INSTITUTO FEDERAL
Sul-rio-grandense

LÓGICA DE PROGRAMAÇÃO NO CURSO TÉCNICO EM INFORMÁTICA

BOAS PRÁTICAS DE ENSINO NA EPT

Contribuições para a Permanência e o Êxito no IFSul - Campus Gravataí

Heitor Boeira dos Reis Filho

Orientadora: Walkiria Helena Cordenonzi

Produto Educacional – Mestrado Profissional em Educação Profissional e Tecnologia

2026

Reis Filho, Heitor Boeira dos
Lógica de programação no curso técnico em informática [livro eletrônico] : boas práticas de ensino na EPT : contribuições para a permanência e o êxito no IFSul : campus Gravataí / Heitor Boeira dos Reis Filho. -- Gravataí, RS : Ed. do Autor, 2026. PDF

Produto educacional (Mestrado profissional em Educação Profissional e Tecnológica). Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense, Campus Gravataí.
Orientadora: Walkiria Helena Cordenonzi Bibliografia.
ISBN 978-65-02-05812-1

1. Educação 2. Educação profissional e tecnológica 3. Linguagem de programação (Computadores) 4. Programação (Computadores)
I. Cordenonzi, Walkiria Helena. II. Título.

26-353450.0 CDD-005.1

Índices para catálogo sistemático:

1. Lógica de programação : Computadores : Processamento de dados 005.1

Eliete Marques da Silva - Bibliotecária - CRB-8/9380

APRESENTAÇÃO

Prezados(as) educadores(as) e gestores(as) pedagógicos(as),

A Lógica de Programação constitui um dos componentes curriculares fundamentais nos cursos técnicos em Informática, desempenhando papel estruturante para a aprendizagem de diferentes linguagens de programação e para o desenvolvimento do pensamento computacional.

Diante da relevância desse componente na formação técnica e tecnológica dos estudantes, torna-se pertinente refletir sobre práticas pedagógicas que favoreçam processos de ensino e aprendizagem mais significativos, capazes de promover o desenvolvimento do raciocínio lógico, da autonomia intelectual e da capacidade de resolução de problemas.

Este produto educacional foi desenvolvido no âmbito do Mestrado Profissional em Educação Profissional e Tecnológica (ProfEPT) na linha de pesquisa 2 no macroprojeto 5 e tem como objetivo apresentar um conjunto de reflexões teóricas, estratégias pedagógicas e sugestões de práticas didáticas voltadas ao ensino de Lógica de Programação no contexto da Educação Profissional e Tecnológica.

O material reúne discussões sobre aspectos relevantes do ensino e da aprendizagem de programação, bem como estratégias pedagógicas e possibilidades de organização das aulas que podem apoiar o trabalho docente. Ao longo do texto, são apresentadas diferentes sugestões de práticas pedagógicas e exemplos de sequências didáticas, elaborados com a finalidade de ilustrar formas de abordagem de conteúdos e de condução das atividades em sala de aula.

O material está organizado em capítulos que abordam diferentes dimensões do ensino de Lógica de Programação. Inicialmente, são discutidos aspectos relacionados à importância desse componente curricular na formação dos estudantes e aos processos envolvidos em sua aprendizagem. Em seguida, são apresentadas reflexões sobre abordagens pedagógicas e estratégias didáticas que podem contribuir para tornar as aulas mais dinâmicas, participativas e significativas. Posteriormente, o material reúne sugestões de práticas pedagógicas, estratégias de ensino e exemplos de sequências didáticas, que buscam ilustrar possibilidades de organização das atividades em sala de aula e apoiar o planejamento das práticas docentes.

Embora os capítulos estejam organizados de forma a construir um percurso de leitura progressivo, cada seção foi elaborada de modo a poder ser consultada de forma independente, permitindo que professores utilizem o material conforme suas necessidades de planejamento, reflexão ou elaboração de atividades.

As propostas apresentadas não constituem modelos rígidos, mas referências que podem ser adaptadas às diferentes realidades institucionais, perfis de turma e estilos de ensino, contribuindo para o planejamento e a diversificação das práticas pedagógicas.

Assim, este produto educacional busca oferecer subsídios teóricos e práticos que apoiem professores no planejamento de suas aulas, estimulando a reflexão sobre o ensino de programação e favorecendo a construção de experiências de aprendizagem mais significativas para os estudantes da Educação Profissional e Tecnológica.

Este material não tem como finalidade orientar passo a passo a prática docente, mas contribuir para a compreensão dos fatores que impactam a aprendizagem e a permanência dos estudantes na disciplina de lógica de programação. A partir dessa compreensão, o professor pode selecionar e articular as estratégias apresentadas de forma contextualizada, considerando o momento pedagógico, o nível de complexidade dos conteúdos e as dificuldades específicas identificadas em sua turma.

Este produto educacional foi desenvolvido a partir da pesquisa realizada no âmbito do Mestrado Profissional em Educação Profissional e Tecnológica (ProfEPT), intitulada "Causas da evasão no curso Técnico de Informática: um olhar sobre a disciplina de Lógica de Programação no Ensino Profissional e Tecnológico". A dissertação apresenta detalhadamente o referencial teórico e as análises que fundamentam as reflexões, estratégias pedagógicas e sugestões de práticas apresentadas neste material.

Heitor Boeira dos Reis Filho
2026

SUMÁRIO

GUIA DE UTILIZAÇÃO DO E-BOOK

Este material não é apenas um texto teórico, mas um recurso didático modular. Veja como aproveitar cada parte:

Entender o Problema: Leia os Capítulos 1 e 2 para compreender por que os estudantes sentem dificuldade na transição para a lógica de programação.

Estratégias de apoio ao estudante: ao longo do material são apresentadas propostas como **"Programando Juntos, Crescendo Juntos"** e **"Diário de Bordo do Codificador"**, que buscam fortalecer a confiança, a autonomia e a capacidade de reflexão dos estudantes durante o processo de aprendizagem.

Refletir sobre a avaliação: no Capítulo 4, é discutida a avaliação ligando-a aos problemas descritos no Capítulo 2.

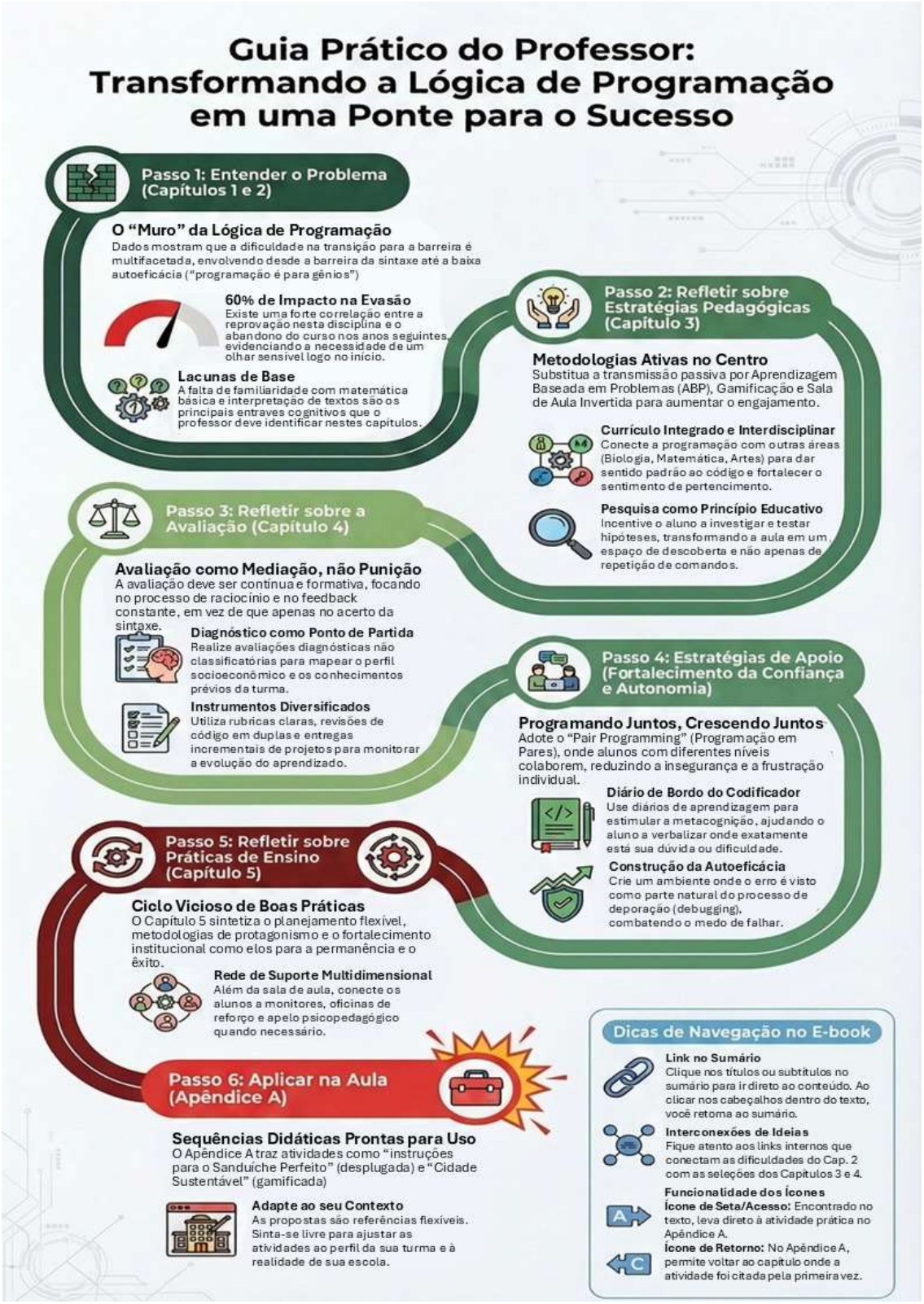
Refletir sobre estratégias pedagógicas: no Capítulo 3, são discutidas algumas estratégias pedagógicas que já são ou podem ser aplicadas em sala de aula.

Refletir sobre as práticas de ensino: o Capítulo 5 traz alguns tópicos relacionados aos problemas encontrados na pesquisa que podem ajudar a refletir a respeito do problema.

Aplicar na Aula: consulte o Apêndice A para encontrar sugestões de sequências didáticas, estratégias de ensino e atividades práticas. Estas sugestões estão apresentadas ao longo do texto, contudo podem ser acessadas diretamente. Sinta-se livre para adaptar as sequências, estratégias e atividades práticas a seu contexto de aula e perfil da turma.

Com relação à navegação pelo material, você pode usar os links do sumário para acessar diretamente o conteúdo desejado. Ao clicar nos títulos dos capítulos ou em seus subtítulos você sempre será redirecionado ao sumário. Também existem dentro do texto links para interconectar as ideias. No apêndice A, você encontrará o ícone , que permitirá que você retorne ao capítulo onde ocorreu a chamada original da atividade. Quando encontrar este ícone no fluxo do texto , clicando nele você acessará diretamente a atividade prática contida no apêndice A. Quando encontrar esta imagem sempre haverá uma dica para seu aprimoramento. Na Figura 1 apresento um guia prático de utilização do material.

Figura 1 -GUIA PRÁTICO DE UTILIZAÇÃO DO E-BOOK



Fonte: Elaborado pelo autor (2026)

CAPÍTULO 1 – A IMPORTÂNCIA DA LÓGICA DE PROGRAMAÇÃO

A disciplina de Pensamento Computacional e Algoritmos constitui um dos pilares da formação nos cursos técnicos em Informática, pois contribui diretamente para o desenvolvimento do raciocínio lógico e das habilidades de resolução de problemas, competências essenciais para a atuação na área de Tecnologia da Informação. No entanto, estudos sobre o ensino de programação e os dados analisados nesta pesquisa indicam que esse componente curricular frequentemente concentra dificuldades significativas de aprendizagem, podendo impactar a trajetória acadêmica dos estudantes e contribuir para processos de desmotivação, reprovação e evasão.

Nesse contexto, compreender os fatores que influenciam o processo de aprendizagem na disciplina de Pensamento Computacional e Algoritmos torna-se fundamental para o desenvolvimento de estratégias pedagógicas mais eficazes. A literatura sobre o ensino de programação aponta que estudantes iniciantes frequentemente enfrentam desafios relacionados à abstração, à lógica de programação e à compreensão da estrutura dos algoritmos, o que pode gerar insegurança, dificuldades de acompanhamento das atividades e, em alguns casos, afastamento do processo de aprendizagem.

A partir dessa perspectiva, este capítulo discute a importância da disciplina de Pensamento Computacional e Algoritmos na formação dos estudantes dos cursos técnicos em Informática. Inicialmente, são apresentados os fundamentos do pensamento computacional, suas principais habilidades e as dificuldades de aprendizagem associadas, bem como estratégias pedagógicas voltadas à sua superação. Em seguida, analisa-se o papel dessa disciplina como base para outras áreas do curso e sua relação com o mundo do trabalho, evidenciando seus impactos na trajetória acadêmica dos estudantes. Por fim, discute-se sua contribuição para a formação integral no âmbito da Educação Profissional e Tecnológica, destacando seu potencial na construção de uma formação crítica, cidadã e alinhada às demandas da sociedade contemporânea.

1.1 Pensamento Computacional: A Arte de Resolver Problemas

De acordo com Papert (1993), a lógica de programação consiste na organização coerente e sequencial de instruções que permitem ao computador executar uma tarefa.

Essa sequência, conhecida como algoritmo, é o ponto de partida para transformar problemas do mundo real em soluções computacionais. Mais do que ensinar comandos ou sintaxe de linguagens específicas, a disciplina visa desenvolver habilidades cognitivas como raciocínio lógico, pensamento analítico e criatividade.

Segundo Wing (2006), ao dominar a lógica de programação, os estudantes aprendem a decompor problemas complexos em partes menores, identificar padrões, pensar de forma algorítmica e testar soluções de forma sistemática.

Essas habilidades são valiosas não apenas para a área técnica, mas também para o desenvolvimento pessoal e intelectual dos estudantes.

Nesse sentido, o pensamento computacional pode ser compreendido como um conjunto estruturado de habilidades cognitivas, frequentemente organizado em quatro pilares fundamentais: **decomposição, reconhecimento de padrões, abstração e elaboração de algoritmos**. A decomposição refere-se à capacidade de segmentar problemas complexos em partes menores e mais manejáveis, favorecendo sua análise e resolução progressiva. O reconhecimento de padrões, por sua vez, envolve a identificação de regularidades e similaridades entre diferentes situações-problema, possibilitando a transferência e a generalização de soluções. A abstração consiste na seleção dos elementos essenciais do problema, com a consequente desconsideração de aspectos secundários, permitindo a construção de representações mais eficientes. Por fim, a elaboração de algoritmos diz respeito à organização lógica e sequencial de procedimentos necessários à resolução de um problema, constituindo a base operacional da programação (WING, 2006).

A articulação desses pilares é central para o desenvolvimento do raciocínio lógico e da capacidade de resolução de problemas, competências indispensáveis à formação em

Tecnologia da Informação (WING, 2006). No entanto, conforme evidenciado nos dados desta pesquisa, dificuldades relacionadas à interpretação dos enunciados, à compreensão dos problemas propostos e à ausência de estratégias de resolução indicam fragilidades no desenvolvimento dessas habilidades. Em particular, limitações nos processos de abstração e decomposição tendem a comprometer a compreensão inicial dos problemas, enquanto a dificuldade no reconhecimento de padrões restringe a capacidade de reaproveitamento de soluções previamente construídas.

Tais lacunas não apenas impactam o desempenho acadêmico dos estudantes, mas também contribuem para a construção de um percurso marcado por insegurança, desmotivação e, em muitos casos, evasão. Dessa forma, torna-se fundamental que o ensino de Lógica de Programação contemple, de maneira intencional e sistematizada, o desenvolvimento desses pilares desde os momentos iniciais da formação, por meio de estratégias pedagógicas que favoreçam a construção gradual dessas habilidades.

Essa abordagem encontra respaldo nas orientações da Base Nacional Comum Curricular, que reconhece o pensamento computacional como uma competência essencial, associada ao desenvolvimento do raciocínio lógico, da resolução de problemas e do uso crítico das tecnologias digitais (BRASIL, 2018).

A pesquisa corroborou a importância das habilidades descritas, identificando entre os alunos evadidos que as principais dificuldades estavam na **“**Falta de conhecimento a respeito dos problemas apresentados nos exercícios" e na "Interpretação dos exercícios que eram apresentados". Essas dificuldades reforçam a necessidade de um foco pedagógico robusto na construção do pensamento computacional desde o início. Além disso, compreender a lógica de programação desde o início do curso é fundamental, pois essa disciplina serve de base para diversas outras áreas da formação em informática.

Como estratégia de nivelamento inicial, o uso de atividades desplugadas (*Unplugged*) mostra-se eficaz para introduzir o pensamento computacional de forma acessível e lúdica. Ao explorar conceitos por meio de papel, caneta e raciocínio lógico, sem a exigência imediata de sintaxe, essas atividades reduzem barreiras cognitivas relacionadas à

abstração, à decomposição de problemas e à ausência de experiência prévia em programação.

A **sequência didática 1** apresenta um exemplo para o professor iniciar seus trabalhos com a turma, podendo ser adaptada para outros problemas tradicionais, como por exemplo, os passos para trocar uma lâmpada entre outros.

Sequência Didática 1:
Instruções para o Sanduíche Perfeito

Este tipo de atividade sempre deve ser aplicada nas primeiras aulas e pode ser bastante reveladora do nível dos estudantes.

Esta prática, ainda pode ajudar os estudantes que não tiveram contato com a área de desenvolvimento de software a se inserir no ambiente e também pode favorecer a integração em pares ou grupos, estimulando o trabalho em equipe. Também permitirá ao professor identificar e diagnosticar as dificuldades dos estudantes iniciantes.

1.2 Pilar para Outras Disciplinas e o Mundo do Trabalho

No contexto do Curso Técnico em Informática para Internet do IFSul – Campus Gravataí, a disciplina, atualmente denominada Pensamento Computacional e Algoritmos, possui papel ainda mais decisivo. Ela serve de base para outras unidades curriculares como Programação Web, Banco de Dados, Redes de Computadores e Desenvolvimento de Aplicativos Móveis. A falta de domínio nessa fase inicial compromete diretamente o desempenho dos estudantes ao longo do curso e, em muitos casos, contribui para sua evasão.

Essa perspectiva é reforçada por Morais, Neto e Osório (2020), que apontam que interpretar problemas computacionais e traduzi-los em código constitui um desafio frequente para os estudantes, especialmente aqueles em seu primeiro contato com linguagens de programação.

Essas dificuldades impactam negativamente sua motivação e autoestima acadêmica, sendo um fator crítico na repetência e na desistência de estudantes em cursos da área de TI. A pesquisa realizada junto aos docentes do IFSul – Campus Gravataí valida essa percepção, com professores e supervisores descrevendo a disciplina como um **"primeiro filtro" ou "muro"** para a permanência dos estudantes.

É fundamental mostrar aos estudantes, de forma concreta, como os conhecimentos desenvolvidos em Pensamento Computacional e Algoritmos serão aplicados nas próximas disciplinas e, futuramente, no mundo do trabalho. Isso ajuda a reduzir o **desalinhamento de expectativas** e a **falta de identificação com o curso**, problemas identificados na pesquisa com estudantes evadidos.

Para fortalecer o engajamento e a percepção de relevância da disciplina, é crucial mostrar como a disciplina de Pensamento Computacional e Algoritmos se integra ao contexto mais amplo do curso e do mundo do trabalho. Entretanto, sua importância não se limita à formação técnica, estendendo-se também à formação humana e cidadã proposta pela Educação Profissional e Tecnológica. Desta forma pode ser muito interessante a aplicação da **sequência didática 2**: “De onde Viemos, Para Onde Vamos”.

Sequência Didática 2:
De Onde Viemos, Para Onde Vamos

Esta atividade pode ser aplicada em sala de aula, ou em um auditório de forma integrada com outras turmas!

1.3 Lógica e EPT: Formação Integral e Cidadania Digital

Além da sua função técnica, a Lógica de Programação oferece uma oportunidade singular para promover uma formação crítica e integrada, conforme propõe a Educação Profissional e Tecnológica (EPT). Esta abordagem visa o desenvolvimento omnilateral dos sujeitos, articulando trabalho, ciência, tecnologia e cultura na formação humana (FRIGOTTO; CIAVATTA; RAMOS, 2005). Quando ensinada de forma articulada com a realidade dos estudantes, com apoio de metodologias ativas e uso de tecnologias educacionais, ela pode contribuir para o desenvolvimento de cidadãos mais conscientes e preparados para atuar em um mundo digital cada vez mais complexo.

> Essa abordagem não só engaja estudantes cuja motivação inicial era o **"Gostar de Tecnologia"**, mas também expande seu horizonte, mostrando que a tecnologia é uma ferramenta poderosa para transformar a realidade social e não apenas um meio técnico.

Como sugestão, usando a sequência didática 3, estimule os estudantes a verem a tecnologia como uma ferramenta para transformar a realidade e a desenvolverem um olhar crítico sobre seu uso e suas implicações éticas e sociais.

Esta atividade fortalece a conexão com a realidade do estudante, promove a formação omnilateral e pode atenuar a desmotivação ao mostrar o impacto social da tecnologia.

Sequência Didática 3:
Tecnologia para o Bem

Como sugestão apresente vídeos que possam instigar a discussão. Com sugestão apresente um vídeo de cidadania digital, ou outro que você julgar necessário.

Considerações Finais do Capítulo

O Pensamento Computacional e Algoritmos, portanto, é mais do que uma disciplina técnica. É um instrumento de transformação que qualifica o estudante para o mundo do trabalho, desenvolve seu raciocínio lógico e crítico, e o prepara para ser um cidadão atuante em um mundo digital. Ao valorizar e contextualizar essa disciplina em suas múltiplas dimensões – técnica, cognitiva, profissional e social – os educadores podem mitigar a desmotivação e o desalinhamento de expectativas, fatores identificados na pesquisa como contribuintes significativos para a evasão, promovendo assim maior permanência e sucesso escolar.

Dessa forma, compreender a importância dessa disciplina vai além da sua função técnica. Trata-se de reconhecer seu potencial como instrumento de transformação da trajetória escolar e profissional dos estudantes. Uma abordagem mais contextualizada, interdisciplinar e significativa da Lógica de Programação não apenas favorece o aprendizado, mas também constitui uma estratégia pedagógica para combater a evasão e garantir o êxito formativo dos alunos da EPT.

Diante desse cenário, torna-se relevante propor estratégias pedagógicas que contribuam para tornar o processo de aprendizagem mais significativo e acessível aos estudantes dos cursos técnicos em Informática. Nesse sentido, este produto educacional foi elaborado com o objetivo de apresentar reflexões, orientações e possibilidades de abordagem da

disciplina de Pensamento Computacional e Algoritmos, considerando suas aplicações práticas e as perspectivas de atuação profissional na área da informática.

Capítulo 2 – Dificuldades de Aprendizagem

A baixa performance e a consequente evasão em disciplinas introdutórias da área de Tecnologia da Informação não são meros eventos isolados; representam um fenômeno complexo e multifacetado, amplamente documentado na literatura acadêmica e, mais crucialmente, revelado pela pesquisa conduzida no Curso Técnico em Informática para Internet do IFSul – Campus Gravataí. Este capítulo analisa as principais dificuldades de aprendizagem na disciplina de Pensamento Computacional e Algoritmos, contextualizando-as com os resultados obtidos por meio dos questionários aplicados aos estudantes evadidos e das entrevistas com docentes e supervisores.

A compreensão aprofundada desses desafios é o alicerce para a construção de estratégias pedagógicas eficazes, que transcendam a abordagem superficial e alcancem as raízes dos problemas identificados. São exploradas não apenas as barreiras cognitivas e metodológicas, mas, também os fatores emocionais, sociais e institucionais que se entrelaçam, criando um cenário de complexidade que exige uma intervenção pedagógica holística e consciente.

2.1 Altas taxas de reprovação e evasão

A aprendizagem da Lógica de Programação é frequentemente identificada como uma das principais barreiras nos cursos de TI, culminando em elevadas taxas de reprovação e, por conseguinte, de evasão. Estudos na área apontam que a taxa média de insucesso em disciplinas introdutórias de programação no ensino superior pode variar drasticamente, chegando a patamares preocupantes que evidenciam a magnitude do problema (Cheah, 2021). Esse padrão é igualmente observado nos cursos técnicos, onde a Lógica de Programação atua como um divisor de águas, testando a resiliência e a capacidade de adaptação dos estudantes desde as fases iniciais.

Evidências da Pesquisa no IFSul – Campus Gravataí:

A análise dos dados institucionais da CORAC (Coordenadoria de Registros Acadêmicos) do IFSul – Campus Gravataí corrobora dramaticamente essa realidade.

O relatório revelou que aproximadamente 60% dos estudantes reprovados em Algoritmos e Programação eventualmente se desligaram do curso nos anos subsequentes (CORAC 2025a). Este dado é alarmante e aponta para uma forte correlação entre o insucesso nesta disciplina e o abandono dos estudos.

Ainda, os dados da CORAC mostraram (CORAC 2025b):

Variação Anual: Entre 2020 e 2024, o número de reprovações variou de 10 a 27 estudantes, com um pico em 2021, "possivelmente relacionado a dificuldades de adaptação no pós-pandemia"

Reprovação por Desempenho: A reprovação acadêmica (por desempenho) manteve-se como a causa principal em muitos anos, atingindo 75% das reprovações em 2022, mesmo com a melhora na frequência dos estudantes. Isso evidencia que a mera presença não garante o sucesso, e que o conteúdo da disciplina, por si só, apresenta desafios significativos.

Reprovação com Dependência: O aumento expressivo de estudantes "aprovados com dependência" em 2023 e 2024 (63,6% das reprovações em 2024) indica uma estratégia institucional para evitar a evasão imediata. Contudo, essa medida pode apenas postergar o problema, contribuindo para a desmotivação a longo prazo, conforme apontam Giraffa e Mora (2013).

Momento da Evasão: Curiosamente, nenhum dos estudantes reprovados solicitou desligamento no primeiro ano, com todos os desligamentos ocorrendo a partir do segundo ano. Este comportamento sugere que os estudantes tentam persistir, talvez na esperança de superar as dificuldades encontradas, mas a decisão final de abandono se manifesta mais tardiamente.

A percepção dos professores do curso reforça que a disciplina funciona como um "primeiro filtro" ou "muro". Segundo um dos professores entrevistados, isso acontece porque muitos estudantes se frustram ao perceber a dificuldade do raciocínio lógico sem

uma base sólida em matemática e sem uma rotina de estudos adequada. Essa constatação alinha-se perfeitamente com a literatura, que já descreve as disciplinas iniciais de lógica e programação como um "gargalo" em cursos de TI (Reis Filho, 2025; Deters et al., 2008; Morais, Neto e Osório, 2020). Além dos dados relacionados à reprovação e evasão, os estudos já citados também apontam que muitos desses insucessos estão associados a desafios cognitivos inerentes ao aprendizado da programação.

2.2. Dificuldades cognitivas: sintaxe e abstração

Aprender lógica de programação vai muito além de memorizar comandos ou a sintaxe de uma linguagem específica. Envolve um complexo processo cognitivo de abstração, decomposição de problemas e construção de um raciocínio lógico que nem sempre é intuitivo. Como apontam Robins et al. (2003), além dos erros sintáticos – comuns entre iniciantes –, os estudantes frequentemente enfrentam obstáculos mais profundos relacionados à compreensão conceitual e à capacidade de planejar a estrutura lógica de um programa, bem como de entender a diferença entre o que se *espera* que o código faça e o que ele realmente faz.

A pesquisa desenvolvida no campus reforça que as dificuldades cognitivas são um pilar central para a evasão, manifestando-se em diversas frentes, a partir das contribuições de 13 estudantes evadidos que participaram da pesquisa:

- ***Ausência de Conhecimento Prévio:*** *Tanto os docentes quanto os estudantes evadidos confirmam a chegada de estudantes com pouca ou nenhuma experiência em programação. Um dos professores destacou que: "A imensa maioria dos estudantes não teve contato algum com programação e a disciplina é fundamental para que aprendam a desenvolver sistemas no futuro.". De fato, a pesquisa com evadidos mostrou que 9 dos 13 estudantes (69,23% dos estudantes) não possuíam conhecimento prévio em programação antes de ingressar no curso. Essa lacuna inicial exige um esforço adaptativo considerável.*
- ***Déficits em Habilidades Fundamentais:*** *A matemática e a interpretação de textos emergem como barreiras significativas. Um dos professores destacou: "Matemática, Interpretação de textos, dificuldades em inglês em especial com os comandos da*

linguagem C.", enquanto o outro professor mencionou a "Deficiência em matemática (conceitos básicos)" como um fator que dificulta o acompanhamento dos temas. Essas habilidades são pré-requisitos essenciais para o desenvolvimento do raciocínio lógico-computacional, que, por sua vez, é a espinha dorsal da lógica de programação (Wing, 2006).

- ***Interpretação e Modelagem de Problemas:*** *A dificuldade em traduzir problemas do mundo real para uma lógica computacional é um obstáculo crítico. Os estudantes evadidos apontaram a "Falta de conhecimento a respeito dos problemas apresentados nos exercícios" (30,77% dos estudantes) e a "Interpretação dos exercícios que eram apresentados" (23,08% dos estudantes) como suas principais dificuldades na disciplina. Essa constatação alinha-se à pesquisa de Morais, Neto e Osório (2020), que já indicava a interpretação de problemas computacionais como um dos maiores desafios para os estudantes.*
- ***Complexidade de Estruturas Lógicas:*** *Estudos, como o de Santos, N. R. M. et al. (2022), revelaram que os estudantes apresentaram níveis mais elevados de confusão em questões que envolviam estruturas condicionais e de repetição. Isso sugere que a dificuldade não se restringe a pequenos detalhes, mas se estende a conceitos fundamentais que exigem um raciocínio lógico mais elaborado.*

Para minimizar essas dificuldades, é crucial que os professores empreguem estratégias que facilitem a visualização da lógica e a conexão entre a abstração computacional e o cotidiano do aluno, antes de introduzir a sintaxe.

Uma forma de tentar atenuar esta situação é a aplicação de sequências didáticas como, por exemplo a sequência didática 4 **Processos do dia a dia em fluxograma**.

Sequência Didática 4:
Processos do Dia a Dia em Fluxograma

Realize diversas atividades com fluxogramas em suas aulas. De preferência em momentos iniciais, isso ajuda a melhorar o raciocínio lógico.

Apesar desta sugestão de sequência didática propor uma atividade desplugada, ela também pode ser aplicada em ambiente computacional disponibilizado pelo professor, ou

mesmo servir de base para uma atividade preliminar a uma atividade com uma linguagem computacional, ficando a critério do professor.

2.3. Falta de experiência prévia e contexto disciplinar

A ausência de experiências anteriores com programação impacta significativamente a aprendizagem inicial. Estudantes sem familiaridade com lógica computacional ou algoritmos costumam apresentar maior resistência à abstração e maior insegurança na resolução de problemas (Cheah, 2021; Demirci et al., 2022). Em cursos não específicos da área de TI, a falta de contextualização agrava ainda mais essa dificuldade. No entanto, mesmo em cursos técnicos e superiores da área, a chegada de estudantes com diferentes níveis de preparo, muitas vezes sem qualquer contato prévio, cria um cenário desafiador para a disciplina de Pensamento Computacional e Algoritmos.

Os achados da pesquisa no campus confirmam que a falta de experiência prévia e, crucialmente, o desalinhamento de expectativas são fatores substanciais que contribuem para a desmotivação e, em última instância, para a evasão:

- ***Preponderância de Iniciantes:*** *A pesquisa com os estudantes evadidos revelou que a maioria – 9 de 13 participantes – não possuía conhecimento prévio em programação antes de ingressar no curso. Essa carência de base exige que o ensino comece do zero para muitos, o que pode ser um choque para quem esperava um avanço mais rápido ou uma aplicação mais imediata.*
- ***Expectativas Desalinhadas:*** *Um ponto de destaque na análise dos questionários com estudantes evadidos foi o desalinhamento entre a expectativa inicial e a realidade do curso. Depoimentos como "Inicialmente, eu tinha interesse em tecnologia, mas o curso foi diferente do que eu imaginava. Tive muita dificuldade e acabei desanimando" ou "nunca tive interesse na área da programação" são alarmantes. Eles revelam que, mesmo com a motivação de "Gostar de tecnologia" (citada por 5 estudantes evadidos), a falta de clareza sobre o que o curso realmente demandaria pode levar à frustração.*
- ***Impacto de Fatores Externos:*** *A pesquisa também destacou como fatores externos e institucionais podem intensificar essa sensação de desalinhamento. Alunos mencionaram problemas como "a troca de professores nas disciplinas do técnico era constante" e "a greve e a falta dos professores nas disciplinas de programação", que, somados às dificuldades da disciplina, solidificaram a decisão de evadir. O contexto da pandemia, com a dificuldade de aprendizado remoto ("Não consegui aprender em casa, em meio à pandemia..."), também foi um fator relevante, alterando a experiência e as expectativas sobre a modalidade de ensino.*

Essa lacuna entre o que o aluno espera e o que ele encontra na disciplina e no curso como um todo é um terreno fértil para a desmotivação, o baixo desempenho e, consequentemente, para a evasão.

Uma sugestão prática para reduzir o desalinhamento de expectativas e a falta de experiência prévia, é a realização de um diagnóstico inicial abrangente e promover a transparência sobre o percurso de aprendizagem. Para tanto podemos utilizar a **sequência didática 5 "Meu futuro na Ti: Diagnóstico e Realidade"**.

Sequência Didática 5:
Meu Futuro na TI: Diagnóstico e Realidade

Você pode adaptar e aplicar esta sequência trimestralmente, para avaliar se o nível de desalinhamento dos alunos melhorou em relação ao inicio do ano!

2.4. AUTOEFICÁCIA, AUTOCONCEITO E SENSO DE PERTENCIMENTO

As dificuldades em disciplinas como Pensamento Computacional e Algoritmos não se restringem apenas a aspectos cognitivos ou de conhecimento prévio. Fatores psicológicos, como a baixa autoeficácia (a crença do aluno sobre sua capacidade de aprender), um autoconceito negativo em relação à área e a ausência de um senso de pertencimento ao ambiente acadêmico ou à comunidade de programação, são igualmente determinantes para a desmotivação e a evasão. Muitos estudantes internalizam a ideia de que "programação é para gênios", o que reduz sua persistência diante das dificuldades e os afasta da busca por ajuda (Seda et al., 2023).

A pesquisa ofereceu um panorama claro de como esses fatores intangíveis contribuem para a problemática da evasão:

- ***Insegurança e Desmotivação:*** *os depoimentos dos docentes revelam uma percepção da insegurança dos estudantes, especialmente em relação à matemática. Um dos professores apontou a "insegurança diante da matemática" como um dos aspectos emocionais que dificultam o desempenho. Essa insegurança, combinada com a percepção de que a disciplina funciona como um "filtro", contribuindo para um ciclo de desmotivação que, segundo Giraffa e Mora (2013), culmina em desistências.*
- ***Falta de Identificação com a Área:*** *A pesquisa com estudantes evadidos mostrou que a falta de identificação com o curso ou a área de programação é um forte motivador para o abandono. Frases como* ***"****Não. Este tipo de curso não é algo que me interessa." ou "não me identifiquei de maneira alguma com a área da programação" ecoam a preocupação de um dos professores entrevistados, que observou: "Alunos não se identificam com o curso muitas vezes". Essa desconexão impede o desenvolvimento de um autoconceito positivo como futuro profissional de TI.*
- ***Dificuldade em Buscar Apoio Contínuo:*** *Embora a maioria dos estudantes evadidos (9 de 13) tenha procurado ajuda, essa busca foi muitas vezes direcionada à "Atendimento com o professor" ou "Colegas", e não de forma contínua, especialmente próximo às avaliações. Foi mencionado por um dos professores que a "dificuldade em procurar apoio" e a "falta de rotina de estudos" como fatores que contribuem para a persistência das dificuldades, indicando que a autoeficácia e o senso de pertencimento não eram fortes o suficiente para sustentar a busca por suporte de longo.*
- ***Melhora com Identificação Prévia:*** *Em contraste, os três professores entrevistados notaram que o número de estudantes que chegam mais preparados e se identificam com a área (já com conhecimento em linguagens como JavaScript e Python) está aumentando, o que facilita a curva de aprendizagem geral no curso e, consequentemente, reduz as reprovações e à evasão. Isso sublinha à importância da identificação e da autoeficácia no processo de aprendizagem.*

Para combater a baixa autoeficácia, o autoconceito negativo e a falta de pertencimento, é vital criar um ambiente de aprendizagem colaborativo e de apoio mútuo, onde o erro é visto como parte do processo e o sucesso é compartilhado.

Como ferramentas de apoio algumas estratégias podem ser aplicadas, foi selecionada a estratégia "**Programando juntos, crescendo juntos**".

Estratégia 1:
Programando Juntos, Crescendo Juntos

A estratégia de aprendizagem por pares pode ser uma grande aliada no ensino de programação! Quer conhecer mais, sugiro a leitura do artigo:
Utilização da técnica de aprendizagem por pares para ensinar algorítmos e programação

2.5. Falta de alinhamento entre avaliação e prática

A avaliação é uma peça fundamental no processo de ensino e aprendizagem, mas, quando desalinhada com as particularidades do ensino de Lógica de Programação, pode se tornar um obstáculo em vez de um instrumento de apoio. Métodos tradicionais, que focam excessivamente na reprodução de comandos ou na memorização de sintaxe, muitas vezes falham em captar a real compreensão do estudante sobre os conceitos de abstração, raciocínio lógico e depuração (Mtaho e Mselle, 2023). Essa inadequação avaliativa pode gerar um "falso senso de domínio" para alguns estudantes e dificultar o diagnóstico preciso das suas verdadeiras lacunas de aprendizagem, contribuindo para a frustração e a desmotivação (Mtaho e Mselle, 2023).

Os dados da pesquisa revelam que a percepção dos estudantes sobre as metodologias e materiais didáticos, que estão intrinsecamente ligados às práticas avaliativas, contribui para essa problemática:

- ***Insatisfação com Material e Metodologia:*** *A pesquisa com estudantes evadidos demonstrou uma lacuna na percepção da eficácia das estratégias pedagógicas e do material didático. A maioria dos estudantes evadidos (6 de 13) considerou o material didático apenas "Parcialmente adequado", e muitos avaliaram a metodologia como "Razoável, mas poderia ser mais prática" (5 estudantes) ou "Difícil de acompanhar" (4 estudantes. Isso sugere que a forma como o aprendizado era proposto e, consequentemente, avaliado, não atendia plenamente às necessidades dos estudantes, especialmente na construção de um entendimento mais profundo e prático.*

- ***Reprovação com Dependência e Diagnóstico:*** *A estratégia institucional de "aprovado com dependência", embora busque reduzir a evasão imediata, pode mascarar a necessidade de um diagnóstico mais aprofundado das dificuldades. Ao permitir que o aluno avance com uma lacuna não resolvida, o desafio é apenas postergado, podendo levar a um acúmulo de frustração e, por fim, à desistência, como já observado em outros contextos (Giraffa e Mora, 2013). Tal cenário reflete a necessidade de avaliações que não apenas verifiquem o aprendizado, mas que sirvam como ferramentas de diagnóstico contínuo e apoio pedagógico.*
- ***Foco na Codificação, não na Compreensão:*** *A pesquisa de Santos, N. R. M. et al. (2022) ao analisar o nível de confusão de estudantes, observou que a maioria dos relatos de confusão estava relacionada ao processo de codificação, e não à compreensão do enunciado. Isso aponta para a necessidade de estratégias de ensino que auxiliem os estudantes na tradução do problema em código, e, consequentemente, de avaliações que reflitam essa transição, indo além da mera validação de sintaxe para focar na lógica subjacente.*

A falta de um alinhamento claro entre as ferramentas de avaliação e os reais objetivos de desenvolvimento do pensamento computacional podem, portanto, agravar as dificuldades dos estudantes, impedindo um feedback construtivo e o desenvolvimento das habilidades essenciais.

Para superar o desalinhamento entre avaliação e prática, é fundamental adotar uma abordagem mais formativa e contextualizada, onde a avaliação se torna parte integrante do processo de aprendizagem, oferecendo feedback contínuo e direcionado. Como forma de minimizar essas dificuldades, propõe-se a estratégia **"A avaliação que ensina"**.

Estratégia 2:
A avaliação que ensina

Aprimore seus conhecimentos sobre avaliação em pares lendo o artigo: Uma avaliação de aprendizagem cooperativa e em pares na disciplina de Programação de Computadores: um relato de experiência.

2.6. DESAFIOS DE AUTODIAGNÓSTICO

Mesmo em níveis mais avançados, como nas disciplinas de estrutura de dados e programação intermediária, muitos estudantes não conseguem explicar como seus próprios códigos funcionam, ou o porquê de um determinado erro. Essa limitação revela uma compreensão superficial dos conceitos fundamentais e limita a evolução do pensamento computacional (Alharbi et al., 2021; Patitsas et al., 2022). A incapacidade de autodiagnosticar a origem de suas dificuldades impede que o aluno formule perguntas eficazes, procure o tipo certo de apoio e, consequentemente, se torne um aprendiz autônomo e resiliente. O aluno pode sentir-se confuso ou frustrado, mas não consegue articular a raiz de seu problema, tornando a intervenção pedagógica mais desafiadora.

Embora a dissertação não trate explicitamente do conceito de autodiagnóstico, diversos achados da pesquisa apontam indiretamente para essa dificuldade. Em diferentes momentos, observa-se que os estudantes apresentam limitações para identificar a natureza de seus próprios problemas de aprendizagem, conforme evidenciado nos aspectos descritos a seguir:

- ***Confusão na Codificação vs. Compreensão:*** *A análise de Santos, N. R. M. et al. (2022) é particularmente relevante aqui. Ela revelou que a maioria dos relatos de confusão dos estudantes estava relacionada ao processo de codificação, e não à compreensão do enunciado. Isso sugere que os estudantes podem estar focando na "sintaxe" ou na "escrita do código" como fonte de seu problema, quando a real dificuldade reside na "interpretação" ou "lógica" subjacente ao problema. Essa má atribuição da causa dificulta o autodiagnóstico correto e, por extensão, a busca por uma solução efetiva.*

- ***Natureza das Dificuldades Relatadas pelos Evadidos:*** *Quando questionados sobre suas principais dificuldades na Lógica de Programação, os estudantes evadidos citaram "Falta de conhecimento a respeito dos problemas apresentados nos exercícios" e "Interpretação dos exercícios que eram apresentados". Essas respostas podem indicar que o problema não era apenas a dificuldade em si, mas também a incapacidade de entender plenamente o que estava sendo solicitado ou o conhecimento necessário para abordar a questão. Sem essa clareza, o aluno não consegue direcionar seus esforços de estudo ou pedir ajuda de forma assertiva.*

- ***Busca por apoio próximo às avaliações:*** *A observação de um dos professores entrevistados de que a busca por apoio era "principalmente próximos às avaliações,*

embora ainda haja pouca procura contínua ao longo do semestre ou ano letivo" pode ser um sintoma da dificuldade de autodiagnóstico. Muitos estudantes só percebem à extensão de suas lacunas quando a avaliação se aproxima, em vez de conseguir monitorar seu próprio aprendizado de forma contínua e identificar as dificuldades à medida que surgem.

Esses pontos sugerem que os estudantes podem estar enfrentando uma espécie de "cegueira cognitiva", entendida aqui como uma limitação metacognitiva na identificação da origem de suas próprias dificuldades de aprendizagem. Tal condição compromete a progressão no aprendizado e configura-se como um fator silencioso, porém potente, de desmotivação e evasão.

Para desenvolver a capacidade de autodiagnóstico, é fundamental que os estudantes sejam incentivados a refletir sobre seu processo de aprendizado e a verbalizar suas dificuldades de forma estruturada. Isso transforma a confusão em perguntas específicas e direcionadas. Uma boa estratégia é criar um diário de bordo colaborativo ou mesmo um *Debugging* colaborativo.

Estratégia 3:
Diário de Bordo do Codificador e Debugging Colaborativo

Aprimore seus conhecimentos sobre como criar um diário de bordo no site Rhema Neuroeducação.

Considerações Finais do Capítulo

Em suma, as barreiras identificadas neste capítulo — que vão desde a complexidade da abstração e sintaxe até os desafios socioemocionais de autoeficácia — revelam que o insucesso na aprendizagem da lógica de programação raramente é fruto de um único

fator. A pesquisa realizada no IFSul Campus Gravataí demonstra que, quando o estudante não consegue realizar um autodiagnóstico preciso de suas lacunas, a desmotivação se instala, tornando a evasão um desfecho recorrente diante do “muro” da reprovação.

Portanto, reconhecer essas dificuldades não deve servir para estigmatizar o estudante, mas para fundamentar uma revisão profunda das estratégias de ensino. É a partir desse entendimento que o próximo capítulo apresenta metodologias e abordagens práticas voltadas a transformar o ambiente de sala de aula em um espaço de aprendizagem significativa, interdisciplinar e, acima de tudo, inclusiva."

CAPÍTULO 3 – ESTRATÉGIAS PEDAGÓGICAS

A superação das dificuldades de aprendizagem na disciplina de Pensamento Computacional e Algoritmos, identificadas no Capítulo 2, demanda a adoção de estratégias pedagógicas e alinhadas aos princípios da Educação Profissional e Tecnológica (EPT). O modelo tradicional, frequentemente centrado na transmissão passiva de conteúdo e na memorização de comandos, tem se mostrado insuficiente para engajar os estudantes e desenvolver as habilidades cognitivas e socioemocionais cruciais para a área de Tecnologia da Informação.

No âmbito da EPT, a centralidade do trabalho como princípio educativo exige que o ensino esteja contextualizado em situações reais e significativas, articulando teoria, prática, ciência, cultura e tecnologia (FRIGOTTO; CIAVATTA; RAMOS, 2005). Dessa forma, o ensino da lógica de programação não deve ser tratado como um conteúdo isolado, mas como parte de uma formação omnilateral, integral e interconectada, que prepare o indivíduo para o mundo do trabalho e para o exercício pleno da cidadania.

Neste capítulo, serão exploradas abordagens pedagógicas que transcendem a mera técnica, buscando promover uma formação integral e significativa. O foco estará em metodologias que transformam o estudante em protagonista de seu próprio processo de aprendizagem, conectando o conhecimento teórico à prática e à realidade do mundo do trabalho. Neste capítulo são apresentadas e discutidas estratégias pedagógicas que buscam:

Combater as dificuldades cognitivas, como a abstração e o raciocínio lógico, por meio de abordagens mais interativas e concretas.

Reduzir o impacto da falta de experiência prévia e alinhar as expectativas, tornando o aprendizado mais relevante e acessível desde o início.

Fortalecer a autoeficácia e o senso de pertencimento dos estudantes, criando um ambiente de apoio e colaboração.

Promover o autodiagnóstico e a reflexão sobre o próprio aprendizado, capacitando o aluno a identificar e superar suas lacunas.

Diminuir as altas taxas de reprovação e evasão, transformando o "filtro" da lógica de programação em uma ponte de acesso ao sucesso e à permanência estudantil.

Ao integrar teoria e prática, ciência e cultura, e ao contextualizar o ensino em situações reais e significativas, este capítulo propõe um caminho para reencantar os estudantes com a lógica de programação, tornando-a um instrumento de empoderamento e transformação, conforme preconiza a EPT.

3.1 Metodologias Ativas e Aprendizagem Significativa

A superação das dificuldades de aprendizagem em lógica de programação exige a adoção de estratégias pedagógicas inovadoras que transcendam a mera transmissão de conteúdo. A literatura aponta que metodologias tradicionais, centradas na exposição teórica e na memorização de comandos, têm se mostrado insuficientes para engajar os estudantes e promover o desenvolvimento do pensamento computacional (Robins et al., 2003; Cheah, 2021). As metodologias ativas, ao colocarem o aluno no centro do processo, estimulam a participação, a resolução de problemas e a construção colaborativa do conhecimento.

Metodologias como a Aprendizagem Baseada em Problemas (ABP), gamificação, sala de aula invertida e ensino por pares têm sido amplamente utilizadas no ensino de programação por promoverem maior engajamento e protagonismo discente (Silva; Fernandes, 2021; Santos et al., 2020). Autores como Ausubel (2003) defendem que a aprendizagem significativa ocorre quando o novo conhecimento se ancora de maneira substantiva em estruturas cognitivas já existentes. Complementarmente, Freire (1998, apud Vieira et al., 2019) destaca que a construção do saber exige curiosidade, observação crítica e vínculo com a realidade do educando. Nesse contexto, propor desafios de

programação baseados em problemas do cotidiano torna-se uma prática potente para desenvolver não apenas habilidades técnicas, mas também reflexivas e criativas.

Conectando com as Dificuldades Identificadas no Capítulo 2:

A incorporação de metodologias ativas e a ênfase na aprendizagem significativa constituem não apenas alternativas metodológicas, mas respostas estruturais aos desafios mapeados anteriormente. Essas abordagens reposicionam o estudante no centro do processo educativo, oferecendo condições para enfrentar, de modo sistemático e articulado, os obstáculos identificados no ensino introdutório de programação.

- **Altas Taxas de Reprovação e Evasão (Seção 2.1):** O engajamento ativo do estudante — por meio de atividades práticas, colaboração e resolução de problemas reais — mitiga a sensação de distanciamento em relação ao conteúdo e reduz a percepção de inutilidade imediata da disciplina. Ao atribuir significado ao aprendizado, metodologias ativas reforçam a motivação intrínseca e oferecem um caminho realista para diminuir tanto a reprovação quanto o abandono, frequentemente associados à falta de conexão entre teoria e prática.

- **Dificuldades Cognitivas (Seção 2.2):** Muitas das barreiras cognitivas observadas decorrem da ausência de contextos que permitam ao estudante construir relações, elaborar hipóteses e validar soluções. Ao trabalhar com problemas abertos, desafios progressivos e atividades que exigem experimentação, as metodologias ativas favorecem o desenvolvimento de habilidades de abstração e pensamento computacional. A lógica deixa de ser uma sequência rígida de comandos para se tornar um instrumento de construção de sentido, facilitando a internalização de conceitos complexos.

- **Falta de Experiência Prévia e Desalinhamento de Expectativas (Seção 2.3):** A distância entre a experiência prévia dos estudantes e as exigências iniciais da disciplina cria fricção cognitiva e emocional. Ao trazer problemas conectados ao cotidiano, a cenários conhecidos ou a interesses pessoais, o docente reduz a assimetria entre expectativa e realidade. Isso não apenas facilita a entrada dos estudantes no universo da programação, como também constrói uma ponte entre seus conhecimentos prévios e os novos esquemas conceituais que precisam ser desenvolvidos.

- **Autoeficácia e Senso de Pertencimento (Seção 2.4):** O protagonismo do aluno e o trabalho em grupo fomentam um ambiente colaborativo, onde o erro é parte do aprendizado e o sucesso é compartilhado. Isso fortalece a autoeficácia, reduz a sensação de que "programação é para gênios" e promove o senso de pertencimento à comunidade de aprendizagem.

- **Falta de alinhamento entre avaliação e prática** (Seção 2.5): Quando as avaliações se distanciam das atividades realizadas em sala, reforçam a ansiedade e a sensação de imprevisibilidade. As metodologias ativas, ao integrarem avaliação formativa, ciclos curtos de feedback e práticas avaliativas baseadas em desempenho, diminuem esse hiato. O estudante é continuamente convidado a refletir, ajustar e testar suas soluções, aproximando avaliação e experiência de aprendizagem.

- **Desafios de Autodiagnóstico** (Seção 2.6)**:** A capacidade de avaliar o próprio progresso é fundamental para o desenvolvimento da autonomia no processo de aprendizagem. Em ambientes que utilizam *feedback* contínuo — seja por meio de jogos educacionais, rubricas claras ou sistemas automáticos de verificação — o aluno consegue identificar seus pontos fortes e fragilidades de forma mais precisa. Esse movimento favorece a metacognição e fortalece a autorregulação, elementos centrais para o avanço consistente na programação.

Esse movimento favorece a metacognição e fortalece a autorregulação, elementos centrais para o avanço consistente na programação. Uma forma de operacionalizar esses aspectos é apresentada na **Sugestão Prática 1 – Desafio de Programação Gamificado**.

Sugestão Prática 1:

Desafio de Programação Gamificado

Veja aqui, 7 exemplos de Gamificação para aplicar em sala de aula.

CAIXA – RECURSOS PARA O PROFESSOR

Esta seção reúne materiais de referência, guias, artigos e plataformas digitais que podem auxiliar o docente no aprofundamento das metodologias ativas e de estratégias pedagógicas contemporâneas aplicadas à Educação Profissional e Tecnológica. Os links indicados são de acesso público e foram selecionados por sua confiabilidade e utilidade pedagógica.

A seguir, apresenta-se um conjunto de ferramentas pedagógicas que podem subsidiar o trabalho docente no planejamento e na condução de práticas voltadas ao desenvolvimento da aprendizagem ativa no contexto da Educação Profissional e Tecnológica. Cada abordagem mobiliza princípios específicos e contribui, de maneira complementar, para responder às demandas formativas contemporâneas. As metodologias ativas, de modo geral, reposicionam o estudante como agente central do processo educativo, favorecendo sua autonomia e ampliando oportunidades de participação qualificada. A sala de aula invertida reorganiza o tempo pedagógico, permitindo que o momento presencial seja dedicado à resolução de problemas e à mediação direta por parte do professor (Bergmann; Sams, 2016). A aprendizagem baseada em problemas incentiva investigação, análise e tomada de decisões (Barrows, 1986), enquanto a aprendizagem baseada em projetos integra diferentes saberes em atividades aplicadas e contextualizadas (Dewey, 1938). A gamificação introduz elementos motivacionais que intensificam o engajamento (Kapp, 2012); a aprendizagem colaborativa promove interações ricas e fundamentais à construção partilhada do conhecimento (Vygotsky, 1991); e a aprendizagem significativa orienta a organização dos conteúdos por meio de vínculos consistentes com os conhecimentos prévios dos estudantes (Ausubel, 2003).

Articuladas ao uso intencional de tecnologias educacionais, essas ferramentas compõem um repertório didático e diversificado, capaz de apoiar o professor na implementação de experiências formativas mais dinâmicas, coerentes com os princípios da EPT e alinhadas às necessidades de aprendizagem identificadas ao longo deste estudo.

METODOLOGIAS ATIVAS VISÃO GERAL	SALA DE AULA INVERTIDA FLIPPED CLASSROOM

Metodologias ativas de aprendizagem: o que são e 15 tipos: https://www.totvs.com/blog/instituicao-de-ensino/metodologias-ativas-de-aprendizagem	Material do Instituto Federal de Goiás, produzido por Carlos Roberto da Silveira Junior: https://ifg.edu.br/attachments/article/19169/Sala%20de%20aula%20invertida_%20por%20onde%20come%C3%A7ar%20(21-12-2020).pdf
APRENDIZAGEM BASEADA EM PROBLEMAS (PBL)	**APRENDIZAGEM BASEADA EM PROJETOS (ABP / PBL)**
Universidade de Maastricht – Referência mundial no modelo PBL: https://www.maastrichtuniversity.nl/education/why-um/problem-based-learning	PBLWorks (Buck Institute for Education) – Estrutura e exemplos de projetos: https://www.pblworks.org/what-is-pbl
GAMIFICAÇÃO	**APRENDIZAGEM SIGNIFICATIVA (AUSUBEL)**
Laboratório de Gamificação da UNICAMP – artigos, tutoriais e estudos: https://liag.ft.unicamp.br/jogos-educacionais/	Portal do professor -MEC: https://portaldoprofessor.mec.gov.br/storage/materiais/0000012381.pdf

TECNOLOGIAS EDUCACIONAIS PARA O ENSINO DE PROGRAMAÇÃO	
Code.org – Introdução à lógica e à programação em ambiente visual: https://code.org/	Replit – Ambiente online para múltiplas linguagens: https://replit.com/
Scratch – Programação visual para atividades introdutórias: https://scratch.mit.edu/	Jupyter Notebook – Ferramenta para programação e ciência de dados: https://jupyter.org/
QUESTÕES PARA EXERCÍCIOS DE AULA, DESAFIOS OU PROVAS	
Bebras Brasil – https://www.bebrasbrasil.com.br/	

3.2 CURRÍCULO INTEGRADO E INTERDISCIPLINARIDADE

Conforme Silva (2016), a efetiva operacionalização do currículo integrado é essencial para combater a fragmentação do conhecimento e fomentar a permanência estudantil. No contexto da EPT, a formação omnilateral do estudante implica na articulação entre diferentes áreas do saber – técnico, científico e cultural – rompendo com a visão disciplinar isolada. Em vez de tratar a lógica de programação como uma disciplina descolada da realidade, sua abordagem deve dialogar com outros componentes curriculares, projetos interdisciplinares e contextos concretos do mundo do trabalho.

Exemplo disso é o projeto Workshop de Informática, realizado no IFSC – Câmpus Chapecó, em que diferentes disciplinas técnicas foram articuladas para desenvolver soluções reais a partir de problemas identificados pelos próprios estudantes (Silva, 2016). Essa proposta, além de aplicar os conceitos de forma prática, favoreceu o sentimento de

pertencimento e de utilidade do aprendizado — elementos essenciais para reduzir a evasão. A interdisciplinaridade permite que o aluno perceba a aplicabilidade da lógica em diversas esferas, fortalecendo a conexão entre o conhecimento adquirido e as demandas profissionais e sociais.

Conectando com as Dificuldades Identificadas no Capítulo 2:

A integração curricular e a interdisciplinaridade oferecem respostas diretas a vários dos problemas identificados:

- **Falta de Experiência Prévia e Desalinhamento de Expectativas** (Seção 2.3)**:** Conectar a lógica de programação a outras disciplinas ou áreas de interesse do aluno (matemática, física, química, biologia, até mesmo artes ou ciências sociais) pode tornar o conteúdo mais relevante e menos abstrato para aqueles sem experiência prévia, ajustando suas expectativas sobre a amplitude da área.
- **Autoeficácia, Autoconceito e Senso de Pertencimento** (Seção 2.4)**:** Projetos interdisciplinares permitem que os estudantes vejam o "sentido" e a "utilidade" do que estão aprendendo, fortalecendo sua percepção de competência (autoeficácia) e de como eles se encaixam no mundo profissional (autoconceito). A colaboração com outras áreas também pode gerar um senso de pertencimento mais amplo.
- **Dificuldades Cognitivas** (Seção 2.2) **e Desafios de Autodiagnóstico** (Seção 2.6)**:** Ao aplicar conceitos de lógica em diferentes contextos, o aluno é desafiado a utilizar seu raciocínio de maneiras variadas, o que pode aprofundar sua compreensão e ajudá-lo a identificar onde suas lacunas residem em um cenário mais amplo, e não apenas no código puro.

Uma forma de ampliar essa abordagem consiste na integração da lógica de programação com outras áreas do conhecimento, conforme exemplificado na **Sugestão Prática 2 – Projeto Interdisciplinar**.

Sugestão Prática 2:
Projeto interdisciplinar

interdisciplinaridade é um conhecimento muito importante para um professor saiba mais sobre o assunto.

3.3 A Pesquisa como Princípio Educativo

A pesquisa integrada ao ensino permite que o estudante se torne sujeito do próprio processo formativo, uma abordagem fundamental na EPT. Quando os estudantes são incentivados a investigar, testar hipóteses e construir soluções por meio da programação, a disciplina deixa de ser uma sequência de comandos e se transforma em um espaço de investigação e descoberta. Isso fomenta o desenvolvimento do pensamento crítico e da autonomia intelectual.

Freire (1998) e Vieira et al. (2019) destacam que a pesquisa desenvolve autonomia intelectual, curiosidade crítica e capacidade de enfrentamento de problemas. Santos (2019) complementa que, a partir de uma problemática real, os estudantes constroem saberes por meio de um movimento dialético de análise, confronto e superação do senso comum. Na lógica de programação, isso significa ir além da mera resolução de exercícios propostos, buscando a compreensão das diferentes abordagens, a otimização de algoritmos e a investigação de novas tecnologias.

Conectando com as Dificuldades Identificadas no Capítulo 2:

A pesquisa como princípio educativo atua na superação de diversas dificuldades:

- **Dificuldades Cognitivas** (Seção 2.2) **e Desafios de Autodiagnóstico** (Seção 2.6)**:** A investigação ativa de problemas de programação e suas soluções desenvolve a capacidade do aluno de analisar, testar e depurar. Esse processo de descobrir por si mesmo a origem de um problema e a melhor forma de solucioná-lo é essencial para aprimorar a lógica e a capacidade de autodiagnóstico.
- **Autoeficácia e Senso de Pertencimento** (Seção 2.4)**:** Ao se engajar em um processo de pesquisa, o aluno se sente mais capaz e produtivo, fortalecendo sua autoeficácia. A sensação de estar contribuindo com o conhecimento ou com a solução de um problema real fomenta o senso de pertencimento e a motivação.
- **Falta de Experiência Prévia e Desalinhamento de Expectativas** (Seção 2.3)**:** A pesquisa permite que o aluno explore áreas de seu interesse dentro da programação, construindo conhecimento de forma autônoma e conectando a teoria a aplicações que ele mesmo descobriu, o que pode alinhar as expectativas com a profundidade da área.

Uma forma de operacionalizar essa abordagem investigativa é apresentada na **Sugestão Prática 3 – Mini Investigação: Algoritmos para Otimização**.

Sugestão prática 3:
Mini-Investigação: Algoritmos para Otimização

Você pode encontrar ideias para algoritmos nesta playlist.

3.4 Teoria da Atividade e Aprendizagem Expansiva

A Teoria da Atividade, especialmente na formulação de Engeström (2001) sobre a aprendizagem expansiva, oferece um referencial metodológico robusto para a inovação

pedagógica na lógica de programação. A ideia central é que o processo de aprender não se limita à assimilação de conteúdos existentes, mas envolve a criação de novas práticas e significados a partir de contradições vivenciadas no cotidiano. Essa perspectiva é fundamental para a EPT, que busca formar profissionais capazes de intervir e transformar a realidade.

Nesse sentido, ao invés de se restringir à reprodução de algoritmos padronizados, a disciplina pode ser organizada em torno de projetos colaborativos, nos quais os estudantes identificam problemas reais, modelam soluções em equipe e implementam protótipos utilizando linguagem de programação. Essa abordagem transforma a sala de aula em um espaço de "co-configuração", onde o conhecimento é continuamente reconstruído de forma coletiva e contextualizada (Santos, 2019; Paixão; Pinto, 2020). A aprendizagem expansiva ocorre quando os estudantes, confrontados com contradições (desafios, problemas sem solução óbvia), expandem seu próprio sistema de atividade para superá-las, desenvolvendo novas ferramentas e modos de pensar.

Conectando com as Dificuldades Identificadas no Capítulo 2:

A Teoria da Atividade e a aprendizagem expansiva respondem de forma abrangente às dificuldades:

- **Dificuldades Cognitivas** (Seção 2.2)**:** Ao trabalhar com problemas do cotidiano e "reais", os estudantes são forçados a expandir seu raciocínio lógico e sua capacidade de abstração para lidar com contradições e desenvolver soluções inovadoras.
- **Falta de Experiência Prévia e Desalinhamento de Expectativas** (Seção 2.3)**:** A aprendizagem expansiva motiva os estudantes a ir além do que já sabem, pois as contradições do problema exigem a criação de novos conhecimentos e práticas, superando a barreira da falta de experiência inicial.

- **Autoeficácia, Autoconceito e Senso de Pertencimento** (Seção 2.4)**:** O processo colaborativo de superar desafios reais e de criar algo novo ("co-configuração") fortalece a crença na própria capacidade (autoeficácia) e constrói um senso de comunidade e pertencimento ao grupo.
- **Desafios de Autodiagnóstico** (Seção 2.6)**:** Ao serem confrontados com contradições nos projetos, os estudantes são naturalmente levados a diagnosticar a origem dos problemas (seja na lógica, na implementação ou na concepção da solução) e a buscar meios de superá-los de forma autônoma e em equipe.

Uma forma de aplicar esses princípios no ensino de lógica de programação é apresentada na **Sugestão Prática 4 – Projeto de "Solução para um Desafio Escolar" com Abordagem Expansiva**.

Sugestão Prática 4:
Projeto de "Solução para um Desafio Escolar" com Abordagem Expansiva.

Como a ideia aqui é um projeto de longo prazo procure aplicar alguma adaptação de uma metodologia ágil, como por exemplo Scrum ou XP.

3.5 Tecnologias Educacionais e Ferramentas Visuais

O uso de ambientes de codificação visual, como Blockly, Scratch, Visualg e simuladores de fluxogramas como por exemplo Flowgorithm, draw.io e Lucidchart, facilita o aprendizado de lógica por estudantes iniciantes, permitindo a visualização das estruturas lógicas de forma concreta e interativa (Cheah, 2021). Essas ferramentas são pontes

importantes entre o raciocínio abstrato e a representação concreta do código, especialmente para estudantes que estão tendo o primeiro contato com a programação.

O *feedback* imediato fornecido por essas ferramentas ajuda os estudantes a compreenderem seus erros de maneira formativa, favorecendo o processo de autoaprendizagem. Além disso, plataformas como o Replit, Code.org e o VSCode com extensões didáticas permitem experiências práticas de programação com maior autonomia e flexibilidade, proporcionando um ambiente de experimentação seguro e acessível. A escolha da ferramenta adequada deve considerar o nível de conhecimento prévio dos estudantes e os objetivos de aprendizagem, progredindo do visual para o textual de forma gradual.

Conectando com as Dificuldades Identificadas no Capítulo 2:

A aplicação estratégica de tecnologias educacionais e ferramentas visuais é uma resposta direta a várias dificuldades:

- **Dificuldades Cognitivas** (Seção 2.2)**:** As ferramentas visuais e simuladores reduzem a carga cognitiva inicial, tornando conceitos abstratos (como loops, condicionais, variáveis) mais tangíveis. Isso é fundamental para estudantes com dificuldades em matemática e interpretação, pois o erro se torna visível e mais fácil de compreender e corrigir.
- **Falta de Experiência Prévia e Desalinhamento de Expectativas** (Seção 2.3)**:** Para estudantes sem contato prévio com programação, o início com ferramentas visuais ou ambientes simplificados (como Visualg) oferece uma "rampa de entrada" suave, tornando a experiência menos intimidante e mais engajadora, alinhando as expectativas com um caminho de aprendizado gradual.
- **Desafios de Autodiagnóstico** (Seção 2.6)**:** O feedback imediato e a representação visual das ferramentas permitem que o aluno identifique

rapidamente onde e por que seu código não funciona, desenvolvendo a capacidade de autodiagnóstico de maneira mais eficiente e menos frustrante.

- **Autoeficácia e Senso de Pertencimento** (Seção 2.4)**:** O sucesso inicial proporcionado por ferramentas mais acessíveis aumenta a autoeficácia do aluno, que se sente capaz de programar. A possibilidade de criar algo funcional rapidamente, mesmo que simples, gera motivação e um senso de pertencimento à comunidade de programadores.

Uma forma de integrar essas ferramentas de maneira pedagógica é por meio de uma sequência estruturada de aprendizagem, conforme apresentado na **Sugestão Prática 5 – Jornada da Lógica**.

Sugestão Prática 5: Jornada da Lógica

Recomendo o uso de iniciativas como por exemplo o desafio do código. Isto pode facilitar seu trabalho.

A partir das dificuldades identificadas no capítulo anterior e das estratégias pedagógicas discutidas nesta seção, o Quadro 1 apresenta uma síntese das principais relações entre desafios de aprendizagem e possíveis intervenções didáticas no ensino de Pensamento Computacional e Algoritmos.

Quadro 1 – Síntese das dificuldades de aprendizagem e estratégias pedagógicas propostas

Dificuldades identificadas no ensino de Pensamento Computacional e Algoritmos	Estratégias pedagógicas sugeridas	Contribuições para o processo de aprendizagem
Dificuldades na compreensão de conceitos abstratos da programação.	Uso de **ferramentas visuais de programação**, como Scratch e Portugol Studio.	Favorece a visualização do fluxo lógico dos algoritmos e facilita a compreensão de estruturas fundamentais como sequência, decisão e repetição.
Falta de experiência prévia com programação.	**Atividades desplugadas** para introdução de conceitos computacionais.	Permite que os estudantes compreendam conceitos de lógica e algoritmos por meio de situações concretas e acessíveis.
Baixa autoconfiança e insegurança diante da disciplina.	**Aprendizagem colaborativa** e resolução de problemas em grupo.	Estimula a troca de conhecimentos entre os estudantes, reduz a ansiedade e fortalece o processo de construção coletiva do conhecimento.
Dificuldade em relacionar teoria e prática.	**Projetos interdisciplinares** envolvendo situações do mundo real.	Promove a contextualização dos conteúdos e reforça a aplicação prática dos conhecimentos adquiridos.
Desmotivação ou percepção da disciplina como excessivamente técnica.	**Uso de plataformas digitais interativas** e desafios progressivos.	Torna o processo de aprendizagem mais dinâmico, estimulando o engajamento e a participação ativa dos estudantes.

Fonte: Elaborado pelo Autor (2026)

A síntese apresentada evidencia que diferentes dificuldades de aprendizagem podem ser enfrentadas por meio da diversificação de estratégias pedagógicas, reforçando a importância de práticas docentes que integrem metodologias ativas, recursos digitais e situações contextualizadas de aprendizagem.

CONSIDERAÇÕES FINAIS DO CAPÍTULO

As estratégias apresentadas ao longo deste capítulo evidenciam que o ensino de programação na Educação Profissional e Tecnológica demanda abordagens capazes de articular teoria, prática e mediação pedagógica qualificada. As metodologias ativas, quando fundamentadas em princípios da aprendizagem significativa e apoiadas por tecnologias educacionais adequadas, ampliam as possibilidades de engajamento e favorecem processos de aprendizagem mais contextualizados e autônomos. Ao diversificar práticas, reorganizar tempos pedagógicos e incorporar investigação, colaboração e resolução de problemas, o docente fortalece a construção de sentido e contribui para superar os desafios identificados na formação inicial dos estudantes. Assim,

o conjunto de ferramentas discutidas neste capítulo constitui uma base sólida para a implementação de práticas pedagógicas coerentes com os princípios formativos da EPT e com as demandas contemporâneas do ensino de programação.

CAPÍTULO 4 – AVALIAÇÕES E INTERVENÇÕES

A avaliação constitui uma dimensão essencial do processo educativo. Contudo, quando limitada à verificação pontual de acertos e erros, ela tende a ocultar os verdadeiros obstáculos enfrentados pelos estudantes, especialmente em disciplinas como Pensamento Computacional e Algoritmos, onde as dificuldades cognitivas nem sempre são evidentes nas respostas objetivas. Avaliar com foco exclusivamente na execução correta de algoritmos pode reforçar a exclusão de estudantes que, embora não dominem ainda a sintaxe, estejam em processo de construção do raciocínio lógico.

Na perspectiva da Educação Profissional e Tecnológica (EPT), a avaliação deve ser compreendida como um processo contínuo, formativo e emancipador, que contribua para o desenvolvimento integral do estudante e para a transformação de sua prática formativa (Frigotto; Ramos; Ciavatta, 2005). Isso implica repensar os instrumentos e métodos avaliativos utilizados nas disciplinas técnicas, buscando práticas mais coerentes com os princípios da formação omnilateral.

4.1 AVALIAÇÃO DIAGNÓSTICA: PONTO DE PARTIDA

Uma avaliação diagnóstica no início da disciplina é fundamental para identificar o nível de conhecimento prévio, familiaridade com lógica matemática, fluência em leitura e capacidade de abstração. Essa etapa não deve ter caráter classificatório, mas sim orientar o planejamento docente e a definição de estratégias diferenciadas de ensino.

Aplicações como formulários *online*, *quizzes* e desafios simples podem ser utilizados para levantar dados de forma leve e acessível. Além disso, conversar com os estudantes sobre suas experiências com tecnologia pode revelar muito sobre suas trajetórias e suas expectativas.

4.2 Avaliação formativa: feedback como aprendizagem

A avaliação formativa propõe o acompanhamento constante do progresso do aluno. Ao invés de esperar provas formais, o professor pode adotar práticas como:

- Revisões de código em duplas;
- Avaliações por meio de rubricas claras (ex: clareza, lógica, organização);
- Quizzes com feedback automático;
- Entregas incrementais de projetos de programação;
- Diários de aprendizagem (o que aprendi, o que ainda tenho dúvida).

Essa abordagem transforma o erro em oportunidade de aprendizagem. Como defendem Luckesi (2011) e Anastasiou e Alves (2006), a avaliação deve deixar de ser instrumento de punição para se tornar instrumento de mediação pedagógica.

4.3 Avaliação prática e contextualizada

Na disciplina de Pensamento Computacional e Algoritmos, a avaliação prática baseada em situações-problema permite observar se o estudante compreende as estruturas lógicas e consegue aplicá-las para resolver desafios reais. Por exemplo:

- Criar um algoritmo para um sistema de fila de atendimento;
- Desenvolver um pseudocódigo para cálculo de consumo de energia;
- Simular, com fluxogramas, um processo logístico básico;
- Ler e realizar testes de mesa/depuração em algoritmos preparados pelos professores ou pelos colegas;

Essas tarefas mobilizam o conhecimento técnico e favorecem a percepção de sentido, conectando o conteúdo à prática profissional. Esse tipo de proposta também está em

consonância com a abordagem sociotécnica da EPT, que integra trabalho, ciência e cultura.

A diversidade de estratégias avaliativas discutidas nas seções anteriores evidencia que a avaliação no ensino de Pensamento Computacional e Algoritmos deve contemplar diferentes momentos do processo de aprendizagem. O **Quadro 2** apresenta uma síntese dos principais tipos de avaliação e de suas possíveis aplicações no contexto da disciplina.

Quadro 2 – Síntese dos tipos de avaliação no ensino de Pensamento Computacional e Algoritmos

Tipo de avaliação	Objetivo pedagógico	Exemplos de aplicação
Diagnóstica	Identificar conhecimentos prévios e possíveis dificuldades iniciais dos estudantes.	Questionários iniciais, quizzes online, desafios simples de lógica, conversas sobre experiências com tecnologia.
Formativa	Acompanhar o processo de aprendizagem e oferecer feedback contínuo.	Revisões de código, rubricas avaliativas, quizzes com feedback automático, entregas incrementais de projetos.
Prática e contextualizada	Verificar a capacidade de aplicar conceitos de lógica e algoritmos em situações reais.	Desenvolvimento de pseudocódigos, criação de algoritmos para problemas do cotidiano, elaboração de fluxogramas e testes de mesa.

Fonte: Elaborado pelo autor (2026).

A articulação entre diferentes formas de avaliação contribui para uma compreensão mais ampla do processo de aprendizagem dos estudantes. A partir dessas informações, o professor pode planejar intervenções pedagógicas mais adequadas às necessidades da turma.

4.4 Intervenções pedagógicas e estratégias de apoio

Além da avaliação, é necessário prever intervenções sistemáticas para apoiar os estudantes em dificuldade. Entre as estratégias sugeridas estão:

- **Monitorias** com estudantes de semestres avançados;
- **Rodas de conversa** para escuta ativa de desafios enfrentados;
- **Material de apoio acessível**, como tutoriais em vídeo, exercícios resolvidos e guias rápidos;
- **Oficinas de reforço** em contraturno;
- **Uso de pares programadores**, onde dois estudantes trabalham juntos no desenvolvimento de código, trocando ideias e aprendendo mutuamente (*pair programming*).

Como destacam Santos (2019) e Vieira et al. (2019), práticas pedagógicas orientadas pela escuta e pela mediação crítica promovem não só o aprendizado técnico, mas, também, o fortalecimento da autoestima acadêmica e da permanência escolar.

4.5 Avaliação institucional e ações coletivas

Cabe ressaltar que a avaliação do processo formativo não deve se restringir à sala de aula. A coordenação pedagógica, os supervisores e demais setores institucionais também têm papel fundamental na construção de mecanismos institucionais de apoio à aprendizagem, como programas de nivelamento, integração entre disciplinas e análise de dados sobre evasão.

Tais medidas colaboram para que a avaliação transcenda o indivíduo e se torne um instrumento de aperfeiçoamento coletivo da prática educativa.

Considerações Finais do Capítulo

As reflexões apresentadas neste capítulo reafirmam a importância de práticas avaliativas comprometidas com o desenvolvimento dos estudantes, especialmente em contextos de elevada evasão nos cursos técnicos de informática.

A integração entre avaliação diagnóstica, formativa e prática, aliada a estratégias de escuta e apoio, pode contribuir significativamente para a permanência e o êxito acadêmico dos estudantes na disciplina de Pensamento Computacional e Algoritmos.

Capítulo 5 – Boas Práticas Recomendadas

A jornada de aprendizagem de lógica de programação, conforme explorado nos capítulos anteriores, revela-se um pilar fundamental na formação dos estudantes do curso técnico em informática, mas, paradoxalmente, também um dos maiores desafios à sua permanência e êxito escolar. Ao identificar as complexas dificuldades enfrentadas, que vão desde obstáculos cognitivos e metodológicos até questões de autoeficácia, desalinhamento de expectativas e o impacto de fatores institucionais e sociais, este e-book propõe um olhar aprofundado e soluções articuladas.

Neste capítulo, as reflexões teóricas e as análises empíricas são condensadas em um conjunto de Boas Práticas Recomendadas. O objetivo é oferecer um guia prático para docentes, supervisores e gestores educacionais do IFSul – Campus Gravataí e de instituições de Educação Profissional e Tecnológica (EPT) com desafios semelhantes. Estas práticas são desenhadas para transformar a disciplina de Pensamento Computacional e Algoritmos de um "filtro" em uma ponte para o sucesso, fomentando uma formação omnilateral que integra trabalho, ciência, tecnologia e cultura, e que reconhece o estudante como sujeito ativo e protagonista de sua trajetória.

As recomendações são agrupadas em cinco eixos principais, que abordam o planejamento, as metodologias, a avaliação, o apoio ao estudante e o fortalecimento institucional, buscando oferecer um ciclo virtuoso de melhoria contínua.

5.1 Planejamento e Diagnóstico Inicial

A fase inicial do processo de ensino Eaprendizagem é determinante para o sucesso do estudante. Um planejamento cuidadoso, precedido por um diagnóstico abrangente, permite ao professor compreender a realidade da turma, suas lacunas e seus potenciais,

ajustando a abordagem pedagógica desde o primeiro contato. Isso minimiza o impacto da falta de experiência prévia e alinha as expectativas, reduzindo a desmotivação inicial.

Conexão com os Capítulos Anteriores: Este pilar atende diretamente à Falta de Experiência Prévia e Desalinhamento de Expectativas (Seção 2.3) e as Dificuldades Cognitivas (Seção 2.2), conforme discutido no Capítulo 2. Complementa a Avaliação Diagnóstica (Seção 4.1) detalhada no Capítulo 4, garantindo que o ensino seja adaptado à realidade do aluno.

Práticas recomendadas: Mapeamento Holístico do Estudante – Ir além do diagnóstico de conhecimentos técnicos, buscando compreender o perfil socioeconômico, as experiências de vida, as expectativas e as dificuldades cognitivas de cada aluno.

Dicas Práticas:

Realizar uma avaliação diagnóstica não classificatória no início do curso, levantando conhecimentos prévios em lógica (não necessariamente de programação), matemática básica e interpretação de texto, utilizando questionários mistos (abertos e fechados) e dinâmicas interativas.

Promover momentos de acolhimento e escuta ativa (ex: rodas de conversa, entrevistas rápidas) para entender as trajetórias dos estudantes, seus interesses e suas expectativas em relação ao curso e à área de TI.

Planejar o componente curricular de forma flexível, com módulos introdutórios que nivelam o conhecimento da turma e exemplos que dialoguem com a realidade e os interesses dos estudantes. Considerar a implementação do "Mapeamento de Saberes e Expectativas" (Seção 4.1).

5.2 Metodologias que Promovem o Protagonismo

As metodologias ativas são essenciais para engajar os estudantes e desenvolver as habilidades práticas e cognitivas necessárias para a lógica de programação. Ao colocar o aluno no centro do processo, essas abordagens transformam a sala de aula em um ambiente dinâmico de construção do conhecimento, onde o erro é uma etapa da aprendizagem e a colaboração é incentivada. Isso fortalece a autoeficácia e o senso de pertencimento, combatendo a desmotivação.

Conexão com os Capítulos Anteriores: Este ponto é a aplicação prática das Metodologias Ativas e Aprendizagem Significativa (seção 3.1), Currículo Integrado e Interdisciplinaridade (Seção 3.2) e A Pesquisa como Princípio Educativo (Seção 3.3), detalhadas no Capítulo 3. Responde às Dificuldades Cognitivas (Seção 2.2), à Falta de Experiência Prévia (Seção 2.3) e à Autoeficácia e Senso de Pertencimento (Seção 2.4).

Práticas recomendadas: Aprendizagem Ativa e Contextualizada – Utilizar abordagens que estimulem o aluno a investigar, criar e resolver problemas reais, conectando a lógica de programação a seu cotidiano e a outras áreas do conhecimento.

Dicas Práticas:

Aplicar metodologias ativas como Aprendizagem Baseada em Problemas (ABP), gamificação e ensino por pares, por meio de desafios e projetos que simulem situações do mundo real (ex: "Cidade Sustentável" ou "Solução para um Desafio Escolar" – Seções 3.1 e 3.4).

Estimular a pesquisa como princípio educativo, propondo que os estudantes investiguem e comparem diferentes soluções algorítmicas para um mesmo problema (ex: "Mini-Investigação: Algoritmos para Otimização" – Seção 3.3).

Utilizar a Teoria da Atividade e a Aprendizagem Expansiva para organizar projetos colaborativos, nos quais os estudantes identifiquem contradições e criem novas práticas

e conhecimentos, transformando a sala de aula em um espaço de "co-configuração" (Engeström, 2001).

Fomentar a interdisciplinaridade, conectando a lógica de programação com outras disciplinas do currículo e projetos que explorem diferentes áreas do conhecimento (ex: "Monitoramento Ambiental Inteligente" – Seção 3.2).

5.3 Avaliação como Ferramenta de Aprendizagem

A avaliação deve transcender seu caráter classificatório para se tornar um instrumento de mediação pedagógica, focado no processo de aprendizagem e no desenvolvimento do estudante. Um sistema avaliativo que valoriza o feedback contínuo, a prática contextualizada e a autoavaliação contribui para o autodiagnóstico, fortalece a autoeficácia e alinha a prática pedagógica com as demandas reais da programação.

Conexão com os Capítulos Anteriores: Este eixo está diretamente ligado à necessidade de um Alinhamento entre Avaliação e Prática (Capítulo 2, introdução) e às propostas de Avaliação Formativa (Seção 4.2) **e** Avaliação Prática e Contextualizada (Seção 4.3) discutidas no Capítulo 4. Responde aos Desafios de Autodiagnóstico (Seção 2.6) e à Autoeficácia e Senso de Pertencimento (Seção 2.4)**.**

Práticas recomendadas: Avaliação Formativa e Centrada no Processo. Implementar um sistema de avaliação que forneça feedback contínuo, transparente e construtivo, valorizando o percurso do aprendizado e a aplicação prática do conhecimento.

Dicas Práticas:

Priorizar a avaliação formativa, com *feedback* contínuo e específico sobre o processo de raciocínio, não apenas sobre o resultado final. Implementar a estratégia de "Revisão de Código e *Feedback* Orientado" (Seção 4.2).

Utilizar instrumentos variados que permitam ao aluno demonstrar sua compreensão da lógica de diferentes maneiras (ex: mapas mentais de algoritmos, fluxogramas, pseudocódigo, resolução de bugs em códigos existentes, diários de bordo – Seção 2.6).

Garantir espaço para autoavaliação e coavaliação, estimulando a reflexão crítica sobre o próprio processo de aprendizagem e o dos colegas.

Implementar avaliações práticas e contextualizadas por meio de problemas reais ou simulados, como um "Mini-Hackathon de Problemas Locais" (Seção 4.3), que simulem os desafios do ambiente profissional.

5.4 Estratégias de Apoio e Acompanhamento

A complexidade da lógica de programação e a diversidade do perfil discente exigem um sistema de apoio robusto que transcenda o ambiente da sala de aula. Intervenções pedagógicas e estratégias de acompanhamento personalizadas são cruciais para detectar e resolver dificuldades precocemente, combatendo a desmotivação e reduzindo as taxas de reprovação e evasão. O suporte contínuo fortalece o senso de pertencimento e a autoeficácia do estudante.

Conexão com os Capítulos Anteriores: Este eixo aborda diretamente as Altas Taxas de Reprovação e Evasão (Seção 2.1), os Desafios de Autodiagnóstico (Seção 2.6) e a Autoeficácia e Senso de Pertencimento (Seção 2.4), conforme analisado no Capítulo 2. É a operacionalização das Intervenções Pedagógicas e Estratégias de Apoio (Seção 4.4) propostas no Capítulo 4.

Práticas recomendadas: Rede de Suporte Multidimensional. Oferecer um leque diversificado de apoio (acadêmico, psicopedagógico, social) que atenda às necessidades individuais dos estudantes, promovendo a identificação e a superação de suas dificuldades de forma proativa.

Dicas Práticas:

Organizar monitorias com estudantes de semestres avançados (ex: "Clube de Lógica e Programação" – Seção 4.4), incentivando o "pair programming" e a troca de conhecimentos.

Disponibilizar materiais de apoio acessíveis e multiformatos (vídeos curtos, tutoriais interativos, guias visuais, listas de exercícios resolvidas passo a passo), utilizando tecnologias educacionais e ferramentas visuais (Seção 3.5).

Criar oficinas de reforço em contraturno para tópicos específicos, focando nas dificuldades mais comuns identificadas nos diagnósticos e nas avaliações formativas.

Fomentar rodas de conversa regulares para escuta ativa dos desafios enfrentados pelos estudantes, criando um espaço seguro para verbalização das dificuldades e troca de experiências (Seção 4.4).

Encaminhar estudantes para apoio psicopedagógico quando identificadas questões emocionais ou de saúde mental que impactam o desempenho acadêmico, conforme a observação de estudantes evadidos.

5.5 Fortalecimento Institucional

A efetividade das boas práticas pedagógicas e das estratégias de apoio depende de um ambiente institucional coeso e engajado. O fortalecimento institucional envolve a articulação entre diferentes setores, a análise sistemática de dados de permanência e êxito e a promoção de uma cultura de melhoria contínua. Sem um compromisso coletivo, os esforços individuais dos professores podem ser mitigados por fatores sistêmicos, como instabilidade docente ou falta de recursos.

Conexão com os Capítulos Anteriores: Este é o pilar que consolida a Avaliação Institucional e Ações Coletivas (Seção 4.5) do Capítulo 4 e busca combater os Fatores

Extrínsecos e o Contexto da Evasão, como a instabilidade dos professores ou a falta de bolsas, que foram apontados por docentes e estudantes evadidos (Seção 2.4).

Prática recomendadas: Gestão Colaborativa para Permanência e Êxito. Estabelecer mecanismos institucionais que promovam a comunicação, a análise de dados e a implementação coordenada de políticas e ações que visem à permanência e ao sucesso acadêmico de forma transversal.

Dicas Práticas:

Promover o diálogo contínuo entre docentes, coordenação pedagógica, supervisores e gestão escolar, por meio de reuniões periódicas e canais de comunicação eficazes.

Criar um Comitê de Acompanhamento da Permanência e Êxito (CAPE) (conforme sugerido na Seção 4.5), composto por representantes de diferentes setores, para analisar os dados de reprovação e evasão da disciplina de Pensamento Computacional e Algoritmos e propor ações integradas.

Integrar a análise dos dados de evasão e desempenho diretamente nos planos de ação pedagógica da instituição, garantindo que as intervenções sejam baseadas em evidências.

Estimular e apoiar a formação continuada dos professores em metodologias ativas, tecnologias educacionais, avaliação formativa e temas psicopedagógicos, reconhecendo que a atualização docente é chave para a inovação.

Garantir a estabilidade e o suporte aos docentes, buscando minimizar a rotatividade e oferecendo os recursos (financeiros, materiais, tecnológicos) necessários para à implementação das boas práticas, como o acesso a bolsas estudantis e infraestrutura adequada, que foram pontos de vulnerabilidade identificados nos resultados da pesquisa realizada na dissertação (Seção 2.4).

Considerações Finais do Capítulo

As discussões apresentadas ao longo deste capítulo evidenciam a relevância do produto educacional enquanto instrumento de apoio ao trabalho docente e ao aprimoramento das práticas pedagógicas no ensino de programação. A análise do material produzido demonstra que ele dialoga diretamente com as dificuldades identificadas nos capítulos anteriores e oferece orientações concretas, embasadas teoricamente, para a implementação de práticas mais articuladas, dinâmicas e contextualizadas. Além disso, o produto educacional se mostra coerente com os princípios da Educação Profissional e Tecnológica, valorizando a integração entre teoria e prática, a autonomia dos estudantes e a mediação docente qualificada. Assim, o conjunto de elementos apresentados neste capítulo reforça a consistência conceitual e a aplicabilidade do material, destacando seu potencial de contribuir para o fortalecimento das práticas pedagógicas na formação inicial em programação.

CONSIDERAÇÕES FINAIS

A disciplina de Pensamento Computacional e Algoritmos, central na formação em tecnologia da informação, apresenta-se frequentemente como um ponto crítico para a permanência e o sucesso dos estudantes em cursos técnicos. Este e-book, fruto de uma pesquisa aprofundada, buscou não apenas diagnosticar as complexas dificuldades que levam à evasão – desde barreiras cognitivas e metodológicas até fatores socioeconômicos e institucionais – mas, sobretudo, oferecer caminhos concretos para sua superação.

As **Boas Práticas Recomendadas,** apresentadas aqui, abrangendo planejamento, metodologias ativas, avaliação formativa, estratégias de apoio e fortalecimento institucional, convergem para uma visão unificada: transformar a lógica de programação de um "filtro" em uma ponte sólida para o êxito estudantil. Essas práticas, embasadas nos princípios da Educação Profissional e Tecnológica (EPT), visam uma formação que integra teoria e prática, ciência e cultura, capacitando o estudante não apenas com habilidades técnicas, mas com autonomia, pensamento crítico e um profundo senso de propósito.

Este material foi concebido como uma ferramenta de apoio institucional e pedagógico. Ele representa um convite a docentes, supervisores e gestores para reavaliar e aprimorar suas abordagens, construindo um ambiente de aprendizagem mais acolhedor, engajador e eficaz. Ao adotar uma postura proativa e integrada, focada no desenvolvimento integral do estudante, as instituições podem contribuir significativamente para reduzir a evasão e para formar profissionais qualificados, prontos para os desafios de um mundo do trabalho em constante evolução.

Que as reflexões e as sugestões aqui contidas inspirem a construção de práticas pedagógicas inovadoras, impulsionando a permanência e o sucesso acadêmico, e reafirmando o papel transformador da educação em lógica de programação para o desenvolvimento socioeconômico de nossa comunidade.

REFERÊNCIAS

ALHARBI, N. et al. Debugging difficulties among novice programming students: a mixed methods study. arXiv preprint arXiv:2104.06710, 2021.

AUSUBEL, David Paul. Aquisição e retenção de conhecimentos: uma perspectiva cognitivista. Lisboa: Plátano Editora, 2003.

ALVES, Lynn. Game on: jogos eletrônicos na escola e na vida. Salvador: SEED, 2006.

BARROWS, Howard S. A taxonomy of problem-based learning methods. *Medical Education*, v. 20, n. 6, p. 481–486, 1986.

BERGMANN, Jonathan; SAMS, Aaron. *Sala de aula invertida: uma metodologia ativa de aprendizagem*. Rio de Janeiro: LTC, 2016.

BRASIL. Base Nacional Comum Curricular. Brasília: Ministério da Educação, 2018. Disponível em: https://www.gov.br/mec/pt-br/escolas-conectadas/BNCCComputaoCompletodiagramado.pdf Acesso em: 16 abr. 2026.

CHEAH, Chin Soon. Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming. Contemporary Educational Technology, v. 13, n. 4, 2021.

CORAC, Relatório de cruzamento entre reprovações em Algoritmos e Programação e procedimentos de desligamento (2020-2024). 2025a.

CORAC, Reprovação na disciplina Algoritmos e Programação (2020-2024). 2025b.

DETERDING, Sebastian et al. From game design elements to gamefulness: defining “gamification”. In: INTERNATIONAL ACADEMIC MINDTREK CONFERENCE, 15., 2011, Tampere. Proceedings… New York: ACM, 2011. p. 9–15.

DEWEY, John. *Experiência e educação*. 2. ed. São Paulo: Nacional, 1979.

DEMIRCI, Sinem et al. Learning difficulties of introductory data science students. Proceedings of the International Association for Statistical Education (IASE), 2022.

DETERS, Janice Inês; SILVA, Júlia Marques Carvalho da; MIRANDA, Elisângela Maschio de; FERNANDES, Anita Maria da Rocha. O Desafio de Trabalhar com Alunos Repetentes na Disciplina de Algoritmos e Programação. Simpósio Brasileiro de Informática na Educação SBIE, 2008. Disponível em:

https://sol.sbc.org.br/index.php/wei/article/view/27746.

ENGESTRÖM, Yrjö. Expansive Learning at Work: Toward an activity theoretical reconceptualization. Journal of Education and Work, v. 14, n. 1, p. 133–156, 2001.

FREIRE, Paulo. Pedagogia da autonomia: saberes necessários à prática educativa. 26. ed. São Paulo: Paz e Terra, 1998.

FRIGOTTO, Gaudêncio; CIAVATTA, Maria; RAMOS, Marise. A educação profissional e tecnológica na sociedade do conhecimento: a perspectiva do trabalho como princípio educativo. Brasília: MEC/SETEC, 2005.

GIRAFFA, Lucia Maria Martins; Mora, Michael da Costa. *Evasão na Disciplina de Algoritmo e Programação: Um Estudo a partir dos Fatores Intervenientes na Perspectiva do Aluno*. Porto Alegre: PUCRS, 2013. Disponível em : Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do aluno | Congresos CLABES (utp.ac.pa)

KAPP, Karl M. The gamification of learning and instruction: game-based methods and strategies for training and education. San Francisco: Pfeiffer, 2012.

LUCKESI, Cipriano Carlos. Avaliação da aprendizagem escolar: estudos e proposições. 22. ed. São Paulo: Cortez, 2011.

MORAIS, Ceres Germanna Braga; NETO, Francisco Milton Mendes; OSÓRIO, António José Meneses. Dificuldades e desafios do processo de aprendizagem de algoritmos e programação no ensino superior: uma revisão sistemática de literatura. 2020.

MTAHO, Adam B.; MSELLE, Leonard J. Difficulties in Learning the Data Structures Course. Tanzania Journal of Informatics and Communication Technology, v. 2, n. 1, 2023.

PAIXÃO, Márcia Valéria; PINTO, Leandro Rafael. Educação Profissional e Tecnológica: a Teoria da Atividade como possível caminho metodológico. Debates em Educação, v. 12, n. 27, p. 1–21, 2020. DOI: https://doi.org/10.28998/2175-6600.2020v12n27p1-21.

PAPERT, Seymour. A máquina das crianças: repensando a escola na era da informática. Tradução de Sandra Costa. Porto Alegre: Artes Médicas, 1993.

PATITSAS, Elizabeth et al. Exploring the "cold start" problem in CS1. International Journal of STEM Education, v. 9, 2022.

REIS FILHO, Heitor Boeira dos. O potencial das inovações pedagógicas na redução da evasão em cursos técnicos de informática: perspectivas para a lógica de programação. In: IX MOVACI – Mostra Venâncio-airense de Cultura e Inovação, 9., 2025, Venâncio Aires. Evento realizado de 17 a 19 de setembro e 1º de setembro de 2025. Venâncio Aires: Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense (IFSul), Câmpus Venâncio Aires, 2025.

ROBINS, Anthony; ROUNTREE, Jennifer; ROUNTREE, Nathan. Learning and teaching programming: A review and discussion. Computer Science Education, v. 13, n. 2, 2003.

SANTOS, Amanda et al. Gamificação no ensino de algoritmos: um relato de experiência. Revista Novas Tecnologias na Educação (RENOTE), v. 18, n. 2, 2020. Disponível em: https://seer.ufrgs.br/index.php/renote/article/view/108290. Acesso em: 06 jul. 2025.

SANTOS, André. A Teoria da Atividade como possibilidade de enfrentamento das contradições da Educação Profissional e Tecnológica. Debates em Educação, v. 11, n. 24, p. 466–485, 2019.

SILVA, Júlia L.; FERNANDES, Vinícius C. Metodologias ativas no ensino de lógica de programação: uma revisão de literatura. RENOTE, v. 19, n. 1, 2021. DOI: https://doi.org/10.22456/1679-1916.118785.

SEDA, Maria Cecília et al. Autoeficácia e ensino de programação: um estudo com estudantes do ensino superior. Journal of Computing Sciences in Colleges, v. 38, n. 2, 2023.

SILVA, A. L. et al. O currículo integrado. Florianópolis: IFSC, 2016.

VIEIRA, J. A.; VIEIRA, M. M. M.; PASQUALLI, R.; CASTAMAN, A. S. Ensino com pesquisa na educação profissional e tecnológica: noções, perspectivas e desafios. Revista Tempos e Espaços em Educação, v. 12, n. 29, p. 279–298, 2019. DOI: https://doi.org/10.20952/revtee.v12i29.9306.

VYGOTSKY, Lev Semenovich. *A formação social da mente*. 7. ed. São Paulo: Martins Fontes, 2007.

WING, Jeannette M. Computational thinking. Communications of the ACM, v. 49, n. 3, p. 33–35, 2006.

APÊNDICE A

SEQUÊNCIA DIDÁTICA "INSTRUÇÕES PARA O SANDUÍCHE PERFEITO"

Objetivo: Desenvolver nos estudantes a capacidade de decompor problemas cotidianos em sequências claras e ordenadas de instruções, compreendendo a importância da precisão, do sequenciamento lógico e da clareza na elaboração de algoritmos.

Materiais: papel, lápis e canetas de diversas cores, opcional: cartões com instruções prontas para intervenção do professor , quadro branco (para discussão final).

Competências Desenvolvidas:

- Pensamento sequencial;
- Clareza e precisão na escrita de instruções;
- Decomposição de problemas;
- Depuração e melhoria incremental;
- Trabalho em pares;
- Comunicação clara.

Tempo total previsto: 45 min

Passos:

1. Desafio (10 min): Peça aos estudantes para escreverem instruções detalhadas para fazer um sanduíche, como se estivessem ensinando um robô ou uma pessoa que nunca fez isso.
2. Teste e Depuração (20 min): Alunos trocam as instruções e um "executa" o algoritmo do colega literalmente, sem inferências. O "robô" (aluno que executa) aponta as falhas (instruções ambíguas, passos ausentes, falta de ingredientes etc.).
3. Discussão (15 min): Reflita sobre a importância da clareza, precisão e sequenciamento lógico para que o "programa" funcione. Conecte com a necessidade de detalhar os problemas e interpretar as instruções.

Como avaliar o aluno?

A avaliação será realizada de forma formativa, por meio da observação da participação dos estudantes, da análise da clareza e da organização lógica das instruções produzidas, da identificação de falhas durante a etapa de teste e depuração, bem como da capacidade de revisão e reflexão sobre o processo. Será considerado, ainda, o envolvimento nas discussões em grupo e a contribuição para o trabalho colaborativo.

Que instrumentos usar?

Serão utilizados observação direta, lista de verificação, registro das produções iniciais e revisadas, roda de conversa final e autoavaliação breve dos estudantes.

Os estudantes devem tentar fazer isto em linguagem natural, na sequência pode ser introduzido o conceito de português estruturado ou fluxograma, mas ainda usando papel.

SEQUÊNCIA DIDÁTICA "DE ONDE VIEMOS, PARA ONDE VAMOS"

Objetivo: Apresentar aos estudantes a estrutura do curso e contextualizar a disciplina de Lógica de Programação dentro da formação técnica em informática, destacando suas relações com outras disciplinas e aplicações profissionais, de modo a favorecer o engajamento e a compreensão da relevância dos conteúdos a serem estudados.

Materiais: projetor multimídia, computador e sala de aula ou auditório, dependendo do número de estudantes.

Competências desenvolvidas:

- Compreensão do percurso formativo;
- Pensamento computacional;
- Visão sistêmica da área de Informática;
- Autoeficácia e confiança para aprender;
- Projeto de vida e orientação profissional;
- Comunicação e participação;
- Aprendizagem por observação e inspiração.

Tempo total previsto: 45min

Passos:

I. **Currículo Visual (15 min):** Apresente aos estudantes a estrutura curricular do curso, destacando disciplinas que serão cursadas posteriormente e suas relações com conceitos de Pensamento Computacional e de programação, como estruturas de repetição, variáveis e lógica de controle, evidenciando como esses conceitos são utilizados em diferentes áreas do desenvolvimento de software.

II. **Depoimentos Rápidos (20 min):** Sempre que possível, convide brevemente ex-alunos bem-sucedidos ou profissionais da área para compartilhar como a base em lógica foi essencial em suas carreiras. Isso ajuda a combater a percepção de que "programação é para gênios" e fortalece a autoeficácia dos estudantes.

III. **Sessão de Perguntas (15 min):** Abra espaço para que os estudantes apresentem dúvidas e questionamentos sobre a disciplina, suas aplicações práticas e as possibilidades de atuação profissional na área de informática.

Como avaliar o aluno?

Você pode usar critérios tais como:

1. Participação e engajamento;
2. Compreensão da estrutura do curso;
3. Reflexão sobre a carreira;
4. Comunicação e expressão.

Que instrumentos posso usar?

- Observação direta durante a aula
- Registro de participação em debate e sessão de perguntas
- Pequena atividade diagnóstica ao final, como:

- escrever o que entendeu sobre a disciplina;
- indicar uma área da informática que despertou interesse;
- relacionar um conceito de programação com algo prático.

OUTRAS SUGESTÕES DE ATIVIDADES

a) Atividades de Pesquisa Interdisciplinar

Proposta: Incentive os estudantes a conversarem com colegas de disciplinas mais avançadas (ex: Programação Web, Banco de Dados) para entenderem como o pensamento computacional é aplicado em seus projetos. Eles podem, então, apresentar essas descobertas aos demais.

Benefícios: Esta prática traz uma visão mais concreta das fases futuras do curso, ajuda na integração entre turmas e permite que os estudantes "visualizem" a progressão de suas habilidades.

b) Atividades de Recepção com Palestras de Profissionais

Proposta: Organize palestras ou rodas de conversa com profissionais da área de TI logo no início do curso ou da disciplina. Eles podem compartilhar suas experiências, as diversas atuações no mundo do trabalho e a importância da lógica para qualquer papel.

Benefícios: Essas palestras são valiosas para motivar e ajudar os estudantes que ingressam sem um conhecimento aprofundado da área a se encontrarem dentro das possibilidades da profissão, alinhando expectativas e inspirando.

SEQUÊNCIA DIDÁTICA "TECNOLOGIA PARA O BEM"

Objetivo: Promover um momento de diálogo com os estudantes, possibilitando o esclarecimento de dúvidas sobre a disciplina de Lógica de Programação, suas aplicações práticas e as possibilidades de atuação profissional na área de informática, contribuindo para o engajamento e a compreensão da relevância dos conteúdos abordados no curso.

Materiais: Projetor multimídia (caso sejam utilizados slides ou exemplos visuais), computador, quadro branco ou lousa, marcadores ou giz, papel e caneta para registro de dúvidas (opcional).

Competências desenvolvidas:

- Pensamento crítico sobre o uso da tecnologia e seus impactos sociais;
- Compreensão inicial de Lógica de Programação aplicada a problemas reais;
- Capacidade de identificar problemas e propor soluções tecnológicas;
- Decomposição de problemas em etapas lógicas;
- Comunicação oral e argumentação em debates e apresentações;
- Trabalho colaborativo em pequenos grupos;
- Consciência ética e social sobre o desenvolvimento e uso de tecnologias;
- Engajamento e percepção de relevância da disciplina para a formação técnica e profissional.

Tempo total previsto: 1h e 10 min

Passos:

Notícia e Debate (15 min): Iniciar a aula com uma notícia atual sobre o impacto da tecnologia (positiva ou negativa) na sociedade (ex: uso de IA, privacidade de dados, algoritmos de redes sociais, carros autônomos). Promova um breve debate.

Identificação de Problemas (20 min): Em pequenos grupos, os estudantes identificam problemas em sua comunidade ou no mundo que poderiam ser minimizados ou resolvidos com soluções tecnológicas. Ex: otimização de rotas de transporte público, sistema de alerta para desastres naturais, aplicativo de doação de alimentos.

Proposta Algorítmica (25 min): Cada grupo esboça a lógica de um possível software ou sistema para uma das soluções propostas, descrevendo os passos lógicos e as interações. O foco é no raciocínio e na estrutura, não na codificação.

Apresentação e Reflexão (10 min): Os grupos apresentam suas ideias e discutem as implicações éticas, sociais e práticas de suas propostas, fomentando o senso crítico.

Como avaliar o aluno?

A avaliação será realizada de forma formativa, com base na participação dos estudantes nas discussões, na identificação de problemas reais, na coerência da proposta algorítmica elaborada em grupo e na capacidade de reflexão sobre as implicações éticas, sociais e práticas das soluções apresentadas. Também será considerada a habilidade de comunicar ideias com clareza e de trabalhar colaborativamente.

Que instrumentos posso usar?

Observação direta, lista de verificação, registro da participação, roteiro de análise da proposta, autoavaliação.

SEQUÊNCIA DIDÁTICA "PROCESSOS DO DIA A DIA EM FLUXOGRAMA"

Objetivo: Desenvolver a capacidade de decomposição de problemas, raciocínio lógico e abstração, transformando processos familiares em representações algorítmicas visuais.

Materiais: Cartolinas, canetas coloridas, post-its, ou ferramentas online de criação de fluxogramas (ex: Draw.io).

Competências desenvolvidas:

- Pensamento lógico e sequencial;
- Decomposição de problemas;
- Abstração;
- Representação visual de processos;
- Identificação de decisões e desvios;
- Clareza na comunicação de instruções;
- Trabalho colaborativo;
- Depuração de erros em algoritmos visuais.

Tempo total previsto: 1h e 35min

Passos:

Escolha do Processo (15 min): Peça aos estudantes, em grupos pequenos, para escolherem um processo simples do dia a dia (ex: fazer um café, atravessar a rua, emprestar um livro da biblioteca, resolver uma equação matemática simples).

Decomposição (20 min): Oriente-os a listar todas as etapas necessárias para completar esse processo, de forma sequencial e detalhada.

Construção do Fluxograma (30 min): Utilizando símbolos básicos de fluxograma (início/fim, processo, decisão, entrada/saída), os grupos deverão desenhar o algoritmo visual do processo escolhido. Enfatize a necessidade de clareza, à existência de desvios (decisões) e repetições (loops) se aplicável.

"Execução" e Depuração (20 min): Grupos trocam os fluxogramas e tentam "executar" o processo desenhado por outro grupo, identificando possíveis falhas lógicas ou ambiguidades.

Discussão (10 min): Promova uma reflexão sobre como a clareza e a precisão do fluxograma são essenciais para que qualquer um possa seguir as instruções, e como isso se relaciona com à escrita de um programa. Conecte com à importância da interpretação de problemas para criar a solução correta.

Como avaliar o aluno?

A avaliação deve ser formativa e processual, observando a participação nos grupos, a capacidade de decompor o processo escolhido em etapas lógicas, a coerência do fluxograma produzido, a identificação de falhas durante a troca entre grupos e a reflexão final sobre a relação entre fluxograma e programação.

Que instrumentos posso usar?

Observação direta; lista de verificação; rúbrica de desempenho; análise do fluxograma produzido; registro das interações em grupo; autoavaliação breve; socialização oral das soluções.

SEQUÊNCIA DIDÁTICA: "MEU FUTURO NA TI: DIAGNÓSTICO E REALIDADE"

Objetivo: Avaliar o conhecimento prévio dos estudantes, compreender suas motivações e expectativas, e apresentar uma visão realista do que será exigido na disciplina e no curso.

Materiais: Questionário online (Google Forms), apresentação multimídia, recursos visuais.

Competências desenvolvidas:

- Autoconhecimento acadêmico e profissional: identificar conhecimentos prévios, interesses, expectativas e possíveis caminhos na área de TI.
- Pensamento reflexivo: comparar expectativas pessoais com a realidade do curso e da disciplina.
- Compreensão do percurso formativo: perceber a função da Lógica de Programação dentro da formação técnica.
- Raciocínio lógico inicial: reconhecer a importância da lógica, da persistência e da resolução de problemas.
- Comunicação e expressão: formular dúvidas, expor percepções e participar das discussões.
- Postura investigativa: responder ao diagnóstico com sinceridade e analisar o próprio ponto de partida.
- Projetividade: relacionar o curso a possibilidades futuras de atuação profissional.

Tempo total previsto: 1h e 25min

Passos:

1 - Questionário Diagnóstico e de Expectativas (20 min): Nos primeiros dias de aula, aplique um questionário (anônimo, se possível, para maior sinceridade) que inclua:

- Perguntas sobre experiência prévia com computação, lógica, matemática e programação.
- Motivações para escolher o curso e a disciplina (o que esperam aprender, onde se veem atuando).
- Percepções sobre o que é "programar" ou "pensar computacionalmente".
- Nível de conforto com o ambiente virtual de aprendizagem, caso haja.

2 - Devolutiva e Realidade do Curso (30 min): Apresente os resultados agregados do questionário (sem identificar os estudantes). Use estes dados como ponto de partida para discutir a realidade da disciplina e do curso:

- Desmistifique a programação, explicando que não é preciso ser um "gênio", mas exige dedicação e raciocínio lógico.
- Apresente o currículo de forma detalhada, mostrando a progressão e a interconexão das disciplinas, reforçando o papel da lógica.

- Discuta as dificuldades comuns (como as abordadas neste capítulo) e como a disciplina abordará cada uma delas.
- Destaque a importância da persistência e do processo de "tentativa e erro".

3 - "Dia a Dia do Desenvolvedor" (20 min): Mostre vídeos curtos ou convide um profissional (virtualmente, se necessário) para descrever a rotina de um desenvolvedor, os desafios e as recompensas. Inclua exemplos de como a lógica é aplicada em tarefas cotidianas de TI.

4 **- Sessão de Perguntas e Respostas (15 min):** Abra espaço para que os estudantes tirem dúvidas, expressem suas preocupações e reajustem suas expectativas.

Como avaliar o aluno?

Observando a participação nas atividades, a sinceridade e consistência das respostas ao questionário, a capacidade de relacionar expectativas com a realidade apresentada e o envolvimento na sessão de perguntas e respostas. Mais do que “acertar” respostas, importa verificar se o estudante consegue reconhecer seu ponto de partida, compreender melhor a proposta do curso e ajustar suas expectativas de forma consciente.

Que instrumentos posso usar?

- Questionário diagnóstico inicial
- Registro de participação nas discussões
- Observação direta do engajamento
- Lista de verificação com itens como: participou, refletiu, expressou dúvidas, compreendeu a proposta do curso
- Produção breve de fechamento, como um parágrafo ou resposta escrita sobre o que aprendeu com a devolutiva
- Autoavaliação simples sobre expectativas e grau de compreensão do curso

Benefícios: Esta abordagem permite ao professor um diagnóstico preciso da turma, adaptando a metodologia. Para os estudantes, ajuda a construir expectativas mais realistas, reduz a ansiedade de quem não tem experiência e aumenta a motivação de quem entende a relevância da base.

ESTRATÉGIA: "PROGRAMANDO JUNTOS, CRESCENDO JUNTOS"

Objetivo: Fomentar a colaboração, reduzir à insegurança individual, fortalecer a autoeficácia e construir um senso de comunidade.

O que resolve: Insegurança, desmotivação, falta de busca por apoio contínuo, percepção de que a programação é difícil demais.

Materiais: Ambientes de desenvolvimento integrados (IDEs) que permitem colaboração em tempo real (ex: VS Code Live Share, Repl.it), exercícios práticos, projetos em duplas.

Tempo total previsto: o tempo dependerá do nível dos exercícios que serão escolhidos para a execução da estratégia, mas pode varias de 45 a 90 min.

Passos:

1 - Pair Programming (Programação em Pares) (Contínuo): Implementar a programação em pares regularmente. Um dos professores entrevistados que já adota esta prática comentou que: *"Juntar os alunos em duplas alunos com poucas dificuldades, com alunos mais fracos. Melhora a aprendizagem e integração entre alunos."*. Esta técnica consiste em dois alunos trabalhando juntos em um único computador: um "piloto" que escreve o código e um "navegador" que revisa, sugere e pensa estrategicamente. Trocar os papéis frequentemente.

Benefícios: Compartilhamento de conhecimento, detecção precoce de erros, desenvolvimento de habilidades de comunicação e trabalho em equipe, redução da frustração individual e aumento da confiança.

2 - Círculos de Ajuda e Mentoria (Semanal/Quinzenal): Organizar encontros informais (presenciais ou online) onde estudantes mais experientes (ou mesmo os que estão se destacando na turma) possam mentorar colegas com mais dificuldades. Pode ser estruturado como "Rodas de Conversa .

Benefícios: Cria uma rede de apoio natural, normaliza a busca por ajuda, e permite que os "mentores" consolidem seu próprio conhecimento ao explicá-lo.

ESTRATÉGIA: "A AVALIAÇÃO QUE ENSINA"

Objetivo: Transformar a avaliação em uma ferramenta de aprendizado e diagnóstico, que mensure não apenas o conhecimento, mas também as habilidades de resolução de problemas e o processo de raciocínio.

O que resolve: Falta de clareza sobre o que é esperado, foco excessivo em sintaxe, dificuldade em diagnosticar lacunas reais, desmotivação por avaliações punitivas.

Materiais: Rubricas de avaliação claras, plataformas de submissão de código com feedback automático (se possível), problemas práticos contextualizados, ferramentas de versionamento de código (ex: Git/GitHub).

Tempo total previsto: Como se trata de uma estratégia de aplicação contínua, não existe um tempo total previsto.

Passos:

1 - Avaliações Incrementais e Contínuas (Durante a Unidade): Em vez de provas únicas e finais, dividir a avaliação em pequenas entregas. Cada entrega pode ser um pequeno módulo de um projeto, um exercício de depuração (debugging) ou a criação de um fluxograma para um problema.

- **Exemplo:** Após a introdução de estruturas condicionais, o aluno deve implementar um miniprograma que use essas estruturas para resolver um problema simples do cotidiano, com foco na lógica, e não apenas na sintaxe.

2 - Feedback Qualitativo e Oportuno (Após cada entrega): Fornecer feedback detalhado e rápido, focando não apenas no que está errado, mas no ***porquê*** e ***como*** o aluno pode melhorar. Utilizar rubricas transparentes que contemplem:

- **lógica:** O algoritmo resolve o problema proposto? É eficiente?
- **Estrutura e Organização:** O código (ou pseudocódigo/fluxograma) está bem organizado e legível?
- **Interpretação:** O aluno demonstrou compreender o enunciado do problema?
- **Depuração:** O aluno consegue identificar e corrigir erros?

3 - Avaliação por Pares e Autoavaliação: Incentivar os estudantes a avaliar o trabalho dos colegas (e receber feedback deles) e a refletir sobre seu próprio processo de aprendizado. Isso promove a metacognição e o senso de responsabilidade sobre o próprio aprendizado.

4 - **Desafios de Depuração (Debugging Challenges):** Criar exercícios onde o objetivo não é escrever um código do zero, mas corrigir um código com erros (bugs) intencionais. Isso desenvolve a capacidade analítica e de resolução de problemas, habilidades cruciais na programação real.

ESTRATÉGIA: "DIÁRIO DE BORDO DO CODIFICADOR E DEBUGGING COLABORATIVO"

Objetivo: Promover a metacognição e a capacidade de autoanálise, permitindo que os estudantes identifiquem a origem de seus erros e compreendam melhor seu próprio processo de aprendizado.

O que resolve: Falta de clareza sobre onde estão as dificuldades, confusão entre sintaxe e lógica, busca ineficaz por apoio.

Materiais: Caderno/documento digital para o diário, problemas de programação, ferramentas de depuração (debuggers), colega de dupla/grupo.

Tempo total previsto: Como se trata de uma estratégia de aplicação contínua, não existe um tempo total previsto.

Passos:

1 - Diário de Bordo do Codificador (Contínuo): Após cada atividade de programação (exercício, projeto), o aluno preenche um pequeno "diário" com as seguintes perguntas:

- Qual era o objetivo do exercício/projeto?
- Quais foram os principais desafios que encontrei? (Descreva especificamente: foi a lógica, a sintaxe, a interpretação do problema, a ferramenta?)
- Que estratégias usei para superar esses desafios? Quais funcionaram e quais não?
- Se ainda tenho dificuldades, quais são as perguntas exatas que faria ao professor ou a um colega? (Incentivar a formulação de perguntas claras, evitando "Não entendi nada").
- que aprendi com essa atividade, mesmo com os erros?

Benefícios: A escrita força a reflexão e a organização do pensamento, ajudando o aluno a articular suas dificuldades e a rastrear seu próprio progresso. O professor pode, ocasionalmente, coletar esses diários para identificar padrões de dificuldades na turma.

2 - Debugging Colaborativo (Sessões Periódicas): Em duplas (idealmente formadas por estudantes com níveis ligeiramente diferentes, ou através da metodologia de pair programming), os estudantes trabalham para encontrar e corrigir erros em um código pré-definido (com bugs intencionais, como na Sugestão Prática da seção 2.5) ou em um código que um deles escreveu e que não está funcionando.

- Um aluno "pilota" o debugger, explicando o que está acontecendo linha a linha.
- outro aluno "navega", questionando as suposições, propondo hipóteses e ajudando a rastrear a lógica.

Benefícios: Ao explicar para o outro e ter que verbalizar o raciocínio, o aluno desenvolve a autoconsciência de seu processo mental. A colaboração ajuda a desmistificar o "bug" e a torná-lo um desafio superável, fortalecendo a autoeficácia e o senso de pertencimento.

SUGESTÃO PRÁTICA: DESAFIO DE PROGRAMAÇÃO GAMIFICADO – "CIDADE SUSTENTÁVEL"

Para fomentar a aprendizagem significativa e o uso de metodologias ativas, propõe-se um desafio gamificado que conecta a lógica de programação a problemas sociais e ambientais.

Estratégia: "Cidade Sustentável: Programando Soluções Reais"

Objetivo: Aplicar conceitos de lógica de programação para resolver problemas do cotidiano em um contexto gamificado, promovendo a colaboração, o pensamento crítico e a formação para a cidadania digital.

O que resolve: Dificuldades cognitivas (abstração, lógica), falta de contextualização, baixa motivação, desalinhamento de expectativas.

Materiais: Plataforma online de gamificação ou um sistema de pontuação/badges (ex: Google Classroom com add-ons de gamificação, Kahoot! para quizzes), ferramenta de pseudocódigo/fluxogramas (ex: Visualg, Lucidchart) ou ambiente de programação em blocos (ex: Scratch, Blockly, App inventor), problemas do cotidiano da comunidade escolar ou local.

Tempo total previsto: 4 semanas

Passos:

1. **Apresentação do Cenário (30 min):** Apresentar o "desafio da Cidade Sustentável". Dividir os estudantes em equipes (3-4 integrantes). Cada equipe será uma "Startup de Tecnologia Cidadã".
2. **Identificação de Problemas e Missões (45 min):** Cada equipe escolhe um problema real da comunidade que possa ser resolvido ou minimizado com lógica computacional (ex: sistema de coleta de lixo inteligente, monitoramento de consumo de água, organização de caronas solidárias). O professor fornece "missões" com pontos por cumprimento (ex: 100 pontos para identificar o problema, 200 para o algoritmo inicial).
3. **Desenvolvimento do Algoritmo (Semanas):**
    - **Fase 1 – Pseudocódigo/Fluxograma (Missão 1):** Equipes desenvolvem o algoritmo da solução usando pseudocódigo ou fluxogramas. O professor atua como mentor, fornecendo feedback formativo e desafios adicionais ("mini-missões") para aprimorar a lógica.
    - **Fase 2 – Prototipagem em Blocos/Linguagem Simples (Missão 2):** Se a turma já tiver alguma base, as equipes podem traduzir parte do algoritmo para uma linguagem de programação em blocos ou uma

linguagem mais simples, focando na implementação das estruturas lógicas.

- **Fase 3 – Testes e Debugging (Missão 3):** Equipes testam seus protótipos, identificam e corrigem erros. Feedback entre pares é incentivado, gerando pontos adicionais.

4. **Apresentação Final e Votação (60 min):** As equipes apresentam suas soluções e os desafios que enfrentaram. Uma votação (gamificada) pode premiar a solução mais criativa, mais eficiente ou com maior impacto social.
5. **Reflexão Crítica (30 min):** Discutir como a lógica e a programação podem ser usadas para o bem comum e as implicações éticas das soluções propostas.

SUGESTÃO PRÁTICA: PROJETO INTERDISCIPLINAR – "MONITORAMENTO AMBIENTAL INTELIGENTE"

Um projeto que integra a lógica de programação com outras disciplinas pode oferecer um contexto rico para a aplicação dos conhecimentos e o desenvolvimento de habilidades.

Estratégia: "Monitoramento Ambiental Inteligente: Um Projeto Interdisciplinar"

- **Objetivo:** Desenvolver uma solução de monitoramento ambiental simples utilizando lógica de programação, integrando conhecimentos de Biologia, Química, Física e Matemática.
- **O que resolve:** Descontextualização da lógica, falta de percepção da utilidade prática, fragmentação do conhecimento, dificuldade em ver a aplicabilidade da lógica em problemas reais.
- **Materiais:** Sensores básicos (temperatura, umidade, luminosidade – podem ser simulados ou de baixo custo, como os de kits Arduino), softwares de simulação ou ambientes de programação em blocos/texto, dados climáticos ou ambientais (reais ou simulados).

Tempo total previsto: 5 semanas

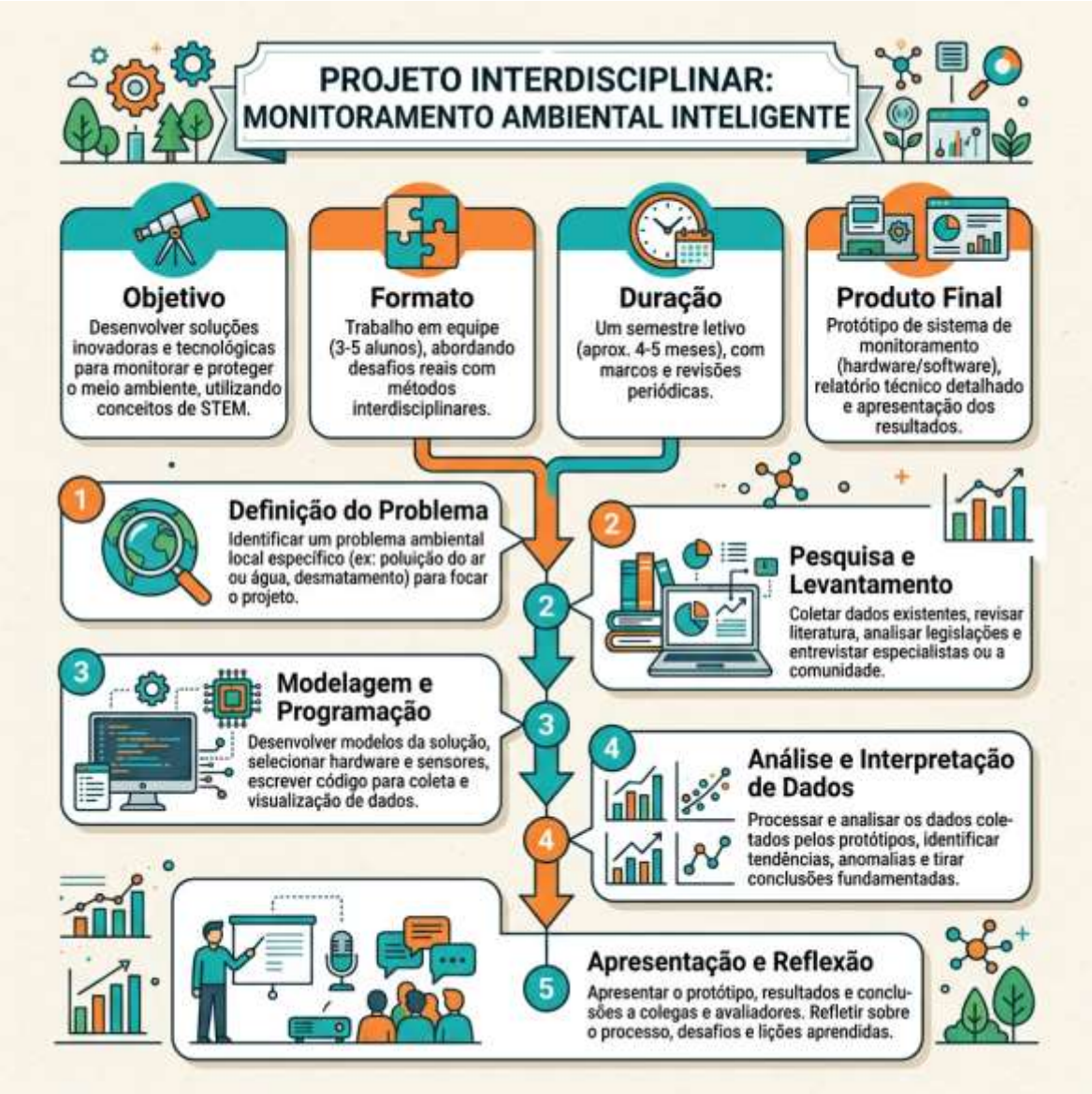
Passos:

1. **Definição do Problema (com áreas parceiras - 60 min):** Em colaboração com professores de Biologia (ou Química/Física/Matemática), definir um problema ambiental relevante para a comunidade ou para a escola (ex: monitorar a temperatura de uma estufa, a umidade do solo em uma horta escolar, a qualidade do ar em uma sala).
2. **Pesquisa e Levantamento (em grupos - Semanal):** Os estudantes pesquisam (orientados pelos professores de cada disciplina) sobre:
    - **Biologia/Química:** Quais parâmetros ambientais são importantes? Qual a faixa ideal para plantas/humanos? Como esses parâmetros se relacionam?
    - **Física/Matemática:** Como os sensores funcionam? Como converter dados brutos em informações úteis? Quais cálculos são necessários?
    - **Programação:** Quais estruturas lógicas são necessárias para ler os dados, comparar com limites, e emitir alertas?
3. **Modelagem e Programação da Lógica (Semanas):**
    - **Desenho do Sistema:** Alunos criam um fluxograma ou pseudocódigo do sistema de monitoramento, definindo as variáveis, condições e ações.
    - **Implementação:** Programam a lógica em um ambiente adequado (simulação, blocos ou linguagem de texto, dependendo do nível).

- **Integração:** Se houver hardware, integram o software aos sensores para leitura de dados.

4. **Análise e Interpretação de Dados (com áreas parceiras - 45 min):** Com os dados coletados (reais ou simulados), os estudantes aplicam os conhecimentos das outras disciplinas para analisar as informações, identificar tendências e propor melhorias.
5. **Apresentação e Reflexão (45 min):** Os grupos apresentam suas soluções, discutindo tanto a lógica de programação quanto os conhecimentos das outras áreas. A reflexão final foca na importância da integração do saber.

O Infográfico Projeto Interdisciplinar: **Monitoramento Ambiental Inteligente**, traz um resumo visual da prática sugerida.



SUGESTÃO PRÁTICA: "MINI-INVESTIGAÇÃO: ALGORITMOS PARA OTIMIZAÇÃO"

Propõe-se uma atividade de mini-investigação onde os estudantes exploram e comparam diferentes algoritmos para resolver um mesmo problema, desenvolvendo habilidades de pesquisa, análise e pensamento crítico.

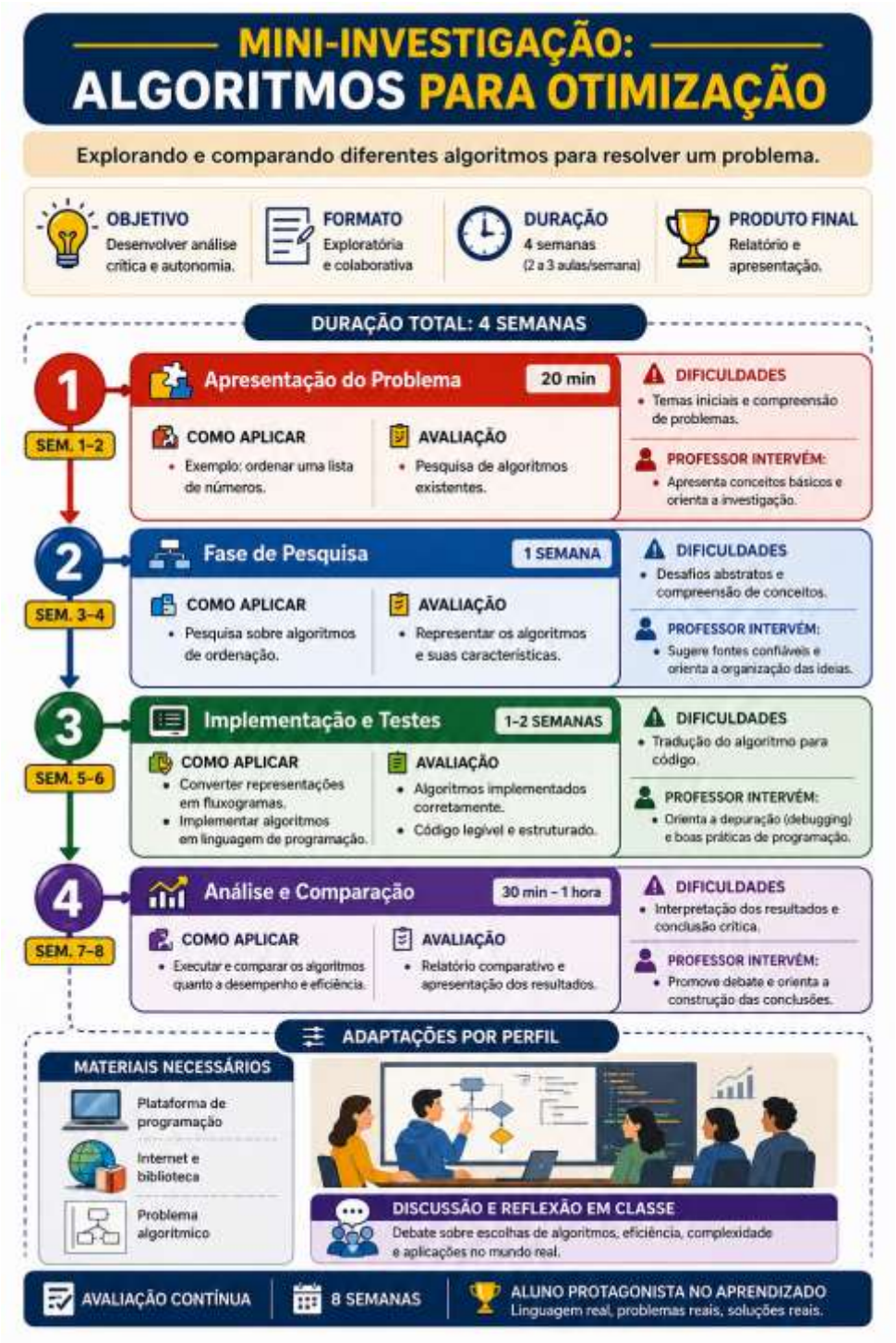
Estratégia: "Mini-Investigação: Algoritmos para Otimização"

- **Objetivo:** Incentivar os estudantes a pesquisar, implementar e comparar a eficiência de diferentes algoritmos para um problema específico, desenvolvendo a autonomia intelectual e a capacidade de análise.
- **O que resolve:** Memorização passiva, falta de pensamento crítico sobre a "melhor" solução, dificuldades em autodiagnóstico sobre a eficiência de um código.
- **Materiais:** Plataforma de programação online (ex: Repl.it, VSCode), ambiente de pesquisa (internet, biblioteca), problema com múltiplas soluções algorítmicas (ex: ordenação de dados, busca, cálculo de Fibonacci).

Tempo total previsto: 4 semanas

- **Passos:**
    1. **Apresentação do Problema (20 min):** Apresentar um problema de programação que possua diferentes abordagens algorítmicas para sua solução (ex: "Como ordenar uma lista de números?" – Bubble Sort, Selection Sort, Insertion Sort, etc.).
    2. **Fase de Pesquisa (45 min - 1 semana):** Em grupos (2-3 estudantes), os estudantes pesquisam pelo menos duas abordagens algorítmicas distintas para resolver o problema. Devem entender a lógica de cada um, seus prós e contras teóricos.
    3. **Implementação e Testes (1-2 semanas):** Cada grupo implementa em código (linguagem definida pelo professor ou em pseudocódigo avançado) as duas abordagens pesquisadas. Em seguida, criam um pequeno conjunto de dados de teste para comparar o desempenho (tempo de execução, número de operações) de seus algoritmos.
    4. **Análise e Comparação (30 min - 1 hora):** Os grupos analisam os resultados dos testes e preparam um breve relatório ou apresentação, discutindo:
        - Qual algoritmo foi mais eficiente para os dados testados? Por quê?
        - Quais as vantagens e desvantagens de cada abordagem?
        - Como o tamanho dos dados afeta o desempenho?
        - Quais foram os desafios na implementação e nos testes?
    5. **Discussão em Classe e Reflexão (45 min):** Apresentação dos achados dos grupos e discussão coletiva sobre a importância de escolher o algoritmo certo para cada situação, a relação entre teoria e prática e a complexidade computacional.

O Infográfico **Mini-Investigação: Algoritmos para otimização**, traz um resumo visual da prática sugerida.



Sugestão Prática: Projeto de "Solução para um Desafio Escolar" com Abordagem Expansiva

Propõe-se um projeto de longo prazo onde os estudantes, em equipe, desenvolvem uma solução de software para um desafio real vivenciado na própria instituição de ensino, seguindo o ciclo da aprendizagem expansiva.

Estratégia: "Solução para um Desafio Escolar: Aprendizagem Expansiva na Prática"

- **Objetivo:** Desenvolver soluções de software para problemas reais da comunidade escolar, aplicando a lógica de programação em um ciclo contínuo de identificação de contradições, modelagem e desenvolvimento de novas práticas e conhecimentos.
- **O que resolve:** Descontextualização, falta de propósito, dificuldades cognitivas avançadas, autoeficácia e pertencimento.
- **Materiais:** Ferramentas de prototipagem (ex: Figma, Balsamiq), ambiente de desenvolvimento integrado (IDE), ferramentas de comunicação e colaboração (ex: Trello, Slack), acesso a dados ou informações do problema real, feedback de "stakeholders" (usuários reais do problema na escola).

Tempo total previsto: 14 semanas

- **Passos (Ciclo Expansivo):**
    1. **Identificação de Contradições/Problemas (Semana 1-2):** Em reuniões com a direção, coordenação ou outros setores da escola, os estudantes identificam um problema real que o campus enfrenta e que poderia ser resolvido ou melhorado com uma aplicação de software (ex: sistema de agendamento de laboratórios, gestão de eventos estudantis, portal de comunicação interna).
    2. **Análise e Modelagem (Semana 3-4):** Os grupos analisam profundamente o problema, entrevistam usuários, mapeiam fluxos de trabalho e modelam a solução. Neste ponto, surgem as primeiras contradições: a solução ideal é muito complexa? Os dados são acessíveis? O que falta no conhecimento da equipe para implementar?
    3. **Desenvolvimento da Ferramenta e Novos Conhecimentos (Semanas 5-10):** Os estudantes iniciam o desenvolvimento do protótipo da solução. As contradições identificadas no passo anterior guiam a busca por novos conhecimentos (novas linguagens, frameworks, padrões de design, algoritmos específicos) e a criação de novas práticas de trabalho em equipe. O professor atua como facilitador e consultor, ajudando a "desembalar" essas contradições.
    4. **Testes e Feedback (Semana 11-12):** O protótipo é testado com os usuários reais na escola. O feedback (muitas vezes com novas contradições entre a solução proposta e a necessidade real) é coletado.

5. **Reflexão e Expansão (Semana 13-14):** Os estudantes refletem sobre o processo, os aprendizados, as contradições superadas e as que ainda persistem. O projeto pode ser expandido para uma nova versão, incorporando os aprendizados e novas funcionalidades. Essa reflexão consolida o aprendizado expansivo, mostrando como o conhecimento e as práticas da equipe evoluíram.

A figura **Projeto: Solução para um Desafio Escolar**, traz uma representação gráfica do ciclo de projeto.

SUGESTÃO PRÁTICA: "JORNADA DA LÓGICA: DO BLOCO AO CÓDIGO"

Propõe-se uma sequência didática progressiva baseada na transição entre diferentes formas de representação da lógica (blocos, fluxogramas, pseudocódigo e linguagem de programação), estruturada para reduzir a complexidade inicial e favorecer a construção gradual do pensamento computacional.

ESTRATÉGIA PARA EXECUÇÃO DA: "JORNADA DA LÓGICA: DO BLOCO AO CÓDIGO"

- **Objetivo:** Facilitar a compreensão dos fundamentos da lógica de programação para iniciantes, utilizando ferramentas visuais e de simulação para uma transição suave para a codificação textual.
- **O que resolve:** Dificuldades iniciais com sintaxe, abstração, falta de conhecimento prévio, autodiagnóstico de erros.
- **Materiais:** Plataformas de programação em blocos (ex: Scratch, Blockly), simuladores de fluxogramas (ex: Draw.io, Lucidchart), ambiente de pseudocódigo (ex: Visualg) ou IDE online (ex: Repl.it).

Tempo total previsto: 8 semanas

Passos (Sequência Progressiva):

1. **Blocos (Semanas 1–2)**: Introdução aos conceitos básicos (sequência, decisão e repetição) por meio de programação em blocos. Foco na construção do raciocínio lógico com feedback visual imediato.

2. Fluxogramas (Semanas 3–4): Representação gráfica dos algoritmos desenvolvidos anteriormente.Ênfase na organização e compreensão do fluxo lógico.

3. Pseudocódigo (Semanas 5–6): Conversão da lógica para uma forma textual estruturada. Foco na transição da representação visual para a escrita de algoritmos.

4. Código (Semanas 7–8): Implementação em linguagem de programação real, com apoio de ferramentas. Aplicação da lógica desenvolvida em ambiente de codificação.

Avaliação:

Acompanhamento contínuo com base na:

- Compreensão dos conceitos
- Coerência das soluções
- Capacidade de explicação do raciocínio

O infográfico abaixo detalha o caminho de execução da sequência didática.


JORNADA DA LÓGICA
DO BLOCO AO CÓDIGO
Sequência didática progressiva para ensinar Lógica de Programação com múltiplas representações e foco no pensamento computacional
OBJETIVO
Construir a lógica passo a passo
FORMATO
Progressivo e cumulativo
DURAÇÃO
8 semanas (2 a 4 aulas/semana)
PRODUTO FINAL
Algoritmo em linguagem real
ETAPAS DA JORNADA
1
SEM. 1–2
EXPLORAÇÃO COM BLOCOS
(Scratch | Code.org | Blockly)
4 a 6 AULAS
COMO APLICAR
• Desafios visuais simples
• Trabalho em duplas
AVALIAÇÃO
• Resolução dos desafios
• Explicação da lógica
DIFICULDADES
• Confusão sequência x repetição
PROFESSOR INTERVÉM:
Peça para explicar antes de executar
2
SEM. 3–4
FLUXOGRAMAS
4 AULAS
COMO APLICAR
• Representar soluções dos blocos
• Comparar estratégias
AVALIAÇÃO
• Coerência do fluxo
• Uso correto dos símbolos
DIFICULDADES
• Tradução do visual para o simbólico
PROFESSOR INTERVÉM:
Exemplos coletivos guiados
3
SEM. 5–6
PSEUDOCÓDIGO
(Visualg | Portugol)
4 a 6 AULAS
COMO APLICAR
• Converter fluxogramas
• Modelos iniciais prontos
AVALIAÇÃO
• Clareza da lógica
• Correspondência com o fluxograma
DIFICULDADES
• Dificuldade com a escrita
• Estrutura de comandos
PROFESSOR INTERVÉM:
Exercícios parcialmente prontos
4
SEM. 7–8
CODIFICAÇÃO TEXTUAL
(Python | Replit | VSCode)
4 a 6 AULAS
COM AUTOSSUPORTE
COMO APLICAR
• Reescrever algoritmos
• Ambiente com feedback
AVALIAÇÃO
• Funcionamento do código
• Correção de erros
DIFICULDADES
• Frustração com erros de sintaxe
PROFESSOR INTERVÉM:
Debug guiado (foco na lógica)
5
CONSOLIDAÇÃO
DESAFIO INTEGRADOR
EM GRUPOS
Sistema simples do cotidiano (contextualizado)
COMO APLICAR
• Problema aberto
• Turmas em grupos
AVALIAÇÃO
• Processo + resultado
• Apresentação da solução
OBJETIVO:
Consolidar todo o pensamento construído
ADAPTAÇÕES POR PERFIL
TURMAS COM MAIS DIFICULDADE
• + Tempo nas Etapas 1 e 2
• + Exemplos guiados
TURMAS AVANÇADAS
• Acelerar para Pseudocódigo
• Desafios mais abertos
ADAPTAÇÕES POR PERFIL
TURMAS COM MAIS DIFICULDADE
• + Tempo nas Etapas 1 e 2
• + Exemplos guiados
TURMAS AVANÇADAS
• Acelerar para Pseudocódigo
• Desafios mais abertos
AVALIAÇÃO CONTÍNUA | 8 SEMANAS | ALUNO PROGRAMA EM LINGUAGEM REAL