v7.n11.2026

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

A ESCADA DE PROGRAMAÇÃO: EXPERIÊNCIA PEDAGÓGICA NO ENSINO DE ALGORITMOS NA AMAZÔNIA

THE PROGRAMMING LADDER: PEDAGOGICAL EXPERIENCE IN TEACHING ALGORITHMS IN THE AMAZON

LA ESCALERA DE PROGRAMACIÓN: EXPERIENCIA PEDAGÓGICA EN LA ENSEÑANZA DE ALGORITMOS EN LA AMAZONÍA

Delfa Huatuco Zuasnábar[1], Miguel Flores Santibanez[2], Leonardo Vinícius Lima Castro[3], João Sidney Rodrigues[4], Salvador Malavé Campos[4]

e7119442
https://doi.org/10.47820/recima21.v7i11.9442

PUBLICADO: 11/2026

RESUMO

O ensino introdutório de programação em cursos de Tecnologia da Informação enfrenta desafios globais relacionados a altas taxas de retenção e dificuldades na aprendizagem, fenômeno frequentemente associado à sobrecarga cognitiva gerada pelo aprendizado simultâneo de lógica e sintaxe complexa. Este relato de experiência com avaliação descritiva de abordagem quantitativa documenta a implementação do projeto de extensão "Trajetória de Aprendizagem para Programação: de iniciantes a expertos" na Universidade Federal de Roraima (UFRR), em Boa Vista-RR, que adotou a metodologia da "Escada de Programação". A estratégia fundamenta-se na fragmentação do aprendizado em três níveis de abstração: (1) fortalecimento do pensamento computacional via programação visual baseada em blocos no Code.org; (2) formalização algorítmica em língua materna através do Portugol Studio; e (3) consolidação técnica em linguagem profissional utilizando a linguagem Python. A intervenção, estruturada em 40 horas de oficinas presenciais (avaliada via questionário estruturado de autopercepção aplicado em delineamento de amostras pareadas com N=64 participantes, sendo 25 da comunidade acadêmica e 39 da comunidade externa), contou com o suporte de monitoria por pares para mediar a transição cognitiva. Os resultados indicam que 85% dos discentes participantes manifestaram maior segurança técnica para enfrentar as disciplinas curriculares após as oficinas. Além do impacto acadêmico, o projeto promoveu ações de extensão em Boa Vista-RR, desmistificando o ato de codificar para a comunidade e contribuindo para o fortalecimento da confiança dos participantes em atividades introdutórias de programação. Conclui-se que a dissociação gradual entre lógica e sintaxe constitui uma alternativa pedagógica promissora para a introdução à programação.

PALAVRAS-CHAVE: Ensino de Programação. Pensamento Computacional. Code.org. Portugol Studio. Python.

[1] Mestra em Ciência da Computação pelo Instituto Tecnológico de Aeronáutica (ITA). Professora dos cursos de Bacharelado em Ciência da Computação e Licenciatura em Informática, na modalidade a distância, da Universidade Federal de Roraima (UFRR).
[2] Doutor em Informática na Educação pela Universidade Federal do Rio Grande do Sul (UFRGS) e mestre pelo Instituto Tecnológico de Aeronáutica (ITA). Professor dos cursos de Bacharelado em Ciência da Computação e Licenciatura em Informática, na modalidade a distância, da Universidade Federal de Roraima (UFRR).
[3] Graduando em Ciência da Computação pela Universidade Federal de Roraima (UFRR), com atuação nas áreas de desenvolvimento de software e tecnologias educacionais.
[4] Graduando em Ciência da Computação pela Universidade Federal de Roraima (UFRR).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

ABSTRACT
Introductory programming education in Information Technology courses faces global challenges related to high failure rates and learning difficulties, a phenomenon often associated with the cognitive overload caused by simultaneous learning of logic and complex syntax. This experience report with descriptive quantitative evaluation documents the implementation of the extension project "Learning Trajectory for Programming: from beginners to experts" at the Federal University of Roraima (UFRR), in Boa Vista, Roraima, Brazil, which adopted "The Programming Ladder" methodology. The strategy is based on fragmenting learning into three abstraction levels: (1) strengthening computational thinking via block-based visual programming in Code.org; (2) algorithmic formalization in the native language through Portugol Studio; and (3) technical consolidation in a professional language using Python. The intervention, structured in 40 hours of face-to-face workshops (evaluated through a structured self-perception questionnaire applied in a paired-samples design with N=64 participants, including 25 academic community members and 39 external community members), relied on peer mentoring to mediate the cognitive transition. Results indicate that 85% of participating students expressed greater technical confidence to face curricular subjects after the workshops. In addition to academic impact, the project promoted extension activities in Boa Vista, demystifying coding for the community and contributing to strengthening participants' confidence in introductory programming activities. It is concluded that the gradual dissociation between logic and syntax constitutes a promising pedagogical alternative for introductory programming.

KEYWORDS: *Programming Education. Computational Thinking. Code.org. Portugol Studio. Python.*

RESUMEN
La enseñanza introductoria de programación en cursos de Tecnología de la Información enfrenta desafíos globales de reprobación y deserción, asociados a la sobrecarga cognitiva por el aprendizaje simultáneo de la lógica y la sintaxis compleja. Este relato de experiencia con evaluación descriptiva de enfoque cuantitativo documenta el proyecto de extensión "Trayectoria de Aprendizaje para la Programación" en la Universidad Federal de Roraima (UFRR), Boa Vista, que adoptó la metodología de la "Escalera de Programación". La estrategia se fundamenta en tres niveles de abstracción: (1) fortalecimiento del pensamiento computacional mediante programación visual en bloques en Code.org; (2) formalización algorítmica en lengua materna con Portugol Studio; y (3) consolidación técnica en Python. La intervención, estructurada en 40 horas de talleres presenciales (evaluada mediante un cuestionario de autopercepción aplicado en un diseño de muestras pareadas a N=64 participantes, de los cuales 25 pertenecen a la comunidad académica y 39 a la externa), contó con tutoría entre pares para mediar la transición cognitiva. Los resultados indican que el 85% de los estudiantes manifestaron mayor seguridad técnica para enfrentar las asignaturas curriculares. Además del impacto académico, el proyecto promovió acciones de extensión en Boa Vista, desmitificando la codificación y fortaleciendo la confianza de los participantes. Se concluye que la disociación gradual entre lógica y sintaxis constituye una alternativa pedagógica prometedora para la introducción a la programación.

PALABRAS CLAVE: *Enseñanza de la Programación. Pensamiento Computacional. Code.org. Portugol Studio. Python.*

INTRODUÇÃO

O ingresso no ensino superior em cursos de Tecnologia da Informação (TI) é marcado por desafios de adaptação acadêmica que influenciam a permanência e o desempenho dos

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

estudantes nos semestres iniciais. De um lado, encontram-se as expectativas dos ingressantes, motivados pela expansão do mercado profissional e pela ubiquidade da tecnologia; de outro, surgem exigências de raciocínio abstrato e rigor formal que demandam reconfigurações nas estratégias de estudo. O descompasso entre a percepção inicial do estudante e a complexidade conceitual da formação tecnológica exige abordagens pedagógicas estruturadas, visando favorecer a transição para o ambiente universitário e reduzir os índices de retenção e evasão.

No cenário nacional, os dados consolidados pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira – INEP (Brasil, 2024) nos relatórios do Censo da Educação Superior revelam que os cursos das áreas de Computação e Tecnologia da Informação registram taxas históricas de evasão acumulada que chegam a ultrapassar 50% ao longo da trajetória acadêmica. A descontinuidade do percurso curricular concentra-se criticamente no primeiro ano de graduação, período em que os índices de desistência e trancamento são acentuados pelas elevadas taxas de retenção em disciplinas introdutórias, a exemplo de Algoritmos e Programação (Morais; Neto; Osório, 2020). Levantamentos educacionais e as estatísticas plurianuais do INEP (Brasil, 2024) apontam que a combinação entre a alta seletividade conceitual e a complexidade sintática precoce gera barreiras que contribuem para o abandono antes mesmo da consolidação do ciclo básico. Essa realidade impõe a necessidade urgente de reestruturar as abordagens pedagógicas tradicionais, substituindo modelos excludentes por estratégias de suporte instrucional que favoreçam a permanência e o aprimoramento contínuo da aprendizagem.

Na Universidade Federal de Roraima (UFRR), essa dinâmica assume relevância estratégica diante da necessidade de promover a retenção de talentos e a formação de profissionais no contexto da Amazônia Setentrional. No curso de Bacharelado em Ciência da Computação, a disciplina introdutória de Algoritmos e Programação apresenta-se historicamente como um ponto crítico do percurso curricular, acompanhando o panorama mapeado pelas estatísticas do INEP em que taxas expressivas de reprovação e abandono afetam a fluidez acadêmica e a motivação discente (Medeiros; Ramalho; Falcão, 2019). Para enfrentar essa limitação, a reestruturação das estratégias de ensino pauta-se nos princípios do pensamento computacional, empregando a decomposição de problemas para organizar o aprendizado em etapas de complexidade gradualmente crescente (Santana Filho, 2025).

A proposta pedagógica organiza-se em três fases encadeadas para acompanhar o desenvolvimento lógico do estudante: inicia-se pela abstração visual no Code.org, progride para a formalização algorítmica em língua materna por meio do Portugol Studio e consolida-se com a implementação técnica em Python. Essa progressão possibilita a assimilação prévia de

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

estruturas lógicas fundamentais — como fluxo de controle e repetição — antes da introdução à sintaxe e aos formalismos de uma linguagem profissional. Ao adotar esse suporte pedagógico escalonado (*scaffolding*), busca-se não apenas o domínio da codificação, mas a consolidação de esquemas cognitivos necessários à continuidade da formação acadêmica e ao aprimoramento do processo de ensino-aprendizagem.

O objetivo geral deste trabalho é relatar e analisar a aplicação da metodologia "Escada de Programação" no ensino introdutório de computação, enquanto os objetivos específicos contemplam investigar a mitigação da sobrecarga cognitiva, avaliar a autopercepção de segurança técnica dos discentes e verificar o alcance social das ações de extensão universitária em Boa Vista, Roraima. Justifica-se a investigação pela necessidade de propor alternativas fundamentadas para mitigar a retenção escolar decorrente da exigência simultânea de lógica de programação e sintaxe complexa nas etapas iniciais do curso, reconhecendo o caráter exploratório e formativo da intervenção sem assumir premissas de causalidade estrita sobre o desempenho longitudinal.

1. REFERENCIAL TEÓRICO

A Teoria da Carga Cognitiva (*Cognitive Load Theory – CLT*), formulada no final da década de 1980 pelo psicólogo educacional australiano John Sweller (1988), constitui um arcabouço da psicologia cognitiva dedicado a explicar como a arquitetura da memória humana processa e armazena informações durante a aprendizagem. A teoria fundamenta-se nas limitações estritas da memória de trabalho — que possui capacidade e duração restritas para manipular novos conteúdos simultaneamente —, estabelecendo uma relação direta entre o design instrucional e o fenômeno da sobrecarga cognitiva. Segundo Souza (2010), a sobrecarga cognitiva ocorre precisamente quando a soma das demandas de processamento imposta aos alunos ultrapassa a capacidade máxima dessa memória temporária.

Para estruturar essa análise, a CLT categoriza a carga cognitiva em três dimensões estruturantes: a intrínseca (inerente à complexidade conceitual do conteúdo), a extrínseca (gerada por ruídos, formatações inadequadas ou exigências sintáticas prematuras) e a germânea (dedicada à construção e automação de esquemas mentais duradouros). No campo educacional, a teoria demonstra que métodos tradicionais de ensino que expõem o aprendiz a múltiplas complexidades em paralelo disparam a sobrecarga cognitiva extrínseca, bloqueando o processamento da memória de trabalho e prejudicando a retenção. Por essa razão, a CLT fundamenta o desenvolvimento de suportes instrucionais graduais (*scaffolding*) e de abordagens

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

por níveis de abstração, orientados a gerenciar a carga mental e prevenir a sobrecarga no ensino de disciplinas complexas como a programação de computadores.

No âmbito da aprendizagem inicial de programação, o problema central identificado no estado inicial deste relato relaciona-se diretamente com o fenômeno da sobrecarga cognitiva na memória de trabalho durante a exposição precoce a formalismos sintáticos. Ao iniciar o aprendizado de programação diretamente por linguagens caracterizadas por sintaxes complexas e de alta densidade de regras formais (como C ou Java), o estudante é submetido ao processamento concorrente de duas dimensões de complexidade. A primeira compreende a dimensão lógico-conceitual (carga cognitiva intrínseca), associada à estruturação do raciocínio algorítmico e à resolução de problemas via pensamento computacional. A segunda refere-se à dimensão sintática (carga cognitiva extrínseca), que impõe o domínio simultâneo de convenções gramaticais formais, pontuação rigorosa e tradução de comandos em língua estrangeira.

Essa abordagem tradicional, pautada na exposição direta e precoce à escrita de código textual, estabelece uma barreira inicial que frequentemente desmotiva os estudantes nas primeiras semanas do curso. A frustração decorrente de erros de compilação por falhas sintáticas pontuais (como a omissão de delimitadores ou caracteres especiais) é, muitas vezes, equivocadamente associada a uma suposta limitação do raciocínio lógico. Como consequência, parcela expressiva dos ingressantes abandona a formação por considerar a programação uma aptidão inata. Nesse cenário, o curso de Bacharelado em Ciência da Computação da UFRR desenvolveu o projeto de extensão intitulado "Trajetória de Aprendizagem para Programação: de iniciantes a expertos", visando reconfigurar essa percepção e considerando que determinadas dificuldades de aprendizagem estão relacionadas às estratégias pedagógicas empregadas.

Para mitigar os índices de retenção e evasão mapeados no ensino superior de computação (Morais; Neto; Osório, 2020), torna-se fundamental a estruturação de suportes pedagógicos baseados no conceito de suporte instrucional progressivo (*scaffolding*). Diferentemente de um mero parcelamento cronológico de tópicos curriculares, a transição mediada por ferramentas organizadas em níveis crescentes de abstração possibilita o desacoplamento temporário entre a lógica de programação e a sintaxe formal, permitindo que os andaimes pedagógicos sejam retirados gradualmente à medida que a autonomia do estudante se consolida.

Sob a perspectiva da Teoria da Aprendizagem Significativa (Ausubel, 1963), cada etapa pedagógica atua mobilizando e consolidando ancoradouros cognitivos prévios (subsunçores), fornecendo a sustentação necessária para que novos conceitos lógicos sejam incorporados de forma não arbitrária e substantiva na estrutura cognitiva do aprendiz. Na intervenção proposta, o

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

raciocínio sequencial e espacial do cotidiano é mobilizado inicialmente na programação visual (Code.org); em seguida, o domínio da gramática em língua materna (Português) atua como subsunçor na transição para o pseudocódigo (Portugol Studio); por fim, a lógica já consolidada serve de ancoradouro para a escrita em linguagem profissional (Python). Essa estratégia otimiza a capacidade de processamento da memória de trabalho e favorece a consolidação de esquemas cognitivos duradouros. Complementarmente, a incorporação da instrução por pares (Mazur, 2015), viabilizada pela atuação de monitores discentes, atua como elemento facilitador da aprendizagem significativa e fortalecedor da autopercepção de segurança técnica dos estudantes.

Em consonância com a missão institucional de integração social e promoção do desenvolvimento regional, a UFRR mapeou na comunidade de Boa Vista, Roraima, uma crescente demanda por capacitação na área de Tecnologia da Informação. No entanto, constatou-se que o acesso inicial a esses conhecimentos frequentemente enfrentava barreiras decorrentes de abordagens pedagógicas predominantemente tecnicistas e focadas precocemente em formalismos sintáticos, o que desestimulava novos aprendizes. Diante desse panorama, o projeto foi concebido sob a premissa de que uma metodologia fundamentada em níveis graduais de abstração é capaz de abrandar a sobrecarga cognitiva e democratizar o aprendizado de computação para públicos heterogêneos, contemplando desde jovens em busca de inserção no mercado de trabalho até adultos em transição profissional. Assim, a iniciativa busca ir além da instrução técnica em programação, visando desmistificar o uso da tecnologia e posicioná-la como um recurso efetivo para o desenvolvimento do pensamento crítico e a resolução de problemas no contexto local.

2. METODOLOGIA

O presente estudo caracteriza-se como um relato de experiência de abordagem quantitativa e delineamento descritivo-exploratório, voltado a documentar e analisar a implementação da intervenção pedagógica na UFRR. A execução do projeto de extensão envolveu uma reestruturação da abordagem pedagógica, fundamentada na organização do aprendizado em três níveis distintos de abstração, cada um planejado para reduzir a incidência de sobrecarga cognitiva na memória de trabalho. Sob a perspectiva da Teoria da Aprendizagem Significativa (Ausubel, 1963), cada etapa busca mobilizar e consolidar ancoradouros cognitivos prévios (subsunçores), fornecendo a sustentação necessária para que os novos conceitos lógicos sejam incorporados de forma não arbitrária e substantiva. Denominada institucionalmente

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

"Escada de Programação", essa estratégia visa assegurar o domínio prévio das estruturas do pensamento computacional antes da introdução à sintaxe e ao formalismo das linguagens de programação profissionais.

Para viabilizar essa transição, estruturou-se um cronograma composto por 40 horas de oficinas presenciais — inseridas na estrutura global de 60 horas do projeto de extensão —, divididas em cinco marcos de aprendizagem organizados para acompanhar o desenvolvimento técnico progressivo dos participantes, conforme detalhado na Tabela 1. A condução das atividades contou com o suporte de ciclos regulares de *feedback* para acompanhamento do aprendizado, alinhando-se aos princípios de estratégias ativas de ensino (Mazur, 2015).

Tabela 1. Cronograma das oficinas (40 horas efetivas)

Marco de Aprendizagem	Ferramenta Base	Carga Horária	Foco Principal
Alfabetização Computacional	Code.org	8 horas	Lógica de blocos e algoritmos visuais
Formalização Algorítmica I	Portugol Studio	7 horas	Variáveis, Entrada/Saída e Decisão
Formalização Algorítmica II	Portugol Studio	7 horas	Laços de Repetição, Matriz e Função
Implementação Técnica I	Python	9 horas	Sintaxe, Operadores, Condição e Repetição
Implementação Técnica II	Python	9 horas	Listas, Dicionários, Funções, Arquivos e Projeto Final

Fonte: Elaborado pelos autores (2026).
Nota: Esta carga horária de 40 horas de interação direta é complementada pelas atividades de planejamento e gestão, totalizando a estrutura de 60 horas prevista para o módulo do projeto de extensão.

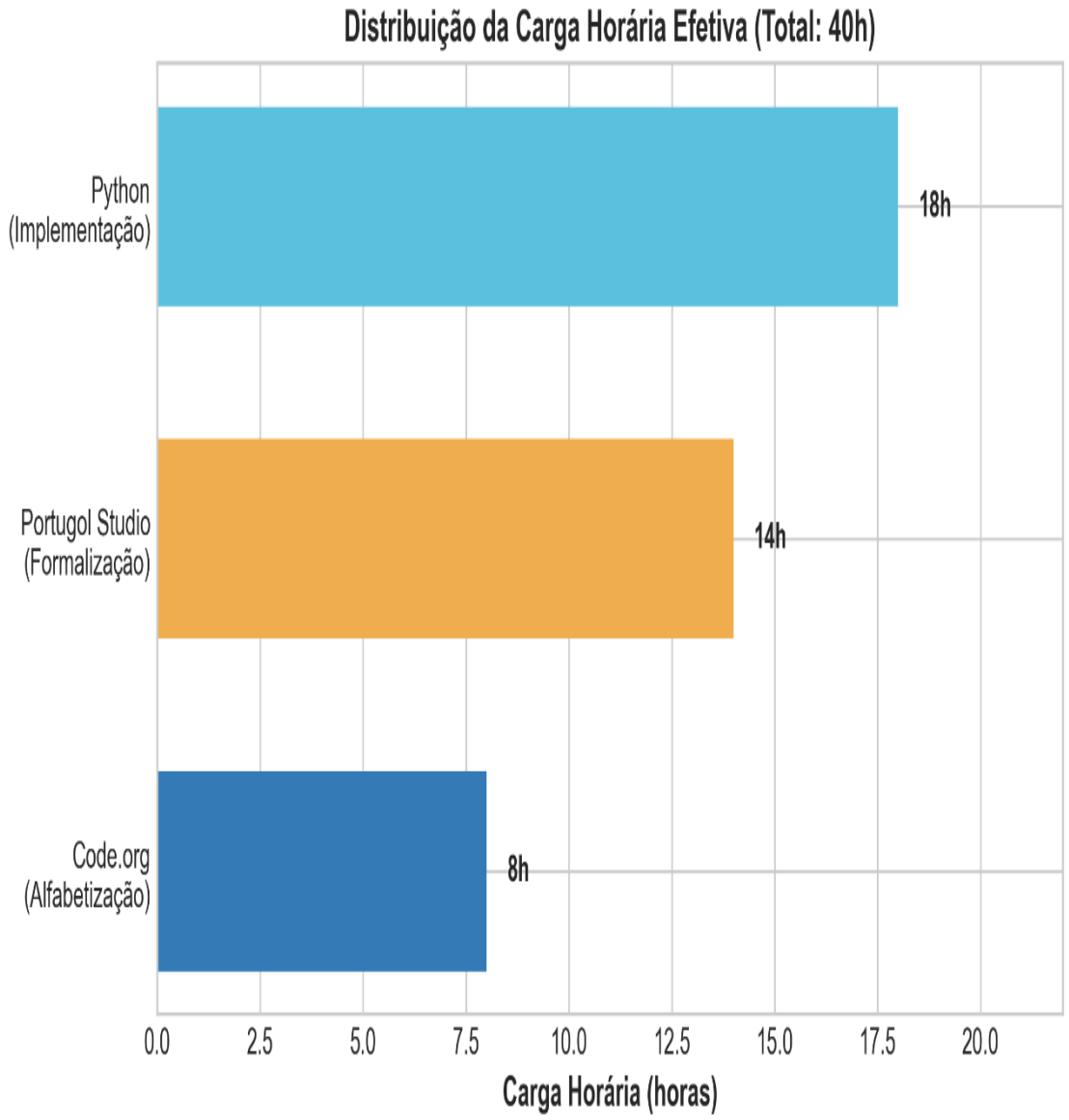
A intervenção pedagógica foi realizada com uma amostra de N=64 participantes, composta por 25 discentes do público interno da instituição e 39 membros da comunidade externa de Boa Vista, Roraima. A coleta de dados empíricos fundamentou-se na aplicação de um formulário estruturado de autopercepção técnica, administrado em dois momentos distintos: pré-intervenção (anterior ao início do módulo de introdução no Code.org, referente ao primeiro marco da Tabela 1) e pós-intervenção (ao término do ciclo de oficinas em Python). O instrumento utilizou uma escala ordinal composta por três níveis para mensurar a evolução da percepção de segurança dos participantes: (i) Inseguro, (ii) Neutro e (iii) Seguro. Esse procedimento possibilitou mensurar e comparar a variação na autopercepção de autoeficácia técnica diante dos conceitos de algoritmos e lógica de programação.

A organização do tempo pedagógico foi planejada para adequar gradualmente o nível de complexidade em cada etapa da intervenção. A Figura 1 apresenta a distribuição da carga

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

horária efetiva dedicada a cada nível de abstração contemplado na metodologia da "Escada de Programação".

Figura 1. Distribuição da carga horária efetiva por nível de abstração na metodologia da Escada de Programação



Fonte: Elaborado pelos autores (2026).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2.1. O estágio de alfabetização computacional (Fase 1: Code.org)

A intervenção inicia-se com o Estágio de Alfabetização Computacional, dedicando as primeiras 8 horas ao fortalecimento dos fundamentos do pensamento computacional (Wing, 2006) por meio da plataforma Code.org. Nesta etapa inicial, a programação estruturada em blocos visuais substitui representações abstratas estáticas que carecem de interatividade e retorno imediato. A utilização dessa ferramenta para iniciantes possibilita a assimilação de conceitos estruturantes de forma tátil e interativa, atuando como elemento facilitador no ensino inicial de computação (Suwarso, 2025). Ao manipular blocos visuais, os discentes resolvem problemas lógicos que introduzem, de modo gradual, estruturas fundamentais de controle, como laços de repetição e desvios condicionais.

O aspecto técnico central desse estágio reside na eliminação prévia de erros de sintaxe ou digitação: ao prescindir da memorização de comandos textuais, o participante direciona sua capacidade de processamento cognitivo prioritariamente à resolução do problema proposto. A interface gráfica e os elementos de gamificação reconfiguram a percepção do erro sintático — frequentemente associado à frustração em ambientes textuais tradicionais —, convertendo-o em uma etapa previsível de análise e depuração lógica. A criação desse ambiente de aprendizagem controlado é fundamental para estabelecer a segurança inicial necessária, especialmente para os participantes da comunidade externa sem experiência prévia em lógica de programação.

2.2. O estágio de formalização algorítmica (Fase 2: Portugol Studio)

Consolidada a percepção do fluxo lógico inicial, a proposta avança para o Estágio de Formalização Algorítmica, compreendendo 14 horas de atividades estruturadas no ambiente Portugol Studio. Este estágio desempenha papel intermediário no processo de transição, viabilizando a passagem da representação visual por blocos para a estruturação textual em pseudocódigo, em conformidade com as diretrizes do raciocínio algorítmico (Forbellone; Eberspächer, 2022). O emprego do Portugol favorece essa etapa ao permitir a assimilação da estrutura e da semântica dos comandos sem a interferência imediata de uma barreira idiomática (Mendes; Muniz, 2022). A adoção de instruções em língua materna (como leia() e escreva()) isola a construção conceitual da lógica de programação em relação às exigências de um idioma estrangeiro (Manzano; Oliveira, 2019).

Nessa etapa, os participantes são apresentados aos elementos formais da computação: tipagem e declaração de variáveis, manipulação de fluxos de entrada e saída de dados, estruturas condicionais e aninhamento de laços de repetição. A relevância pedagógica desta fase consiste em habituar o estudante à sistematização do raciocínio em formato textual,

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

utilizando a gramática em língua materna como suporte instrucional intermediário (*scaffolding*) antes da introdução aos formalismos de linguagens de programação profissionais.

2.3. O estágio de implementação e consolidação técnica (Fase 3: Python)

A etapa final da metodologia compreende o Estágio de Implementação em Linguagem de Alto Nível, no qual, ao longo de 18 horas, os conceitos previamente assimilados são consolidados na linguagem Python. A seleção do Python fundamenta-se na literatura sobre o ensino de computação (Coutinho, 2024), uma vez que sua sintaxe enxuta e elevada legibilidade apresentam-se como uma transição gradual em relação ao pseudocódigo utilizado na etapa anterior. Em função do nivelamento lógico e semântico construído nos estágios precedentes, verifica-se menor resistência técnica na transição para a linguagem textual profissional.

Nesta fase, a escrita do código passa a ser realizada como uma tradução direta da lógica algorítmica previamente estruturada. Os participantes progridem para a elaboração de scripts de automação, manipulação de estruturas de dados (como listas e dicionários) e desenvolvimento de pequenos projetos integradores voltados à resolução de problemas práticos. Essa aplicação em uma linguagem amplamente empregada nos meios acadêmico e profissional (em domínios como Ciência de Dados e Inteligência Artificial) contribui para o fortalecimento da autopercepção de capacidade técnica e atribui utilidade prática aos conhecimentos adquiridos.

2.4. A gestão pedagógica: monitoria e suporte por pares

Complementando essa estrutura tripartida, a atuação de discentes veteranos do curso de Ciência da Computação da UFRR como monitores desempenhou papel central na mediação pedagógica das oficinas. Por compartilharem um contexto acadêmico e geracional próximo ao dos ingressantes, esses discentes facilitaram a comunicação técnica e reduziram a distância hierárquica na relação de ensino-aprendizagem (Mazur, 2015). Essa abordagem alinha-se às estratégias contemporâneas de ensino de programação no ensino superior (Holanda; Freire; Coutinho, 2019) e às diretrizes do pensamento computacional para a educação básica e tecnológica (Brasil, 2024).

A atuação da monitoria possibilitou o acompanhamento contínuo e a adaptação do ritmo das oficinas. Pautando-se nos princípios de flexibilidade instrucional, a execução das atividades passou por ajustes periódicos com base em retornos formativos semanais e no acompanhamento do desempenho das turmas. Esse monitoramento visou assegurar a progressão contínua entre os níveis de abstração da "Escada de Programação" mediante o domínio prévio dos conceitos de cada etapa, promovendo a consolidação da aprendizagem.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

A dinâmica prática dessa mediação pedagógica em ambiente de laboratório é ilustrada na Figura 2, que apresenta a atuação discente durante o acompanhamento dos participantes em uma das oficinas presenciais de formalização algorítmica.

Figura 2. Atividade de monitoria por pares durante oficina de programação em laboratório



Fonte: Acervo dos autores (2026).

2.5. Aspectos éticos

O presente relato de experiência decorre das atividades acadêmicas vinculadas ao projeto de extensão institucional intitulado "Trajetória de Aprendizagem para Programação: de iniciantes a expertos", desenvolvido na UFRR. O enquadramento ético da investigação fundamenta-se nas diretrizes do Art. 1º, Parágrafo Único, da Resolução CNS nº 510/2016 do Conselho Nacional de Saúde, por tratar-se de avaliação de processos educacionais no âmbito de atividades regulares de ensino e extensão, cujos dados possuem caráter secundário e procedem de registros formativos e de acompanhamento pedagógico das oficinas.

A aplicação inicial dos instrumentos de autopercepção teve como finalidade primária o acompanhamento pedagógico e o aprimoramento formativo do projeto. A utilização analítica posterior das informações ocorreu de forma totalmente anonimizada, assegurando o sigilo e a confidencialidade dos dados dos N=64 participantes envolvidos. Previamente ao preenchimento dos formulários, os participantes receberam orientações acerca dos objetivos educacionais das oficinas e manifestaram anuência voluntária para o uso pedagógico e investigativo dos registros

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

consolidados, preservando-se o direito de não participação sem qualquer prejuízo acadêmico ou institucional.

Quanto à inclusão de registros fotográficos da dinâmica em laboratório (Figura 2), adotaram-se procedimentos de preservação da imagem e identidade dos sujeitos. As fotografias foram obtidas mediante consentimento informado para fins acadêmicos e institucionais, sendo editadas e enquadradas de modo a impedir a identificação facial dos discentes e monitores presentes, em estrito cumprimento às salvaguardas de privacidade e aos preceitos éticos vigentes.

2.6. Instrumento de coleta e tratamento dos dados de autopercepção

Para avaliar a evolução da percepção técnica dos participantes, utilizou-se um formulário estruturado de autopercepção técnica composto por 10 itens afirmativos direcionados à mensuração da competência frente ao aprendizado de algoritmos (disponibilizado integralmente no Apêndice A). O instrumento utilizou uma escala ordinal de três níveis: (1) Inseguro / Com Medo, (2) Neutro e (3) Seguro / Confiante.

Para avaliar a evolução temporal, adotou-se um delineamento de amostras pareadas (*paired-samples design*), garantindo que os mesmos N=64 participantes respondessem ao formulário em dois momentos distintos de aplicação: no pré-teste (anterior ao início do módulo no Code.org) e no pós-teste (ao término do ciclo de oficinas em Python), sem a ocorrência de perdas amostrais ao longo do projeto. O tratamento dos dados pautou-se na análise estatística descritiva, com apuração de frequências absolutas e percentuais das respostas em cada nível da escala ordinal, permitindo confrontar diretamente a distribuição prévia e posterior à intervenção pedagógica e fundamentando a análise comparativa apresentada na Figura 3.

3. RESULTADOS E DISCUSSÃO

Esta seção apresenta e discute os resultados obtidos a partir da intervenção pedagógica fundamentada na metodologia da "Escada de Programação", articulando a análise descritiva dos dados quantitativos de autopercepção com as observações qualitativas colhidas durante a execução do projeto de extensão na UFRR.

3.1. Avaliação da autopercepção técnica e análise do impacto pedagógico

A análise comparativa dos dados coletados através do formulário estruturado de autopercepção (N=64) nos dois momentos de aplicação (pré e pós-intervenção) revelou uma evolução expressiva na confiança dos participantes frente aos fundamentos de algoritmos e lógica de programação.

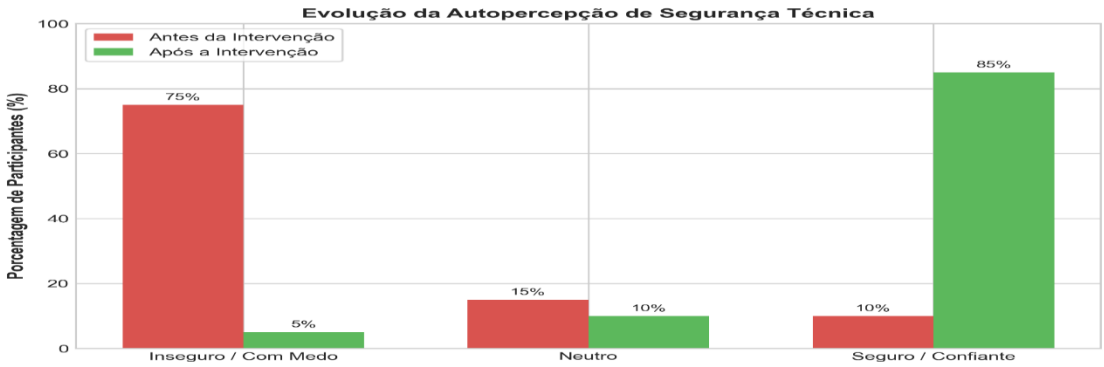
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

A Figura 3 ilustra a alteração percentual nas categorias da escala ordinal, indicando uma redução acentuada de 75% para 5% na proporção de respondentes alocados na categoria de insegurança/medo, em contrapartida a um salto expressivo de 10% para 85% na categoria de segurança/confiança técnica. Esses quantitativos absolutos e percentuais demonstram que a fragmentação do aprendizado em níveis de abstração progressivos — transcorrendo do Code.org e do Portugol Studio até a consolidação em Python — contribuiu de maneira efetiva para mitigar a barreira cognitiva inicial associada à aprendizagem simultânea de lógica e sintaxe complexa.

Cumpre ressaltar, contudo, que os resultados consolidados na autopercepção representam um indicador direto do nivelamento e da segurança formativa imediata dos discentes, não constituindo, isoladamente, comprovação exaustiva de aquisição definitiva de competências em programação ou de redução longitudinal de evasão acadêmica (Barichello, 2023). A validação da eficácia restringe-se ao escopo da intervenção proposta.

Ademais, observações qualitativas registradas em diários de campo ao longo das oficinas indicaram impressões favoráveis por parte da equipe docente e dos monitores quanto ao engajamento ativo dos alunos e à fluidez na transição entre os ambientes de ensino. Tais relatos da prática pedagógica, embora compreendidos como impressões qualitativas da vivência e não como métricas formais de eficácia isolada, corroboram a pertinência do alinhamento metodológico adotado para apoiar discentes ingressantes e membros da comunidade em suas primeiras experiências com a computação.

Figura 3. Comparativo da autopercepção de segurança técnica dos participantes pré e pós-intervenção no projeto de extensão



Fonte: Dados da pesquisa (2026).
Nota: Os percentuais apresentados refletem as proporções calculadas sobre o total da amostra de N=64 respondentes pareados (pré e pós-intervenção).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3.2. Reflexões teórico-metodológicas sobre os "andaimes" de aprendizagem

Os resultados obtidos viabilizam reflexões críticas fundamentadas no conceito de suporte instrucional progressivo (*scaffolding*) e na Teoria da Aprendizagem Significativa (Ausubel, 1963). Mapeou-se que a temporalidade na retirada gradual desses suportes constitui aspecto determinante para o sucesso da abordagem. A retenção excessiva no estágio de pseudocódigo pode gerar resistência motivacional em relação à sintaxe formal em língua estrangeira; assim, a transição para a linguagem Python deve ocorrer logo após a consolidação das estruturas de controle de fluxo, repetição e manipulação de variáveis (Mendes; Muniz, 2022; Coutinho, 2024).

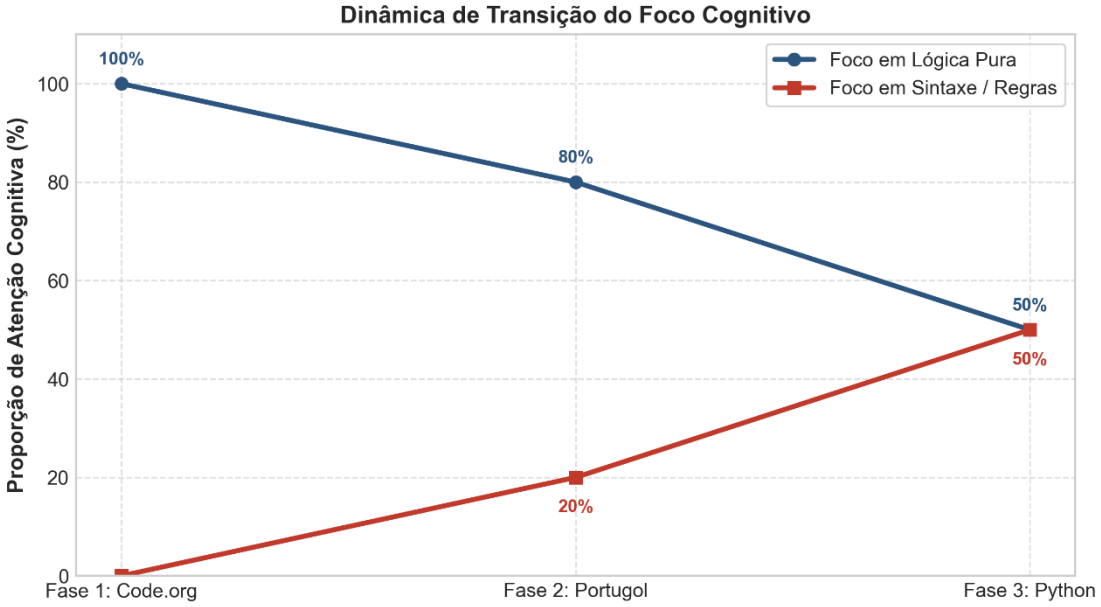
Outro aspecto relevante refere-se à reconfiguração da percepção do erro no processo formativo. O estágio inicial em ambiente visual (Suwarso, 2025) estabeleceu um contexto de experimentação protegida, favorecendo a internalização de estratégias de depuração contínua. Essa atitude reflexiva no diagnóstico de falhas lógicas mostrou-se relevante na etapa final do projeto, na qual o formalismo da linguagem Python exigia maior precisão sintática e conceitual, em alinhamento às diretrizes do ensino de programação no contexto nacional (Borges et al., 2018; Castro; Siqueira, 2019).

A análise da trajetória evidencia a adequação da progressão graduada de abstração no gerenciamento da carga cognitiva discente. A Figura 4 ilustra essa representação esquemática da dinâmica de alocação cognitiva ao longo dos três estágios metodológicos da "Escada de Programação":

- Fase 1 (Code.org — 100% Lógica / 0% Sintaxe): Foco exclusivo no pensamento computacional via blocos visuais, eliminando restrições sintáticas textuais.
- Fase 2 (Portugol Studio — 80% Lógica / 20% Sintaxe): Introdução da estrutura textual em língua materna, exigindo atenção inicial aos formalismos gramaticais.
- Fase 3 (Python — 50% Lógica / 50% Sintaxe): Consolidação em linguagem profissional, equilibrando a resolução de problemas e o domínio da sintaxe formal.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 4. Representação esquemática conceitual da alocação teórica da carga cognitiva (foco em lógica versus sintaxe) ao longo da jornada metodológica da "Escada de Programação"



Fonte: Elaborado pelos autores (2026).
Nota: O gráfico ilustra a projeção conceitual do design pedagógico, e não uma mensuração psicométrica direta.

3.3. Desafios de percurso e ajustes estratégicos

A execução do projeto também evidenciou obstáculos operacionais e conceituais que demandaram adequações pedagógicas ao longo da intervenção. Mapeou-se que a transição do Portugol Studio para o Python apresentou complexidades específicas relacionadas à abstração dos ambientes de desenvolvimento. Enquanto o Portugol Studio opera como um ambiente integrado e autocontido, a linguagem Python introduz conceitos adicionais, tais como o carregamento de módulos e a estruturação de bibliotecas, o que exigiu esforço cognitivo suplementar por parte dos participantes.

Como ajuste estratégico, definiu-se a utilização exclusiva das funcionalidades nativas da linguagem (*standard library*) durante as etapas iniciais de Python, postergando a introdução de dependências e ambientes virtuais de terceiros. Esse refinamento instrucional manteve a diretriz de progressão gradual de complexidade, assegurando que o foco da atenção permanecesse direcionado à tradução textual da lógica algorítmica. As adaptações efetuadas contribuíram para a estabilização do percurso formativo, atendendo tanto às necessidades de nivelamento dos discentes do curso de Ciência da Computação quanto às demandas de inclusão digital dos participantes da comunidade externa.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3.4. Avaliação de competências, autoeficácia e acessibilidade no contexto amazônico

A consolidação do ensino introdutório de programação requer a articulação entre a mensuração da autopercepção discente e a adoção de métodos objetivos de avaliação de competências computacionais em tarefas práticas de codificação (Medeiros; Ramalho; Falcão, 2019). Embora a elevação da autoeficácia técnica atue como indicador relevante para a retenção estudantil e a adaptação ao ingresso no ensino superior, a sustentabilidade desse processo depende da integração contínua de estratégias de avaliação formativa.

Essas estratégias devem contemplar a heterogeneidade e as disparidades de conhecimento prévio dos ingressantes, identificando lacunas conceituais sem recorrer a abordagens classificatórias que desestimulem a permanência. No contexto da Amazônia Setentrional, essa dinâmica pedagógica intersecta-se com desafios estruturais relativos à acessibilidade e infraestrutura tecnológica, nos quais assimetrias no acesso a equipamentos e conectividade estável influenciam o processo de aprendizagem. Nesse cenário, o emprego de ferramentas computacionais de baixo requisito de processamento, softwares de código aberto e metodologias de progressão graduada atua como mecanismo de facilitação instrucional e de mitigação de barreiras de acesso à educação tecnológica.

4. CONSIDERAÇÕES FINAIS

A execução desta intervenção extensionista na UFRR indicou que as dificuldades frequentemente associadas ao aprendizado inicial de programação podem ser mitigadas por meio de abordagens pedagógicas que consideram a gradação dos níveis de abstração. Configurando-se metodologicamente como um relato de experiência acompanhado de uma avaliação descritiva de abordagem quantitativa, a proposta fundamentou-se na dissociação estruturada entre os componentes lógicos, semânticos e sintáticos do raciocínio algorítmico.

Ao priorizar o desenvolvimento do pensamento computacional na etapa inicial por meio de programação baseada em blocos visuais (Code.org), a abordagem viabilizou a assimilação do raciocínio lógico em momento prévio ao domínio das regras gramaticais das linguagens textuais. A formalização subsequente em pseudocódigo e língua materna (Portugol Studio) atuou como suporte intermediário (*scaffolding*) para a organização conceitual, favorecendo a transição contínua para a linguagem Python. Nessa etapa final, a codificação textual passou a ser desempenhada pelos participantes como a transposição prática de conceitos lógicos previamente consolidados.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Cumpre ressaltar que os resultados apresentados decorrem de um estudo descritivo de caráter formativo com N=64 participantes (compostos por 25 discentes da comunidade acadêmica interna e 39 membros da comunidade externa). Embora os dados demonstrem avanços expressivos na autopercepção de segurança técnica e no aprimoramento do processo de aprendizagem dos participantes, o desenho metodológico empregado não permite inferir causalidade direta sobre o desempenho acadêmico longitudinal ou sobre a redução permanente de taxas de evasão. Recomenda-se, para investigações futuras, o acompanhamento longitudinal dos ingressantes e a condução de avaliações objetivas de desempenho técnico, visando aprofundar a verificação do impacto dessa abordagem na formação em computação e na inclusão digital na Amazônia Setentrional.

4.1. O impacto multidimensional: da academia à sociedade

O presente relato de experiência evidencia a contribuição da UFRR na articulação entre ensino, extensão e formação científica no contexto regional da Amazônia Setentrional. No âmbito do público interno, constituído pelos ingressantes do curso de Bacharelado em Ciência da Computação, a intervenção atuou como estratégia de nivelamento pedagógico, auxiliando no aprimoramento do desempenho inicial e na atenuação de assimetrias de conhecimento prévio decorrentes de diferentes trajetórias escolares — fatores frequentemente associados a dificuldades de acompanhamento e retenção nas disciplinas introdutórias.

Simultaneamente, junto à comunidade externa de Boa Vista, o projeto contribuiu para a disseminação do pensamento computacional e de linguagens de programação, promovendo a alfabetização digital para além do uso passivo de tecnologias. Ao atender participantes de perfis diversos, a aplicação da metodologia "Escada de Programação" favoreceu o desenvolvimento da autonomia técnica e a aproximação da sociedade com os fundamentos da Ciência da Computação.

4.2. Recomendações estratégicas para futuras intervenções

Com base nos aprendizados decorrentes da execução do projeto, recomendam-se diretrizes para investigações futuras voltadas ao aprimoramento da avaliação da aprendizagem e à mensuração do impacto de abordagens graduadas no ensino de computação:

1. Delineamento Comparativo: Propõe-se a realização de estudos experimentais ou quase-experimentais que comparem o desempenho prático, a retenção conceitual e as taxas de aprovação de discentes submetidos à progressão gradual (*Code.org–Portugol–Python*) em relação a grupos expostos diretamente ao ensino textual formal em linguagens profissionais. Esse delineamento permitirá isolar a contribuição específica da sequência pedagógica,

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

diferenciando seus efeitos daqueles decorrentes do aumento da carga horária, do suporte de monitoria por pares e da motivação inicial dos participantes.

2. Acompanhamento Longitudinal: Recomenda-se a implementação de mecanismos de monitoramento contínuo da trajetória acadêmica dos egressos ao longo da graduação, avaliando o impacto da intervenção sobre os índices institucionais de retenção, evasão e desempenho em disciplinas subsequentes da estrutura curricular.
3. Aplicação em Contextos Locais: Sugere-se que as atividades práticas finais da etapa de programação textual sejam orientadas ao desenvolvimento de soluções para problemas reais da comunidade local, fortalecendo a articulação entre a formação técnica e as demandas socioeconômicas e ambientais da região.

4.3. Limitações do estudo

A interpretação dos resultados obtidos neste trabalho deve considerar as limitações metodológicas inerentes ao seu delineamento. Em primeiro lugar, o estudo possui caráter descritivo e exploratório, estruturado como um relato de experiência com avaliação descritiva, sem a inclusão de um grupo de controle comparativo que permitisse isolar o potencial de aprimoramento pedagógico da sequência "Escada de Programação" frente a abordagens de ensino direto de linguagens textuais.

Em segundo lugar, o tamanho da amostra (N=64) reflete uma intervenção de extensão específica realizada na UFRR, o que limita a generalização estatística dos achados para outros contextos educacionais.

Por fim, os indicadores de segurança técnica e autoeficácia fundamentaram-se em relatos de autopercepção discente coletados por meio de um questionário estruturado de 10 itens afirmativos (escala ordinal de três níveis), o que reitera a necessidade de pesquisas futuras apoiadas em avaliações objetivas de desempenho prático e acompanhamento longitudinal dos índices de permanência acadêmica.

APÊNDICE A. Questionário estruturado de autopercepção de segurança técnica

Instruções: Marque a opção que melhor reflete sua percepção atual em relação ao aprendizado de algoritmos e lógica de programação. Escala de Resposta: [1] Inseguro / Com Medo | [2] Neutro | [3] Seguro / Confiante

1. Sinto-me capaz de compreender a lógica necessária para resolver um problema computacional.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2. Consigo acompanhar o fluxo de execução de um algoritmo sem me sentir desmotivado(a).
3. Tenho confiança para criar a sequência de passos necessária para resolver uma tarefa lógica.
4. Sinto segurança ao utilizar estruturas de controle condicional (como se/senão ou if/else).
5. Compreendo o funcionamento de laços de repetição (como enquanto/para ou while/for).
6. Consigo identificar a diferença entre a lógica de um algoritmo e a sintaxe escrita de uma linguagem.
7. Sinto-me preparado(a) para interpretar e corrigir erros de execução durante o aprendizado.
8. Tenho tranquilidade para expressar o raciocínio de programação em forma de código textual.
9. Sinto-me seguro(a) para cursar disciplinas acadêmicas ou realizar projetos práticos de programação.
10. Considero a programação uma ferramenta acessível e viável para o meu desenvolvimento acadêmico ou profissional.

REFERÊNCIAS

AUSUBEL, David Paul. **The psychology of meaningful verbal learning**. New York: Grune & Stratton, 1963.

BARICHELLO, Leonardo. **Pensamento computacional**. 2. ed. Rio de Janeiro: IMPA, 2023.

BORGES, Robinson P. et al. A systematic review of literature on methodologies, practices, and tools for programming teaching. IEEE Latin America Transactions, v. 16, n. 5, p. 1468-1475, 2018. Disponível em: https://doi.org/10.1109/TLA.2018.8408466. Acesso em: 1 set. 2026.

BRASIL. Ministério da Educação. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). **Censo da Educação Superior 2023**: notas estatísticas. Brasília: Inep, 2024. Disponível em: https://www.gov.br/inep/pt-br/areas-de-atuacao/pesquisas-estatisticas-e-indicadores/censo-da-educacao-superior. Acesso em: 1 set. 2026.

CASTRO, Ronney Moreira; SIQUEIRA, Sean. Metodologias, técnicas, ambientes e tecnologias alternativas utilizadas no ensino de algoritmos e programação no ensino superior no Brasil. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (CBIE), 8., 2019, Brasília. **Anais [...].** Brasília: SBC, 2019. p. 228-237. Disponível em: https://doi.org/10.5753/cbie.wcbie.2019.228. Acesso em: 10 set. 2026.

COUTINHO, N. **Introdução à programação com Python**: algoritmos e lógica de programação para iniciantes. 4. ed. São Paulo: Novatec, 2024.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação**: a construção de algoritmos e estruturas de dados com aplicações em Python. 4. ed. Porto Alegre: Bookman, 2022.

HOLANDA, Wallace Duarte; FREIRE, Laís de Paiva; COUTINHO, Jarbele Cássia da Silva. Estratégias de ensino-aprendizagem de programação introdutória no ensino superior: uma revisão sistemática da literatura. **RENOTE – Novas Tecnologias na Educação**, v. 17, n. 1, p. 527-536, 2019. DOI: 10.22456/1679-1916.95905. Disponível em: https://seer.ufrgs.br/index.php/renote/article/view/95905. Acesso em: 1 set. 2026.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 28. ed. São Paulo: Saraiva, 2019.

MAZUR, Eric. **Peer Instruction**: a revolução da aprendizagem ativa. Porto Alegre: Penso, 2015.

MEDEIROS, R. P.; RAMALHO, G. L.; FALCÃO, T. P. A systematic literature review on teaching and learning introductory programming in higher education. **IEEE Transactions on Education**, v. 62, n. 2, p. 77-90, 2019. DOI: 10.1109/TE.2018.2864133. Disponível em: https://ieeexplore.ieee.org/document/8447543. Acesso em: 11 set. 2026.

MENDES, J.; MUNIZ, R. **Lógica de programação com Portugol**: mais de 80 exemplos, 55 exercícios com gabarito e vídeos complementares. São Paulo: Casa do Código, 2022.

MORAIS, C.; NETO, F. M.; OSÓRIO, A. Difficulties and challenges in the learning process of algorithms and programming in higher education: a systematic literature review. **Research, Society and Development**, v. 9, n. 10, p. e9429109287, 2020. DOI: 10.33448/rsd-v9i10.9287. Disponível em: https://rsdjournal.org/rsd/article/view/9287. Acesso em: 2 set. 2026.

SANTANA FILHO, Orlando Soares de. **O pensamento computacional na educação básica:** percepções de docentes do Vale de São Patrício - Goiás. 2025. 51 f. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Sistemas de Informação) — Instituto Federal de Educação, Ciência e Tecnologia Goiano, Ceres, 2025.

SOUZA, Nelson Pinheiro Coelho de. **Teoria da carga cognitiva:** origem, desenvolvimento e diretrizes aplicáveis ao processo ensino-aprendizagem. 2010. Dissertação (Mestrado em Ensino de Ciências e Matemática) – Universidade Federal do Pará, Belém, 2010.

SUWARSO, Winarni. **Application of computational thinking on the Code.org learning platform for beginners.** Seattle: Kindle Direct Publishing, 2025. E-book (Kindle). Disponível em: https://www.amazon.com. Acesso em: 10 set. 2026.

SWELLER, John. Cognitive load during problem solving: Effects on learning. **Cognitive Science**, v. 12, n. 2, p. 257-285, 1988. Disponível em: https://doi.org/10.1016/0364-0213(88)90023-7. Acesso em: 4 set. 2026.

WING, Jeannette M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33-35, 2006. Disponível em: https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf. Acesso em: 8 set. 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.