DIFICULDADES DOS ESTUDANTES NA APRENDIZAGEM DE PROGRAMAÇÃO: UMA REVISÃO SISTEMÁTICA DA LITERATURA

STUDENTS' DIFFICULTIES IN LEARNING PROGRAMMING: A SYSTEMATIC REVIEW OF THE LITERATURE

Francisco Xavier da Silva
ORCID 0000-0001-7028-7623

IFMS - Campus Coxim
Coxim, Brasil
francisco.silva@ifms.edu.br

Crediné Silva de Menezes
ORCID 0000-0002-2709-7135

PPGIE - UFRGS
Porto Alegre, Brasil
credine@gmail.com

Alberto Nogueira de Castro Junior
ORCID 0000-0002-1752-8667

IComp – UFAM
Manaus, Brasil
alberto@icomp.ufam.edu.br

Resumo. O principal objetivo deste estudo foi revisar sistematicamente a literatura sobre as dificuldades dos estudantes no aprendizado de programação, abordando desafios conceituais, emocionais e práticos. A pesquisa seguiu o protocolo PRISMA, com uma busca sistemática de artigos publicados entre 2020 e 2024 nas bases de dados IEEExplore, Web of Science e Online Library Wiley. Os critérios de inclusão abrangeram estudos revisados por pares em português, inglês e espanhol, resultando na análise de 69 artigos. Os principais desafios encontrados incluem a dificuldade dos estudantes em compreender conceitos abstratos, como variáveis e loops, e a falta de recursos interativos que conectem teoria à prática. Estratégias eficazes para superar essas barreiras incluem o uso de metodologias ativas, como jogos sérios e plataformas de programação visual, além de robótica educacional e metodologias ágeis. Ferramentas adaptadas para estudantes com deficiência visual também foram mencionadas como fundamentais para a inclusão. O estudo sugere que a implementação de metodologias interativas e cooperativas, combinada com práticas inclusivas e tecnológicas, podem melhorar significativamente o aprendizado de programação. A formação continuada dos professores e a adaptação dos currículos são importantes para tornar o ensino de programação mais eficaz e acessível a uma ampla gama de estudantes.

Palavras-chave: Programação; Metodologias ativas; Inclusão; Aprendizagem.

Abstract. The main objective of this study was to systematically review the literature on students' difficulties in learning programming, addressing conceptual, emotional and practical challenges. The search followed the PRISMA protocol, with a systematic search of articles published between 2020 and 2024 in the IEEExplore, Web of Science, and Online Library Wiley databases. The inclusion criteria included peer-reviewed studies in Portuguese, English, and Spanish, resulting in the analysis of 69 articles. The main challenges encountered include the difficulty of students in understanding abstract concepts, such as variables and loops, and the lack of interactive resources that connect theory to practice. Effective strategies to overcome these barriers include the use of active methodologies, such as serious games and visual programming platforms, as well as educational robotics and agile methodologies. Tools adapted for visually impaired students were also mentioned as fundamental for inclusion. The study suggests that the implementation of interactive and cooperative methodologies, combined with inclusive and technological practices, can significantly improve programming learning. Continuing teacher training and the adaptation of curricula are important to make programming education more effective and accessible to a wide range of students.

Keywords: Programming; Active methodologies; Inclusion; Learning.

1. INTRODUÇÃO

O ensino de programação é fundamental no contexto atual, especialmente em um mundo orientado pela ciência, tecnologia, engenharia e matemática (STEM). A educação em programação constitui um importante motor de inovação na indústria e, para dominar essa habilidade, é necessário não apenas compreender a teoria, mas também vivenciar o que se aprende na prática (Uehara, 2020).

A solução de problemas, o raciocínio lógico e as construções sociais, comunicativas e interativas são partes essenciais do aprendizado de programação. Isso demanda dos sistemas educacionais contemporâneos uma formação intelectual significativa. Portanto, aprender programação contribui para a formação de redes de ecologia cognitiva, capazes de fomentar uma aprendizagem contínua e descentralizada (Uehara, 2020).

Dessa forma, as construções sociais e as interações podem ser entendidas como abordagens colaborativas e cooperativas no ambiente educacional. A primeira é compreendida como uma interação em que os indivíduos trocam pensamentos, mas suas ações não são necessariamente coordenadas de maneira profunda. Piaget (1973) define colaboração como "a reunião das ações que são realizadas isoladamente pelos parceiros, mesmo quando o fazem na direção de um objetivo comum" (p. 81). A segunda é definida em um nível mais elevado de interação, no qual "cooperar na ação é operar em comum, isto é, ajustar por meio de novas operações de correspondência, reciprocidade ou complementaridade, as operações executadas por cada um dos parceiros" (Piaget, 1973, p. 105).

Todavia, aprender programação não é uma tarefa simples, pois exige do estudante pensamento autônomo, flexível, criativo e autorregulado, características nem sempre presentes nas práticas de sala de aula. Diversas pesquisas, como as de Souza *et al*. (2016) e Ercan e Sales (2020), apontam dificuldades recorrentes enfrentadas pelos estudantes ao aprender e compreender os conceitos de programação, tais como a leitura e interpretação de problemas e a construção de soluções. Além disso, muitos estudantes consideram a programação altamente abstrata e desconectada de seus campos de estudo. Outro desafio é a ansiedade que enfrentam ao lidar com a programação, o que pode resultar em falta de interesse e baixa confiança em sua capacidade de aprender (Ercan & Sales, 2020).

Além das dificuldades dos estudantes, é importante considerar os desafios práticos que os professores encontram ao implementar metodologias de ensino de programação. A separação entre aulas teóricas e práticas compromete frequentemente a eficácia do processo de ensino, resultando na repetição de conceitos que pode desmotivar os estudantes (Ercan & Sales, 2020). Para minimizar esse efeito, os docentes podem adotar abordagens híbridas, utilizando ferramentas acessíveis, como plataformas de programação online gratuitas, a exemplo do Scratch, que permitem a integração entre teoria e prática em um único ambiente (Souza *et al*., 2016). Essa estratégia torna as aulas mais dinâmicas e interativas, mesmo em contextos com recursos limitados.

Outro obstáculo comum é a rigidez dos materiais de ensino, que frequentemente não oferecem a flexibilidade necessária para estimular o pensamento crítico e a criatividade dos estudantes. Nesse sentido, os professores podem personalizar atividades práticas e utilizar simuladores de programação disponíveis gratuitamente, possibilitando que os alunos pratiquem conceitos sem a necessidade de equipamentos físicos caros (Ercan & Sales, 2020). Além disso, a implementação de metodologias ágeis, como o eduScrum, pode ser adaptada a ambientes com menos recursos, por meio de atividades cooperativas simples, porém eficazes, que promovem a cooperação e o desenvolvimento de soluções criativas (Vieira Junior & Reategui, 2018).

Para que essas estratégias sejam bem-sucedidas, é essencial que os professores recebam formação continuada, especialmente no uso de tecnologias educacionais e metodologias inovadoras. Programas de capacitação a distância, oferecidos por universidades ou instituições

de tecnologia, podem apoiar os docentes na implementação dessas mudanças de forma eficaz, mesmo em contextos de recursos limitados (Vieira Junior & Reategui, 2018). Ao focar em soluções acessíveis e práticas, os professores podem transformar as aulas de programação em espaços mais dinâmicos e adaptados às necessidades dos estudantes, mesmo diante de desafios estruturais.

Nesse contexto, evidencia-se a necessidade de analisar a produção científica nacional e internacional sobre as dificuldades de aprendizagem em programação. Para alcançar esse objetivo, serão apresentados os resultados de uma Revisão Sistemática da Literatura (RSL), com a análise de artigos sobre o ensino de programação, considerando as dificuldades de aprendizagem sob a perspectiva do estudante. Serão analisados artigos publicados nos últimos cinco anos (2020–2024). O processo de pesquisa envolveu a busca em bases de dados eletrônicas nacionais e internacionais indexadas, como *IEEE Xplore*, *Web of Science* e *Wiley Online Library*.

Dessa forma, este artigo está organizado da seguinte maneira: a Seção 2 apresenta o planejamento e a condução da RSL; a Seção 3 expõe os resultados e a respectiva discussão; a Seção 4 traz considerações adicionais; e, por fim, a Seção 5 descreve as conclusões finais.

2. PLANEJAMENTO E CONDUÇÃO DO MAPEAMENTO SISTEMÁTICO

Conforme o protocolo PRISMA (2020), uma Revisão Sistemática da Literatura (RSL) segue um processo rigoroso de identificação, avaliação e interpretação das pesquisas disponíveis sobre um tópico ou fenômeno de interesse, com o objetivo de abordar questões relevantes sobre o tema. Esta RSL foi estruturada de acordo com as diretrizes estabelecidas pelos autores do protocolo mencionado.

O processo de busca e seleção dos estudos para esta RSL considerou que os estudos primários estivessem disponíveis na web. As bases de dados eletrônicas identificadas para a busca incluíram tanto bases nacionais quanto internacionais, como *IEEE Xplore, Web of Science* e *Wiley Online Library*. A sistematização dos critérios de inclusão e exclusão foi realizada por meio do Microsoft Excel.

2.1 Questões de pesquisa

Esta Revisão Sistemática da Literatura (RSL) tem como questão central a seguinte:

QP1 – Qual é o panorama das dificuldades enfrentadas pelos estudantes na aprendizagem de programação?

Para responder a essa questão principal, foram estabelecidas as seguintes perguntas complementares de pesquisa:

- **QP2** – Quais são os principais desafios que os estudantes encontram ao compreender os conceitos fundamentais de programação?
- **QP3** – Quais estratégias e recursos têm se mostrado mais eficazes para superar as barreiras na aprendizagem de programação, tanto para estudantes autodidatas quanto para aqueles inseridos em cursos formais?
- **QP4** – Quais reflexões emergem sobre a construção do conhecimento no ensino de programação?

2.2 Definição da estratégia de busca

A expressão de busca foi projetada para analisar os resultados obtidos a cada execução da *string* de busca nas bases selecionadas. As buscas foram construídas com o uso de operadores booleanos (*OR* e *AND*) entre os termos e seus sinônimos, aplicando a busca avançada com o filtro TITLE-ABS-KEY. O período de pesquisa foi limitado aos últimos cinco anos e incluiu artigos em fase final de publicação, escritos em português, inglês e espanhol.

Os resultados desta Revisão Sistemática da Literatura (RSL) apresentarão uma análise do estado da arte sobre as dificuldades enfrentadas pelos estudantes na aprendizagem de programação, tanto no contexto nacional quanto no internacional, considerando publicações em português, inglês e espanhol, no período de 2020 a 2024.

As palavras-chave escolhidas para definir a população de interesse incluíram: *Abordagens Pedagógicas*, *Educação a Distância*, *Didáticas Flexíveis* e Área de Aplicação = {Inteligência Artificial, Educação, Introdução à Programação, Programação, Cooperação}. A partir dessa seleção, os termos foram combinados, resultando na *string* de busca apresentada no Quadro 1.

Quadro 1: String de busca

STRING FINAL
("teaching programming" OR "learning programming" OR "education in programming") AND ("teaching methods" OR "teaching strategies" OR "pedagogical approaches").
("ensino de programação" OR "aprendizagem de programação" OR "educação em programação") AND ("métodos de ensino" OR "estratégias de ensino" OR "abordagens pedagógicas") AND ("linguagens de programação" OR "ferramentas de programação").

Fonte: Autor (2024)

A string de busca foi a mesma para todos os motores de busca, foi estruturada com o uso de conectores como *OR* e *AND*, além de parênteses para organizar as combinações de termos. Esses operadores e a estrutura da busca foram responsáveis por limitar os resultados da seleção. A abordagem de busca foi realizada em duas etapas. A primeira etapa de pré-seleção dos artigos consistiu na seleção manual de todos os artigos completos, com base em seus títulos, resumos e palavras-chave. Na segunda etapa, todos os artigos pré-selecionados foram analisados com a leitura dos resumos e conclusões. Em alguns casos, outras partes dos artigos também foram lidas para proceder com a seleção e aplicação dos critérios de inclusão e exclusão.

Os critérios de inclusão e exclusão foram elaborados para criar uma lógica clara para a pesquisa e selecionar um conjunto de estudos capazes de auxiliar nas respostas às questões de pesquisa.

2.3 Limitações da pesquisa

Esta revisão exigiu a categorização de termos utilizados por diferentes estudos primários para caracterizar os desafios no ensino de programação. No entanto, como é comum em questões relacionadas à educação, as categorias possuem limites difusos; por isso, um determinado termo lexical pode ser utilizado por diferentes autores com significados variados.

2.4 Critérios de exclusão e inclusão

Na sequência, foram definidos os critérios de inclusão e exclusão para que, após os resultados das buscas, fosse possível selecionar os artigos que se enquadram no contexto do estudo. Os critérios estão descritos no Quadro 2.

Quadro 2: Critérios de inclusão e exclusão

CRITÉRIOS DE INCLUSÃO:	CRITÉRIOS DE EXCLUSÃO:
Artigos completos que abordam o processo de ensino-aprendizagem de programação;	Artigos derivados da mesma pesquisa (estudos duplicados);
Artigo publicado entre os anos de 2020 e 2024;	Artigos que não estejam disponíveis em texto completo ou que não sejam acessíveis;

Artigo em português, inglês e espanhol;	Artigos irrelevantes (excluídos pelo título, resumo, palavras-chave que não estão relacionados aos objetivos desta RSL);
Artigo revisado por pares.	Artigos que não fornecem uma análise crítica ou discussão relevante sobre os métodos de ensino de programação.

Fonte: Autor (2024)

2.5 Processo de Seleção dos Estudos Primários

A escolha das bases de dados para a busca dos artigos foi feita de acordo com sua relevância para o ensino de programação e a Informática na Educação, já que essas áreas estão diretamente vinculadas ao tema do estudo. Foram selecionadas as seguintes bases: *IEEE Xplore (http://ieeexplore.ieee.org), Onlinelibrary-wiley (https://onlinelibrary.wiley.com/) e WebofScience (https://www.webofknowledge.com/).* Após a seleção das bases de dados e a elaboração da string de busca, o processo de busca foi iniciado e concluído em agosto de 2024. A busca resultou em um total de 105 estudos potenciais. Assim, a seleção dos estudos necessários a esta revisão seguiu uma abordagem de filtro e seleção em três etapas, conforme ilustrado na Figura 1.

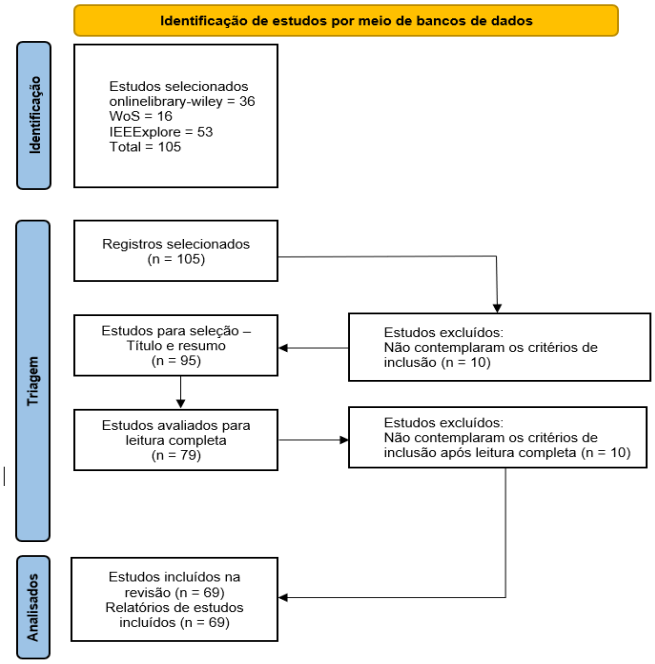

Figura 1. Fluxo da informação com as fases da revisão sistemática, adaptado e traduzido de PRISMA 2020**.** Fonte: Autor (2024)

A pré-seleção dos artigos consistiu na identificação e aplicação de filtros nas bases de dados para selecionar todos os artigos completos, seguindo os critérios: a) últimos cinco anos (2020 - 2024), b) somente artigos, c) revisados por pares ou em fase final de publicação, e d) nos idiomas português, inglês e espanhol. Como resultado, 105 estudos foram selecionados para a próxima etapa.

A triagem dos artigos consistiu em selecionar manualmente todos os artigos completos com base em seus títulos, resumos e palavras-chave. Os estudos que não apresentavam nos títulos, resumos ou palavras-chave alguma indicação de relevância para o estudo foram considerados irrelevantes e removidos, com base nos critérios de exclusão. Nos casos de dúvida sobre a

relevância do artigo para o estudo, ele foi mantido. Após essa etapa, 79 estudos foram selecionados.

Na última etapa, todos os artigos pré-selecionados foram lidos integralmente. Após a leitura, 10 estudos foram excluídos com base nos critérios de exclusão, restando 69 artigos que foram analisados e serviram a esta RSL.

Os estudos incluídos nesta RSL podem ser acessados pelo link: ***Seleção dos artigos da revisão RSL Ensino de programação***, onde foi especificado um identificador (ID) para distinguir o artigo durante a escrita das discussões, com o nome do artigo e sua respectiva citação e referência. O ID foi iniciado com a letra 'E' de estudo, seguida do número arábico para sequenciá-los; por exemplo, (E1) para o estudo número 1 dos achados da pesquisa.

2.6 Sumarização dos Resultados

Diversos estudos investigaram o uso de diferentes abordagens para ensinar programação, com foco em inclusão, gamificação, desenvolvimento de habilidades cognitivas e alinhamento com as demandas do mercado de trabalho. A inclusão no ensino de programação foi um tema recorrente, com estudos como o de (E1, E58), que exploraram o uso de tecnologias assistivas para tornar a programação acessível a estudantes com deficiências visuais e auditivas, respectivamente. Esses estudos demonstram que, mesmo com pequenas amostras, o uso de interfaces visuais e táteis pode melhorar a compreensão de conceitos complexos. Além disso, (E9, E13) focaram na inclusão de estudantes em relação a gênero e multiculturalidade, destacando a importância de adaptar o ensino para lidar com as diferenças entre os estudantes.

O uso de jogos sérios e gamificação também foi explorado como uma abordagem eficaz para o ensino de programação. Estudos como (E5, E2) mostraram como jogos como *PY-RATE ADVENTURES* e *CODECOMBAT* podem aumentar o engajamento dos estudantes, principalmente quando comparados com métodos tradicionais de ensino. Os jogos foram implementados tanto para estudantes do ensino fundamental quanto para recém-graduados, e os estudos indicaram um aumento significativo na assimilação dos conceitos. O estudo (E10) foi além, desenvolvendo o jogo *Rise of the Java Emperor*, que utilizou narrativas textuais para ensinar programação orientada a objetos, mostrando como a gamificação pode ser aplicada em níveis mais avançados de ensino. Da mesma forma, os estudos (E6, E22 e E29) analisaram o uso de plataformas e jogos interativos para fortalecer a construção do conhecimento.

No campo do Pensamento Computacional (PC), estudos como os de (E7, E26, E33, E63) investigaram o uso de ferramentas como *Scratch* e *Blockly* podem desenvolver habilidades de PC desde o ensino fundamental. Esses estudos mostraram que o uso de ferramentas visuais facilita a compreensão de conceitos complexos, como algoritmos e lógica de programação. Em (E4), foi proposto um modelo que combina o ensino de programação com o desenvolvimento de inteligência emocional para crianças de 4 a 8 anos. Este modelo mostrou que o desenvolvimento de habilidades cognitivas e emocionais pode ser potencializado por meio de ambientes interativos adaptativos.

A adoção de metodologias ativas e cooperativas no ensino de programação também é amplamente discutida. Os estudos (E3, E12, E14, E23, E30, E41, E68) embora possam descrever uma abordagem colaborativa, o ambiente educacional descrito envolve uma interação mais profunda entre os estudantes, que discutem estratégias e ajustam seus conhecimentos em grupo para obter melhores resultados. A natureza contínua dessa interação vai além da simples divisão de tarefas e se alinha mais à cooperação segundo Piaget, onde há uma construção conjunta do conhecimento .

O uso de metodologias como a aprendizagem baseada em problemas (PBL) promove a colaboração entre os estudantes e o aprendizado prático. Já (E18, E66) demonstraram como essas metodologias colaborativas podem aumentar o engajamento e o desempenho dos estudantes, principalmente quando aplicadas em contextos universitários.

A robótica educacional desempenha um papel importante no ensino de programação. Estudos como (E19, E53, E62, E69) mostraram como ferramentas, como o Arduino e o *Raspberry Pi*, podem ser usadas para ensinar programação de maneira prática e interativa. Além disso, os estudos (E11, E17, E35) exploraram o uso de robôs como ferramentas para ensinar conceitos de programação e eletrônica, ressaltando que o aprendizado prático pode melhorar significativamente a compreensão de conceitos abstratos.

No que diz respeito às tecnologias emergentes no ensino, estudos como (E32, E36, E43, E60) exploraram o uso de realidade aumentada e inteligência artificial para personalizar e facilitar o ensino de programação. Esses estudos mostraram que essas tecnologias podem melhorar a visualização de algoritmos e fluxos de programação, tornando o aprendizado mais acessível para os estudantes. Além disso, os estudos (E20, E46, E55) investigaram o uso de simuladores e ferramentas baseadas em nuvem como soluções eficazes para o ensino remoto, proporcionando aprendizado prático mesmo em ambientes virtuais.

A inclusão de mulheres e minorias no ensino de programação também foi destaque em estudos como (E39, E51), que promoveram programas de mentoria para incentivar a participação de grupos sub-representações nas áreas de *STEM*. O estudo (E4) também abordou a inclusão de gênero, focando na inteligência emocional como um meio de promover uma educação mais inclusiva e diversa.

Outro foco importante foi o alinhamento do ensino de programação com as demandas do mercado de trabalho. Estudos como (E20, E49, E54) propuseram currículos voltados para o desenvolvimento de competências práticas, preparando os estudantes para enfrentar os desafios reais da indústria de tecnologia. Além disso, os estudos (E45, E64, E67) exploraram o uso de projetos práticos e multidisciplinares como formas eficazes de preparar os estudantes para a complexidade do ambiente de trabalho.

Ambientes virtuais e plataformas online também foram investigados. Estudos como (E8, E50, E61, E65) mostraram como essas ferramentas podem aumentar a flexibilidade e a acessibilidade no ensino de programação. Essas plataformas permitiram que os estudantes tivessem experiências práticas, mesmo em contextos de ensino a distância.

O uso de competições de programação e avaliações lúdicas foi explorado em (E52, E40, E31). Esses estudos mostraram que competições, como *hackathons*, podem aumentar o interesse dos estudantes pela programação, ao mesmo tempo em que melhoram suas habilidades de resolução de problemas.

O uso de ferramentas de código aberto foi explorado em estudos como (E57, E56), que demonstraram como plataformas, como o *GitHub*, podem ser utilizadas para fomentar a colaboração e o desenvolvimento de projetos mais complexos. No estudo E57, observa-se uma dinâmica de cooperação, por outro lado, o estudo E56 enfatiza mais a colaboração, onde os estudantes compartilham informações e contribuem para o projeto de maneira menos integrada, sem necessariamente coordenar suas ações de forma profunda, conforme descrita por Piaget (1973).

O ensino de programação para diferentes faixas etárias e níveis de habilidade foi tema dos estudos (E21, E25), que exploraram o uso de ferramentas baseadas em blocos para ensinar programação a iniciantes, e (E27), que investigou o uso de projetos práticos no ensino superior. Esses estudos destacam que adaptar a metodologia de ensino ao nível de desenvolvimento dos estudantes, respeitando suas habilidades cognitivas, pode aumentar significativamente seu engajamento e compreensão.

Abordagens inovadoras e reflexivas no ensino de programação foram discutidas em estudos como (E24, E38, E42, E44, E47), que investigaram o uso de narrativas em jogos educacionais, mapas mentais e ferramentas de programação visual para ensinar conceitos de programação de forma progressiva, permitindo que os estudantes construam ativamente seu conhecimento.

Adicionalmente, o artigo (E15) investigou o projeto *CodeInnova*, que visa criar um sistema unificado de ensino de programação nas escolas primárias. A iniciativa promoveu a alfabetização digital, introduzindo ferramentas que facilitam o aprendizado desde cedo, permitindo que as crianças construam sua base de conhecimento em programação de forma gradual.

Esses estudos demonstram a diversidade de abordagens no ensino de programação, desde o uso de tecnologias emergentes e gamificação até o desenvolvimento de currículos que estimulam a construção ativa do conhecimento e o alinhamento com as demandas do mercado de trabalho. A inclusão, o desenvolvimento do PC e a adaptação das metodologias para diferentes contextos educacionais mostram como o ensino de programação continua evoluindo para se tornar mais acessível e eficaz, respeitando as necessidades cognitivas e o desenvolvimento de todos os estudantes.

2.7 Qual é o panorama das dificuldades enfrentadas pelos estudantes na aprendizagem de programação?

O panorama das dificuldades enfrentadas pelos estudantes na aprendizagem de programação é complexo e multifacetado. Primeiramente, existem desafios relacionados a questões conceituais e de inclusão, como observado em estudantes com deficiência visual (E1), que enfrentam obstáculos significativos ao interagir com os conceitos de programação. Estratégias de ensino táteis têm se mostrado eficazes para superar algumas dessas barreiras, mas há uma necessidade contínua de práticas pedagógicas que promovam a inclusão, permitindo que os estudantes construam ativamente seu conhecimento ao superar desafios específicos.

Além disso, a comparação entre métodos tradicionais de ensino e o uso de jogos educacionais, como o *CodeCombat* (E2), mostra que esses jogos podem ser mais eficazes para ensinar conceitos de programação do que as metodologias convencionais. Isso ressalta a importância de abordagens centradas no estudante, focadas na aprendizagem ativa, motivadora e interativa. No entanto, implementar esses métodos traz desafios, como a necessidade de capacitar os professores em novas ferramentas e estratégias educacionais. Outro ponto importante é a implementação de estratégias inovadoras, como o modelo 'Programação em Sete Passos' (E3) e o desenvolvimento de jogos sérios (E5), que demonstram que a aprendizagem de programação pode ser aprimorada por meio de métodos que vão além das práticas convencionais e incorporam tecnologias interativas e métodos lúdicos.

O aspecto emocional também desempenha um papel significativo na aprendizagem da programação. A integração do ensino de programação com o desenvolvimento de inteligência emocional mostrou-se eficaz para melhorar o aprendizado das crianças (E4), evidenciando a necessidade de abordagens pedagógicas que considerem o desenvolvimento cognitivo e emocional de forma integrada. As percepções e expectativas dos professores também devem ser levadas em consideração, pois enfrentam desafios ao introduzir conceitos complexos aos estudantes. Assim, é fundamental capacitar os professores em metodologias de ensino adequadas, como a criação de jogos educativos (E6), que podem tornar o processo de aprendizagem mais envolvente.

As dificuldades enfrentadas no ensino de programação estão frequentemente ligadas ao desenvolvimento do pensamento computacional. Abordagens como o uso de mapas mentais (E7) e atividades de programação visual são essenciais para ajudar os estudantes a construir gradualmente sua compreensão dos conceitos subjacentes à programação, indo além do ensino meramente técnico. A experiência e a percepção dos estudantes em relação à programação também são fundamentais. Fatores como autoconfiança, motivação e o valor que atribuem à programação impactam diretamente o processo de aprendizagem (E8), destacando a

necessidade de práticas pedagógicas adaptativas que atendam às diversas formas de experimentação dos estudantes.

Além disso, o desenvolvimento de habilidades práticas é crucial. Abordagens que combinam atividades práticas com a lógica do pensamento computacional, como na aprendizagem de *Python* (E12), têm mostrado eficácia. A autoavaliação e a avaliação entre pares surgem como ferramentas importantes para promover a autonomia dos estudantes e aprofundar o processo de aprendizado, incentivando a cooperação e a construção conjunta do conhecimento. A educação multicultural também influencia na aprendizagem de programação, pois as experiências culturais dos estudantes afetam sua motivação e eficácia na aprendizagem (E13). Assim, métodos de ensino inclusivos e sensíveis às diferenças culturais são necessários para atender às necessidades de uma sala de aula diversificada.

A implementação de metodologias ágeis, como o *eduScrum* (E23), é outra forma de facilitar a aprendizagem de programação, destacando a importância de ciclos iterativos e da cooperação entre estudantes para o desenvolvimento de habilidades de resolução de problemas. O uso de ferramentas e recursos tecnológicos, como robôs (E21) e plataformas de *e-learning* (E18), torna a aprendizagem mais interativa e prática, oferecendo oportunidades para adaptar o ensino às necessidades dos estudantes.

A integração da robótica no ensino de programação (E19) é também uma estratégia eficaz para aprimorar o PC e aumentar a motivação dos estudantes. A aprendizagem baseada em projetos auxilia na comunicação entre professores e estudantes, incentivando a colaboração e a discussão para a resolução de problemas de programação.

Em suma, os estudantes enfrentam uma variedade de desafios ao aprender programação, que vão desde questões conceituais e dificuldades de motivação até barreiras de inclusão e a necessidade de métodos instrucionais inovadores. Os artigos analisados (E1, E69) sugerem que uma abordagem pedagógica diversificada e centrada no estudante, que incorpore tecnologias interativas, jogos sérios, robótica e metodologias ágeis, pode contribuir significativamente para superar essas dificuldades e melhorar a eficácia do ensino da programação. Tais metodologias devem ser adaptativas, inclusivas e considerar a diversidade de experiências e expectativas dos estudantes, promovendo um ambiente de aprendizagem que valorize tanto o desenvolvimento do PC quanto o emocional, preparando-os de forma holística para os desafios

2.8 Quais são os principais desafios que os estudantes encontram ao compreender os conceitos fundamentais de programação?

Os principais desafios enfrentados pelos estudantes na compreensão dos conceitos fundamentais de programação são variados e complexos, abrangendo questões de inclusão, métodos de ensino, motivação e desenvolvimento de habilidades computacionais. Segundo os artigos do (E1 ao E69), uma série de fatores impacta negativamente a aprendizagem, exigindo que abordagens pedagógicas sejam diversificadas e adaptadas às necessidades cognitivas dos estudantes, permitindo que eles construam ativamente seu conhecimento.

Primeiramente, estudantes com deficiências, como aqueles com deficiência visual (E1), enfrentam barreiras significativas ao interagir com conceitos de programação. Apesar do uso de estratégias táteis, há uma necessidade contínua de adaptar práticas pedagógicas para promover a inclusão e superar as dificuldades específicas desses estudantes. O desafio de ensinar programação a diferentes perfis de estudantes é ainda mais acentuado pela natureza abstrata dos conceitos, que se tornam difíceis de assimilar sem metodologias e recursos adequados.

A complexidade dos conceitos de programação e a forma como são introduzidos também geram dificuldades. Por exemplo, a introdução de conceitos como variáveis, loops e condicionais pode ser árdua, principalmente quando ensinada de forma tradicional e sem o apoio de ferramentas interativas (E2). Estudos como o do (E3) sugerem que uma abordagem

mais prática, que demonstre a aplicação gradual da programação em situações reais, pode facilitar a compreensão e tornar o aprendizado mais significativo.

O artigo (E4) destaca a importância de integrar o desenvolvimento emocional com as habilidades de programação. Estudantes frequentemente enfrentam desafios emocionais ao aprender programação, como frustração diante de erros ou dificuldades com conceitos complexos. Uma abordagem pedagógica que integre o desenvolvimento emocional e cognitivo pode proporcionar um ambiente de aprendizagem mais acolhedor e motivador, ajudando os estudantes a superar obstáculos e construir autoconfiança.

No âmbito escolar, especialmente nos níveis fundamental e médio, a falta de recursos e abordagens inovadoras representa outro desafio. O uso de jogos sérios (E5) e atividades gamificadas, como o CodeCombat (E2), mostrou-se eficaz para tornar a aprendizagem mais interativa e envolvente. No entanto, a implementação dessas metodologias depende da formação dos professores e da disponibilidade de recursos tecnológicos. A necessidade de metodologias inovadoras é ainda mais evidente em artigos como o E6, que exploram a introdução da programação por meio de jogos e ferramentas visuais.

Além disso, os estudantes enfrentam dificuldades no desenvolvimento do pensamento computacional. O artigo (E7) explora o uso de mapas mentais para melhorar essas habilidades, demonstrando a importância de estratégias que ajudem os estudantes a construir gradualmente sua compreensão da lógica subjacente à programação, por meio de atividades interativas e colaborativas. O PC envolve decomposição, reconhecimento de padrões e abstração, habilidades que muitos estudantes não possuem naturalmente. Assim, é necessário que as metodologias de ensino sejam adaptadas para desenvolver essas competências de forma progressiva.

A variação das experiências e expectativas dos estudantes em relação à programação é um aspecto fundamental. No artigo (E8), fatores como autoconfiança e motivação afetam diretamente a compreensão dos conceitos fundamentais. Métodos pedagógicos adaptativos e sensíveis à diversidade cultural, como discutido no (E13), são essenciais para criar um ambiente de aprendizagem inclusivo, onde os estudantes possam construir ativamente suas habilidades.

Metodologias ágeis, como discutido no (E14, E23), mostram-se promissoras ao oferecer um ambiente de aprendizagem colaborativo e iterativo. Por meio do *eduScrum* e do uso de microcontroladores Arduino, os estudantes têm a oportunidade de trabalhar em equipe, experimentando ciclos iterativos que facilitam a construção contínua de seu conhecimento.

Em suma, a análise dos artigos do (E1 ao E69) revela que os desafios enfrentados pelos estudantes na compreensão dos conceitos fundamentais de programação são múltiplos e complexos. Eles abrangem desde questões de inclusão, dificuldades conceituais, falta de experiências práticas, barreiras culturais e emocionais até a necessidade de recursos tecnológicos e formação docente. A superação desses desafios requer uma abordagem pedagógica diversificada e centrada no estudante, que faça uso de tecnologias interativas, metodologias ágeis, práticas inclusivas e apoio emocional, promovendo um ambiente de aprendizagem que valorize tanto o desenvolvimento do PC quanto a construção de confiança e motivação para enfrentar os desafios do mundo tecnológico atual.

2.9 Quais estratégias e recursos têm sido mais eficazes para superar as barreiras na aprendizagem de programação, tanto para estudantes autodidatas quanto para aqueles inseridos em cursos formais?

As estratégias e recursos mais eficazes para superar as barreiras na aprendizagem de programação, tanto para estudantes autodidatas quanto para aqueles inseridos em cursos formais, variam conforme as necessidades específicas e o contexto em que a aprendizagem ocorre. Os artigos do (E1 ao E69) exploram diferentes abordagens que têm sido implementadas

com sucesso para facilitar o processo de ensino-aprendizagem e ajudar os estudantes a construir ativamente seu conhecimento, lidando com as diversas dificuldades encontradas.

Um dos aspectos mais destacados é a utilização de metodologias ativas e centradas no estudante, como a aplicação de jogos sérios e a gamificação. O artigo (E2), por exemplo, mostra que o uso de jogos educacionais, como o CodeCombat, se revelou mais eficaz em comparação com métodos tradicionais para ensinar conceitos de programação. Esse tipo de recurso promove uma aprendizagem mais envolvente e interativa, estimulando o interesse, a motivação e a construção ativa do conhecimento. Da mesma forma, o artigo (E5) apresenta o jogo *PY-RATE ADVENTURES*, que ajuda a introduzir conceitos básicos de programação de forma divertida e prática. A abordagem lúdica dos jogos sérios permite aos estudantes praticar a resolução de problemas e aplicar conceitos de maneira ativa, contribuindo para uma compreensão mais profunda.

Além dos jogos, o uso de robótica educacional também se destaca como uma estratégia eficaz para superar as barreiras de aprendizagem. No artigo (E11), a educação em robótica é apresentada como uma ferramenta que não apenas melhora o pensamento computacional, mas também aumenta a motivação dos estudantes ao aprender programação. O aprendizado prático com robôs torna conceitos como loops e condicionais mais tangíveis e observáveis, facilitando a construção ativa do conhecimento e tornando a aprendizagem mais significativa. Essa estratégia é reforçada no artigo (E19), que enfatiza a instrução baseada em robótica como uma forma de aumentar o interesse e o engajamento dos estudantes em ciência da computação.

Outro recurso que tem se mostrado eficaz é a implementação de metodologias ágeis no processo de ensino. O artigo (E14) aborda o uso de métodos ágeis, como o *Scrum*, na educação em programação, destacando que a aprendizagem por meio de ciclos iterativos permite que os estudantes desenvolvam suas habilidades em um ambiente colaborativo e dinâmico. O trabalho em equipe e o desenvolvimento de projetos em incrementos curtos e contínuos incentivam os estudantes a aplicar conceitos fundamentais de programação em situações práticas. O artigo (E23) reforça essa ideia ao apresentar o *eduScrum* como um método eficaz para ensinar programação com microcontroladores Arduino, estimulando a cooperação, o pensamento crítico e a resolução de problemas em grupo.

A integração de tecnologias de ensino, como plataformas de *e-learning* e ambientes interativos, é outro recurso eficaz que tem ajudado a superar barreiras na aprendizagem da programação. O artigo (E18) destaca a reformulação de cursos de programação para estudantes de mecatrônica, enfatizando a combinação de palestras com exercícios práticos e componentes de aprendizado à distância. Essas plataformas possibilitam que os estudantes pratiquem no seu próprio ritmo e revisem conteúdos conforme necessário, o que é especialmente útil para estudantes autodidatas. Além disso, o uso de ferramentas como o *Raspberry Pi* (E17) no ensino de programação oferece aos estudantes a oportunidade de experimentar e aplicar seus conhecimentos em contextos do mundo real, aumentando sua compreensão e engajamento.

Estratégias pedagógicas que utilizam mapas mentais e abordagens visuais também se mostraram eficazes para melhorar a compreensão de conceitos. O artigo (E7) descreve como diferentes métodos de mapeamento mental podem melhorar as habilidades de PC dos estudantes, especialmente quando combinados com ferramentas visuais como o Scratch. Essa estratégia ajuda os estudantes a organizar seus pensamentos, identificar padrões e compreender a lógica da programação, facilitando a transição de conceitos abstratos para aplicações práticas.

Outro recurso eficaz é a aplicação de práticas educacionais inclusivas e adaptativas. O artigo (E1) destaca a importância de estratégias de ensino adaptadas, como a utilização de ferramentas táteis para ensinar programação a estudantes com deficiência visual. A inclusão de métodos multimodais pode facilitar a aprendizagem para estudantes com diferentes necessidades e estilos de aprendizado. Da mesma forma, o artigo (E13) explora como a integração da multiculturalidade em aulas de programação pode afetar positivamente a

expectativa de aprendizado e a motivação dos estudantes. Métodos de ensino que reconhecem e incorporam as experiências culturais dos estudantes são mais eficazes em engajá-los e em criar um ambiente de aprendizagem significativo e acolhedor.

Para estudantes autodidatas, recursos de aprendizagem que oferecem autonomia e a possibilidade de autoavaliação são especialmente valiosos. O artigo (E12) sugere que o uso de ferramentas interativas, como o software Zuvio, permite que os estudantes pratiquem a programação de forma autônoma e recebam feedback imediato, aumentando sua autoconfiança e promovendo a construção autônoma de seu conhecimento. A avaliação entre pares e a autoavaliação também se destacam como formas eficazes de consolidar o aprendizado e promover a reflexão crítica sobre os próprios conhecimentos e processos de pensamento.

O desenvolvimento de competências emocionais também é uma estratégia importante para superar barreiras na aprendizagem. O artigo (E4) aponta que ensinar programação em conjunto com inteligência emocional ajuda os estudantes a lidar com a frustração e a manter a motivação ao enfrentar desafios complexos, promovendo tanto o desenvolvimento cognitivo quanto emocional de forma integrada. Estratégias que promovem a resiliência, a capacidade de resolução de problemas e o pensamento crítico são essenciais para que os estudantes superem obstáculos e desenvolvam uma mentalidade positiva em relação à aprendizagem de programação.

No contexto da educação formal, a construção de um currículo adequado e estruturado é fundamental. O artigo (E15), que aborda a introdução da programação em escolas primárias, destaca a importância de criar planos de aula, materiais didáticos e plataformas online adaptadas às necessidades dos estudantes. A introdução precoce de conceitos de programação, acompanhada de um suporte pedagógico adequado, contribui para o desenvolvimento de competências digitais e para a compreensão de conceitos mais avançados em estágios posteriores.

A utilização de recursos tecnológicos interativos, como jogos, robótica e plataformas online, em conjunto com estratégias pedagógicas inclusivas, metodologias ágeis e práticas baseadas em evidências, mostra-se essencial para superar as barreiras na aprendizagem de programação. Os artigos do (E1 ao E69) reforçam a necessidade de uma abordagem adaptativa e diversificada, centrada no estudante e que inclua tanto a prática quanto o desenvolvimento emocional. Essas estratégias não apenas ajudam os estudantes a entender e aplicar os conceitos fundamentais de programação, mas também a desenvolver uma atitude positiva e proativa em relação à resolução de problemas e ao aprendizado contínuo.

2.10 Quais reflexões emergem sobre a construção do conhecimento no ensino de programação?

As reflexões sobre a construção do conhecimento no ensino de programação, conforme analisadas nos artigos do (E1 ao E69), revelam que o processo de aprendizagem é multifacetado e depende de diversos fatores, incluindo a metodologia de ensino, a inclusão de práticas interativas, a adaptação a diferentes estilos de aprendizagem e o papel das emoções e motivações dos estudantes. O processo de construção do conhecimento é mais eficaz quando o estudante é ativo e central no processo educacional.

Uma das principais reflexões emergentes é a importância da aprendizagem ativa e da participação dos estudantes no processo educacional. O artigo (E3) destaca a eficácia do modelo 'Programação em Sete Passos', que enfatiza que os estudantes se beneficiam ao construir conhecimento de forma incremental, aplicando conceitos fundamentais em atividades práticas. Isso demonstra que a construção do conhecimento em programação é mais eficaz quando os estudantes podem experimentar, testar e corrigir seus próprios erros, transformando a aprendizagem em um processo contínuo e dinâmico. Essa abordagem é reforçada por estudos como o (E2, E5), que mostram como jogos sérios e atividades gamificadas tornam o

aprendizado mais envolvente e significativo, permitindo que os estudantes explorem conceitos complexos de maneira interativa.

A inclusão é outro ponto de reflexão crucial. O artigo (E1) destaca que o ensino de programação deve ser adaptado para atender estudantes com diferentes necessidades, como aqueles com deficiência visual. A inclusão de métodos táteis e ferramentas multimodais é fundamental para a construção do conhecimento, permitindo que esses estudantes desenvolvam habilidades computacionais em igualdade de condições. Da mesma forma, o (E13) explora a influência das experiências culturais no processo de aprendizagem, ressaltando que a construção do conhecimento é influenciada pelo contexto sociocultural do estudante, o que reforça a necessidade de práticas educativas sensíveis a essas diferenças.

A construção do conhecimento em programação também envolve a compreensão e o desenvolvimento do pensamento computacional, que é uma habilidade essencial para resolver problemas de forma lógica e estruturada. O artigo (E7) destaca o uso de mapas mentais como ferramenta eficaz para ajudar os estudantes a organizar seus pensamentos e compreender os processos de programação. Essa abordagem indica que a aprendizagem de programação vai além da memorização de códigos e envolve a construção gradual do pensamento crítico e a capacidade de abstração.

Outro aspecto importante é o papel da prática na consolidação do conhecimento em programação. O artigo (E12) ressalta que a prática contínua, combinada com ferramentas de avaliação formativa, proporciona aos estudantes a oportunidade de refletir sobre seu próprio processo de aprendizagem e ajustar estratégias conforme necessário. Além disso, o artigo (E18) enfatiza a importância de combinar teoria e prática no ensino de programação para estudantes de engenharia, sugerindo que a compreensão dos conceitos é aprimorada quando os estudantes podem ver sua aplicabilidade em cenários do mundo real. Portanto, a construção do conhecimento em programação é impulsionada por práticas interativas e pela aplicação em situações práticas que desafiam os estudantes a utilizar e adaptar seus conhecimentos.

A metodologia de ensino também desempenha um papel central. O uso de metodologias ágeis, como discutido no artigo (E14), proporciona um ambiente de aprendizagem que promove a construção colaborativa do conhecimento. Ao implementar métodos como o Scrum em aulas de programação, os estudantes são incentivados a trabalhar em equipe, compartilhar ideias e refletir sobre os resultados de suas ações. O *eduScrum*, apresentado no artigo (E23), exemplifica como a abordagem colaborativa pode aumentar o engajamento e facilitar a compreensão dos conceitos de programação. Essas metodologias refletem a necessidade de um ensino que valorize não apenas a transferência de conhecimento, mas também a capacidade dos estudantes de construir ativamente seu entendimento por meio da cooperação e da interação.

Os aspectos emocionais e motivacionais são igualmente relevantes para a construção do conhecimento. O artigo (E4) aborda a importância de incorporar a inteligência emocional no ensino de programação, destacando que o aprendizado é impactado pela capacidade dos estudantes de lidar com a frustração e a perseverança ao enfrentar desafios complexos. Promover um ambiente de aprendizagem que apoie os estudantes em seus esforços e valorize suas conquistas é fundamental para que eles construam confiança em suas habilidades. A motivação intrínseca, muitas vezes cultivada através de abordagens lúdicas e interativas, é um motor para o engajamento e a construção contínua do conhecimento em programação.

A introdução precoce à programação é outra reflexão relevante. O artigo (E15) aponta para os benefícios de introduzir conceitos de programação na escola primária, sugerindo que a construção do conhecimento é um processo que deve começar cedo, com materiais e atividades adaptadas ao nível de desenvolvimento dos estudantes. Ao introduzir a programação desde os primeiros anos, os estudantes podem desenvolver uma base sólida que facilitará o aprendizado de conceitos mais complexos em estágios posteriores. Essa perspectiva é reforçada pelo (E16),

que discute a importância de adaptar o ensino de programação em C++ para estudantes do ensino médio, destacando a necessidade de uma transição suave que respeite o ritmo e as experiências prévias dos estudantes.

A construção do conhecimento em programação, conforme refletido nos artigos, é um processo que envolve uma combinação de metodologias centradas no estudante, práticas inclusivas, experiências práticas, motivação e desenvolvimento emocional. O papel do professor é fundamental como facilitador desse processo, oferecendo suporte, recursos e um ambiente de aprendizagem que estimule a curiosidade e a exploração. Os artigos do (E1) ao E69 enfatizam que a construção do conhecimento é mais eficaz quando os estudantes são colocados no centro do processo educacional, encorajados a experimentar, colaborar e refletir sobre seu aprendizado, desenvolvendo assim não apenas habilidades técnicas, mas também uma mentalidade criativa e resiliente.

3. RESULTADOS

As dificuldades enfrentadas pelos estudantes na aprendizagem de programação são multifacetadas, variando conforme o contexto, a infraestrutura disponível e o perfil dos estudantes. Com base nos estudos analisados (E1 ao E69), as estratégias mais eficazes para superar essas barreiras incluem o uso de metodologias ativas, tecnologias inovadoras e práticas pedagógicas adaptadas às necessidades e particularidades dos estudantes, promovendo a construção ativa do conhecimento e levando em consideração também aspectos emocionais e culturais.

O uso de jogos sérios, como *CODECOMBAT* (E2) e *PY-RATE ADVENTURES* (E5), tem demonstrado aumentar o engajamento e a retenção de conceitos, especialmente entre estudantes do ensino fundamental e médio. Esses métodos funcionam bem em ambientes com boa infraestrutura tecnológica, onde o acesso a dispositivos e à internet está garantido. Contudo, sua eficácia tende a diminuir em níveis universitários mais avançados, onde é necessário um enfoque mais técnico e aprofundado, promovendo uma transição gradual para metodologias que exigem maior abstração e domínio técnico.

A robótica educacional, com o uso de ferramentas como Arduino e *Raspberry Pi* (E19, E53, E62, E69), provou ser uma estratégia eficaz para concretizar conceitos abstratos de programação, como loops e condicionais, proporcionando uma experiência prática que facilita a construção ativa do conhecimento pelos estudantes. No entanto, a implementação dessa abordagem depende fortemente da disponibilidade de recursos físicos e da capacitação docente. Em escolas com restrições orçamentárias ou falta de treinamento adequado para professores, o uso da robótica pode ser limitado, o que afeta diretamente sua eficácia como ferramenta educacional.

Metodologias ágeis, como o *eduScrum* (E14, E23), têm se mostrado particularmente eficazes em contextos de ensino superior, principalmente em cursos técnicos ou de engenharia, onde a prática colaborativa é essencial para a aprendizagem. Essas metodologias incentivam a resolução de problemas em equipe e a aplicação prática de conceitos, sendo mais adequadas para turmas menores, onde a personalização do ensino e o feedback constante são possíveis. Em turmas grandes ou em ambientes com menos interação, essas metodologias perdem parte de sua eficácia, dado que a personalização do ensino se torna mais difícil.

A inclusão de estudantes com deficiências no ensino de programação também é destacada nos estudos. Ferramentas táteis para estudantes com deficiência visual (E1, E58) mostraram-se eficazes para tornar o aprendizado acessível, mas sua aplicação enfrenta limitações em ambientes com poucos recursos financeiros. Isso aponta para a necessidade urgente de investimentos e políticas públicas que possibilitem a ampliação dessas tecnologias em mais instituições de ensino, garantindo a inclusão de um público mais diversos.

Além disso, o desenvolvimento emocional tem um papel crucial no aprendizado de programação. Estudos com crianças de 4 a 8 anos (E4) mostraram que combinar o desenvolvimento da inteligência emocional com o ensino de programação aumenta a resiliência dos estudantes e melhora seu desempenho. Essa abordagem integrada é especialmente eficaz em contextos de educação infantil, onde a construção emocional caminha junto com a cognitiva, formando uma base sólida para o aprendizado. No ensino superior, no entanto, essas práticas precisam ser ajustadas para lidar com os desafios específicos desse nível de ensino, como o estresse e a pressão por resultados.

Portanto, as estratégias mais eficazes no ensino de programação variam de acordo com o contexto, os recursos disponíveis e as necessidades dos estudantes. Tecnologias assistivas (E1), robótica (E19), metodologias ágeis (E14) e jogos sérios (E2, E5) são todas estratégias promissoras, mas enfrentam desafios em termos de infraestrutura e implementação. A adaptação dessas estratégias ao contexto específico de cada grupo de estudantes é essencial para garantir seu sucesso, promovendo um ambiente educacional mais inclusivo e adaptativo. Assim, os estudos reforçam a importância de um ensino que valorize tanto a prática quanto o desenvolvimento emocional, proporcionando um ambiente de aprendizagem que prepare os estudantes para os desafios do mundo da programação e da tecnologia.

4. CONCLUSÃO

Concluímos que está Revisão Sistemática da Literatura (RSL) indicam que o estudo conseguiu responder de forma abrangente às perguntas de pesquisa estabelecidas. A primeira questão principal, QP1, sobre o panorama das dificuldades enfrentadas pelos estudantes na aprendizagem de programação, foi amplamente discutida. Os resultados mostram que os estudantes enfrentam desafios complexos, que vão desde dificuldades conceituais e cognitivas até questões de inclusão, falta de recursos adequados e barreiras emocionais e culturais. A diversidade de dificuldades enfrentadas reforça a necessidade de abordagens pedagógicas adaptativas, centradas no estudante e que promovam a construção ativa do conhecimento.

Quanto à QP2, que se refere aos principais desafios que os estudantes encontram ao compreender os conceitos fundamentais de programação, a RSL destacou que os estudantes têm dificuldades particularmente com conceitos abstratos como variáveis, loops e condicionais, especialmente quando esses conceitos são apresentados de forma tradicional e sem a devida conexão com aplicações práticas. A falta de recursos interativos e a ausência de uma abordagem prática agravam essas dificuldades, desmotivando os estudantes e dificultando a construção progressiva do conhecimento.

Quanto à QP3, que busca identificar as estratégias e recursos mais eficazes para superar as barreiras na aprendizagem de programação, a RSL revelou que metodologias ativas, jogos sérios, robótica educacional e o uso de tecnologias emergentes são as abordagens mais eficazes. Jogos sérios como o *CODECOMBAT* e o *PY-RATE ADVENTURES* demonstraram um impacto significativo no aumento do engajamento e da retenção de conceitos pelos estudantes, tornando o aprendizado mais significativo e motivador. Da mesma forma, a robótica educacional e o uso de plataformas interativas e metodologias ágeis, como o *eduScrum*, se mostraram eficazes para conectar conceitos teóricos a práticas reais, promovendo uma aprendizagem mais envolvente e significativa. Essas estratégias não só ajudam a superar as barreiras conceituais, como também melhoram a motivação e o desenvolvimento do PC dos estudantes.

A última pergunta, QP4, aborda as reflexões emergentes sobre a construção do conhecimento no ensino de programação. Os artigos analisados na RSL enfatizam que a construção do conhecimento em programação é um processo multifacetado que envolve práticas pedagógicas interativas, a adaptação a diferentes estilos de aprendizagem e a consideração das dimensões emocionais e motivacionais dos estudantes. A aprendizagem ativa, que permite que os estudantes experimentem, reflitam e corrijam seus próprios erros, se revelou

uma estratégia central para a construção do conhecimento. Além disso, a inclusão de metodologias colaborativas e o uso de recursos tecnológicos são apontados como formas de promover um ambiente de aprendizado mais inclusivo, engajador e voltado para a aplicação prática dos conceitos.

Portanto, a RSL conseguiu responder às perguntas de pesquisa propostas, fornecendo um panorama detalhado das dificuldades enfrentadas pelos estudantes e destacando estratégias e recursos eficazes para superá-las. Ao mesmo tempo, trouxe reflexões importantes sobre a construção do conhecimento no ensino de programação, sugerindo que uma abordagem pedagógica que valorize a prática, a cooperação e o apoio emocional é fundamental para o sucesso dos estudantes na aprendizagem de programação.

REFERÊNCIAS

Ercan, M. F., & Sale, D. (2020, November). Teaching programming: An evidence based and reflective approach. In 2020 IEEE REGION 10 CONFERENCE (TENCON) (pp. 997-1001). IEEE.

Souza, D. M., da Silva Batista, M. H., & Barbosa, E. F. (2016). Problemas e dificuldades no ensino de programação: Um mapeamento sistemático. Revista Brasileira de Informática na Educação, 24(1), 39.

Piaget, J. (1973). Estudos sociológicos. Forense.

PRISMA. (2020). Transparent reporting of systematic reviews and meta-analyses. PRISMA. https://www.prisma-statement.org/

Uehara, M. (2020, November). Programming learning by creating problems. In 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW) (pp. 272-276). IEEE.

Vieira Junior, R. R. M., & Reategui, E. B. (2018). Arquitetura Pedagógica para leitura de textos digitais: uma proposta de intervenções programadas e automáticas.