre em integrações de APIs e configurações de ambiente: exatamente os domínios que exigem julgamento arquitetural e contextual, não geração automática.

A comunidade docente de cursos tecnológicos no Brasil está diante de um momento que exige mais do que atualização de ementa ou adoção de novas ferramentas. Exige repensar o que significa formar um desenvolvedor de *software* quando o código pode ser gerado por máquinas. A aposta deste relato é que a resposta está no processo: no raciocínio que precede o código, na pergunta que antecede a resposta, na compreensão do problema que torna a solução possível. Esse processo não é automatizável, e é aí que reside o argumento mais consistente para continuar ensinando algoritmos, com toda a seriedade e intencionalidade que a disciplina merece. Partilhamos esta experiência na esperança de que outros docentes reconheçam, em seus próprios contextos, situações análogas e contribuam com respostas igualmente situadas para um desafio que é, por ora, de todos nós.

REFERÊNCIAS

- AZAMBUJA, Celso Candido De; FERREIRA DA SILVA, Gabriel. Novos desafios para a educação na Era da Inteligência Artificial. **Filosofia Unisinos**, v. 25, n. 1, p. 1–16, 26 mar. 2024.
- AZEVEDO, Rodolfo; WAINER, Jacques. Ensino de Programação (e Computação) na Era da IA Generativa. In: RIBEIRO, Leila; DURAN, Rodrigo (Orgs.). **Seminário dos Grandes Desafios da Educação em Computação no Brasil**. 1. ed. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 101–105.
- CAMBAZ, Doga; ZHANG, Xiaoling. Use of AI-driven Code Generation Models in Teaching and Learning Programming: a Systematic Literature Review. In: SIGCSE 2024: THE 55TH ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION. **Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1**. Portland OR USA: ACM, 7 mar. 2024. Disponível em: <<https://dl.acm.org/doi/10.1145/3626252.3630958>>. Acesso em: 20 maio. 2026.
- CORRÊA MENDES, William; BEZERRA DE MELO, Alessandra; DO ROSARIO MIRANDA, Marcio Antonio. FRAMEWORK GCED: GESTÃO DO CONHECIMENTO NA EXTENSÃO CURRICULARIZADA MEDIADA POR TECNOLOGIAS DIGITAIS. **REVISTA ELETRÔNICA EXTENSÃO EM DEBATE**, v. 15, n. 25, maio 2026.
- GOMES, Anabela; MENDES, A. J. Learning to Program - Difficulties and Solutions. In: Coimbra, Portugal: ICEE, 2007. Disponível em: <<http://icee2007.dei.uc.pt/proceedings/papers/411.pdf>>.
- JOŠT, Gregor; TANESKI, Viktor; KARAKATIČ, Sašo. The Impact of Large Language Models on Programming Education and Student Learning Outcomes. **Applied Sciences**, v. 14, n. 10, p. 4115, 13 maio 2024.
- KASNECI, Enkelejda *et al.* ChatGPT for good? On opportunities and challenges of large language models for education. **Learning and Individual Differences**, v. 103, p. 102274, abr. 2023.
- LAU, Sam; GUO, Philip. From “Ban It Till We Understand It” to “Resistance is Futile”: How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In: ICER 2023: ACM CONFERENCE ON INTERNATIONAL COMPUTING EDUCATION RESEARCH. **Proceedings of the 2023 ACM Conference on International Computing Education Research V.1**. Chicago IL USA: ACM, 7 ago. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3568813.3600138>>. Acesso em: 20 maio. 2026.

- LI, Jie *et al.* **Vibe Coding in Product Teams: Reconfiguring AI-Assisted Workflows, Prototyping, and Collaboration.** arXiv, 2025. Disponível em: <<https://arxiv.org/abs/2509.10652>>. Acesso em: 20 maio. 2026.
- MESKE, Christian *et al.* Vibe Coding as a Reconfiguration of Intent Mediation in Software Development: Definition, Implications, and Research Agenda. **IEEE Access**, v. 13, p. 213242–213259, 2025.
- MISHRA, Punya; KOEHLER, Matthew J. Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. **Teachers College Record**, v. 108, n. 6, p. 1017–1054, 2006.
- PONTUAL FALCÃO, Taciana; FRANÇA, Rozelma Soares De. Computational Thinking Goes to School: Implications for Teacher Education in Brazil. **Revista Brasileira de Informática na Educação**, v. 29, p. 1158–1177, 2 out. 2021.
- PRATHER, James *et al.* “It’s Weird That it Knows What I Want”: Usability and Interactions with Copilot for Novice Programmers. **ACM Transactions on Computer-Human Interaction**, v. 31, n. 1, p. 1–31, 29 fev. 2024.
- PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software: uma abordagem profissional**. 8. ed. Porto Alegre, RS: AMGH, 2016.
- ROBINS, Anthony; ROUNTREE, Janet; ROUNTREE, Nathan. Learning and Teaching Programming: A Review and Discussion. **Computer Science Education**, v. 13, n. 2, p. 137–172, jun. 2003.
- SARKAR, Advait; DROSOS, Ian. **Vibe coding: programming through conversation with artificial intelligence.** arXiv, 2025. Disponível em: <<https://arxiv.org/abs/2506.23253>>. Acesso em: 20 maio. 2026.
- SILVA, Teresinha Letícia Da *et al.* Potencialidades do uso de Inteligência Artificial Generativa como apoio ao Ensino de Programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. **Anais do XXXV Simpósio Brasileiro de Informática na Educação (SBIE 2024)**. Brasil: Sociedade Brasileira de Computação - SBC, 4 nov. 2024. Disponível em: <<https://sol.sbc.org.br/index.php/sbie/article/view/31369>>. Acesso em: 20 maio. 2026.
- SILVEIRA NASCIMENTO, Márcio; PIRES MARTINS, Sidney; JOSÉ DOS SANTOS, Éber. Professores e inteligência artificial generativa (IAG): mediação, desafios e oportunidades para o engajamento discente. **Devir Educação**, v. 10, n. 1, p. e-1116, 3 mar. 2026.
- WING, Jeannette M. Computational thinking. **Commun. ACM**, v. 49, n. 3, p. 33–35, mar. 2006.
- ZHANG, Ruixin *et al.* **Engineering Pitfalls in AI Coding Tools: An Empirical Study of Bugs in Claude Code, Codex, and Gemini CLI.** arXiv, 2026. Disponível em: <<https://arxiv.org/abs/2603.20847>>. Acesso em: 20 maio. 2026.